

Internetværk og Web-programmering

Applikationslaget

Forelæsning 10
Brian Nielsen

Distributed, Embedded, Intelligent Systems



Agenda

1. Applikationslagsprotokoller, og hvad er deres behov for data-transport
2. HTTP
 1. Struktur af Meddelelser
 2. Responstid og persistente forbindelser
 3. HTTP Caching
3. DNS
 1. Virkemåde
 2. DNS caching
 3. DNS værktøjer
4. (P2P)
 1. Bit-torrent

Applikationslagsprotokoller

Hvad er netværksapplikationer og deres behov for kommunikation?

Hvordan kommunikerer applikationsprogrammer med hinanden?

Hvilke services stiller transport laget i TCP/IP modellen til rådighed for applikationer?

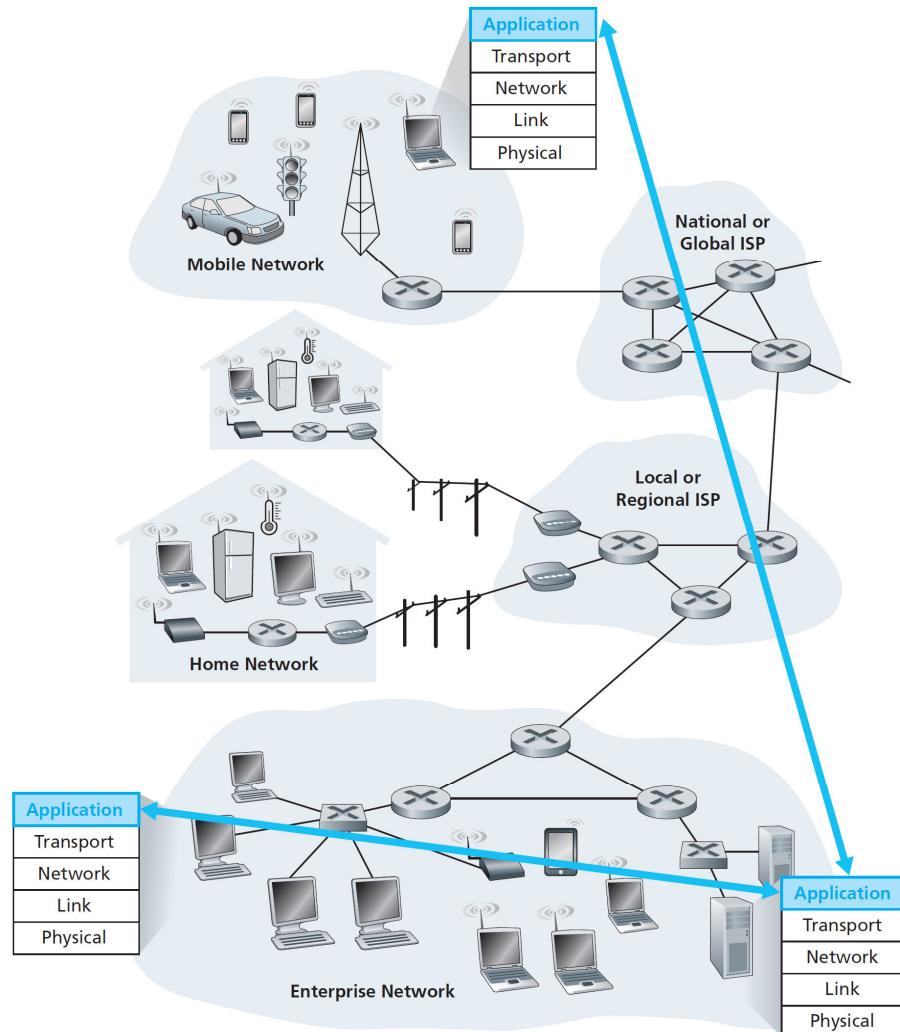
Netværks applikationer

Programmer som:

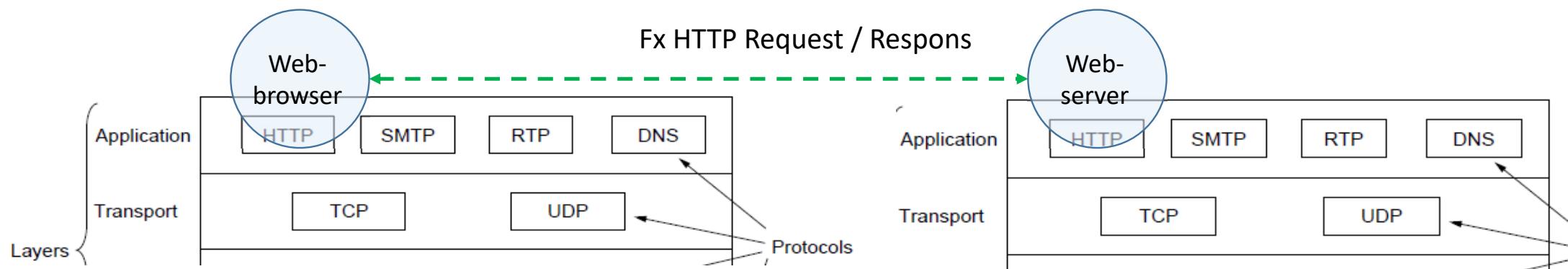
- Afvikles på (flere forskellige) **end systems**
- Kommunikerer over netværket
- Anvender applikations-niveau protokoller

Eksempler

- Web-browser og web-server,
- Discord
- Mail,
- Skype,
- YouTube, Zoom, Teams,
- FTP,
- BitTorrent applikation,
- GoogleDocs, , GitHub, Dropbox,
- CityProbes data-opsamling+dashboard
- ...



Applikationslagsprotokoller



En protokol definerer:

- Hvilken **type meddelelser** bliver udvekslet?
 - fx., request, response, ack,...
- Meddelelses **syntax**:
 - Hvilke felter er der i meddelelsen, og hvordan afgrenses de?
 - Dvs. definerer parametre+struktur
- Meddelelsens **betydning** (semantics)
 - Hvad betyder informationen i meddelelsen?
- **Regler** for hvordan meddelserne skal behandles og besvares

Eksempler:

- Hypertext Transfer Protocol, Simple Mail Transfer Protocol
- Domain Name System Protocol, WebSockets
- Bit Torrent, Network Time Protocol,
- ...

Åbne protokoller:

- defineret i RFCs
- tillader interoperabilitet
- e.g., HTTP, SMTP,

Proprietære protokoller

- e.g., Skype

Applikationskrav til kommunikation

Applikation	Tilladt Data Tab	Throughput krav	Tids- og forsinkelsesfølsom
Fil overførsel	Intet tab	elastisk	nej
e-mail	Intet tab	elastisk	nej
Web documenter	Intet tab	elastisk	nej
real-time/ interaktiv audio/video	tabs-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	ja, 100's msec
stored audio/video	Tabstolerant	samme som ovenfor	ja, få secs
interaktive spil	tabs-tolerant	få kbps og op	ja, 100's msec
text messaging	Intet tab	elastisk	Ja og nej

De fleste applikationer har også behov for “sikkerhed” (security)

- Kryptering, (hemmeligholdelse)
- Data integritet (ingen modificering),
- Authenticitet (Man er den, man giver sig ud for at være / vi ved hvem data kommer fra/skal til)

Services fra Internet transport protokoller

TCP (transmission control protocol) service:

- **pålidelig** transport af en sekvens af bytes (“rørledning” / byte stream) mellem sender og modtager proces
 - Hvis sender og modtager ikke crasher, og netværket ikke fejler permanent, bliver data leveret til modtager korrekt, uden tab, i afsendt rækkefølge
- **flow kontrol:** en hurtig sender overbebyrder ikke modtager
- **congestion kontrol:** sender (ned-)justerer sende raten når netværket er overbelastet (trafikprop i en router)
- **Forbindelses-orienteret** (connection-oriented): Der skal etableres en forbindelse mellem klient og server proces “handshake”

GIVER IKKE

- tids/latens garantier,
- ingen mulighed for kapacitetsreservation (minimum throughput),
- ingen sikkerhed og kryptering
 - (kræver overbygning på applikationslaget “transport layer security”/ “Secure socket layer”)

UDP (“User datagram protokol”) service:

- ” **best-effort** data transport
 - **Datagrammer**
 - **Connection less**
 - **Kan mistes, dublikeres, omordnes af netværket (fx pga forskellig rute)**

GIVER IKKE pålidelighed, flow control, congestion control, timing, throughput garanti, sikkerhed, eller forbindelser

- **SPM: Hva skal vi med den til?** Hvorfor findes den så?

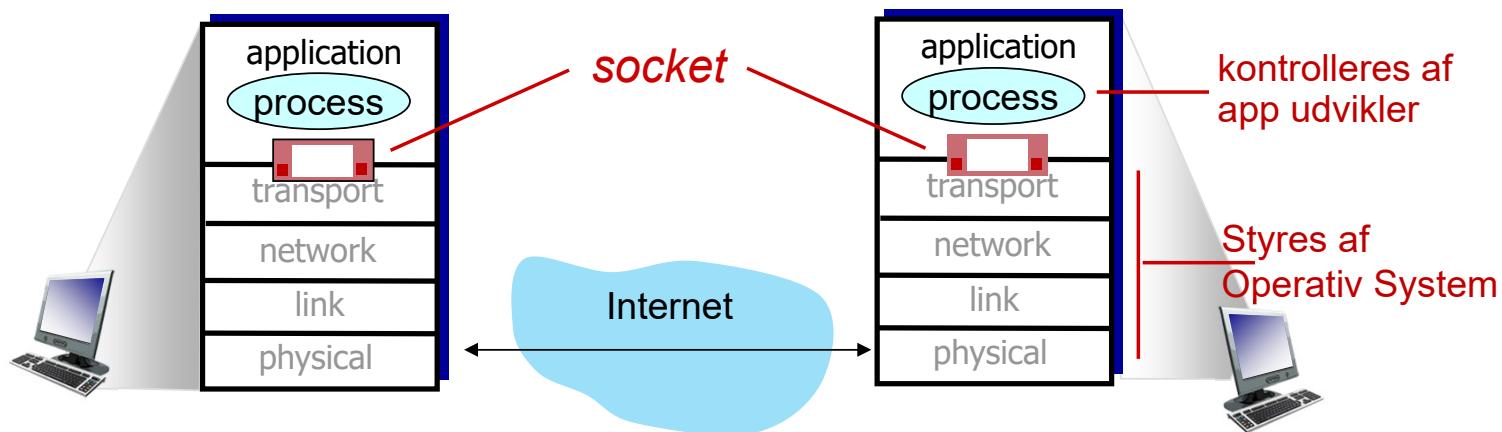
Processer



- Applikationslaget kommunikerer mellem processer på forskellige hosts
- Applikationen afvikles i en "proces" = et kørende program på én host
 - klient proces: proces som initierer kommunikation
 - server proces: proces som afventer at blive kontaktet
 - En given proces kan have begge "roller"
- Processer kan fx vises med "task manager" eller "ps -aux"
- Processer på samme hosts kan kommunikere ved "inter-proces kommunikation"
 - `ls /usr/include | grep "stdio"`
- Mellem forskellige hosts (værtsmaskiner) vha. transport-laget

Sockets

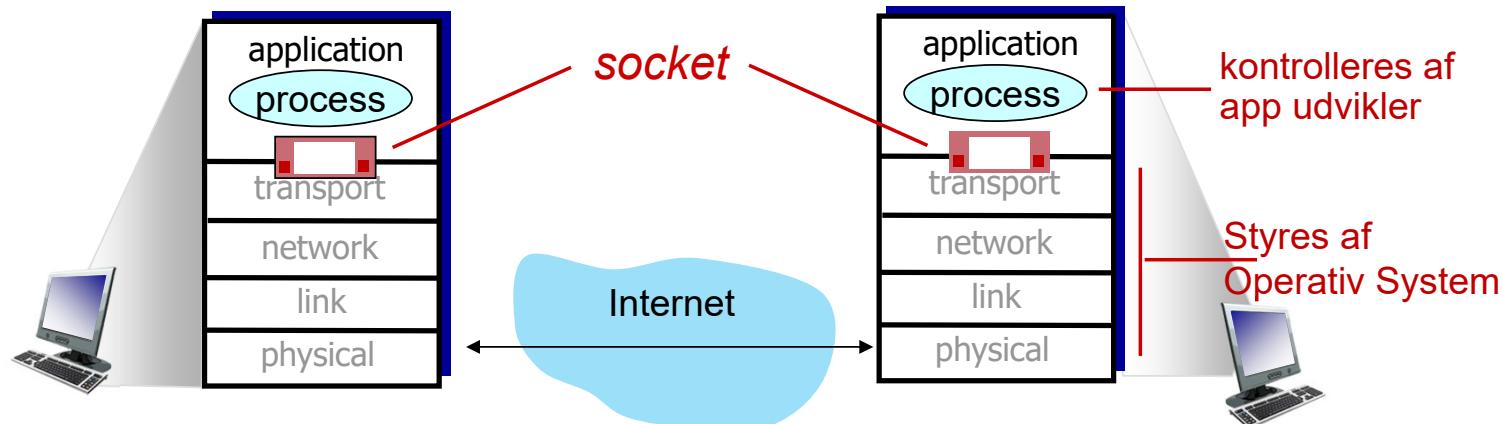
- Processer sender/modtager meddelelser via en ("fatning") **socket**,
- Bindeled/dør mellem applikationslag og transport lag
 - Kræver (mindst) 2-sockets: en på hver side
 - Oprettes af process ved kald til "socket" programmerings-interface



Hvordan kontaktes en Proces?

Fx en server-proces

- Skal have en entydigt identifikation på nettet
- Proces adresseres vha.
 - Hosts **IP-nummer** fx:130.225.63.3
 - Et **port nummer** på host fx: 80
 - CS web-server kontaktes på: 130.225.63.3:80
 - Server process forventes at lave en socket til porten, som den bruge til at "lytte" på
- Mere præcist: Vi kontakter den proces på angivne host, der lytter på en "socket" som er bundet til den angivne port



Velkendte Porte

- Anerkendte services har fast tildelte, reserverede porte
 - Vedligeholdes af [Internet Assigned Numbers Authority](#) (IANA)
 - Portnumre 0-1023 er alle reserverede

Fx,

Port Number	Transport Protocol	Service Name	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS)	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217

HTTP

Hvad bruges HTTP kontrol headers til?

Hvordan struktureres/formateres HTTP meddelelser?

Hvordan effektiviseres HTTP kommunikation?

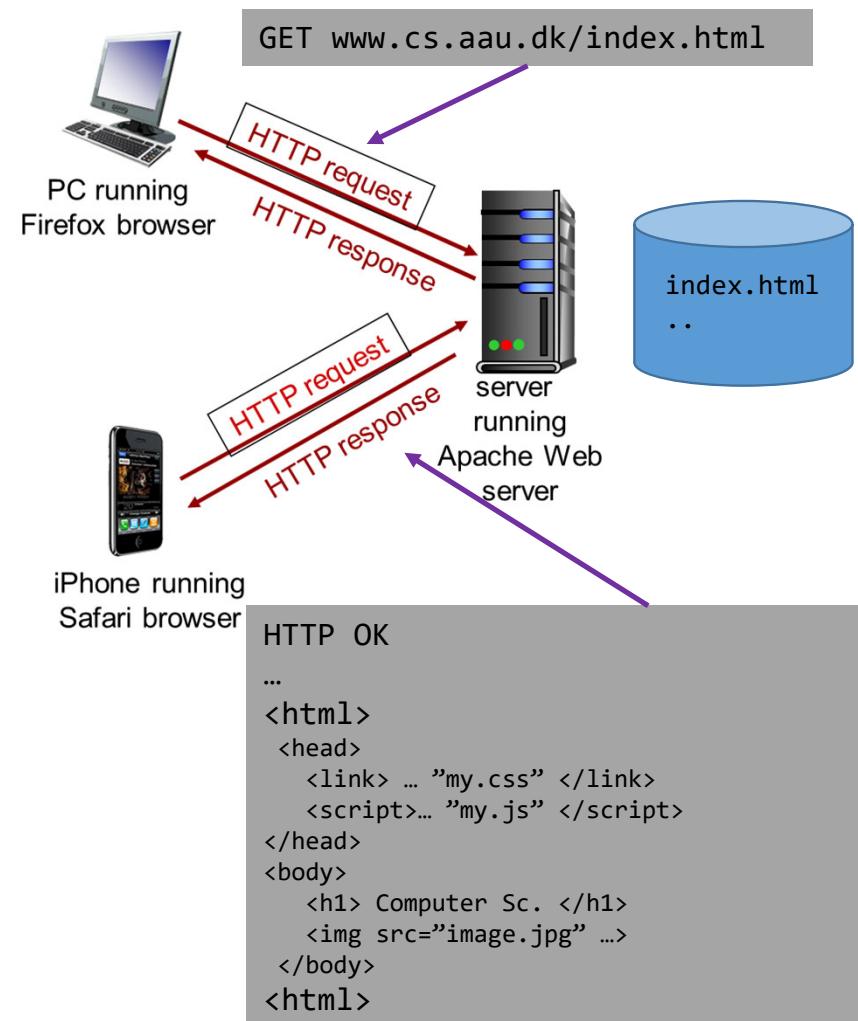
Hvordan håndteres forbindelser til serveren?

