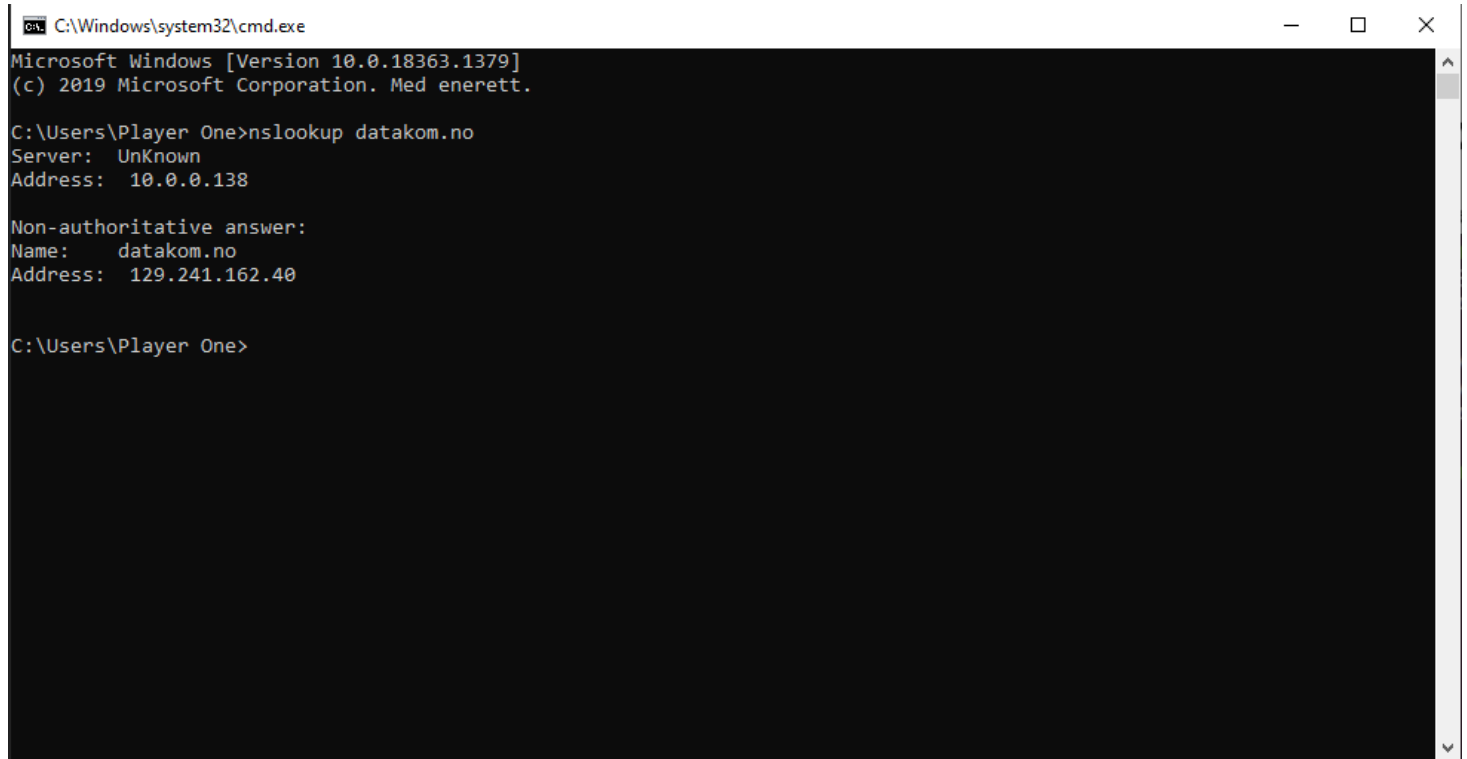


# Datakom øving 1

I denne øvingen har jeg samarbeidet med Mattias Agentoft Eggen

## Oppgave 1: Wireshark og DNS (10%)

a)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. Med enerett.

