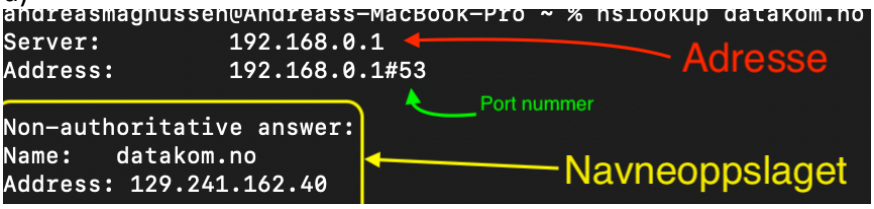
Arbeid 1

Oppgave 1

a)



Ved bruk av nslookup får jeg ikke svar på navnet på min lokale navnetjener. Jeg får bare IPv4 adressen, og også portnummeret som blir brukt på serveren. Jeg har Telenor som min ISP, og jeg har ikke byttet DNS server. Derfor vil jeg tro at det er Telenor sin navnetjener.

b)

```
andreasmagnussen@Andreass-MacBook-Pro ~ % ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 3c:22:fb:aa:ef:0a
        inet6 fe80::1856:ae24:8194:8093%en0 prefixlen 64 secured scopeid 0x6
        inet 192.168.1.6 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

Ut ifra skjermbildet er:

- IPv4 adressen: 192.168.1.6
- IPv6 adressen: fe80::1856:ae24:8194:8093
- MAC adresse: 3c:22:fb:aa:ef:0a

```
andreasmagnussen@Andreass-MacBook-Pro ~ % grep 'nameserver' /etc/resolv.conf
nameserver 192.168.0.1
nameserver 8.8.8.8
```

Skjermbildet viser mine navnetjenere. Som man ser så har jeg 2 navnetjenere listet. Adressen: 8.8.8.8 er en backup navnetjener, i tilfellet den første feiler. Denne navnetjeneren er en av Google sine offentlige navnetjenere.

Som man kan se så er adressen til min primære navnetjener lik som den som har blitt brukt under 'nslookup' i oppgave a.

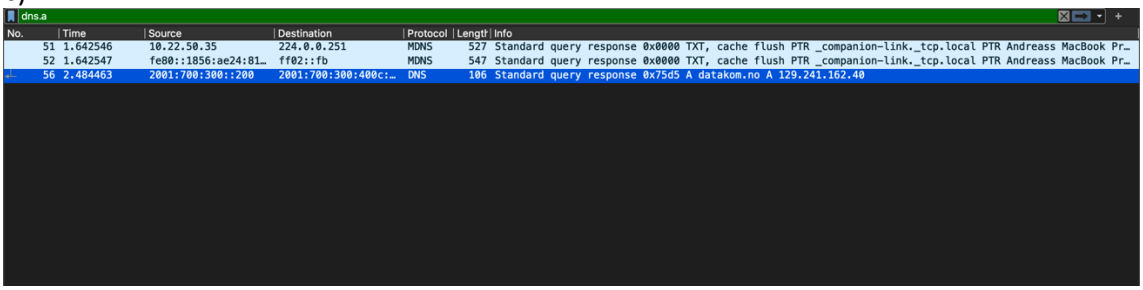
Subnettmasken er: 0xffffff00 (Skjermbildet 1) som er ekvivalent med 255.255.255.0 eller /24 prefix. Det betyr at subnettmasken min har 24 enere.

Min IP adresse er en privat IP-adresse. Dette er fordi adresseblokkene under er satt av for å brukes til private adresser:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

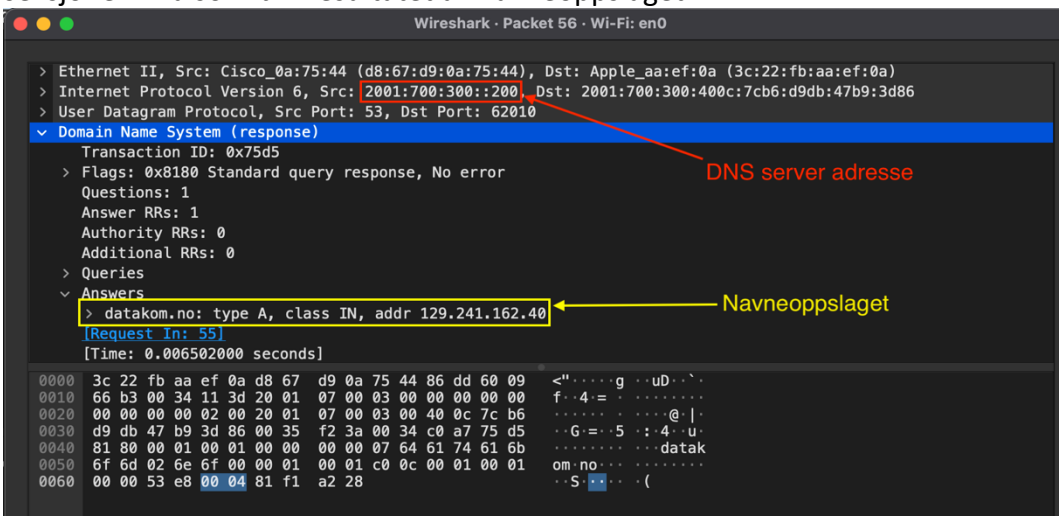
Siden min IP-adresse er: 192.168.1.6, så vil denne adresse gå under den private adresseblokken: 192.168.0.0/16.

c)



Som man kan se på skjermbildet, så har jeg filtrert Wireshark til å vise bare DNS response.

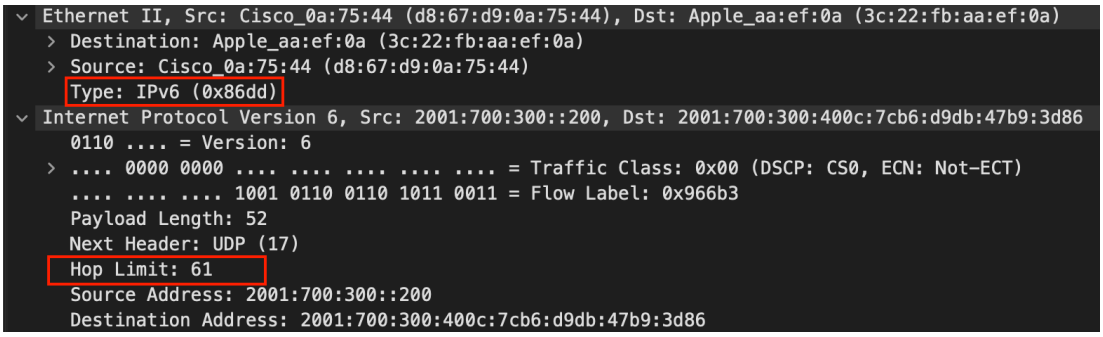
Ved å se nærmere på den markerte responsen, så vil man kunne se under «Answers» seksjonen hva som blir resultatet av navneoppslaget.