Hvorfor og hvordan "cacher" HTTP dokumenter?

Simpel HTTP Scenarie: statisk html fil

HTTP: hypertext transfer protocol

- applikationslags-protokol til web-trafik
- client/server model:
- **klient:** program (typisk browser)
 1. Klient opretter TCP forbindelse til server, typisk port 80
 2. Sender forespørgsel (vha. HTTP GET), om en web side
 3. Afventer respons
 4. Parser respons og optegner siden
 5. Kører evt. skridt 1-4 igen (sideløbende) for at henter nødvendige indlejrede ressourcer (billeder, js-scripts, style sheets)
- **server:**
 1. Afventer
 2. Modtager HTTP forespørgsel fra klient
 3. Behandler den, og beregner svar (fx indlæser filen),
 4. Sender HTTP respons (fx med fil-indhold) til klienten
- Server og ressourcen angives ved et Uniform Resource Identifier, normalt URL
Det eksakte forløb afhænger af HTTP protokol version



HTTP Meddelelser (syntax)

- HTTP forespørgsel
 - Sendes som text (menneske-læsbar)
 - I protokol beskrivelser opstilles formatet som en tabel med felter

request line
(GET, POST,
HEAD commands)

header
lines

carriage return,
line feed at start
of line indicates
end of header lines

```
GET /index.html HTTP/1.1\r\nHost: www-net.cs.umass.edu\r\nUser-Agent: Firefox/3.6.10\r\nAccept: text/html,application/xhtml+xml\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7\r\nKeep-Alive: 115\r\nConnection: keep-alive\r\n\r\n
```

Generelt format for HTTP forespørgsel

method	sp	URL	sp	version	cr	If	request line
header field name	sp	value	cr	If			header lines
~							~
header field name	sp	value	cr	If			
cr	If						
~							entity body

HTTP respons har en tilsvarende format

HTTP Meddelelser: Respons

- HTTP Respons
 - Et lignende format

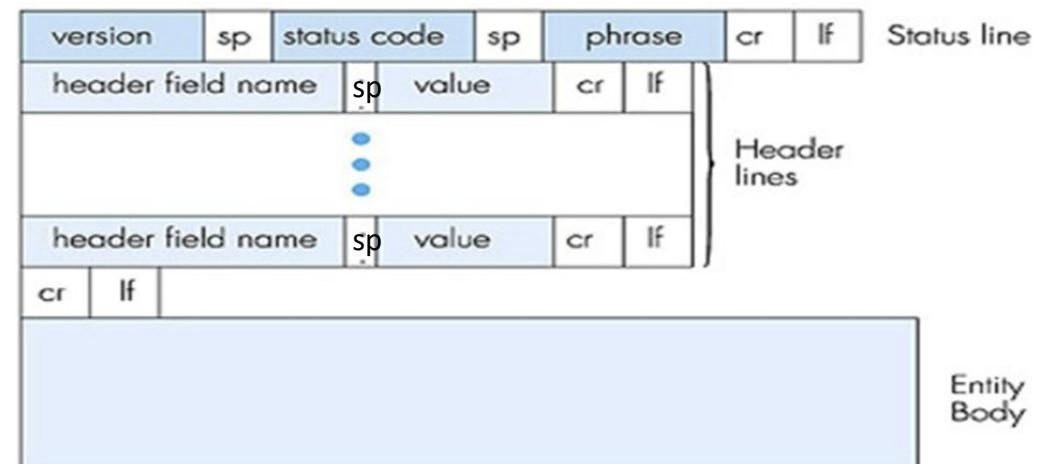
status line
(protocol
status code
status phrase)

header lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK\r\nDate: Sun, 26 Sep 2010 20:09:20 GMT\r\nServer: Apache/2.0.52 (CentOS)\r\nLast-Modified: Tue, 30 Oct 2007 17:00:02  
GMT\r\nETag: "17dc6-a5c-bf716880"\r\nAccept-Ranges: bytes\r\nContent-Length: 2652\r\nKeep-Alive: timeout=10, max=100\r\nConnection: Keep-Alive\r\nContent-Type: text/html; charset=ISO-8859-1\r\n\r\ndata data data data data ...
```

Generelt format for HTTP respons



HTTP klient demo

- `netcat` værktøj: opretter TCP forbindelse
 - Udskriver hvad den modtager
 - Videresender hvad den får på input

```
root@AAU131963:~# nc -C -v www.cs.aau.dk 80
Connection to www.cs.aau.dk 80 port [tcp/http] succeeded!
GET / HTTP/1.1
Host: www.cs.aau.dk

HTTP/1.1 301 Moved Permanently
Date: Wed, 06 Apr 2022 08:19:24 GMT
Server: Apache
Location: https://www.cs.aau.dk/
Content-Length: 230
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://www.cs.aau.dk/">here</a>.</p>
</body></html>
```

- åbner TCP forbindelse til port 80 (std. HTTP server port) på www.cs.aau.dk
- Hvad du nu indtaster sendes til port 80 på www.cs.aau.dk
- Minimalt valid HTTP GET:

GET / HTTP/1.1
Host: www.cs.aau.dk

+2*Enter
- Inspicér responset

<https://blog.desdelinux.net/en/using-netcat-some-practical-commands/>
<https://www.varonis.com/blog/netcat-commands>

Netcat som server

← → × ⌂ ⓘ localhost:3000/index.html

```
root@AAU131963:~# nc -l 3000
GET /index.html HTTP/1.1
Host: localhost:3000
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.75 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8,da;q=0.7,nb;q=0.6,de;q=0.5,en-GB-oxendict;q=0.4,en-AU;q=0.3,en-CA;q=0.2,en-NZ;q=0.1,en-ZA;q=0.1

root@AAU131963:~#
```

Egen protocol over TCP????

The image displays two terminal windows side-by-side, both titled "root@AAU131963: ~".

The left terminal window shows the following session:

```
root@AAU131963:~# netcat -C -v localhost 3000
Connection to localhost 3000 port [tcp/*] succeeded!
hej
hej selv
dumme
selv
```

The right terminal window shows the following session:

```
root@AAU131963:~# nc -l 3000
hej
hej selv
dumme
selv
```

In the left window, the user runs `netcat -C -v localhost 3000` to connect to port 3000. In the right window, the user runs `nc -l 3000` to listen on port 3000. Both windows show the same exchange of messages: "hej", "hej selv", "dumme", and "selv".

HTTP Features

- Indholdsforhandling (foretrukne formation)
- **Forbindelseshåndtering**
- Cookies til sessionsstyring
- Cross Origin Ressource Sharing (CORS)
- Adgangskontrol (Authentication)
- **Caching**
- Data Kompression
- Omdirigering
- Portionslæsning (range requests)

<https://developer.mozilla.org/en-US/docs/Web/HTTP>

HTTP Forbindelseshåndtering

Hvordan effektiviseres HTTP client-server kommunikation?

Hvordan bruges TCP bedst?

HTTP Respons Tid

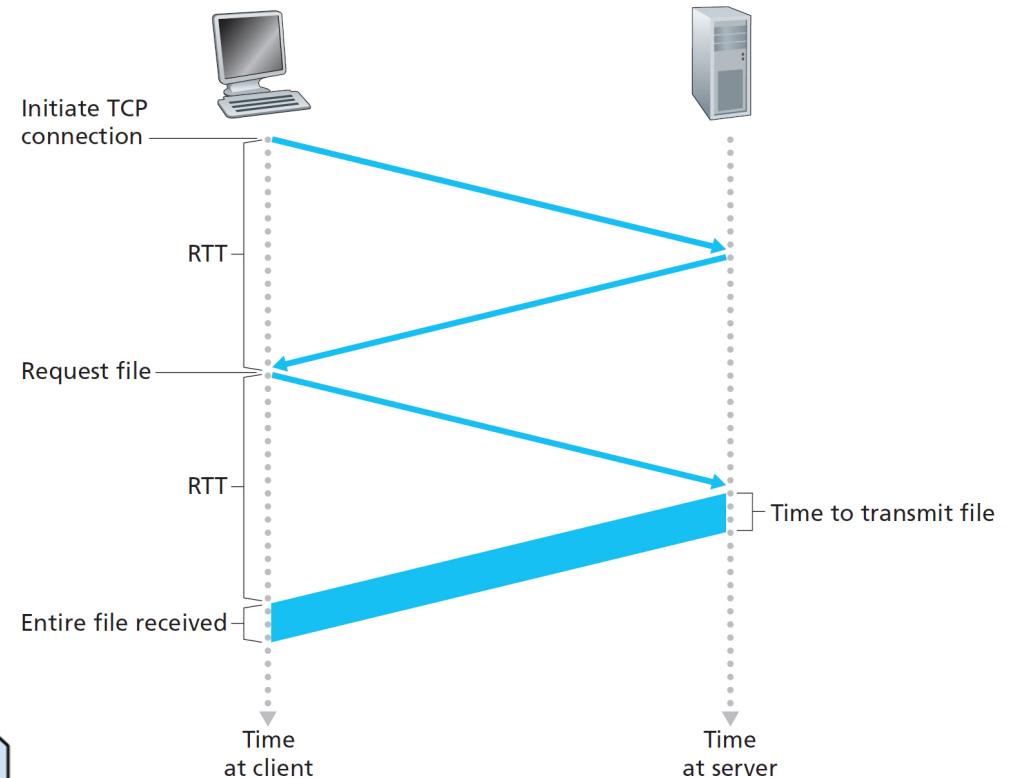
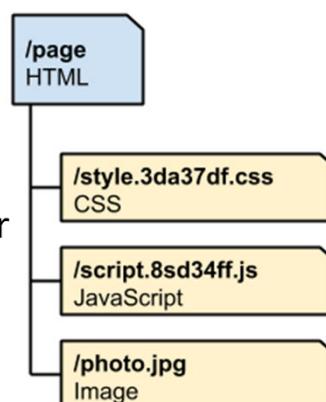
RTT (definition): round-trip-tid: tiden, det tager for klienten at sende en lille pakke til serveren og modtage svaret

HTTP respons tid

- En RTT til at opsætte TCP forbindelsen (handshake)
 - En RTT for HTTP request og første få bytes af HTTP responset
 - Fil transmissionstid
- = 2RTT + file transmission time

Simple browsere henter linkede objekter sekventielt

- Hvis proceduren gentages sekventielt for hvert objekt bliver det **langsomt** at indlæse en side
- I eksemplet: 4 * HTTP Responstid
 - 1 * HTTP Responstid til at hente siden /page.html
 - + 3 * HTTP Responstid til at hente linkede objekter
 - >8 RTT

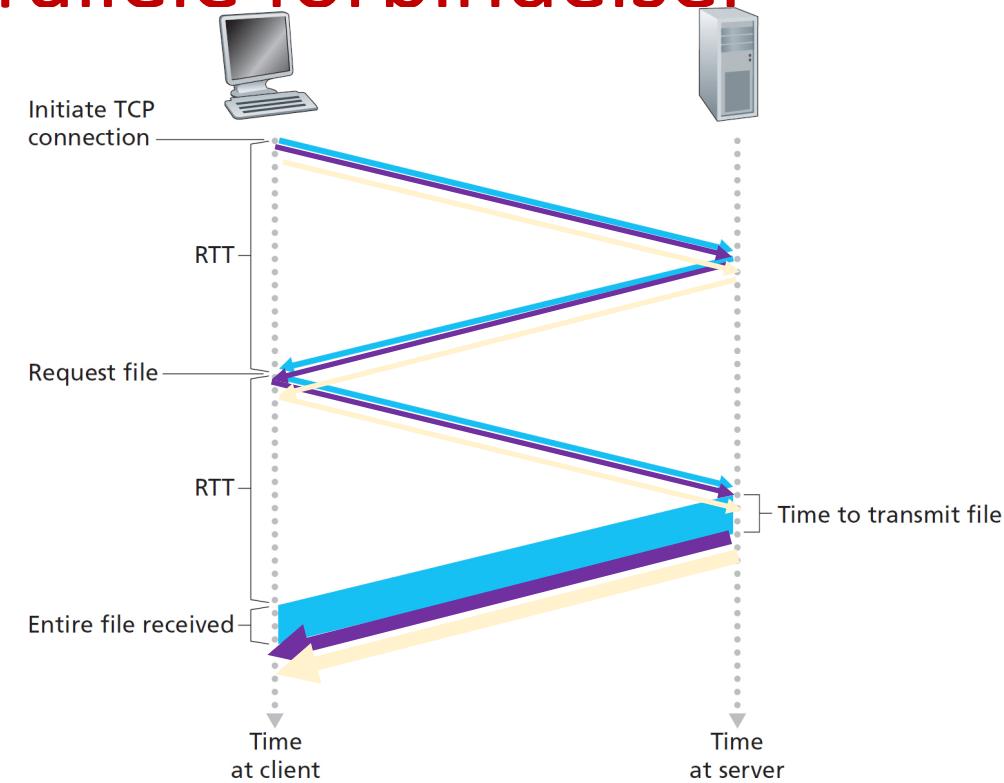
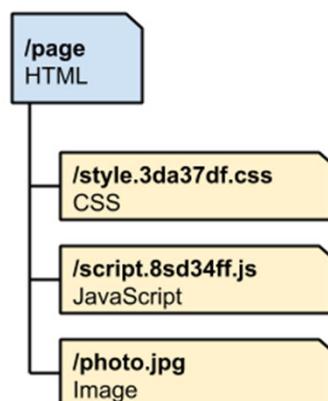


HTTP Responstid med parrallele forbindelser

Klient kan oprette flere “parallelle” forbindelser til server

HTTP respons tid:

- $2 \text{RTT} + \text{file transmission time} + \text{server overhead}$ og evt.
Kapacitetsbegrænsninger downlink til klient
- **Meget hurtigere for klienten**
- **Krævende for server**, da den skal reservere buffer kapacitet til hver
forbindelse (gør dette for mange klinter)
- I eksemplet: $>2 \times \text{HTTP Responstid}$
 - $1 \times \text{HTTP Responstid}$ til at hente siden (/page.html)
 - $+ 1 \times \text{HTTP Responstid}$ til at hente linkede objekter
 - $+ \text{transmissionstid for } 3 \text{ filer}$



3 parallele forbindelser

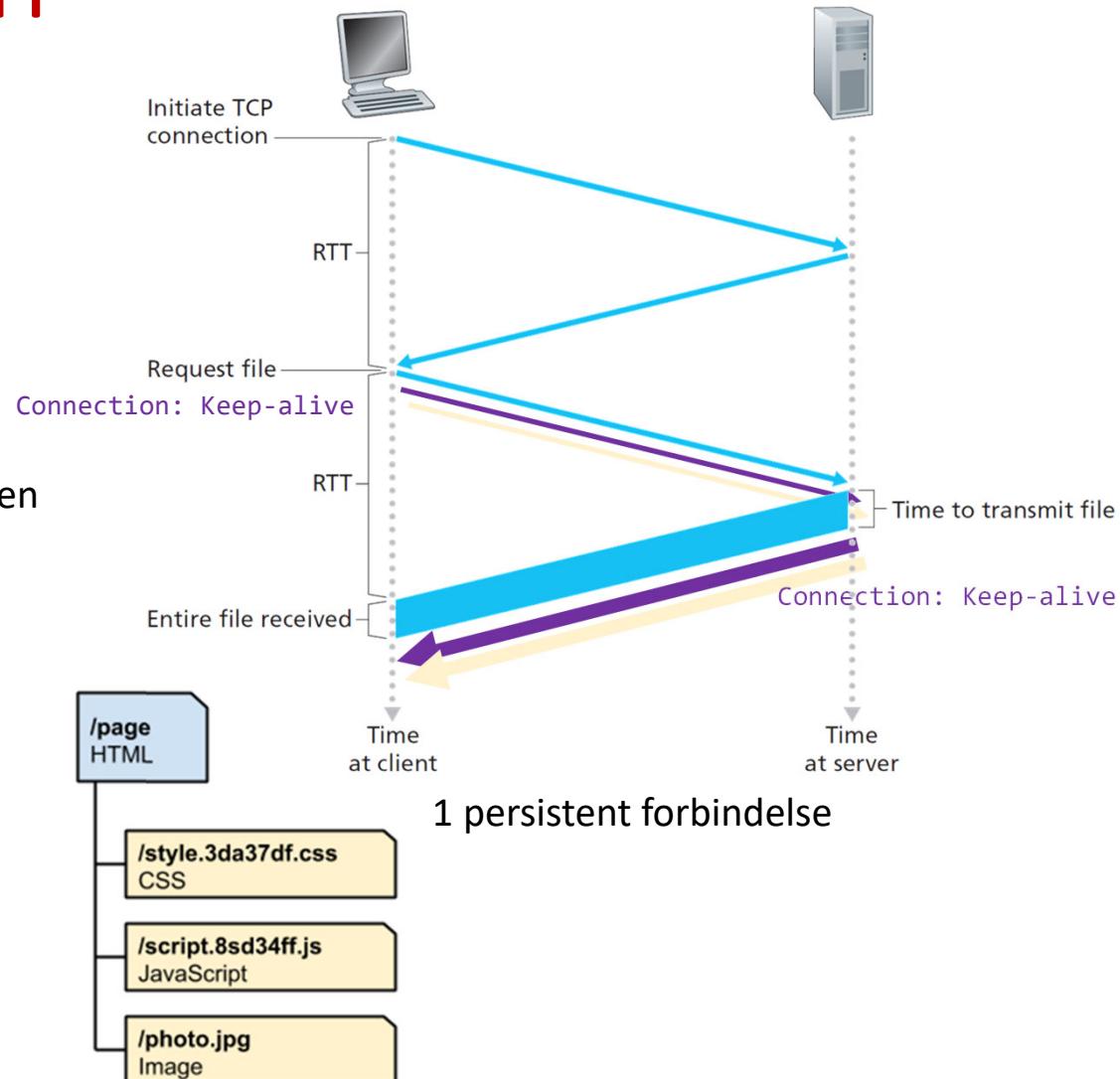
Løsning: Persistent HTTP

Persistent HTTP (default i HTTP/1.1):

- Server holder forbindelsen åben efter afsendelse af respons
- Efterfølgende HTTP forespørgsler og svar sendes på samme åbne forbindelse:
 - Sparer overhead ved oprettelse af TCP forbindelser

HTTP Pipelining

- Forespørgsler kan “pipelines”: sendes efter-hinanden uden at afvente svar “samlebånds princip”



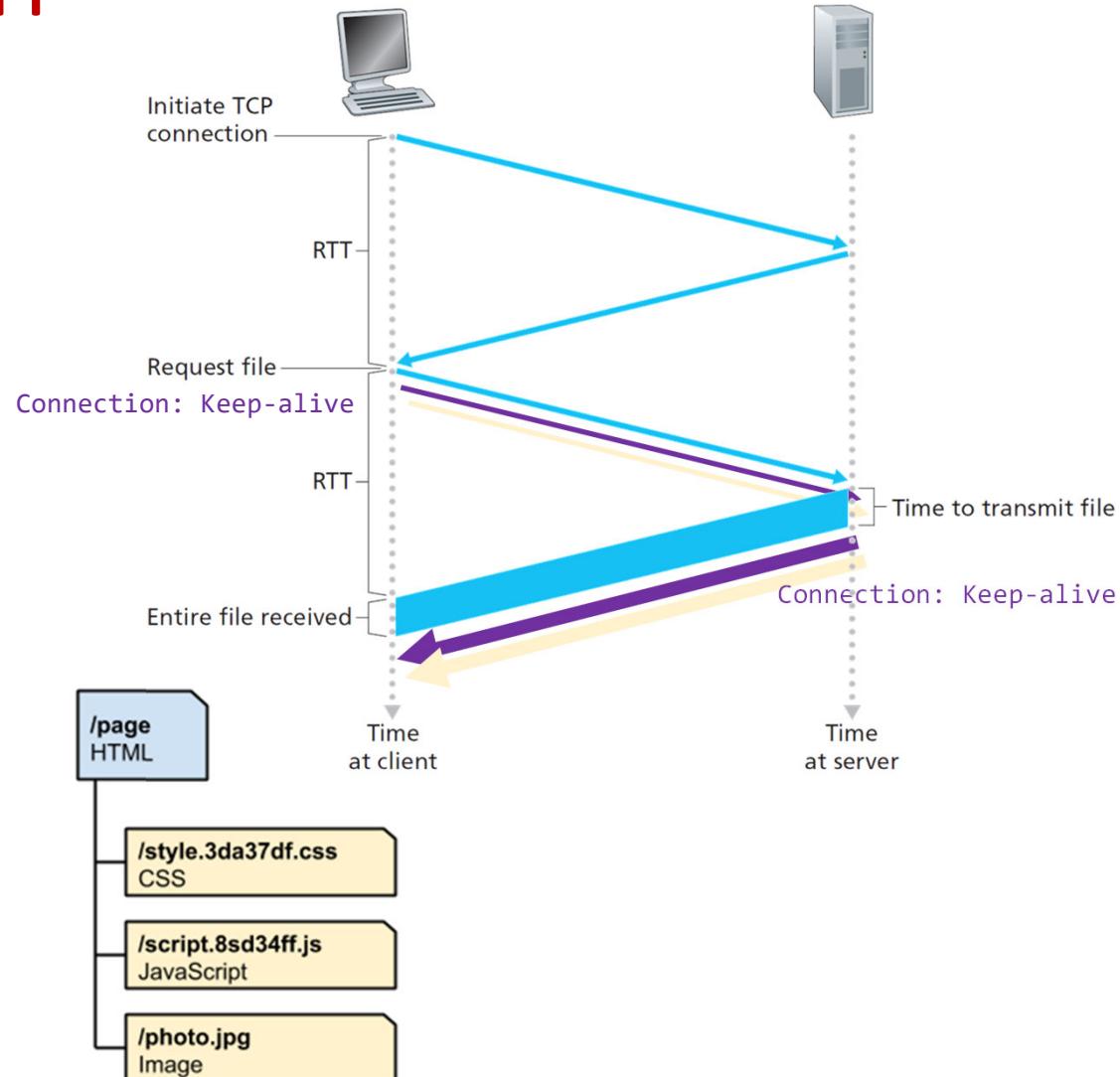
Løsning: Persistent HTTP

HTTP response tid med pipelining:

- $2\text{RTT} + \text{file transmission time} + \text{server overhead}$ og evt. kapacitetsbegrænsninger downlink til klient
- **Meget hurtigere for klienten**
- **Mindre krævende for server**, da den kun skal vedligeholde én forbindelse pr klient, dog skal klienten lukke forbindelsen så snart den er færdig.
- I eksemplet:
 - $1 * \text{HTTP Responstid (GET /page.html)}$
 - $1 * \text{RTT} + 3 * \text{fil transmissionstid}$

GET style.css
GET script.js
GET photo.jpg

- HTTP/2 tager denne ide videre



HTTP/2

Mål: mindske forsinkelsen i multi-objekt HTTP requests

HTTP/2: [RFC 7540, 2015] øget fleksibilitet hos *server* ved afsending af web-objekter til klient:

- metoder, status koder, fleste header felter uforandret fra HTTP 1.1
- Transmissions rækkefølgen af forespurgte objekter baseres på en prioritet angivet af klienten (ikke nødvendigvis FCFS: First-Come-First-Serve)
- Server kan afsende (*push*) objekter til klienten, den (endnu) ikke har forespurgt
- Opdeling af større objekter i "frames", scheduler frames for at formindste HOL ("Head of Line") blokering

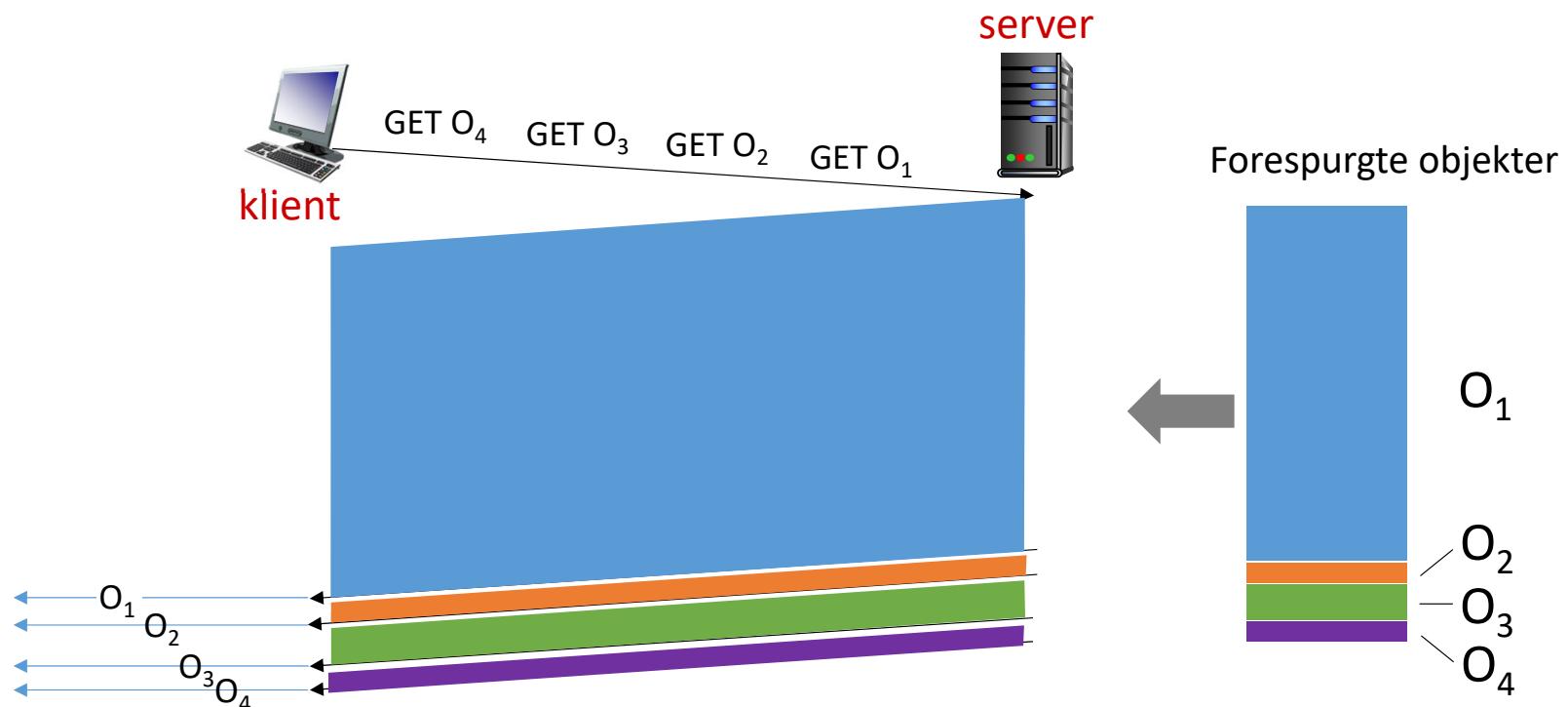
Head-of-Line Blocking

Et stort, langvarigt job blokkerer for fremdrift af (flere) ventende små
Fx i en supermarkeds kø



HTTP/2: reduktion af HOL blokering

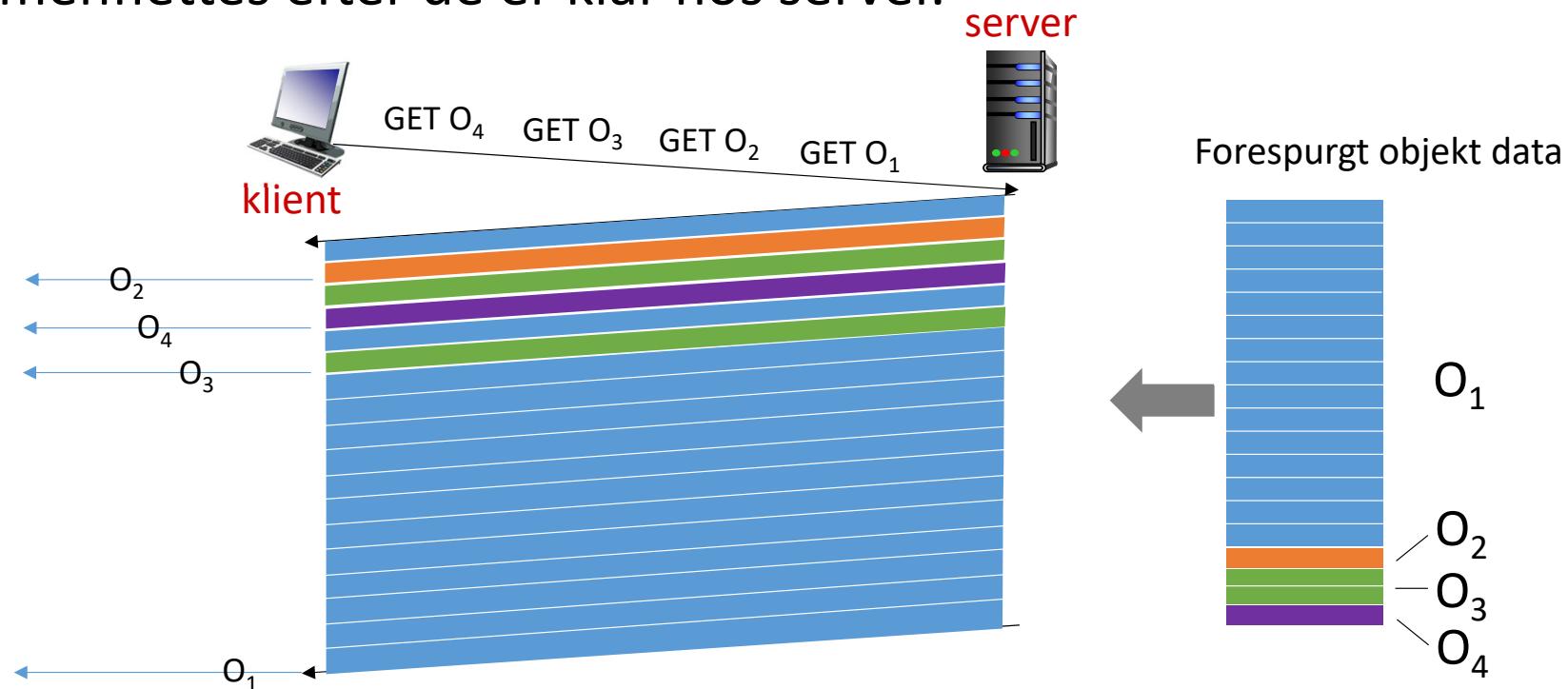
HTTP 1.1: klient beder om 1 stort objekt (e.g., video fil) and 3 mindre



I HTTP1.1 leveres svar objekter i den rækkefølge de efterspørges: O_2 , O_3 , O_4 venter "bag" O_1

HTTP/2: reduktion af HOL blokering

HTTP/2: objekt deles op i bidder ("frames"); afsendelse af frame sammenflettes efter de er klar hos server.



O₂, O₃, O₄ leveres hurtigt, O₁ forsinkes en smule

Fra HTTP/2 til HTTP/3

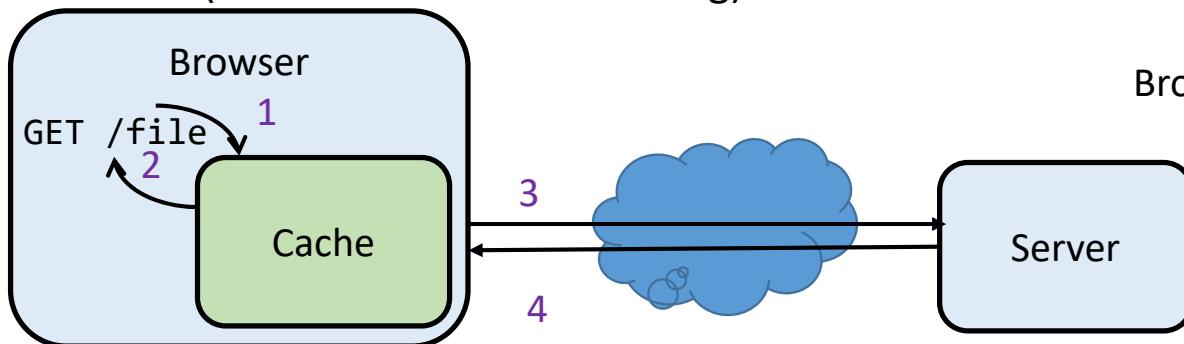
HTTP/2 over enkelt TCP forbindelse betyder

- Gentransmission ved pakketab får transmission af alle objekter til at gå i stå (“stalling) (TCP: ordnet, pålidelig byte-stream service)
 - Som i HTTP 1.1, får browser incitemment til at åbne multiple parallel TCP forbindelser to reducere stalling, og dermed øge samlet throughput
- Ingen sikkerhed over TCP connection
- **HTTP/3:** tilføjer sikkerhed, per objekt fejl- og congestion-kontrol over UDP (QUIC)
 - Lidt mere når vi kommer til transport laget

HTTP Caching

HTTP Caching