C:\Users\Player One>nslookup datakom.no
Server: UnKnown
Address: 10.0.0.138

Non-authoritative answer:
Name: datakom.no
Address: 129.241.162.40

C:\Users\Player One>
```

Min lokale navnetjener er "UnKnown" som er fordi min ruter ikke er konfigurert rett. IP-adressen er 10.0.0.138 som er min ruters IP-adresse.

Resultatet fra nslookup på domenenavnet [datakom.no](https://datakom.no) er IP-adressen 129.241.162.40

b)

```

C:\Users\Player One>ipconfig /all

Windows IP Configuration

Host Name . . . . . : DESKTOP-OUM9NAP
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : home

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : home
Description . . . . . : Intel(R) Ethernet Connection (2) I219-V
Physical Address. . . . . : 2C-56-DC-3C-D1-F4
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::2468:3872:ed4e:ab7e%6(Preferred)
IPv4 Address. . . . . : 10.0.0.13(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : torsdag 18. februar 2021 07:44:56
Lease Expires . . . . . : torsdag 18. februar 2021 11:44:55
Default Gateway . . . . . : 10.0.0.138
DHCP Server . . . . . : 10.0.0.138
DHCPv6 IAID . . . . . : 103569116
DHCPv6 Client DUID. . . . . : 00-01-00-01-24-22-21-D0-2C-56-DC-3C-D1-F4
DNS Servers . . . . . : 10.0.0.138
NetBIOS over Tcpip. . . . . : Enabled

Wireless LAN adapter Wi-Fi:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Description . . . . . : Qualcomm Atheros QCA61x4A Wireless Network Adapter
Physical Address. . . . . : 30-52-CB-49-B1-13
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Lokal tilkobling* 1:

Media State . . . . . : Media disconnected

```

Her står det at DNS servers har samme IP-adresse som vi fikk som lokal navnetjener fra nslookup. I tillegg ser vi at denne IP-adressen er den samme som ruterens IP-adresse som betyr at det er ruterens min som er min lokale navnetjener.

c)

#### Answers

```

datakom.no: type A, class IN, addr 129.241.162.40
Name: datakom.no
Type: A (Host Address) (1)
Class: IN (0x0001)
Time to live: 13931 (3 hours, 52 minutes, 11 seconds)
Data length: 4
Address: 129.241.162.40