Som man kan se er det blitt brukt forskjellig DNS adresse for å gjøre navneoppslaget enn i oppgave a. Dette er fordi jeg har byttet nettverk siden jeg gjorde a oppgaven. Ved å gjøre et raskt søk i terminalen, så vil jeg kunne se at det har blitt brukt den primære DNS adressen som jeg har tilgjengelig på dette nettverket:

```
andreasmagnussen@Andreass-MacBook-Pro ~ % grep 'nameserver' /etc/resolv.conf
nameserver 2001:700:300::200
nameserver 2001:700:300::201
nameserver 129.241.0.200
nameserver 129.241.0.201
```

Som man kan se under, så viser «Type» feltet, at det har blitt brukt IPv6 protokollen. Levetiden på pakken er satt til 61, som man kan se under «Hop Limit»



«Type» feltet brukes til å informere om hvilken Internettprotokoll som har blitt benyttet. Dette gjør at rammen vil bli sendt til riktig protokoll under Internettlaget i TCP/IP stakken når rammen ankommer en node i nettverket. «Hop Limit» feltet benyttes for å sikre at rammen ikke sirkulerer rundt i nettverket til evig tid i tilfellet det er en ruting feil. Derfor settes det et nummer, i dette tilfellet 61, som blir telt ned hver gang rammen blir bearbeidet av en ruter. Når tallet kommer til 0 så vil ruteren forkaste pakken.

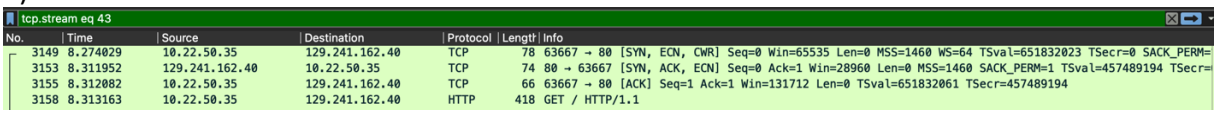
d)

Jeg bruker: macOS Big Sur.

Jeg ser at Wireshark har plukket opp multicast DNS oppslag, som brukes til å gjøre navneoppslag i mindre nettverk.

Oppgave 2

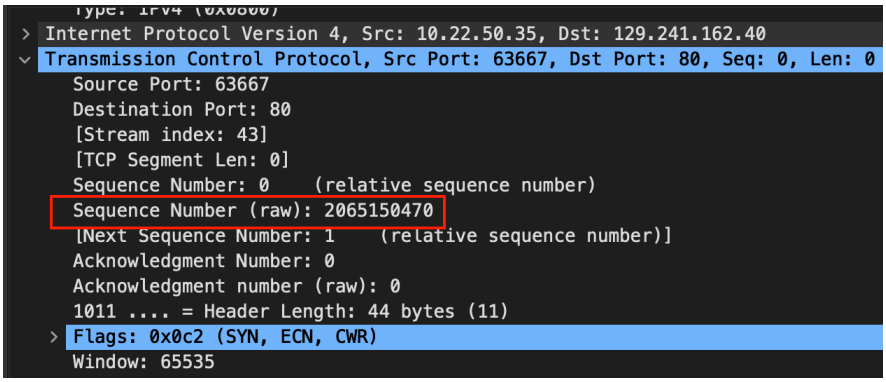
a)

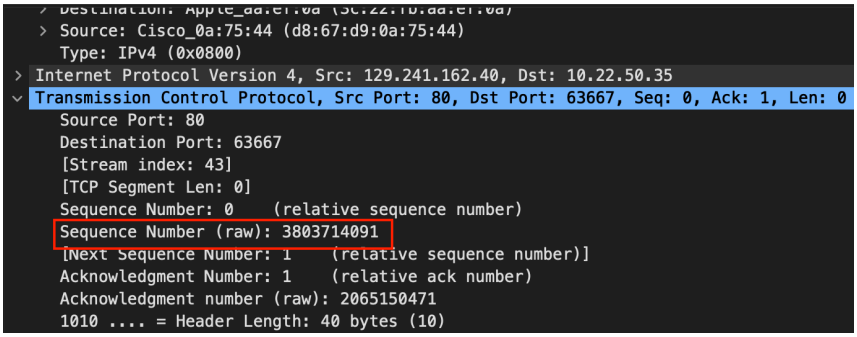


SRC port	→	DST port	FLAGG	Relativt sekvensnummer	Relativt Kvitteringsnummer
63667	→	80	[SYN, ECN, CWR]	0	0
80	→	63667	[SYN, ACK, ECN]	0	1
63667	→	80	[ACK]	1	1

Som man kan se i skjermbildet så er min IP-adresse: 10.22.50.35. Denne IP-adressen er forskjellig fra den IP-adressen jeg fant i oppgave 1b. Grunnen til dette er fordi jeg har koblet meg på et annet nettverk, og dermed har jeg fått en annen IP-adresse.

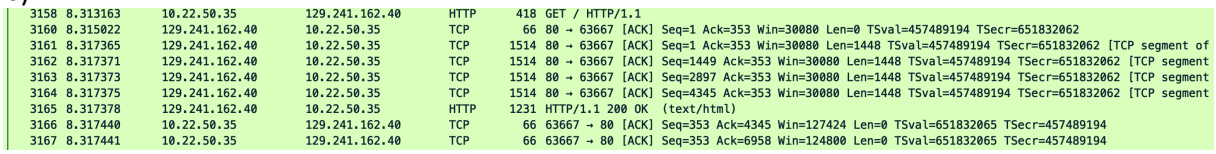
De to bildene under viser hva den faktiske verdien av sekvenstallet er på de to SYN-pakkene som ble sendt.





«Starttallet» som blir satt for sekvensnummeret kalles for ISN (Initial Sequence Number). ISN er et helt tilfeldig tall, som forhindrer potensielle angrep.