- En "cache" er et lokalt lager (forråd), der gemmer kopi af forespurgte objekter for at give ***hurtig*** adgang dertil
 - Optimeringstrick for programmer: gemmer og genbruger nyligt beregnede resultater
 - L1-L3 cache i computer arkitektur, disk/fil-cache,...
 - Web-cache
- I HTTP Web sammenhæng: netværket er "langsomt",
 - **Hurtigere svartid / visning af siden**
 - Mindsker trafikken til server (penge, strøm)
 - (Mindsker server belastning)



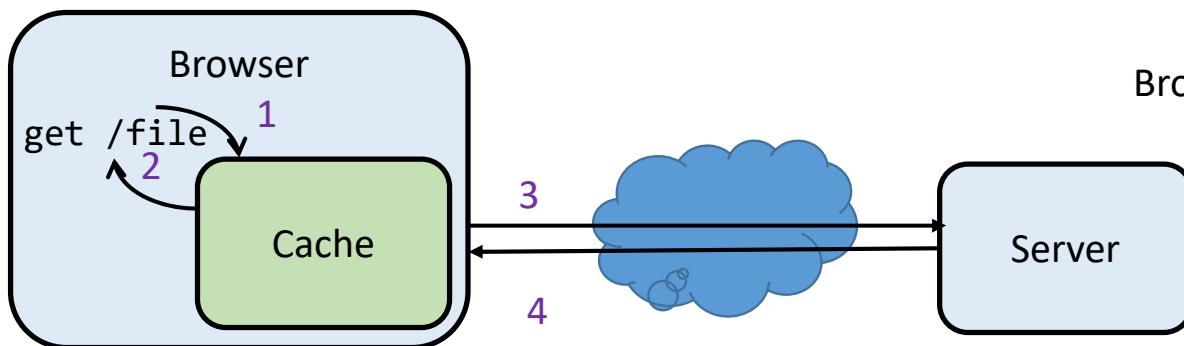
- Browser gemmer nyligt brugte objekter/filer på klient
1. Browser kigger først i cache om det efterspurgte objekt ("file") findes der
 2. Hvis ja, hent objektet derfra
 3. Hvis nej, send forespørgsel til web-server
 4. Gem objektet i cache, og leverer objektet

- **Hvad kan caches? Hvad hvis det ændres på server?**

HTTP Caching

- HTTP headers muliggør at *serveren* kan angive
 - Hvorvidt et objekt kan caches (om det er genbrugeligt)
 - Hvor længe kopien er gyldig, mm
- Kun serveren (og web-app programmør) kender til "foranderligheden" af objektet
 - Sjældent ændres: CSS filer, fleste billeder, iconer, varemærker, statiske html filer
 - Ofte ændres: dynamiske HTML baseret på DB opslag

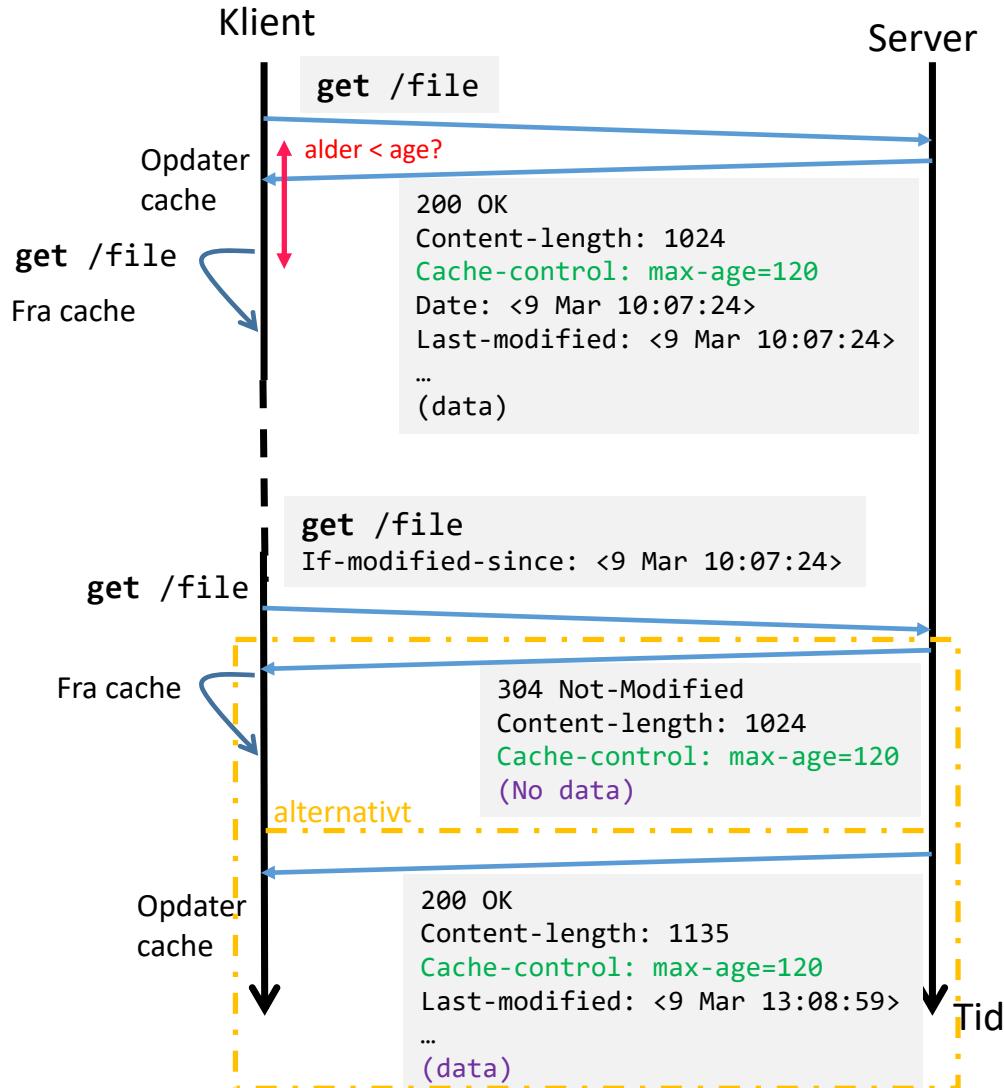
Cache-Control: max-age=<seconds>



- Browser gemmer nyligt brugte objekter/filer på klient
1. Browser kigger først i cache om det efterspurgte objekt ("file") findes der
 2. Hvis ja, hent objektet derfra
 3. Hvis nej, send forespørgsel til web-server
 4. Gem objektet i cache, og leverer objektet

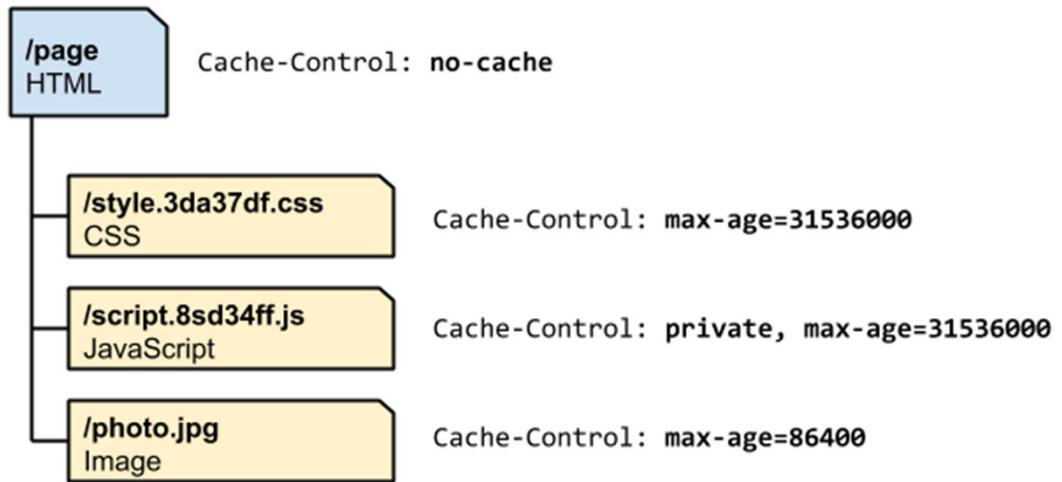
- Hvis objektet ændres, hvordan finder klienten så ud af den ikke bruger forældet info?

HTTP Cache kontrol: Betinget forespørgsel



- **Max-age:** angiver i sekunder hvor lang tid objektet kan betragtes som "frisk" og dermed om klienten må bruge det cachede objekt
 - Fresh: Tidsinterval fra tidspunkt hvor server genererer svar + max-age
 - Typisk beregnet på basis af Date eller Last-modified header i respons
 - Stale: sidste anvendelses dato overskredet
- Hvis Stale: Klient anvender "Conditional Get" som checker med server om cachens objekt er up-to-date
 - Klienten gemmer det modtagne tids-stempel sammen med objektet, og medsender dette
 - Server sender ét af 2 mulige svar:
 - Objekt er ikke ændret siden <tidsstempel>:
 - INGEN data, evt ny forlænget levetid
 - Objektet er ændret siden <tidsstempel>+data

HTTP Cache kontrol

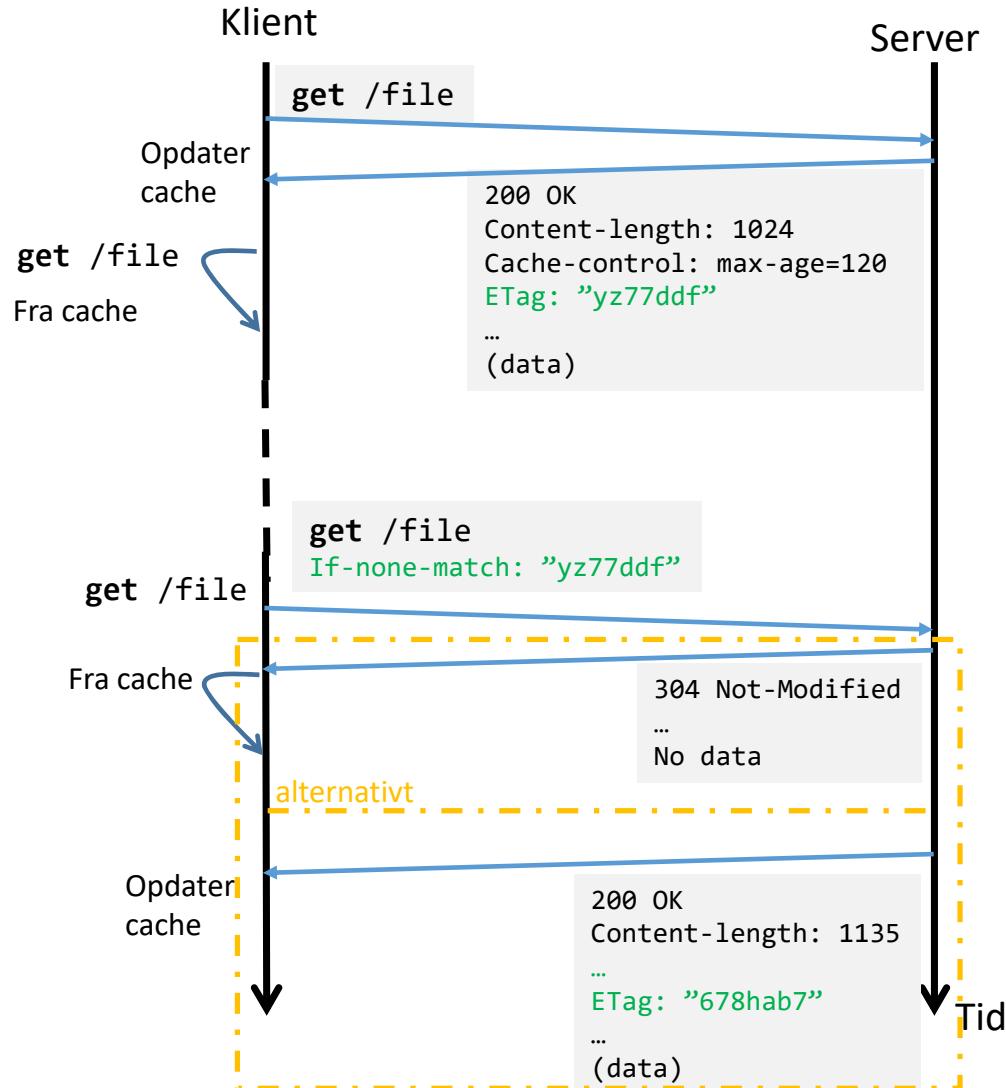


- **No-cache:** lokalt indhold skal altid valideres hos server først (sparer stadig data, hvis objektet må genbruges)
- **No-store:** Objektet må ikke gemmes i cache; skal hentes på ny for hver forespørgsel.
- **Private vs. public.** Privat på kun caches i klientens cache
- **Max-age , If-modified-since**
- ...
- **Hvad caches?**
 - Objekter hentet med GET + status kode (Læsning af ressource)
 - Objekter +status hentet med POST caches normal IKKE (ændring af ressource), skal eksplisit tillades
 - PUT / Delete respons må ikke caches
 - Objekter, der ikke er undtaget vha. cache-kontrol header
- Gennemtvivng reload i browser
- Tøm browser cache

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching>

<https://developer.mozilla.org/en-US/docs/Glossary/cacheable>

HTTP Cache kontrol: Betinget forespørgsel, TAGS

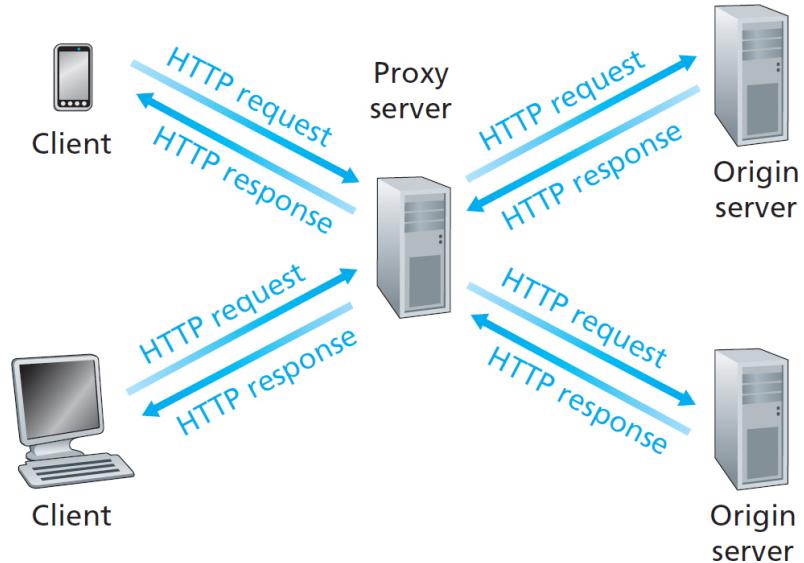


- **ETAG**: en streng, der fungerer som versionsnummer, som klienten kan bruge til at validere om indholdet et objekt den har er "ens" med serverens
- fingeraftryk, teknisk set ofte beregnet som et "hash"
- Conditional Get: checker om cachens version stemmer overens med serverens version
 - Klienten medsender det modtagte versionsnummer
- Flere anvendelser:
 - Tidsopløsning på tidsstemplér er lav (1sek)
 - Undgå "lost-update": Hvis 2 klienter *samtidigt* forsøger med opdatering (put) (overskrivning) kan et afvises

HTTP Proxies

HTTP er designet til at kunne fungere med proxy-servere (mellemliggende servere som er placeret imellem klient og oprindelses-server)

- Web-cache, delt med alle dens klienter
- Load-balancer til en server farm
- Indholds filter
- Anonymizer
- Compression/Encryption



Web-cache (placeret hos ISP)

- Reducerer svar tid til klients
- Reducerer belastning af servers
- Reducerer ekstern trafik brugt af store organisationer / ISP

Proxy og HTTPS

- En proxy er en "man-in-the-middle"
- Et web-fil, der er digitalt-signeret af origin kan ikke uden videre gemmes og sendes fra proxy
- En proxy som web-cache er derfor mindre effektiv ved HTTPS trafik

Domain Name System

Hvad er formålet med DNS?

Hvordan er det struktureret?

Hvordan laver DNS forespørgsler?

Hvilke (slutbruger) værktøjer findes til DNS?

Domain Name System

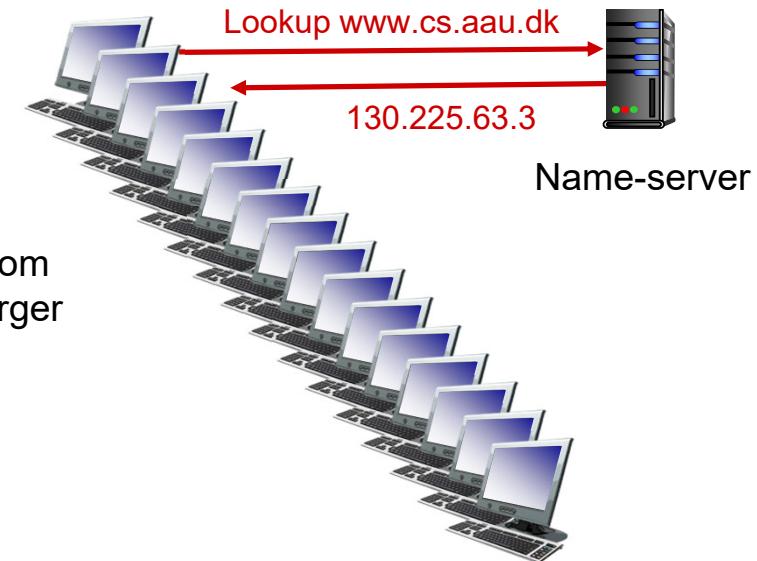
- Navngivning
 - Mennesker foretrækker meningsfylde navne "www.cs.aau.dk"
 - Hosts og routere fortrækker unikke numre (fx 32 bit IP adresse)
- Oversætter hostnavn → IP Adresse: www.cs.aau.dk → 130.225.63.3
- I Internettet barndom:
 - Oversættelserne blev gemt i hver host i en text fil: "/etc/hosts"
 - Ændringer sendt til central organisation på mail
 - Nye versioner blev hentet derfra med ftp.