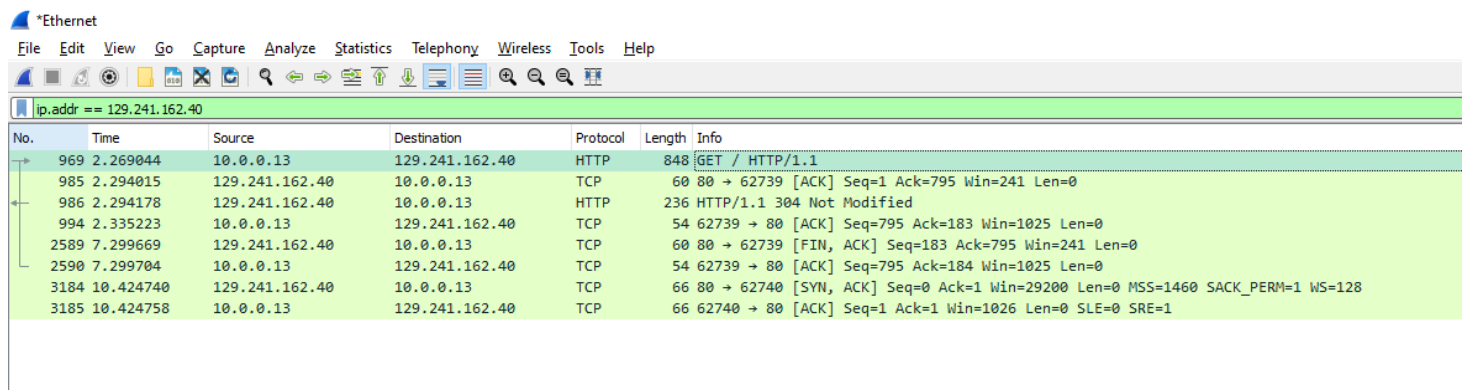
```

Verdien på Type er A og levetid er 3 timer, 52 minutter, 11 sekunder. At det er type A betyr at den kommer fra host addressen som er 129.241.162.40. Levetiden vil si at den får "leve" i så lang tid før pakken blir slettet fra DNS cache.

d) Når jeg bruker kommandoen `nslookup datakom.no` så får jeg både IPv4 og IPv6 adresse til [datakom.no](http://datakom.no), men annet enn dette så er det ikke behov for flere DNS-requester for å finne ut hvilke IP jeg skal koble opp mot.

## Oppgave 2: Wireshark og HTTP (15%)

a)



The image shows a Wireshark packet capture on an Ethernet interface. The filter is set to 'ip.addr == 129.241.162.40'. The packet list shows a GET request (No. 969) and its response (No. 985). The packet details pane shows the structure of the HTTP response, including the status line 'HTTP/1.1 304 Not Modified'.

No.	Time	Source	Destination	Protocol	Length	Info
969	2.269044	10.0.0.13	129.241.162.40	HTTP	848	GET / HTTP/1.1
985	2.294015	129.241.162.40	10.0.0.13	TCP	60	80 → 62739 [ACK] Seq=1 Ack=795 Win=241 Len=0
986	2.294178	129.241.162.40	10.0.0.13	HTTP	236	HTTP/1.1 304 Not Modified
994	2.335223	10.0.0.13	129.241.162.40	TCP	54	62739 → 80 [ACK] Seq=795 Ack=183 Win=1025 Len=0
2589	7.299669	129.241.162.40	10.0.0.13	TCP	60	80 → 62739 [FIN, ACK] Seq=183 Ack=795 Win=241 Len=0
2590	7.299704	10.0.0.13	129.241.162.40	TCP	54	62739 → 80 [ACK] Seq=795 Ack=184 Win=1025 Len=0
3184	10.424740	129.241.162.40	10.0.0.13	TCP	66	80 → 62740 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
3185	10.424758	10.0.0.13	129.241.162.40	TCP	66	62740 → 80 [ACK] Seq=1 Ack=1 Win=1026 Len=0 SLE=0 SRE=1

Header linjen vi sender er en GET request, `GET / HTTP/1.1`, som spør om index fila fra destinasjons IP-adressen: 129.241.162.40.

Svaret vi får er: `HTTP /1.1 304 Not Modified`, som betyr at index-filen vi spør etter ikke har endret seg siden sist vi mottok den, og det er derfor ikke nødvendig å hente samme pakke på nytt.

Dette kommer av at vi først gikk inn på [datakom.no](http://datakom.no) før vi åpnet wireshark, siden da har vi allerede index-fila lagret når vi besøker [datakom.no](http://datakom.no) på nytt.

Vi kan se på skjermbildet at tiden fra siste svar fra webtjeneren til det kommer en TCP pakke med FIN-flagget satt er ganske nøyaktig 5 sekunder. Dette kommer av at HTTP-requesten har en Keep-Alive med timeout verdi = 5, som betyr at den skal holde forbindelsen i 5 sekunder før den kobles ned. Dette er for å slippe å gjennomføre three-way-handshake flere ganger hvis det skal sendes flere pakker kort tid etter hverandre.

b) Klienten sender HTTP request med en headerlinje: `Cache-Control: max-age=0`. `max-age=0` vil si at cachene alltid må revalideres og sjekke om den versjonen av index-fila som er lagret, faktisk er den siste versjonen. GET requesten har også en `If-Modified-Since: Wed, 23 Sep 2020 13:02:56 GMT` som kan sees på som datoen til index-fila som er i mellomlagret på PC'en. Når vi sender GET-requesten med disse headerlinjene, så sjekker webtjeneren om den er endret siden den nevnte datoen, og i dette tilfelle svarer tjeneren: `HTTP/1.1 304 Not Modified` som altså betyr at den IKKE er endret og vi har den nyeste versjonen mellomlagret.