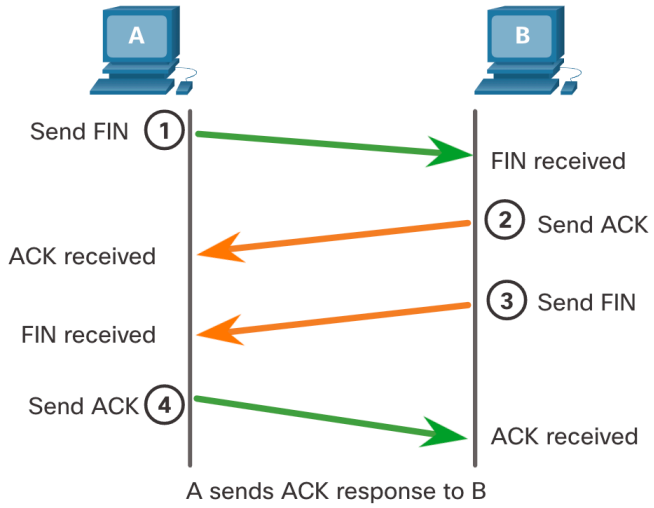
b)



NR	Avsender	DST-port	SEQNR	ACKNR	TCP PAYLOAD i bytes	Eventuelle egne merknader
1	Klient	80	1	1	352	GET-request til webtjener
2	Tjener	63667	1	353		Ingen TCP payload
3	Tjener	63667	1	353	1448	
4	Tjener	63667	1449	353	1448	
5	Tjener	63667	2897	353	1448	
6	Tjener	63667	4345	353	1448	
7	Tjener	63667	5793	353	1165	GET-response 200 OK
8	Klient	80	353	4345		En ack på pakke nr 5
9	Klient	80	353	6958		En ack på GET-responsen. Pakke 7

Etter at 3 way handshake har opprettet kontakt mellom klient og tjener, så sender klienten en GET-request til tjeneren. Her ser vi at når tjeneren skal sende en respons tilbake, så har dataen blitt segmentert inn i 6 forskjellige pakker i transportlaget i TCP/IP stakken. Så har hver pakke fått den nødvendige informasjonen for å kunne sendes individuelt over nettet. Når klienten får pakkene, så vil pakkene vente hos klienten til alle pakkene har ankommet. Til slutt vil TCP protokollen i transportlaget sette sammen pakkene, slik at HTML filen kan vises. 200 betyr at overføringen gikk bra. Etter at klienten har fått alle pakkene som trengs for å sette sammen hele HTML filen, så kan vi se at klienten sender to ACK flagg tilbake til tjeneren. Det siste flagget viser at klienten sier ifra til tjeneren at den har fått hele GET responsen.

c)
For å koble ned en TCP forbindelse, så må FIN flagget bli sendt fra enten klienten eller tjeneren i segmenthodet. Med andre ord, så betyr det at begge kan koble ned en TCP forbindelse. Ved å sende FIN flagget i segment hodet så setter man i gang prosessen som kalles for two-way handshake. Denne prosessen funker ved at enten klienten eller tjeneren sender FIN flagget.



Deretter sender motparten en ACK tilbake, for å vise at den har mottat forespørselen om å koble ned TCP forbindelsen. Motparten sender også en FIN tilbake til A (avsender av første FIN) for å vise at også denne maskinen (B) er klar til å avslutte forbindelsen. Dermed sender A en Ack tilbake til B, for å vise at den har fått bekreftelse om at B også er klar for å avslutte. Med litt andre ord → A sier til B at den vil avslutte, B sier «OK, hadet» til A. A sier «Hadet» til B.

Oppgave 3

a)
En IPv4 adresse består av en nettverksdel og en host del. For å kunne skille mellom disse delene, så må man utføre AND logikk på IPv4 adressen og subnett masken. AND logikk er en metode der man sammenligner én og én bit. Dette vil resultere i nettverksadressen til en gitt IPv4 adresse.

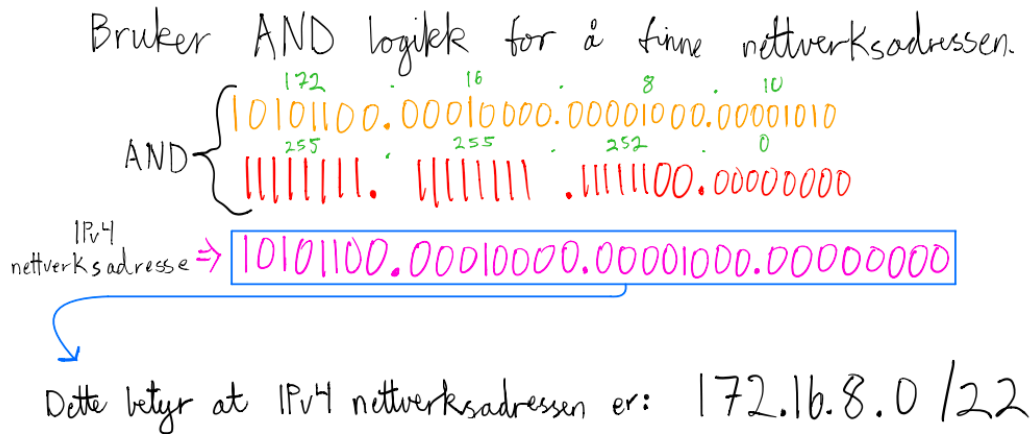
Når AND logikk sammenlikner to bits, så kan man få følgende resultater:

- 1 AND 1 = 1
- 1 AND 0 = 0
- 0 AND 1 = 0
- 0 AND 0 = 0

Her ser man at det er bare 1 AND 1 som kan gi et resultat på 1.

/22 prefix gir en subnett maske tilsvarende: 255.255.252.0, som igjen er ekvivalent med: 11111111.11111111.11111100.00000000 i binær representasjon. IPv4 adressen (172.16.8.10) kan også bli representert av binære tall: 10101100.00010000.00001000.00001010

Utføringen av AND logikk på IPv4 adressen og subnett masken gir:



Broadcast-adressen til et nettverk er alltid den siste adressen i nettverket. Siden dette er et /22 nettverk, så betyr det at våres nettverk er et subnett av et /16 nettverk. Det er fordi /22 har 6 nettverk bits i den tredje oktetten, når vi ser på den binære representasjonen. Den siste nettverks biten viser hvor mye det økes med mellom hvert nettverk. I vårt tilfellet så representerer den siste nettverksbiten 4 i den tredje oktetten. Dette betyr at intervallet i nettverket ovenfor er mellom 172.16.8.0 – 172.16.11.255. Det betyr at broadcast-adressen til dette nettverket er 172.16.11.255/22.