 - Skalerede ikke, langsom opdatering, inkonsistens

Idé 2: Én central navne-server?

- Vi sætter en fed server op som vedligeholder database med alle host navne og tilhørende IP adresser
- Nem at forespørge
 - Lookup(www.cs.aau.dk)
- Kan let opdateres,
 - Update(www.cs.aau.dk=130.22.64.2)
- Flaskehals
 - MANGE klienter (> 1 mia)
 - MANGE forespørgsler (>1 mia / sekund)
- Single-point-of-failure
 - Hvis server bryder ned stopper nettet!!
- Skalerer ikke; Håbløs dårlig ide

=> DNS: Distribueret Hierarkisk Database

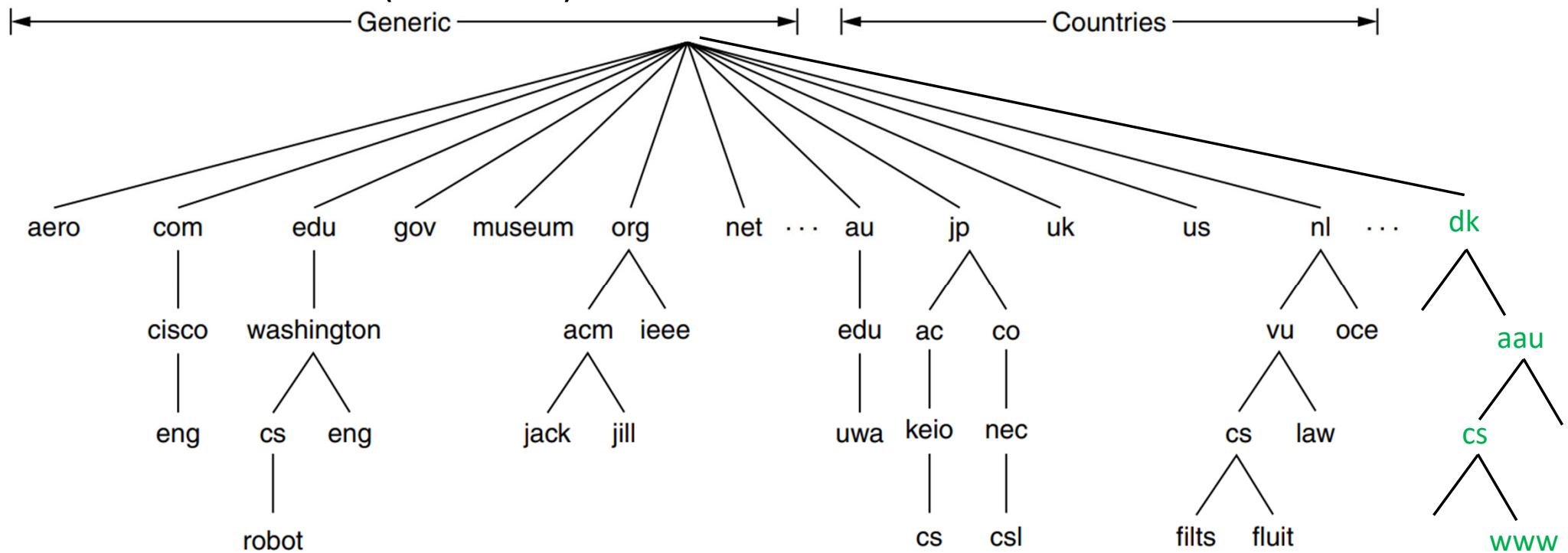
- Fordel belastning på mange servere, organiseret i et hierarki
- Undgå "single-point-of-failure"
- Applikationslagsprotokol: UDP (i enkelte særlige situationer TCP) port 53



navn	værdi
www.cs.aau.dk	130.225.63.3
www.google.com	172.217.4.228
...	
aau.dk	130.225.63.3
gaia.cs.umass.edu	128.119.245.12
dns01-bb.aau.dk	130.225.194.15
...	

Navnerum: www.cs.aau.dk

- Navnene er også organiseret i hierarkier
 - Top domæner (fx .com, .dk) og underdomæner (aau.dk)
 - Et domæne (fx. aau.dk) kontrollerer navnene derunder



DNS Dataposter

- Ressource Record
(Navn, Type, Værdi, TTL)
- Type=A
 - Navn er et hostnavn
 - Værdi er en IP adresse
- Type=NS
 - Navn er et domæne (fx aau.dk)
 - Værdi er navnet på den host, som er "autoritative" navne server for domænet)
- Type=CNAME
 - Navn er et alias navn
 - Værdi er navnet på den "rigtige" ("kanoniske") host
- Type=MX
 - Navn er et alias navn
 - Værdi er mailserveren, der er tilknyttet navnet

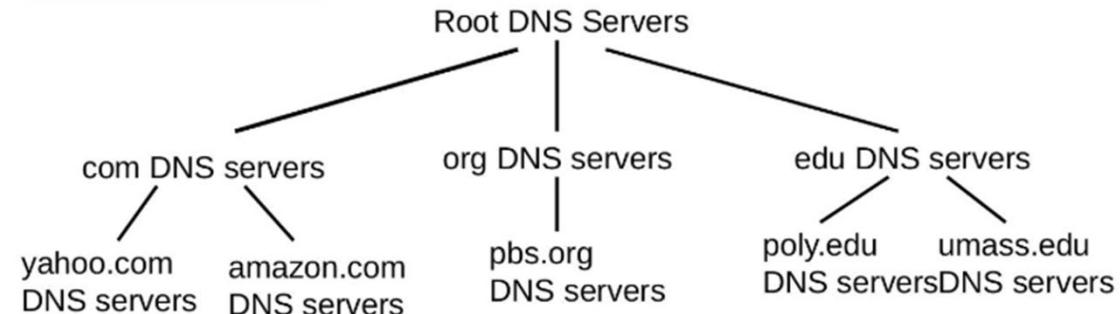
Eksempler

navn	type	værdi	TTL
www.cs.aau.dk	A	130.225.63.3	3599
www.google.com	A	172.217.4.228	299
www.aau.dk	CNAME	aau.dk	3599
aau.dk	A	130.225.63.3	3599
cs.aau.dk	NS	dns01-bb.aau.dk	3600
dns01-bb.aau.dk	A	130.225.194.15	3600
aau.dk	MX	aau-dk.mail. protection.outlook.com.	3599

TTL=Time-to-live (i sekunder)
3599 knap 1 time

DNS Servers

- **Autoritative DNS server:** Hver organisation med offentlige servere har en DNS-server
 - som har til ansvar at hoste DNS poster for dens domæne.
 - Primær og sekundær
- **Top-level-domain DNS:** DNS-Server(e) for hvert TLD
 - Videreformidle forespørgsler til en autoritativ server
- **Root DNS Servers**
 - Videreformidle forespørgsler til en TLD server
- **Lokal DNS server:**
 - En server som klienter bruger for at foretager DNS forespørgsler
 - Opsat af ISP
 - Konfigureret i klient (ofte vha. DHCP)



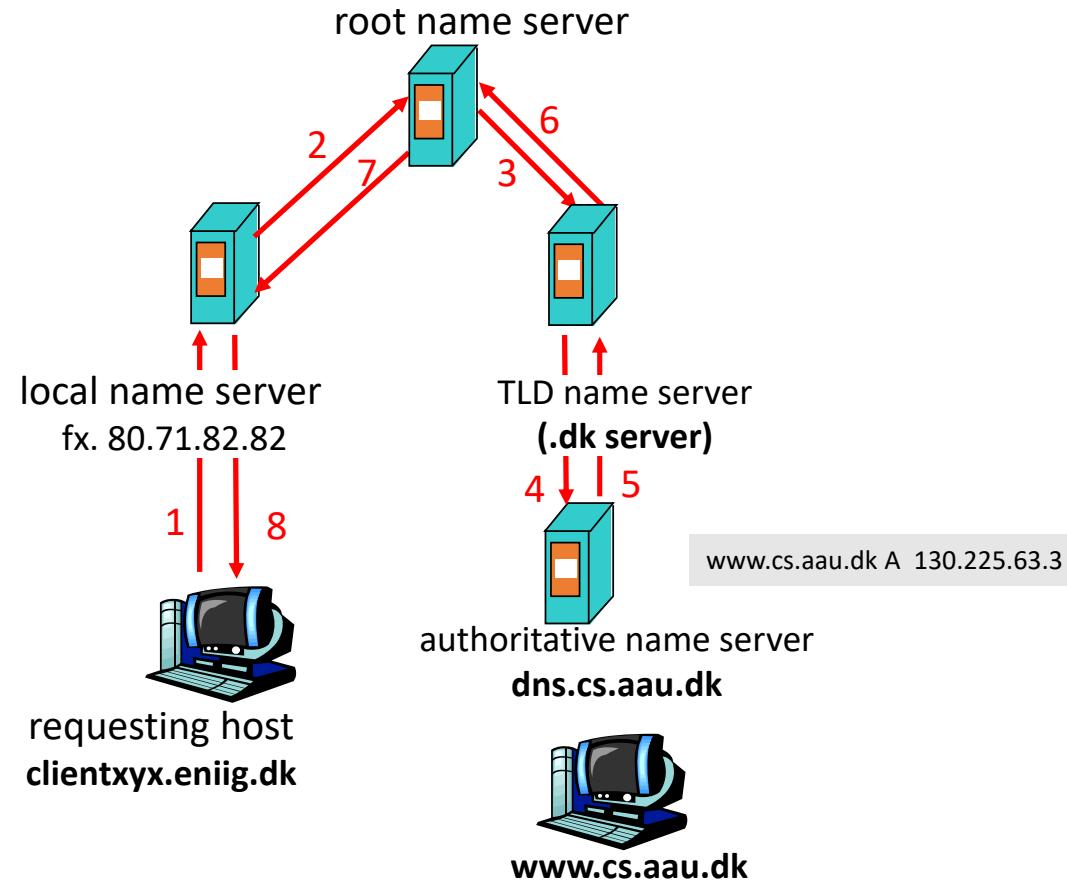
Princip skitse:

1. Klient kontakter lokal DNS server: IP for www.cs.aau.dk?
2. Den lokal kontakter en Root DNS server
3. RootDNS videreformidler til en den ønskede TLD-server
4. TLD videreformidler til den autoritative server
5. Den autoritative server sender svaret tilbage mod klienten 130.225.63.3

Rekursiv DNS Forespørgsel

Rekursiv DNS forespørgsel (query):

1. Klienten forespørger dens lokale NS
 2. Den lokale forespørger root
 3. Root NS forespørger TLD
 4. TLD forespørger en autoritative
 5. Resultat returneres til TLD
 6. Resultat returneres til root
 7. Resultat returneres til lokal
 8. Resultat returneres til klient
- Root name server:
 - Kender ikke nødvendigvis den autoritative, men henvender sig til en på lavere niveau (fx TLD) som den beder om at løse opgaven
 - Metafor: rekursivt procedurekald
 - Giver høj belastning på serverer på højere niveauer



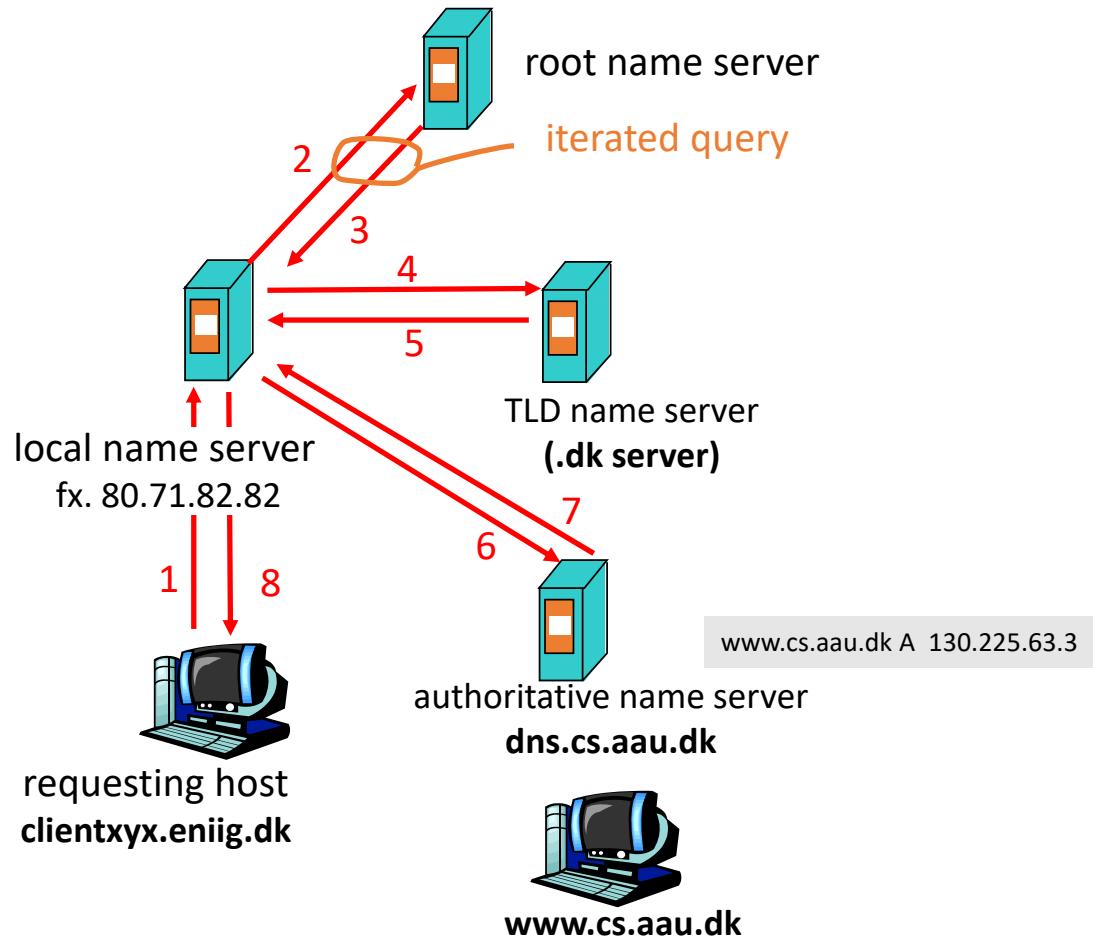
Iterativ DNS Forespørgsel

Iterativ DNS forespørgsel (query):

1. Klienten forespørger dens lokale NS
2. Den lokale forespørger root
3. Root svare tilbage med navn på TLD
4. Den lokale forsørger TLD
5. TLD svarer med navn på autoritative
6. Den lokale forespørger den autoritative
7. Den autoritative svarer den lokale med resultatet
8. Resultat returneres til klient

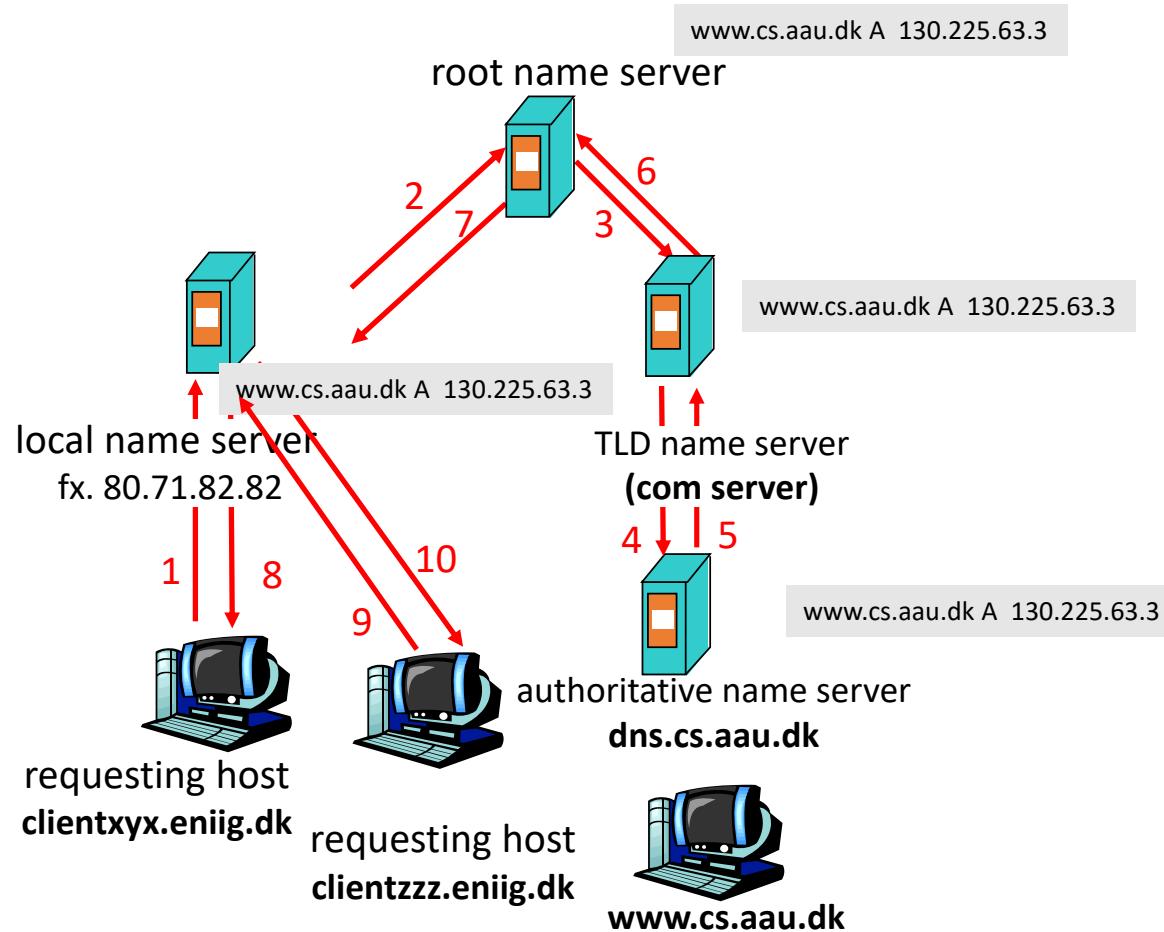
Skridt 3, 5:

- Den kontaktede server svarer med navnet på en server der kan kontaktes”
- “Jeg kender ikke navnet, men prøv denne server”
- Metafor: “while” løkke på den lokale



DNS caching

- Når en name-server NS får kendskab til en oversættelse (DNS post) gemmes denne lokalt hos NS i en cache (5), (6), (7), (8)
- Når NS næste gang får en ny forespørgsel på en host (9), ser den først efter i cache om den allerede kender svaret selv.
- Hvis ja (10), returneres svaret mod klienten, og den rekursive/iterative forespørgsel stoppes
- Fjerner meget belastning fra Root / TLD servere
- Hvad hvis en host får ny IP-adresse (fx www.cs.aau.dk flyttes til ny server) ??
 - Klinter bliver ved med at få gammel information fra caches
 - Derfor har hver DNS post en **TTL (time-to-live)**: et antal sekunder det må caches inden posten skal fjernes (udløbstid)
 - Det kan gå op mod TTL sekunder før den ny server bliver kendt

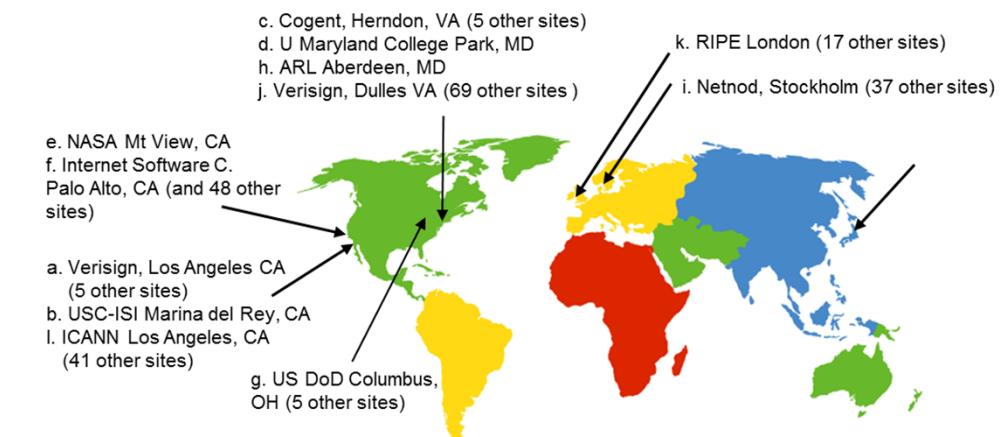


Root Servers

- 12 globale operatorer, 1698 servere
 - a.root-server.net
- En kontaktes altid når den lokale ikke kan oversætte navnet
- Centraliseret funktion ⇒ Spørgsmål:
 - Flaskehals?
 - Enkelt kilde til fejl (single point of failure)?
 - Undgås ved replikering og "caching"

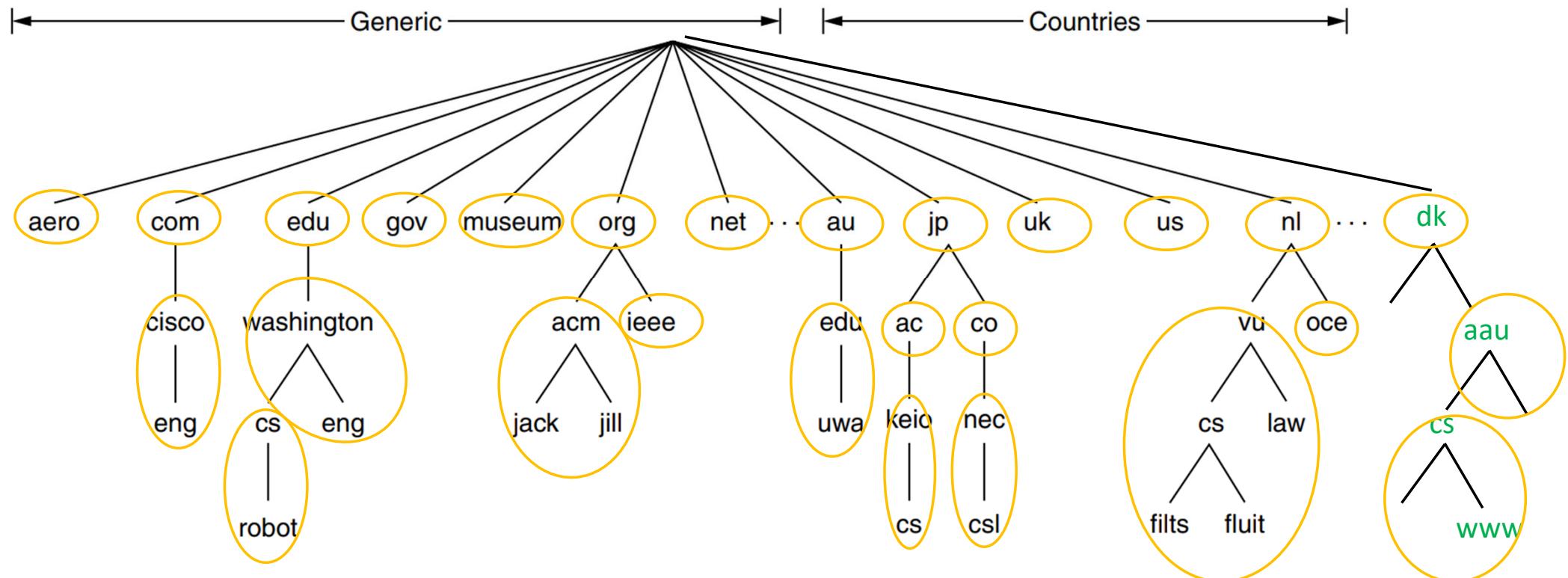
HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University of Maryland
e.root-servers.net	192.203.230.10, 2001:500:a8::e	NASA (Ames Research Center)
f.root-servers.net	192.5.5.241, 2001:500:2f::f	Internet Systems Consortium, Inc.
g.root-servers.net	192.112.36.4, 2001:500:12::d0d	US Department of Defense (NIC)
h.root-servers.net	198.97.190.53, 2001:500:1::53	US Army (Research Lab)
i.root-servers.net	192.36.148.17, 2001:7fe::53	Netnod
j.root-servers.net	192.58.128.30, 2001:503:c27::2:30	VeriSign, Inc.
k.root-servers.net	193.0.14.129, 2001:7fd::1	RIPE NCC
l.root-servers.net	199.7.83.42, 2001:500:9f::42	ICANN
m.root-servers.net	202.12.27.33, 2001:dc3::35	WIDE Project

Interaktivt kort: <https://root-servers.org/>



Navnerum: Zoner

- Zone er en underopdeling med en (eller flere) DNS servere



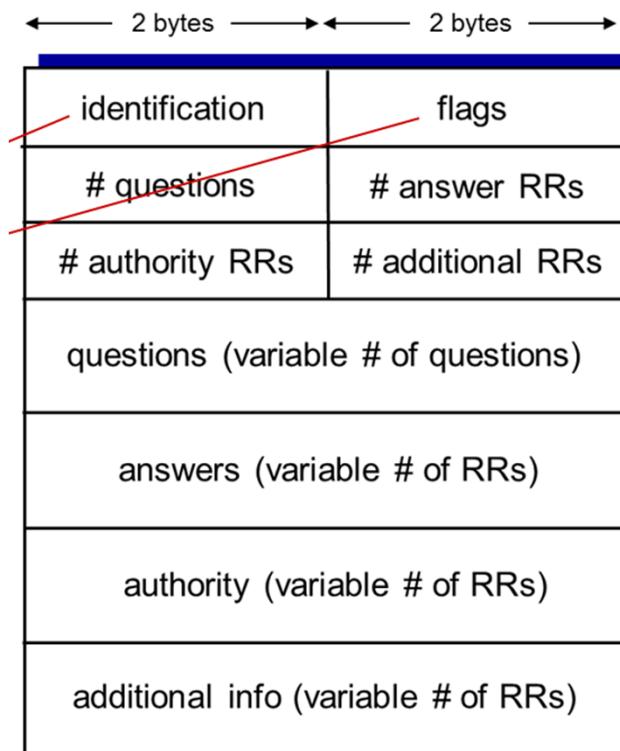
Opdatering af DNS

- Registrering af nye domæne
 - nsmd.dk (Nørresundby Micro Devices)
 - Køb og registrer domænet hos registraror (<https://www.dk-hostmaster.dk/> , minimums information
 - <nsmd.dk , NS, dns1.nsmd.dk >
<dns1.nsmd.dk, A , 123.123.123.123>
 - Info bliver sat ind i TLD server for .dk
- Opdatering af poster
 - Administrations-interface hos primær authoritative server
 - Dynamic-DNS: programmeringsmæssig / automatisk opdatering (fx. Ved opretelse af ny virtuel maskine med en server)

DNS meddelelser

DNS meddelelses format:

- Samme format for forespørgsler og svar
 - Et flag (bit) bestemmer hvilket
- Identifikation: 16 bit tal til at matche forespørgsel med svar
- Flag, fx:
 - Forespørgsel eller svar
 - Rekursivt foretrukket
 - Rekursivt tilgængeligt
 - Svaret er autoritativt
- Antal forespørgsler / svar i meddelelsen
- Navn, type felt for forespørgsel
- Poster for authoritative servere
- Yderligere info
- Transporteres på UDP



DNS meddelelses format: eksempel forespørgsel

Liste med opsnappede pakker,
sorteret efter protokoltype

Wireshark dekodet
info fra UDP payload

The screenshot shows the Wireshark interface with the following details:

- Network Interface:** *Ethernet
- Packets:** 596 total, 569 selected.
- Protocol:** DNS (selected packet 569)
- Decoding:** Transaction ID: 0x9437, Flags: 0x0100 Standard query, Questions: 1 (stanford.edu: type A, class IN), Response In: 5791.
- Hex View:** Shows raw bytes from 0000 to 0040, corresponding to the DNS message structure.
- Text View:** Shows the ASCII representation of the DNS message, including the question "stanford.edu: type A, class IN".

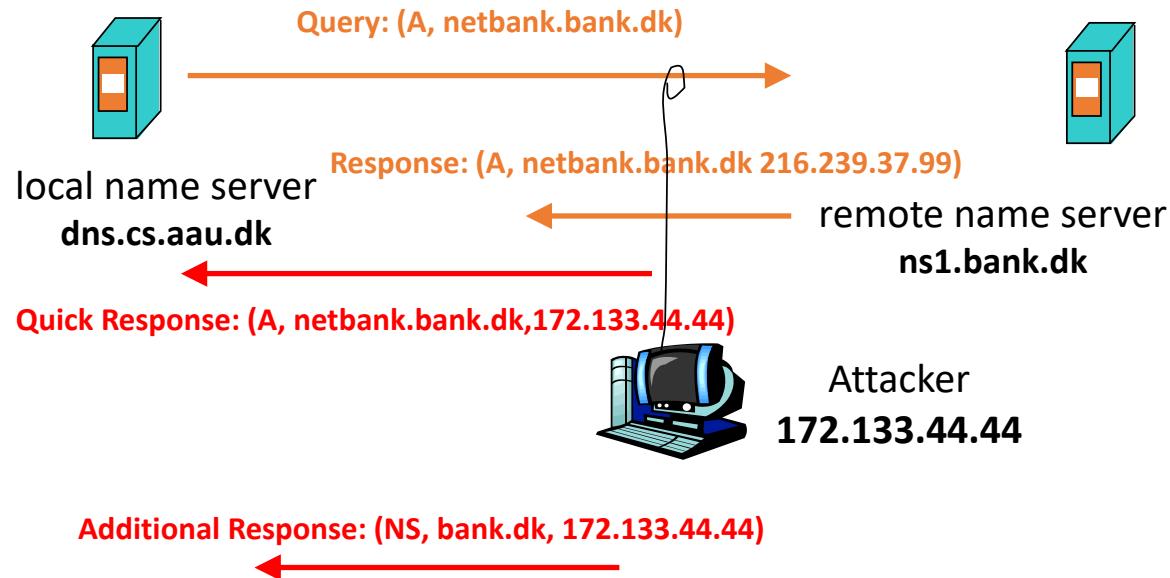
De "rå" bits, vist som
hexadecimale bytes og ascii tekst

DNS meddelelses format: eksempel svar

The screenshot shows a Wireshark capture of network traffic on the "Ethernet" interface. The packet list pane displays several DNS responses (Protocol: DNS) from source 192.168.0.181 to destination 192.168.0.1. The details pane for packet 579 (selected by clicking on its row in the list) provides a detailed breakdown of the DNS message. The selected message is a response (id 0) with transaction ID 0x9437. It has the following flags set: Response (1), Standard query response (0), Authoritative (0), Truncated (0), Recursion desired (0), Recursion available (0), Z reserved (0), Answer authenticated (0), Non-authenticated data (0), and Reply code (0). The message contains one question for "stanford.edu" and one answer pointing to IP address 171.67.215.200. The bytes pane at the bottom shows the raw hex and ASCII representation of the DNS response message.

DNS Sikkerhed

- Spoofing og Cache-forgiftning
 - Forfalske et svar og omdirigere klient til ønsket site
 - Lægge forfalskede oplysninger i en DNS cache
 - Teknisk svært (hastighed)
- Denial-of-service
 - Kan vi nedlægge en name-server ved at bombardere med forespørgsler
 - Nej, caching, og mange replikerede root server
- DNSSEC, en udvidelse baseret på kryptografi

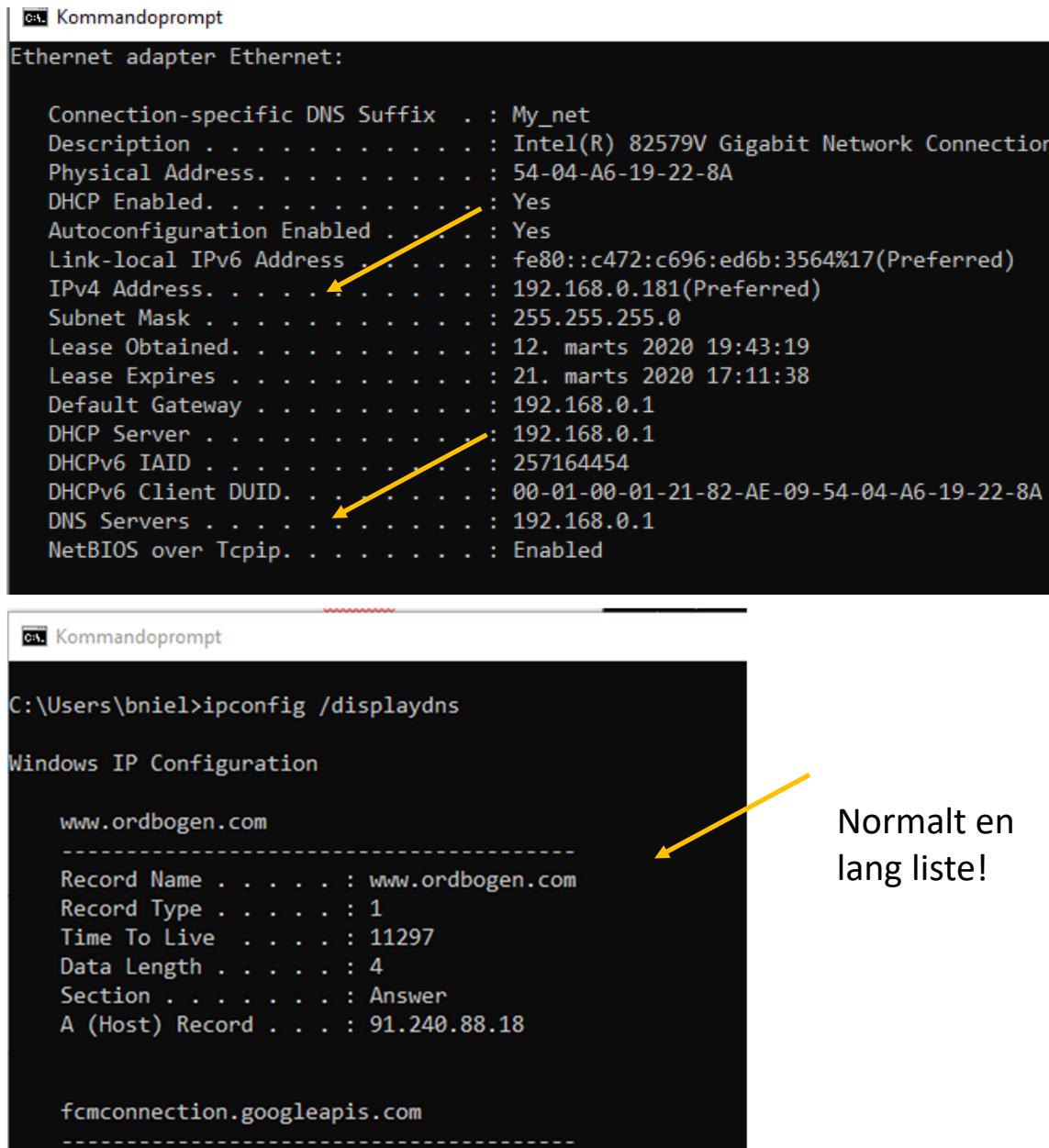


DNS Værktøjer

DNS Værktøjer

- Hvad er min lokale name-server
 - Kommer når din klient maskine konfigureres på nettet, oftest via DHCP (Dynamic Host Configuration Protocol) som også giver maskinen en IP adresse
 - Se Kontrol-panel
 - Eller kommando-linie
`>ipconfig /all`
- Hvilke DNS poster har min maskine cachet?
 - `>ipconfig /displaydns`

Linux / Mac har tilsvarende værktøjer



The image contains two screenshots of Windows Command Prompt windows. The top window shows the output of the command `ipconfig /all`. It lists various network parameters for the 'Ethernet adapter Ethernet' interface, including the IP address (192.168.0.181), subnet mask (255.255.255.0), and default gateway (192.168.0.1). Two yellow arrows point from the text 'Normalt en lang liste!' to the 'DNS Servers' entry and the 'A (Host) Record' entry for the domain 'www.ordbogen.com'. The bottom window shows the output of the command `ipconfig /displaydns`. It displays a list of DNS entries for the domain 'www.ordbogen.com', including the record name, type, time to live, data length, section, and A (Host) record. A yellow arrow points from the same text to the 'A (Host) Record' entry for 'fcmconnection.googleapis.com'.