## Oppgave 3: Wireshark og TCP (20%)

a)

1828	23.200725	10.0.0.13	129.241.162.40	TCP	66	53752 → 80	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	WS=256	SACK_PERM=1	
1829	23.221850	129.241.162.40	10.0.0.13	TCP	66	80 → 53752	[SYN, ACK]	Seq=0	Ack=1	Win=29200	Len=0	MSS=1460	SACK_PERM=1	WS=128
1830	23.221898	10.0.0.13	129.241.162.40	TCP	54	53752 → 80	[ACK]	Seq=1	Ack=1	Win=262656	Len=0			

SRC	→	DST	FLAGG	SEQNR	ACKNR
53752	→	80	SYN	0	
80	→	53752	SYN, ACK	0	1
53752	→	80	ACK	1	1

Den første byten med nyttelast vil alltid ha sekvensnummer 1 fordi pakkene som har blitt sendt fram til da ikke har hatt noe nyttelast altså 0 bytes. Sekvensnummeret øker med antall bytes som er sendt i forrige pakke. Så når sekvensnummeret er 1 (etter three-way-handshake), og pakker som sendes har 0 bytes med nyttelast så blir neste sekvensnummer blir dermed 1 + 0, som er 1. Derfor vil den første byten med nyttelast alltid ha sekvensnummer 1.

Kvitteringsnummeret vi vil få tilbake vil være 2 fordi kvitteringsnummer peker på sekvensnummeret til den neste byten som forventes fra motparten.

b)

1833	23.222052	10.0.0.13	129.241.162.40	HTTP	691	GET / HTTP/1.1								
1838	23.246038	129.241.162.40	10.0.0.13	TCP	60	80 → 53752	[ACK]	Seq=1	Ack=638	Win=30592	Len=0			
1839	23.246569	129.241.162.40	10.0.0.13	TCP	1514	80 → 53752	[ACK]	Seq=1	Ack=638	Win=30592	Len=1460			[TCP segment of a reassembled PDU]
1840	23.246674	129.241.162.40	10.0.0.13	TCP	1514	80 → 53752	[ACK]	Seq=1461	Ack=638	Win=30592	Len=1460			[TCP segment of a reassembled PDU]
1841	23.246674	129.241.162.40	10.0.0.13	TCP	1514	80 → 53752	[ACK]	Seq=2921	Ack=638	Win=30592	Len=1460			[TCP segment of a reassembled PDU]
1842	23.246674	129.241.162.40	10.0.0.13	TCP	1514	80 → 53752	[ACK]	Seq=4381	Ack=638	Win=30592	Len=1460			[TCP segment of a reassembled PDU]
1843	23.246692	10.0.0.13	129.241.162.40	TCP	54	53752 → 80	[ACK]	Seq=638	Ack=5841	Win=262656	Len=0			
1844	23.246955	129.241.162.40	10.0.0.13	HTTP	1171	HTTP/1.1 200 OK								(text/html)

NR	Avsender	DST-port	SEQNR	ACKNR	TCP PAYLOAD	Merknad
1	Klient	80	1	1	637	GET-request til webtjener
2	Webtjener	53752	1	638	0	Kvittering om at GET-request er motatt
3	Webtjener	53752	1	638	1460	Første pakke med nyttelast
4	Webtjener	53752	1461	638	1460	Andre pakke med nyttelast
5	Webtjener	53752	2921	638	1460	Tredje pakke med nyttelast
6	Webtjener	53752	4381	638	1460	Fjerde pakke med nyttelast
7	Klient	80	638	5841	0	Kvittering fra klient
8	Webtjener	53752	5841	638	1117	Beskjed fra server om at forespørselen var en suksess

Det vi ser her er en forespørslen fra en klient om å få index fila til [datakom.no](http://datakom.no).