Resultat:
Med dette har vi funnet ut at IPv4 adressen 172.16.8.10 ikke er nettverksadressen, eller broadcast-adressen. Det betyr at den gitte IP-adressen er en adresse som kan tilordnes en node innenfor det aktuelle nettverket.

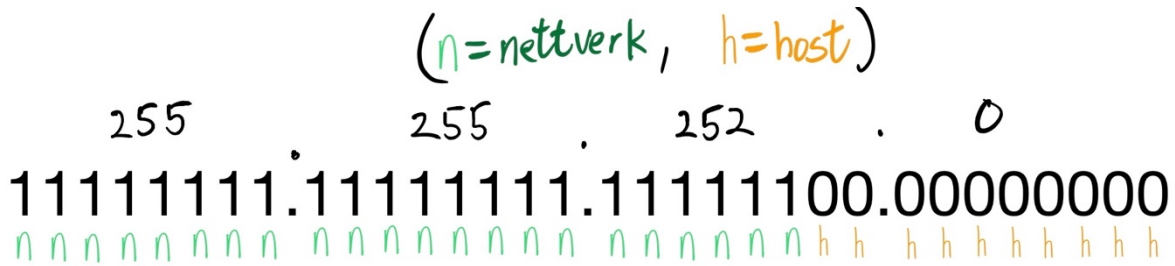
b)
Viser til løsningen i oppgave a:

Nettverksadresse: 172.16.8.0/22
Broadcast-adresse: 172.16.11.255/22
#adresser: 1022

Antall adresser som kan tilordnes noder innenfor /22 nettverket er: $2^{<\text{Antall host bits}>} - 2$.
I et /22 nettverk, så består subnett masken av 10 host bits. Det gir $2^{10} = 1024$ adresser. Siden nettverksadressen og broadcast-adressen er satt, så kan ikke disse tilegnes noder. Derfor blir det:

1024 – 2 = 1022

c)
For å dele 172.16.8.0/22 nettverket inn i 4 like store subnett, så må vi dele opp /22 nettverket til et /24 nettverk. Det er fordi et /22 nettverk gir en subnett maske tilsvarende: 11111111.11111111.11111100.00000000 i binær representasjon. Her kan vi se at nettverksadressen vår allerede er et subnett av et /16 nettverk, siden host bits (0 bits) har blitt lånt og omgjort til nettverk bits (1 bit) i den tredje oktetten.



Den siste nettverksbiten i dette tilfellet viser hvor mye den tredje oktetten skal øke med, mellom nettverksadressene, i vårt tilfelle er dette: 4. Det betyr at intervallet for våres gitte nettverksadresse er mellom 172.16.8.0 (nettverksadressen) – 172.16.11.255 (broadcast-adressen). Når vi nå skal dele opp nettverket i 4 like store adresser, så kommer disse adressene til å ligge innenfor dette intervallet.

Hver host bit som blir lånt av nettverket, gir $2^{<\text{Antall lånte host bits}>}$ subnett. Det betyr at når vi skal dele nettverket vårt opp i 4 like store subnett, så må nettverksdelen låne 2 host bits (0 bits), siden $2^2 = 4$. Derfor blir /22 nettverket vårt omgjort til et /24 nettverk. Dette vil generere følgende subnett:

172.16.8.0/22 → /24

SUBNET	NETTVERKSADRESSE	PREFIX
SUBNETT A	172.16.8.0	24
SUBNETT B	172.16.9.0	24
SUBNETT C	172.16.10.0	24
SUBNETT D	172.16.11.0	24

Alle subnettene vil ha 256 adresser, der 254 adresser kan bli tilegnet noder. Dette er fordi nettverksadressen og broadcast-adressen er fastsatt for hvert subnett.

d)
Ved å analysere kravene til de forskjellige subnettene, så har jeg plassert alle avdelingene i en tabell, sortert etter antall adresser som hver avdeling trenger. Tabellen blir derfor seende slik ut:

AVDELING	KRAV ANTALL ADRESSER
R&D	490
IT	250
PRODUKSJON	120

ADMINISTRASJON	54
HR	25

I tillegg er to adresserom reservert for fremtidig bruk. Disse er:

- 172.16.9.112/28
- 172.16.9.96/29

Dette betyr at bedriften totalt trenger 7 subnets. Det vil nå bli lettere å bruke VLSM til å dele inn nettverket inn i passende subnets. Jeg vil først gå igjennom «utregningen» for hver avdeling, før jeg presenterer resultatene for deloppgavene til slutt.

Nettverksadressen som er gitt er: 172.16.8.0/22

R&D

R&D har et krav til 490 adresser som skal kunne tilegnes noder. Ved å subnette /22 nettverket til to /23 subnets. Så vil dette genere to like store subnets, som vil ha: 2^9 -2 = 510 adresser som kan tilegnes noder.

172.16.8.0/22 → /23

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.8.0	172.16.9.255	23	510
172.16.10.0	172.16.11.255	23	510

Begge disse subnettene oppfyller kravet til R&D med nok adresser, men ved å analysere kravene hele nettverket, så legger man merke til at begge nettverkene som er reservert for fremtidig bruk ligger inne i intervallet til nettverk: 172.16.8.0/23. Det resulterer i at R&D avdelingen ikke kan tilegnes dette nettverket. Derfor blir nettverk 172.16.10.0/23 tilegnet avdeling R&D i stedet. Nettverk 172.16.8.0/23 blir delt opp videre til flere subnets.

IT

IT har et krav til 250 adresser som skal tilegnes noder. Ved å subnette nettverk 172.16.8.0/23 til /24 nettverk, så vil dette generere:

172.16.8.0/23 → /24

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.8.0	172.16.8.255	24	254
172.16.9.0	172.16.9.255	24	254

Begge subnettene oppfyller kravet til IT avdelingen, men igjen så kan man se at de nettverkene som er reservert for fremtidig bruk ligger innenfor nettverk: 172.16.9.0/24. Derfor blir nettverk 172.16.8.0/24 tilegnet IT avdelingen, mens nettverk 172.16.9.0/24 blir delt opp videre.