```
Administrator: Kommandoprompt
Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : My_net
Description . . . . . : Intel(R) 82579V Gigabit Network Connection
Physical Address. . . . . : 54-04-A6-19-22-8A
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::c472:c696:ed6b:3564%17(PREFERRED)
IPv4 Address. . . . . : 192.168.0.181(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 12. marts 2020 19:43:19
Lease Expires . . . . . : 21. marts 2020 17:11:38
Default Gateway . . . . . : 192.168.0.1
DHCP Server . . . . . : 192.168.0.1
DHCPv6 IAID . . . . . : 257164454
DHCPv6 Client DUID. . . . . : 00-01-00-01-21-82-AE-09-54-04-A6-19-22-8A
DNS Servers . . . . . : 192.168.0.1
NetBIOS over Tcpip. . . . . : Enabled

Administrator: Kommandoprompt
C:\Users\bniel>ipconfig /displaydns

Windows IP Configuration

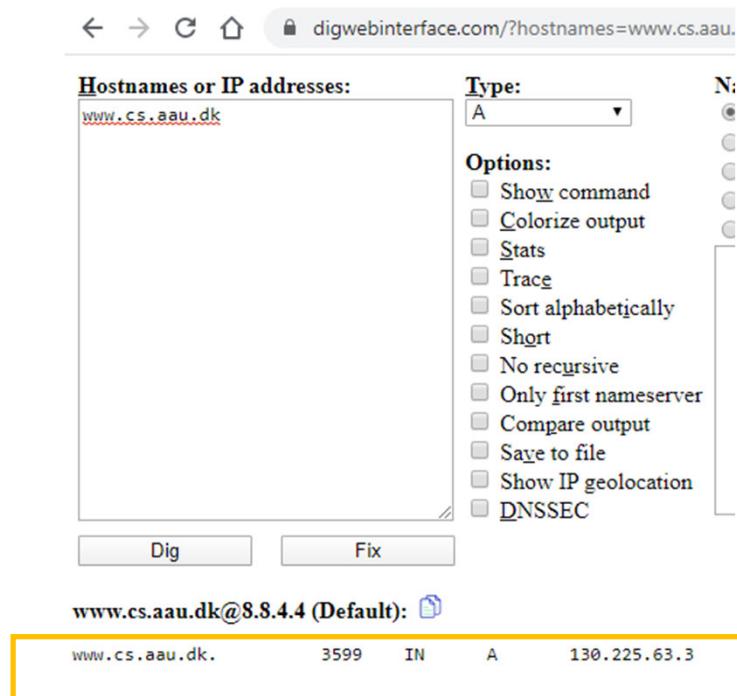
www.ordbogen.com
-----
Record Name . . . . . : www.ordbogen.com
Record Type . . . . . : 1
Time To Live . . . . . : 11297
Data Length . . . . . : 4
Section . . . . . : Answer
A (Host) Record . . . . : 91.240.88.18

fcmconnection.googleapis.com
-----
```

Normalt en lang liste!

DNS Værktøjer

- Dig (domain information groper)
 - Kommando linie
 - Web-interface, fx
<https://www.digwebinterface.com/>
- nslookup



The screenshot shows a web-based DNS query interface. In the search bar, the URL is digwebinterface.com/?hostnames=www.cs.aau.dk. The main area has two sections: "Hostnames or IP addresses:" containing "www.cs.aau.dk" and "Type:" set to "A". Below these are "Options:" checkboxes for Show command, Colorize output, Stats, Trace, Sort alphabetically, Short, No recursive, Only first nameserver, Compare output, Save to file, Show IP geolocation, and DNSSEC. At the bottom are "Dig" and "Fix" buttons. A yellow box highlights the results table below, which shows a single entry: www.cs.aau.dk@8.8.4.4 (Default): followed by a table row with columns: www.cs.aau.dk., 3599, IN, A, 130.225.63.3.

www.cs.aau.dk.	3599	IN	A	130.225.63.3
----------------	------	----	---	--------------



The screenshot shows a Windows Command Prompt window titled "Kommandoprompt". It displays two sets of nslookup results. The first set for "www.cs.aau.dk" shows a non-authoritative answer with Name: www.cs.aau.dk and Address: 130.225.63.3. The second set for "vm-ig-www2.portal.aau.dk" also shows a non-authoritative answer with Name: vm-ig-www2.portal.aau.dk and Address: 130.225.63.3. Both entries show a Server: Unknown and Address: 192.168.0.1.

```
C:\Users\bniel>nslookup www.cs.aau.dk
Server: Unknown
Address: 192.168.0.1

Non-authoritative answer:
Name: www.cs.aau.dk
Address: 130.225.63.3

C:\Users\bniel>nslookup 130.225.63.3
Server: Unknown
Address: 192.168.0.1

Name: vm-ig-www2.portal.aau.dk
Address: 130.225.63.3

C:\Users\bniel>nslookup vm-ig-www2.portal.aau.dk
Server: Unknown
Address: 192.168.0.1

Non-authoritative answer:
Name: vm-ig-www2.portal.aau.dk
Address: 130.225.63.3

C:\Users\bniel>
```

DNS værktøjer: whois

- Information om hvem ejer et givet domæne
- En distribueret database vedligeholdt af registratorer
 - <https://whois.icann.org/en/about-whois>
- FX <https://www.whois.com/whois/aau.dk>

aau.dk

Updated 3 days ago 

```
# Copyright (c) 2002 - 2020 by DK Hostmaster A/S
#
# Version: 4.0.2
#
# The data in the DK Whois database is provided by DK Hostmaster A/S
# for information purposes only, and to assist persons in obtaining
# information about or related to a domain name registration record.
# We do not guarantee its accuracy. We will reserve the right to remove
# access for entities abusing the data, without notice.
#
# Any use of this material to target advertising or similar activities
# are explicitly forbidden and will be prosecuted. DK Hostmaster A/S
# requests to be notified of any such activities or suspicions thereof.

Domain: aau.dk
DNS: aau.dk
Registered: 1997-10-31
Expires: 2023-12-31
Registration period: 5 years
VID: no
DNSSEC: Unsigned delegation, no records
Status: Active

Registrant
Handle: ***N/A***
Name: Aalborg Universitet
Address: Fredrik Bajers Vej 7K
Postalcode: 9220
City: Aalborg Øst
Country: DK
Phone: +4599409940