Først sender klient en HTTP GET request til webtjener. Tilbake får klienten først en kvittering om at "bestilling" er mottatt og deretter 4 pakker med 1460 bytes. Webtjener sitt sekvensnummer økes med 1460 (antall bytes i nyttelasten) for hver pakke, mens kvitteringsnummer står fast på 638 fordi den ikke får noen pakker med nyttelast fra klienten underveis.

Etter de fire pakkene med nyttelast fra webtjeneren, sender klienten en kvittering uten nyttelast tilbake med sekvensnummer 638 og kvitteringsnummer 5841 som er en peker på neste byte som forventes. Dette er en bekreftelse på at den har mottatt de fire første pakkene. Helt til slutt sender webtjeneren tilbake siste delen av index fila og en HTTP response om at GET-requesten var en suksess!

## Oppgave 4: Nettverk subnetting (20%)

I denne oppgaven blir vi gitt et IPv4-nettverk a.b.c.0/24. Subnetting består av å dele IP-adressen inn i to deler, nettadresse og nodeadresse. Tallet bak /, som i dette tilfellet er 4, forteller oss hvor mange bits av IP-adressen som brukes til nettadresse-delen av IP-adressen. Nettadresse-delen til IP-adressen er felles for alle noder i et subnett.

Nodeadressen sin jobb er dermed å skille mellom klientene i subnettet. Siden vi har 24 bits til nettadressen, så har vi 8 bits til nodeadresser. 8 bits tilsvarer 256 ulike nodeadresser, fra a.b.c.0 og helt opp til a.b.c.255. Derimot så er den første adressen referert til nettadressen og den siste adressen reservert til broadcast til alle klienter på subnettet. Dermed så er det kunde a.b.c.1 - a.b.c.254 som er tilgjengelig som nodeadresser.

For hver bit man legger til i nettadressen, så halveres antall plasser i subnettet. En god illustrasjon for hvordan subnettene blir delt opp avhengig av antall bit i nettmasken er dette bildet:

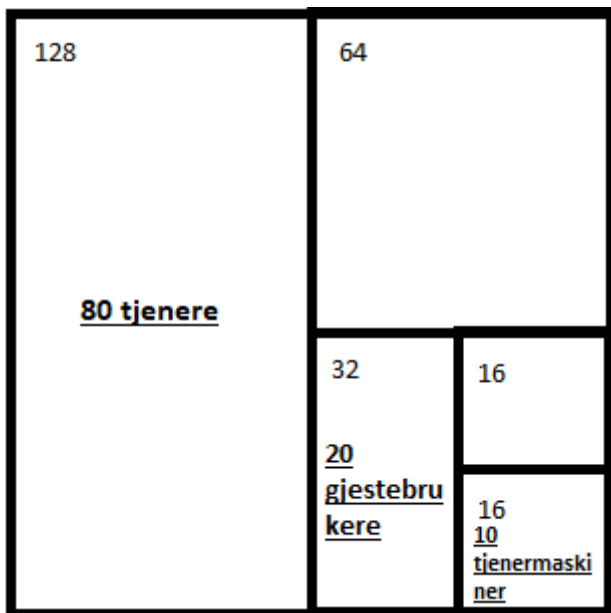
The diagram illustrates the division of a 192.168.50.0/24 network into subnets of various sizes. It shows the following options:

- 192.168.50.0 /24 = 1 network of 256 hosts (minus the network and the broadcast address)
- or
- /25 (255.255.255.128) = 2 subnets of 128 hosts (minus 2)
- or
- /26 (255.255.255.192) = 4 subnets of 64 hosts (minus 2)
- or
- /27 (255.255.255.224) = 8 subnets of 32 hosts (minus 2)
- or
- /28 (255.255.255.240) = 16 subnets of 16 hosts (minus 2)

I denne oppgaven har vi fått beskjed om å dele opp dette IPv4 nettverket opp i subnett med antatt følgende behov:

- 80 ansatte
- 10 tjenermaskiner
- 20 gjestebrukere

Her har vi en figur vi tegnet for å illustrere hvordan vi delte opp subnettene ut i fra oppgavens behov:



Og her er tabellen med informasjon om de forskjellige subnettene.

Nett	Nett-adresse	CIDR	Min host	Max host	Broadcast	Tilgjengelige	Behov	Ledig
A	255.255.255.128	25	1	126	127	126	80	46
B	255.255.255.192	26	129	190	191	62	0	62
C	255.255.225.224	27	193	222	223	30	20	10
D	255.255.255.240	28	225	238	239	14	0	14
E	255.255.255.240	28	241	254	255	14	10	4

## Oppgave 5: Dokumentasjon av Programmeringsøving 1

### - Tråder og socketprogrammering