Produksjon
Produksjon har et krav til 120 adresser som kan tilegnes noder. Ved å subnette nettverk 172.16.9.0/24 til /25 nettverk, så vil dette generere:

172.16.9.0/24 → /25

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.9.0	172.16.9.127	25	126
172.16.9.128	172.16.9.255	25	126

Begge subnettene oppfyller kravet til Produksjon avdelingen, men de fremtidige nettverksadressene ligger i intervallet til nettverk: 172.16.9.0/25. Derfor blir dette nettverket delt opp videre, mens nettverk 172.16.9.128/25 blir tilegnet avdeling Produksjon.

Administrasjon
Administrasjon har et krav på 54 adresser som skal tilegnes noder. Ved å subnette nettverk 172.16.9.0/25 til /26 nettverk, så vil dette generere:

172.16.9.0/25 → /26

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.9.0	172.16.9.63	26	62
172.16.9.64	172.16.9.127	26	62

Siden de reserverte adressene ligger i intervallet til nettverk: 172.16.9.64/26, så vil dette nettverket bli delt opp videre. Derfor blir nettverk 172.16.9.0/26 tilegnet Administrasjon avdelingen.

HR
HR har et krav på 25 adresser som skal kunne tilegnes enheter. Ved å subnette nettverk 172.16.9.64/26 til /27 nettverk, så vil dette generere:

172.16.9.64/26 → /27

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.9.64	172.16.9.95	27	30
172.16.9.96	172.16.9.127	27	30

Siden de reserverte adressene ligger i intervallet til nettverk: 172.16.9.96/27, så blir dette nettverket delt opp videre. Derfor blir nettverket: 172.16.9.64/27 tilegnet HR avdelingen.

Reserverte adresser
Dersom vi tar nettverk: 172.16.9.96/27 og deler dette opp til et /28 nettverk, så vil vi få:

172.16.9.96/27 → /28

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.9.96	172.16.9.111	28	14
172.16.9.112	172.16.9.127	28	14

Her ser vi at subnet 172.16.9.112/28 tilsvarer adresserom I, som er den første reserverte adressen. Derfor blir nettverk: 172.16.9.96/28 delt opp videre.
172.16.9.96/28 → /29

NETTVERKSADRESSE	BROADCAST-ADRESSE	PREFIX	ADRESSER SOM KAN TILEGNES
172.16.9.96	172.16.9.103	29	6
172.16.9.104	172.16.9.111	29	6

Her ser vi at subnet: 172.16.9.96/29 tilsvarer adresserom II, som er den andre reserverte adressen.

Nå har vi totalt delt opp nettverk: 172.16.8.0/22 inn i 8 forskjellige subnets. Dette betyr at kravet om 7 subnets er oppfylt, og vi står igjen med ett subnett til overs.

Oppgave d; I)

AVDELING	NETTVERKSADRESSE	PREFIX
R&D	172.16.10.0	23
IT	172.16.8.0	24
PRODUKSJON	172.16.9.128	25
ADMINISTRASJON	172.16.9.0	26

HR	172.16.9.64	27
ADRESSEROM I (RESERVERT)	172.16.9.112	28
ADRESSEROM II (RESERVERT	172.16.9.96	29

Oppgave d; II)

Administrasjonen: 172.16.9.0/26

BROADCAST-ADRESSE	FØRSTE ADRESSE SOM KAN TILORDNES NODE	SISTE ADRESSE SOM KAN TILORDNES NODE	TOTALE ADRESSER SOM KAN TILORDNES NODER
172.16.9.63	**172.16.9.1**	**172.16.9.62**	**62**

Oppgave d; III)

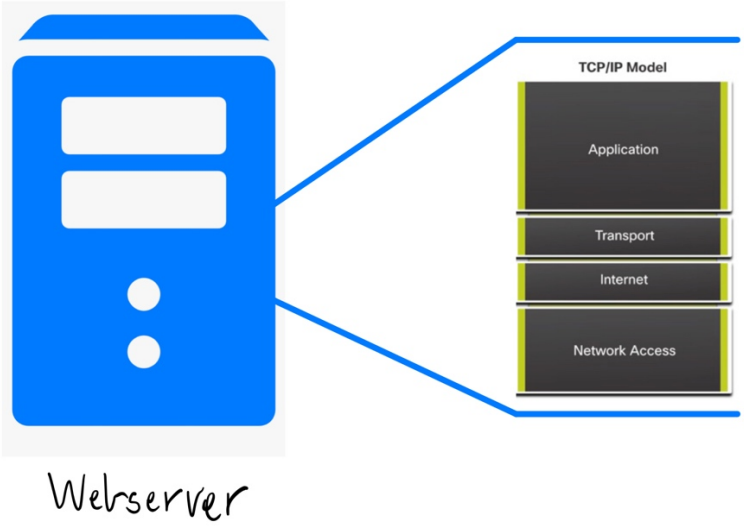
Subnett: 172.16.9.104/29 er hverken tilordnet en avdeling eller blant de to reserverte adressene. Dette subnettet har 6 adresser som kan tilordnes endenoder.

Oppgave 4

a)

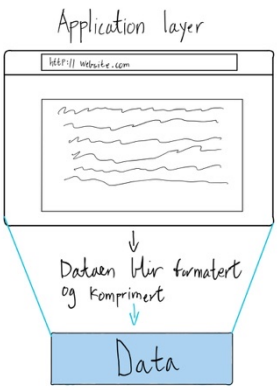
Når en webtjener skal sende en html-side til nettleseren på en pc, i et klient-server nettverk, så betyr det at serveren sender en respons tilbake på en forespørsel fra klienten (PC). TCP/IP modellen er en lagvis modell som gir et godt bilde av hvordan dataen blir behandlet gjennom nettverket.

Vi skal nå se på hvordan dataen hos webtjeneren pakkes inn i flere lag gjennom TCP/IP modellen, og hvilke protokoller som er involvert i hvert lag.