Nameservers
Hostname: auaw.aua.auc.dk
Hostname: noc.aua.auc.dk
```

Peer-to-Peer Systemer

Hvad er P2P arkitekturen?

I modsætning til client-server

Hvilke fordele og ulemper er der ved den?

Hvordan virker BitTorrent i principippet?

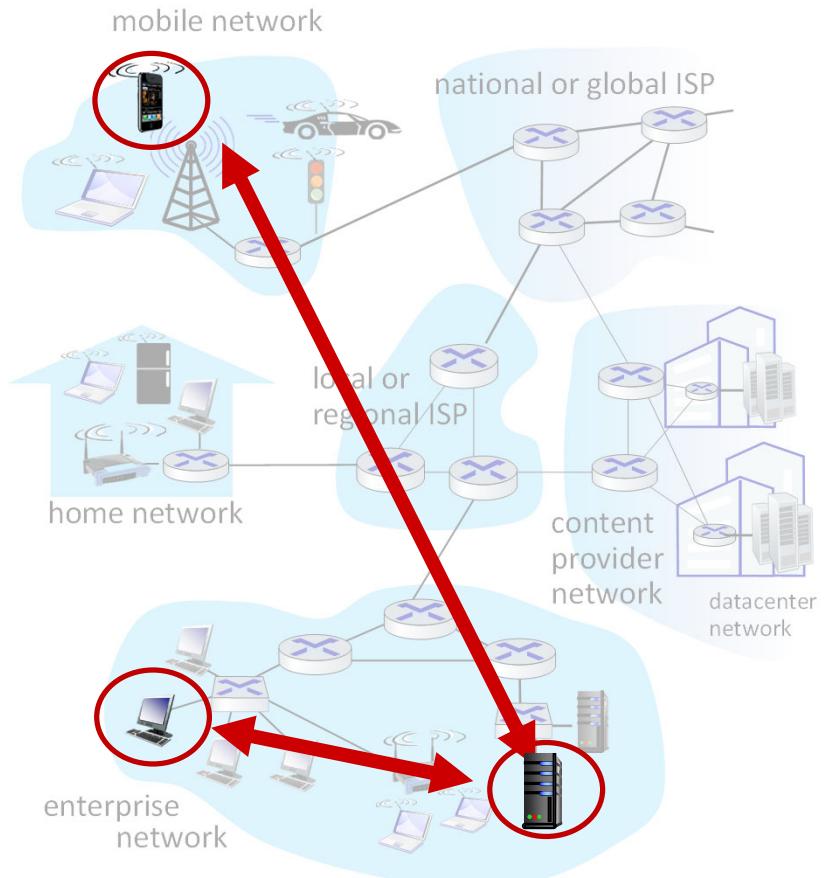
Klient-server modellen

server:

- Always-on host
- Permanent IP adresse
- Ofte i data centre, for skalering
- Tilbyder en speciel funktionalitet eller service
- Servicerer mange klienter

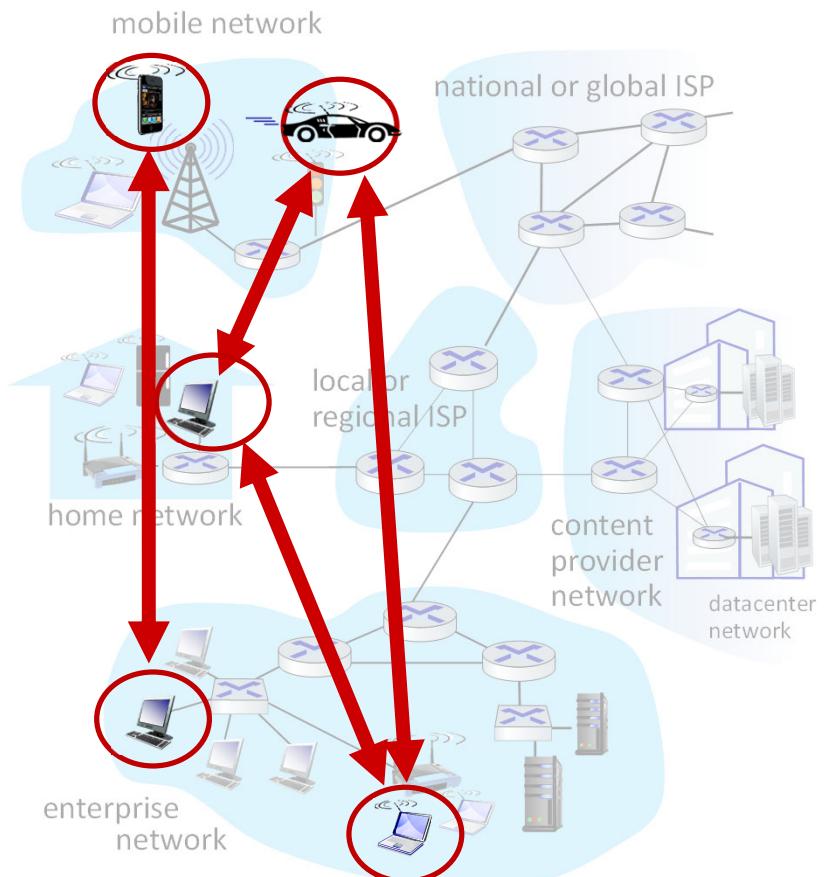
klienter:

- Tager kontakt til server
- Kommer og går
- Kan have dynamiske IP addresser
- Kommunikerer *ikke* direkte indbyrdes
- Fx.: HTTP, IMAP, FTP



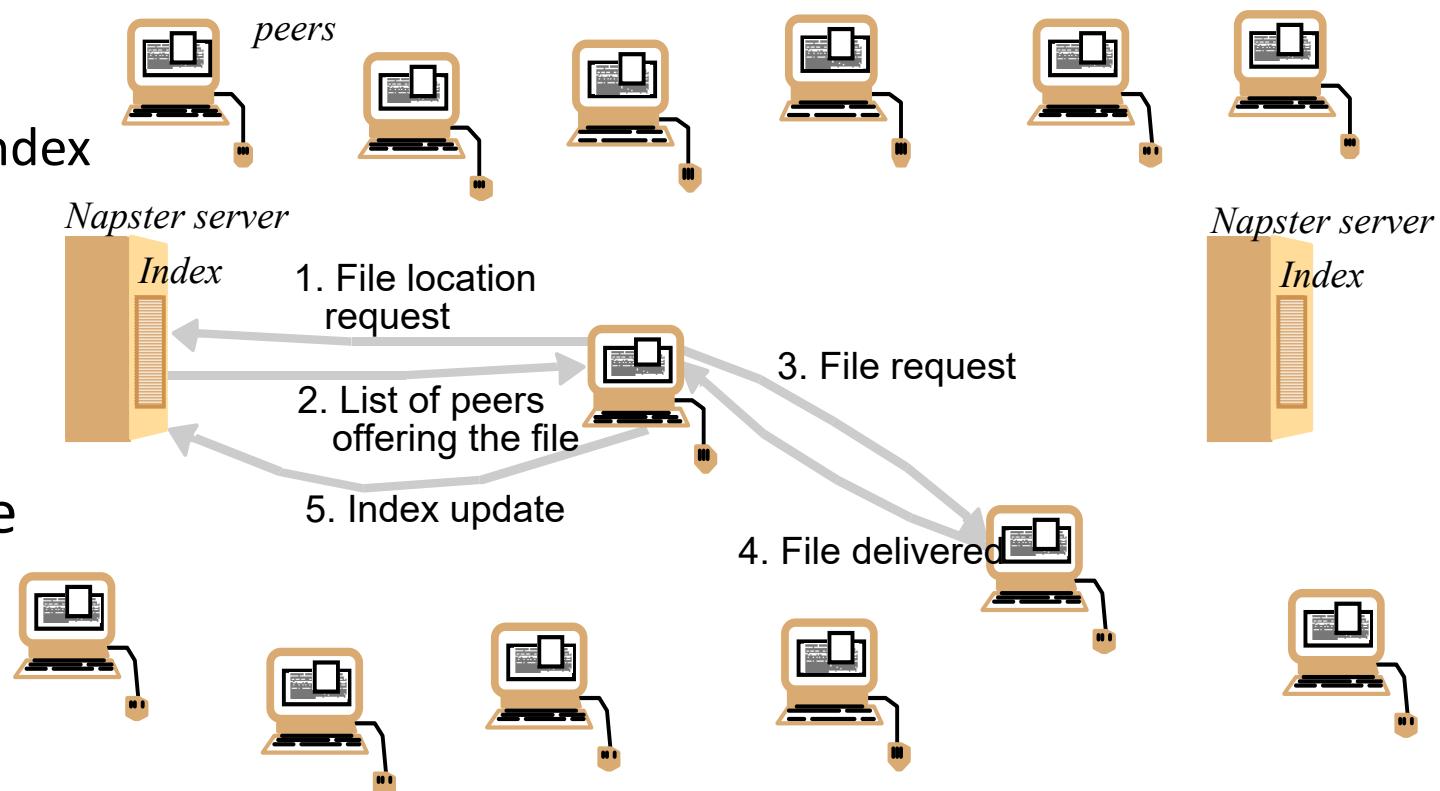
Peer-to-peer Modellen

- Ingen permanent server
- arbitrære hosts kommunikerer direkte
- peers forespørger service direkte fra andre peers, og tilbyder selv servicen i bytte til andre
 - *Selv-skalerende* – nye peers bringer ny kapacitet til service så vel som efterspørgsel på service
 - Ingen enkelt ejer ⇒ *Svært at lukke ned*
 - *Oppetid*: ingen centraliseret server, men klienter kommer og går spontant
- Danner et ”overlejret” netværk
 - peers er midlertidigt og løst forbundne, udveksler IP adresser
 - Kompliceret at vedligeholde af dette
- Fx P2P fildeling: Fil distribution (BitTorrent)
 - Streaming (KanKan), VoIP (Skype), TOR, multicast, spil ...
- Varmt forskningsområde i 0’erne
 - Distribuerede Hash Tabeller til hurtig søgning efter data



Pionererne

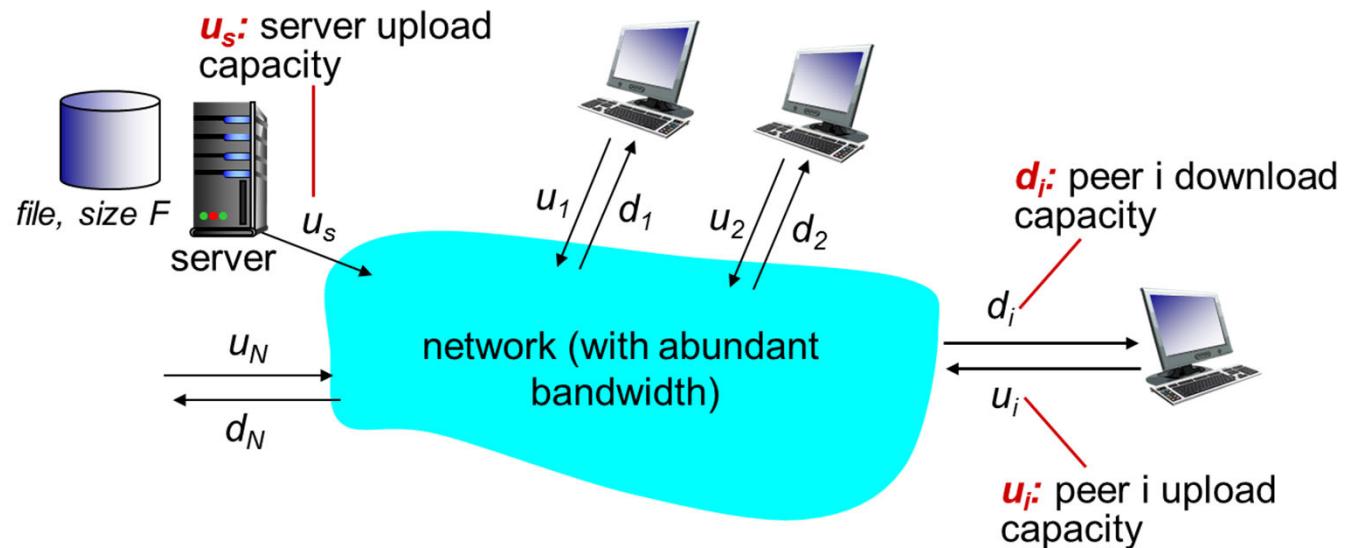
- Napster: 1999-2001
 - Deling af (.mp3) filer
 - Centraliseret, replikeret index
 - Lukket ned efter retskendelse,
 - 24 millioner brugere
- P2P blev synonym med ulovlig deling af materiale beskyttet af ophavsret
- Andre tidlige
 - Gnutella, Freenet



File Distribution: Client-Server vs P2P

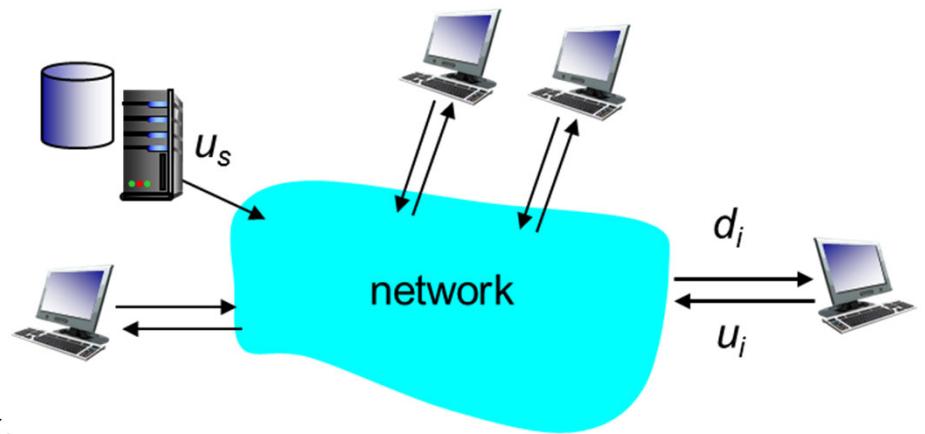
Spørgsmål: Hvor lang tid tager det at fordele en fil (af størrelsen F bits) fra én server til N peers?

- Vi sammenligner baseret på en overslagsberegning
- Antagelse: flaskehalse er i adgangsnetværket, men linkets fulde kapacitet er tilrådighed



Fil Fordelingsstid: Client-Server

- **Server transmission:** skal sende (upload) N kopier af filen med F bits:
 - Tid ialt: NF bits med u_s bps: $\frac{NF}{u_s}$ sek.
- **Klient:** hver klient skal downloade en kopi
 - d_{min} = download-rate fra klienten med mindste hastighed (bps)
 - Denne klient bestemmer samlet download tid: $\frac{F}{d_{min}}$ sek.



- **Samlet fordelingstid** $D_{cs} \geq \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\}$
- Bemærk: behovet stiger lineært, med N men serverens kapacitet er konstant
 - Med stort N bliver dette den dominerende faktor

Fil Fordelingsstid : P2P

- **server transmission:** skal uploadé mindst en kopi

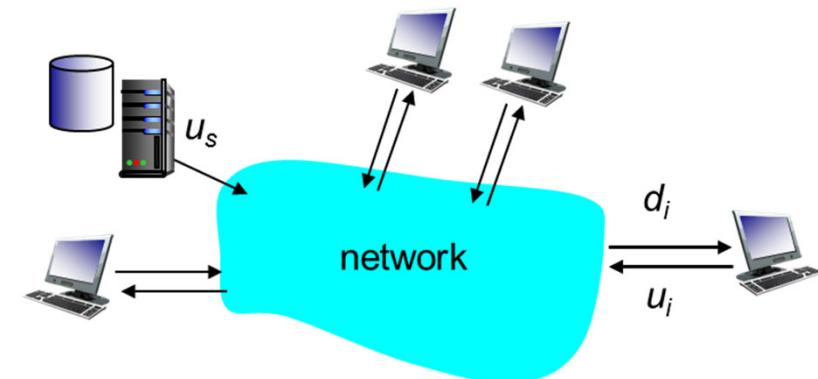
- Tid at sende en kopi: $\frac{F}{u_s}$

- Hver klient skal downloadé en kopi

- Tid til at downloadé en kopi hos langsomste: $\frac{F}{d_{min}}$

- Systemet samlet

- Klienter har et download behov NF bits
- Upload kapacitet: $u_s + \sum u_i$



- **Samlet** fordelingstid $D_{p2p} \geq \max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum u_i} \right\}$

- Bemærk at både behov og kapacitet stiger lineært i antallet af klienter

Client-Server vs P2P: Eksempel

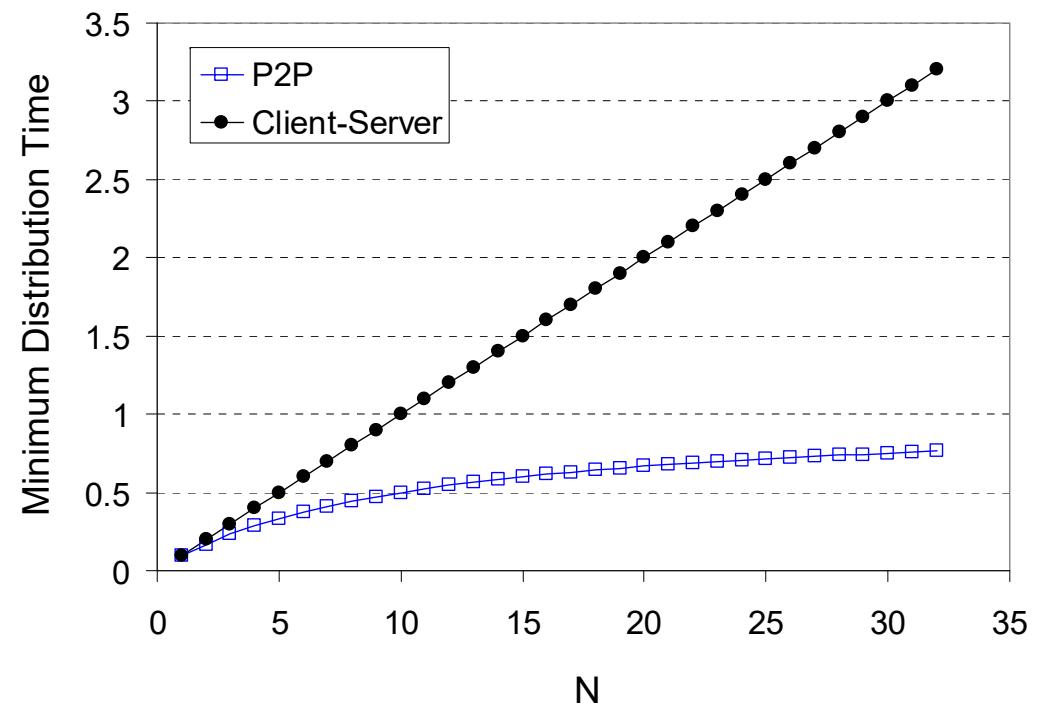
- Hvordan skalerer systemet?

- *Plot af afgørende termer*

$$\text{Server-baseret: } \frac{NF}{u_s}$$

vs.

$$\text{P2P baseret: } \frac{NF}{u_s + \sum u_i}$$



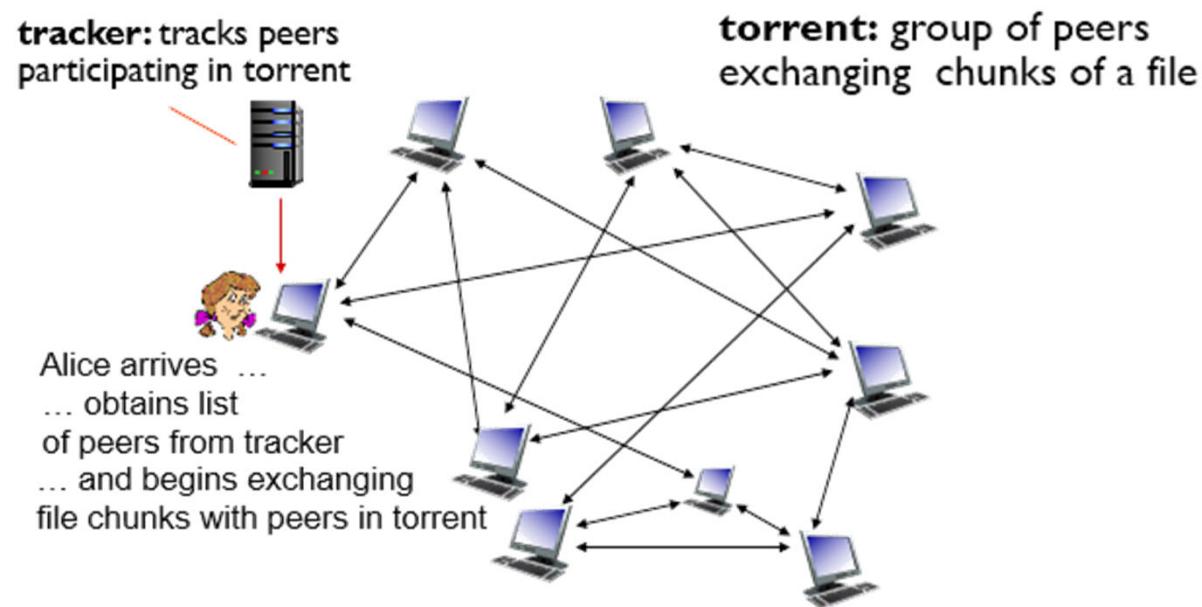
Problemstillinger ved P2P netværk

- Klienter er meget dynamiske: melder sig ofte til og af systemet
 - Afmelding: Resourcer og data forsvinder fra systemet
 - Systemet skal stadig være nogenlunde velfungerende
- Klienter vil gerne nyde; i mindre grad yde
 - Hvordan laver vi en “fair” og jævn belastning af klienterne
- Hvordan kan klienter finde hinanden og “ønskede filer” uden servere?
 - Brug andre web-sider til at annoncere tilstedeværelse

Netværkstekniske problemstilling da fleste klienter ikke har en offentlig fast IP (“NAT”)

P2P Fil Distribution: BitTorrent

- **Torrent** ("rivende strøm"): en gruppe af peers som udveksler en given fil
 - fil opdelt i portioner af 256 Kbytes: chunks
 - peers som deltager i en torrent sender og modtager chunks af filen
- **Tracker**: host, der vedligeholder liste med deltagende peers
- Ny "tom" peer melder sig hos tracker
 - Får tilsendt en liste med peers, den kan forbinde sig til (en delmængde af)
 - Peer kan vælge nye peers
 - Mens den downloader en chunk, uploads tidligere downloadede chunks
 - Vil gradvis (forhåbentligt) akkumulere alle chunks
- Mange lærerige aspekter, som vi ikke når



Opgaverne idag

- Review: Har man forstået grundlæggende begreber
 - Klient og server ”roller”, cookies, DNS forespørgsler
- Øvelser: Kan man anvende dem i nye eksempler?
 - Undersøgelse af HTTP header
 - Tidsforskelt mellem forskellige HTTP forbindelses-strategier
 - HTTP cache (conditional get)
 - P2P download tider
- Praktiske: Kan anvende netværksværktøjerne
 - Wireshark: opsnappe og undersøge de udvekslede DNS meddelelser

SLUT