Applikasjonsslaget

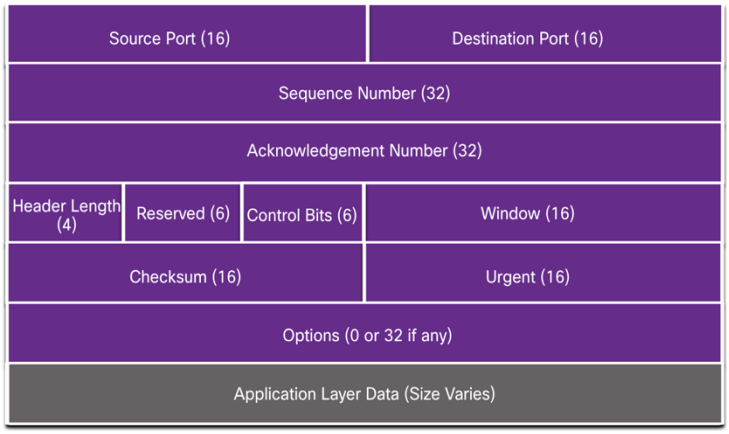
Applikasjonslaget består av mange forskjellige protokoller som kan brukes for å utveksle data mellom en klient og en server. I denne oppgaven så brukes det en webserver som kjører HTTP protokollen. HTTP er en protokoll som gjør det mulig å utveksle hypertekst filer for en webserver og en klient. Når webserveren skal sende en respons tilbake til klienten, så er det HTTP sin oppgave å hente ut den ønskede tjenesten/filen. Etter at dataen er hentet ut, så blir dataen formatert og komprimert, slik at den er klar for innpakkingsfasen som kommer videre nedover i TCP/IP stakken.



Transportlaget

Transportlaget er det neste laget i TCP/IP stakken som PDUen kommer til. Applikasjonslaget hadde som oppgave å genere dataen som skal bli utvekslet mellom klient og server, mens transportlaget har i oppgave å sørge for at denne dataen faktisk blir utvekslet mellom applikasjonene i nettverket. I transportlaget så finner vi to velkjente protokoller, disse er TCP og UDP. Siden en nettside som blir sendt over et nettverk krever en pålitelig leveringsmetode, slik at hele nettsiden kommer frem til destinasjonen, så er TCP den beste løsningen. Det er fordi TCP er en protokoll som har flere tiltak som sørger for en pålitelig overføring av pakker i nettverket. TCP kan også håndtere tap av pakker. UDP er på den andre siden kjent som en best-effort protokoll, som sørger for rask overføring med lite administrativt overheng.

Når PDUen har kommet ned til transportlaget, så vil den bli pakket inn av en TCP header. TCP segmentet er på tilsammen 20 bytes som holder viktig informasjon om PDUen.



Den holder blant annet informasjon om avsender sin port og destinasjonen sin port. Transportlaget benytter seg av porter for å kunne håndtere forskjellige sesjoner som skjer samtidig på serveren.

Segmentet holder også annen viktig informasjon som sikrer en pålitelig overføring mellom applikasjonene. Denne informasjonen er blant annet «sequence number», som sørger for at alle pakkene som ankommer klienten blir satt sammen i riktig rekkefølge, «Checksum», som brukes for å sjekke om dataen har blitt korrupt, og «window» som bestemmer hastigheten av pakkeoverføringen.

Portene som blir lagt i headeren på dette segmentet blir:

- Source port: 80

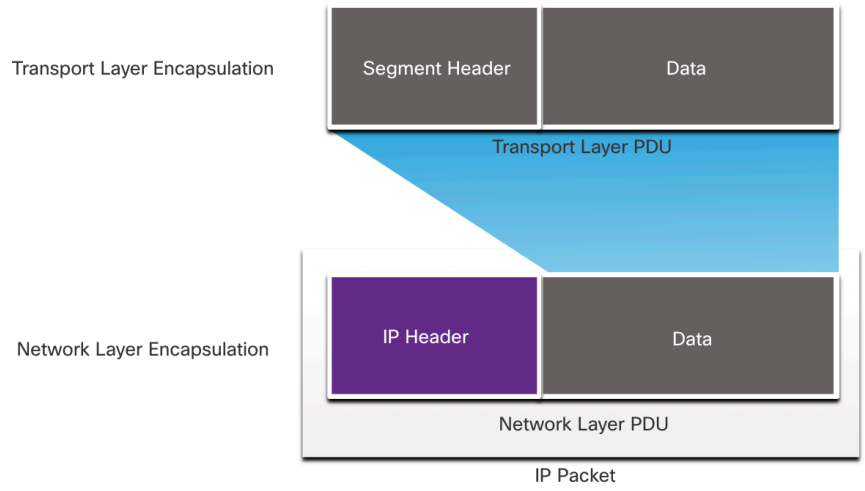
- Destination port: 61123

Etter at segmentet har fått en header med kritisk informasjon angående overføingen, så vil PDUen bli sendt videre til Internettlaget.

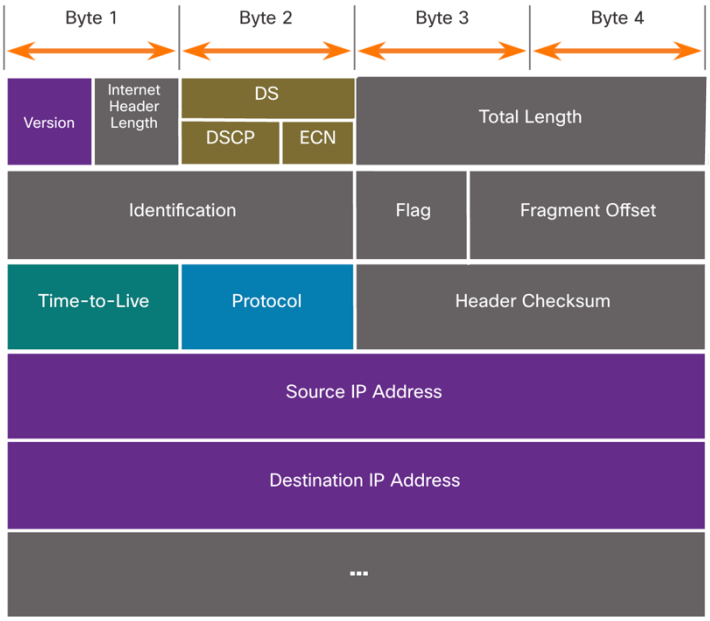
Internettlaget

Til nå har vi sett at applikasjonslaget har formatert og komprimert dataen, transportlaget sørger for pålitelig overføring mellom applikasjonene, slik at hele nettsiden kommer frem til klienten. Nå har internnettlaget ansvaret for å sørge for at webtjeneren kan sende data til andre enheter over nettverk. Internettlaget består av enten IPv4 – eller IPv6 protokollen. Disse protokollene gjør det mulig for en avsender å sende pakker over nettverket til en mottaker. IP protokollene har lavt administrativt overheng, som betyr at det er ingen sporing av pakker. Med andre ord, så er IP protokollene en best-effort protokoll. Derfor er det viktig at transportlaget tar seg av pålitelig overføring gjennom TCP protokollen.

Når PDUen kommer fra transportlaget og ned til internettlaget, så blir segmentet innpakket i en IP pakke.



IP headeren inneholder mye viktig informasjon angående levering av pakken. I denne oppgaven så er det snakk om en IPv4 header.

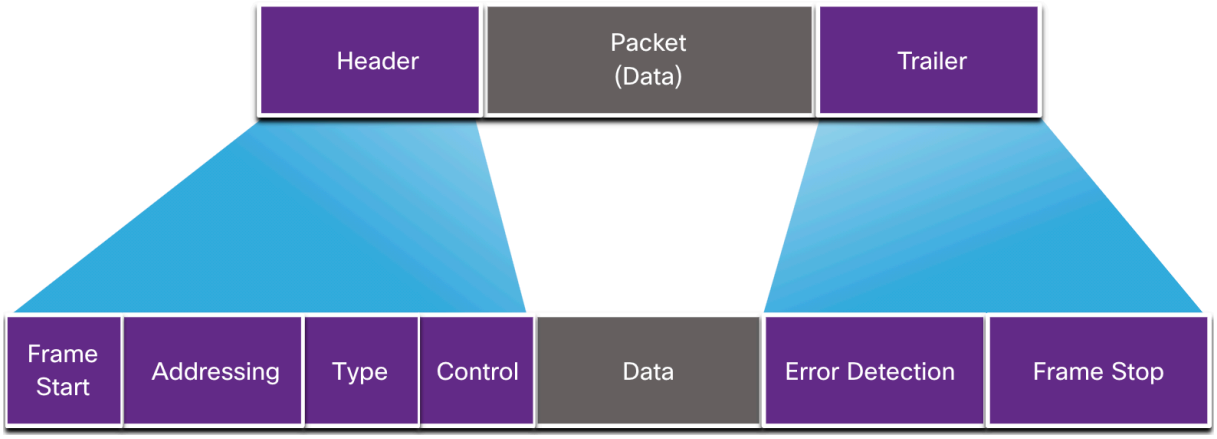


Headeren inneholder blant annet «Source IP» og «Destination IP» , som hendholsvis er avsender og mottaker sin IPv4 adresse. Headeren har også et «Time-to-Live» felt, som er et tall som blir dekrementert med 1 for hvert stopp langs nettverket. Hvis dette feltet når 0, så vil pakken bli kastet. «Header checksum» feltet er med på og bestemme om pakken har blitt korrupt.

«Protocol» feltet forteller hvilken protokoll som blir brukt i transportlaget, dette gjør at mottakeren vil kunne sende dataen til riktig protokoll, som i vårt tilfelle er TCP.

Datalinklaget

Oppgaven til datalinklaget er forbrede PDUen fra Internettlaget til å kunne sendes over det lokale kommunikasjonsmediet. Datalinklaget gjøre det mulig å sende pakker mellom forskjellige nettverksgrensesnittkort i det lokalet nettverket ved å innpakke pakken i en header og en trailer. Denne headeren og traileren vil støtte Ethernet protokollen siden webserveren, switchen og ruteren utgjør et kablet LAN.



Headeren og traileren vil nå inneholde mye informasjon om rammen. Det første og siste feltet vil indikere start og stopp på rammen. Nodene i nettverket vil nå vite at en ramme vil komme, ved hjelp av «frame start» feltet, og noden vil også vite når rammen slutter ved hjelp av «frame stop» feltet. Traileren inneholder også et «error detection» felt, som sjekker at ingen problemer har oppstått.

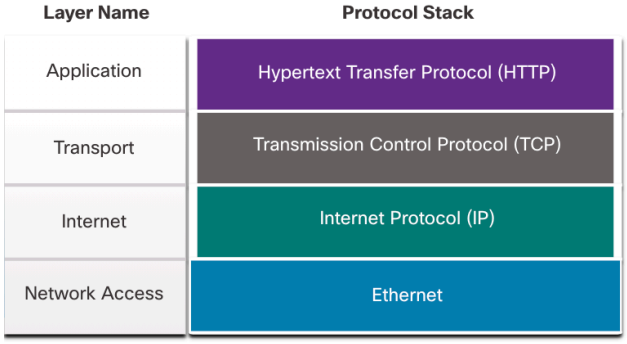
«Adressing» feltet vil inneholde den lokale avsender - og mottaker MAC adressen som trengs for å sende pakken over det lokalet nettverket. MAC adressen gjør det mulig for noder, på det lokalet nettverket, å sjekke om rammen er tiltenkt dem eller ikke ved å bare se på datalinklaget. Dersom rammen må sendes over avsidesliggende nettverk for å nå mottakeren, så vil datalinklaget jevnlig bli byttet ut hos en ruter. Dette fører til at de lokale adressene i «adressing» feltet jevnlig blir oppdatert, slik at rammen kan sendes gjennom det nye lokale nettverket.

I denne oppgaven så vil rammen bli sendt fra webserveren med følgende adresse felt i transportlaget.

Destination Address	Source Address	Data
00-0D	00-0A	Encapsulated data

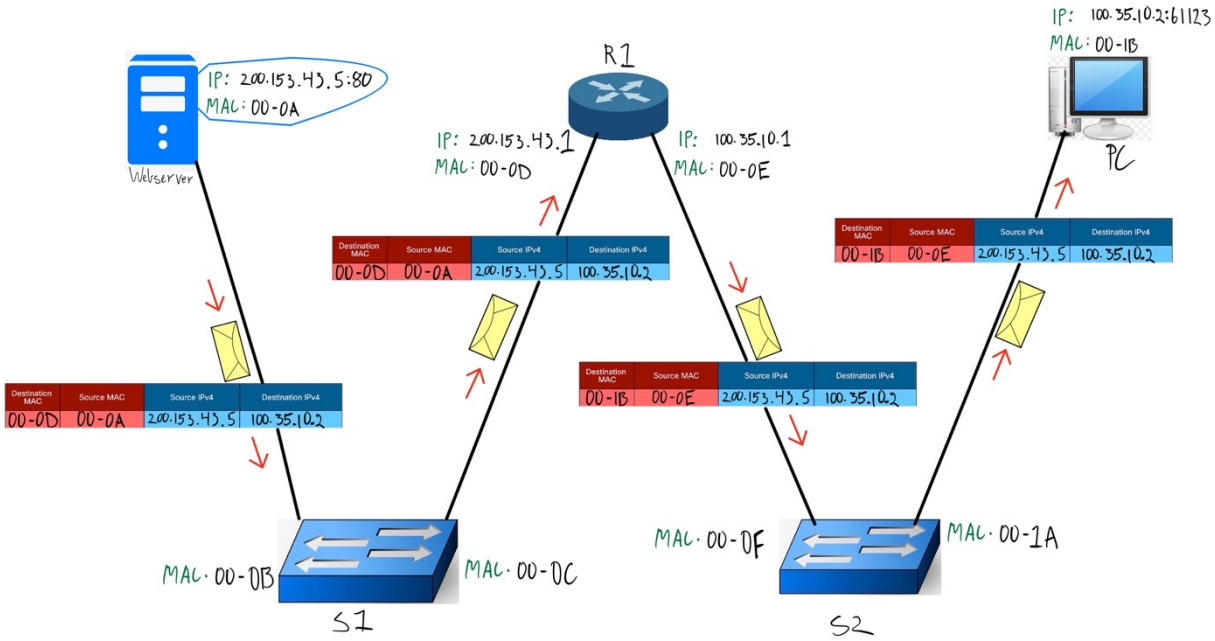
«Default gateway» vil være mottakeradressen siden noden er en enhet som ruter trafikk til andre nettverk. Avsenderadressen vil være webserveren.

Konklusjon



PDUen vil bli pakket inn i flere lag nedover TCP/IP stakken. Hvert lag vil gi PDUen en relevant protokoll med tanke på dataen som sendes. Bildet til venstre viser hvilke protokoller som blir brukt når webserveren skal sende en html-side til klienten, og hvilket lag som støtter de forskjellige protokollene.

b)



b I)

Som man kan se på bildet over, har rammen som blir sendt fra webserveren MAC adressen til R1 som destinasjons MAC adresse, og sin egen MAC adresse som avsender MAC adresse. Internettlaget sine IPv4 adresser inneholder klienten (PC) sin IPv4 adresse som destinasjons IPv4 adresse, og webserveren sin IPv4 adresse som avsender IPv4 adresse.

Internettlaget sine IPv4 adresser blir ikke endret underveis i sendingen, men de lokale adressene blir endret når rammen kommer frem til R1. Da får rammen nye lokale adresser, slik at rammen kan sendes over et nytt lokalt nettverk. Da blir klienten sin MAC adresse satt opp som destinasjons MAC adresse, og R1 sin MAC-adresse blir satt opp som avsender adresse. Et viktig punkt å få med seg er at R1 er konfigurert med to forskjellige MAC adresser. Hver MAC adresse er tilknyttet hvert sitt lokale nettverk. Derfor får rammen som sendes fra R1 MAC adressen; 00-0E, og ikke adressen 00-0D.

B II)
Hos begge switchene (S1 og S2) benyttes bare Datalinklaget for å bestemme hvor pakkene skal bli sendt videre. Når porten på switchen fanger opp rammen, så vil switchen se på de lokale adressene som ligger i rammen på Datalinklaget. Switchen vil så bruke MAC-adress tabellen for å vite hvor rammen skal sendes videre.

Ruteren (R1) benytter seg først av Datalinklaget, for å se at pakken er ment for den. Deretter «skrelles» Datalinklaget, slik at ruteren kan se på Internettlaget sine IPv4 adresser for å vite hvilket nettverk pakken skal sendes til. Deretter får pakken et nytt ramme hode og hale av Datalinklaget, før rammen sendes videre.

c)
Når rammen ankommer klienten (PC), så er det først Datalinklaget som vil se på innholdet i rammehodet. Datalinklaget vil da se på «adresse» feltet, som inneholder de lokale MAC adressene, for å bestemme om denne pakken er ment for denne maskinen eller ikke. Når datalinklaget har sett at destinasjons MAC adressen matcher sin egen MAC adresse, så betyr det at PDUen skal sendes oppover TCP/IP stakken. Datalinklaget vil da først se på «Type» feltet i rammehodet, siden dette feltet sier hvilken protokoll som har blitt benyttet på laget over(Internettlaget) i stakken. I dette tilfellet vil det være en IPv4 protokoll. Etter at «Error Detection» har sørget for at rammen ikke har blitt korrupt under sendingen, vil rammen bli sendt opp til Internettlaget.

Internettlaget vil da skrelle av PDU hodet og halen til Datalinklaget. Internettlaget vil så videre se på i informasjonen som befinner seg i hodet på pakken. Etter at Internettlaget har sørget for at PDU hodet i pakken ikke er korrupt ved å se på «header checksum» feltet, så vil internettlaget så se på «protokoll» feltet. Dette feltet vil fortelle hvilken protokoll som har blitt benyttet i laget over(transportlaget) i stakken, for å vite hvilken protokoll PDUen skal bli sendt til. I dette tilfellet så vil det være en TCP protokoll. Etter at Internettlaget har sett på IP adressene og forsikret seg igjen at pakken er ment for denne maskinen, så vil pakken bli sendt opp til TCP protokollen i Transportlaget.

Transportlaget vil skrelle av pakkehodet, slik at den nå kan se på TCP hodet til segmentet. Segment hodet inneholder viktig informasjon som skal sørge for en pålitelig overføring av dataen. De to første feltene som transportlaget vil se på er «avsender port» og «destinasjon

port». Destinasjonsporten viser hvilken port segmentet skal bli sendt til for å komme til riktig applikasjon. I dette tilfellet så vil destinasjonsporten være 61123. Transportlaget vil også se på «sekvensnummer» feltet som er et veldig viktig felt for pålitelig overføring. «Sekvensnummeret» benyttes for å kunne levere alle segmentene som ble sendt fra webserveren i riktig rekkefølge. Dersom noen segmenter kommer i forskjellig rekkefølge, så vil segmentene bli lagret i en buffer, mens de venter på de resterende segmentene. Når alle segmentene har ankommet, så vil TCP omorganisere segmentene etter sekvensnummeret, slik at rekkefølgen blir riktig. Etter at TCP har alle segmentene i riktig rekkefølge og «Checksum» feltet har sjekket at segment hodet og dataen ikke har blitt korrupt, så vil dataen bli sendt til riktig port, som i dette tilfellet er port 61123. Applikasjonslaget kan nå dekomprimere html filen før nettleseren presenterer dataen, som i dette tilfellet vil være den etterspurte nettsiden.