

Internetwork og Web-programmering

Applikationslaget

Forelæsning 10

Brian Nielsen

Distributed, Embedded, Intelligent Systems



Agenda

1. Applikationslagsprotokoller
2. HTTP
 1. Meddelels-struktur
 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?

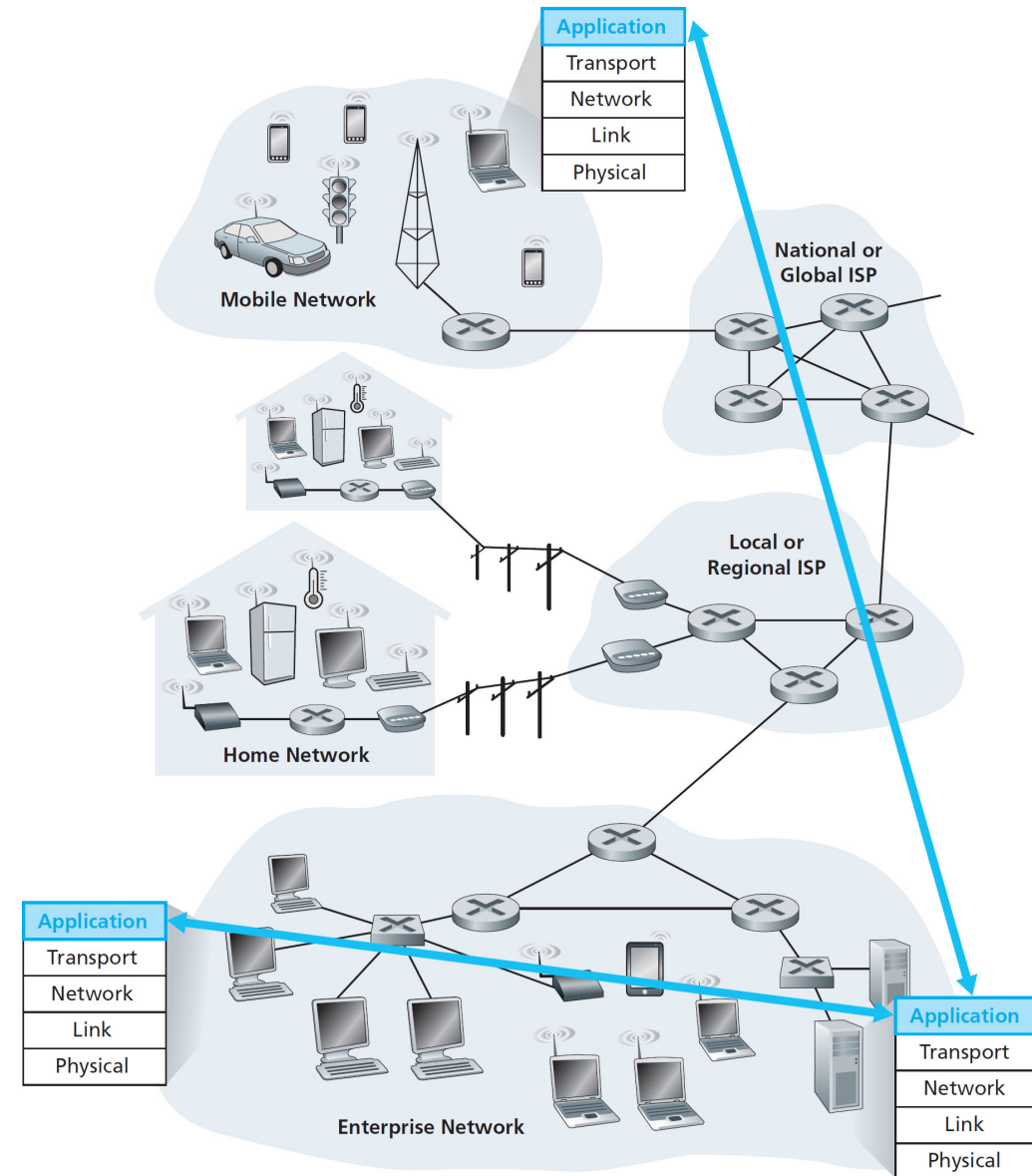
Netværks applikationer

Programmer som:

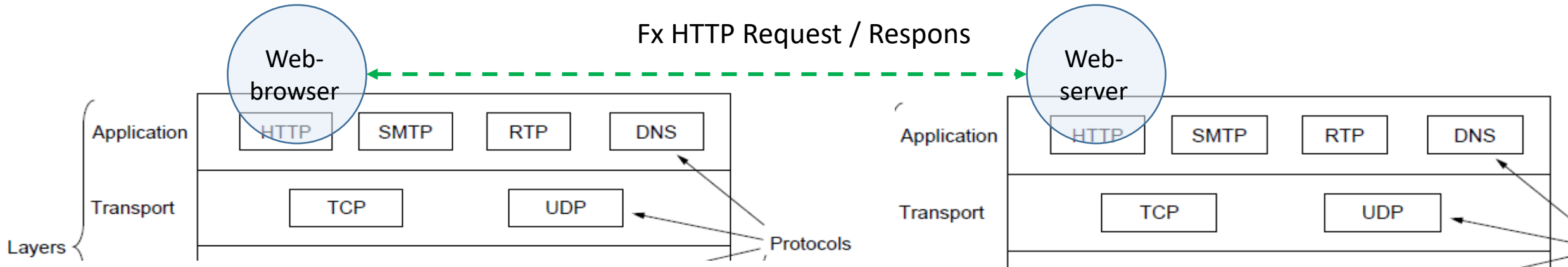
- Afvikles på (flere forskellige) **end systems**
- Kommunikerer over netværket
- Anvender applikations-niveau protokoller

Eksempler

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



Applikationslagsprotokoller



En protokol definerer:

- Hvilken **type meddelelser** bliver udvekslet?
 - fx., request, response
- Meddelelses **syntax**:
 - Hvilke felter er der i meddelelsen, og hvordan afgrænses 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

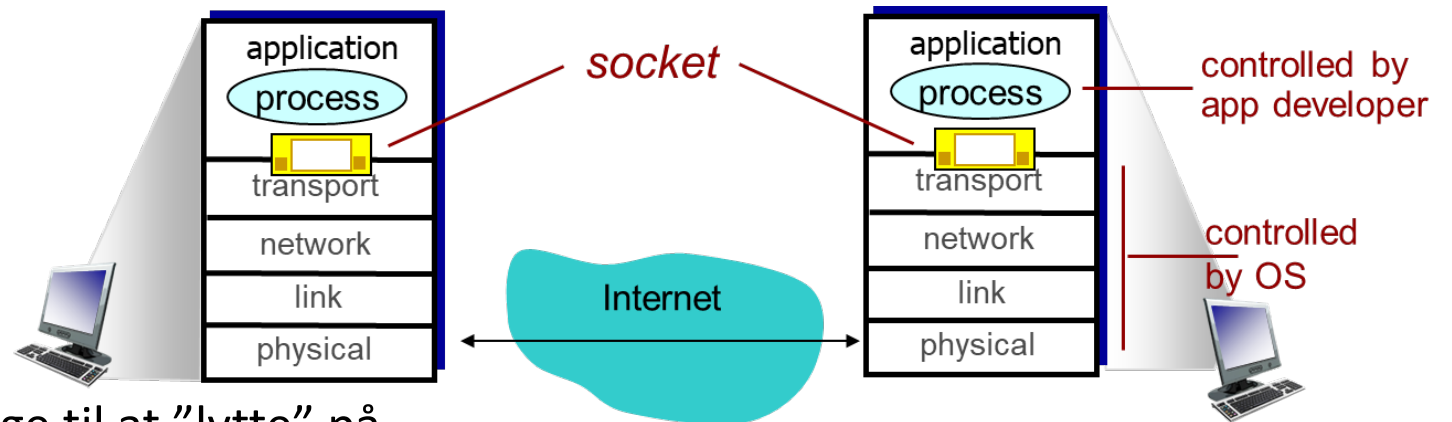
- e.g., Skype

Processer

- 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"
- Applikations niveau protokol gives ofte som et program bibliotek / "package", som kan kompileres sammen med applikationen, fx i node.js: `require('http');`
- Processer sender/modtager meddelelser via en **socket**, et bindeled/dør mellem applikationslag og transport lag

- Proces adresseres vha.

- Hosts IP-nummer
- Et port nummer på host
- Fx 130.225.63.3:80 (cs web-server)
- Server forventes at lave en socket til porten, som den bruger til at "lytte" på



Velkendte Porte

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

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

Applikationskrav til kommunikation

Application	Data Loss	Throughput	Time-Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time/ interactive audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
text messaging	no loss	elastic	yes and no

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

- kryptering,
- data integritet (ingen modificering),
- Authentication

Internet transport protokoller

TCP service:

- **pålidelig** transport af en byte stream (“rørledning”) 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 service:

- ~~“upålidelig”~~ **best-effort** data transport
 - **Datagrammer**
 - **Kan mistes, dublikeres, omordnes af netværket**

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

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

HTTP

Hvad bruges HTTP kontrol headers til?

Hvordan struktureres/formateres HTTP meddelelser?

Hvordan effektiviseres HTTP kommunikation?

- Hvordan håndteres forbindelser til serveren?

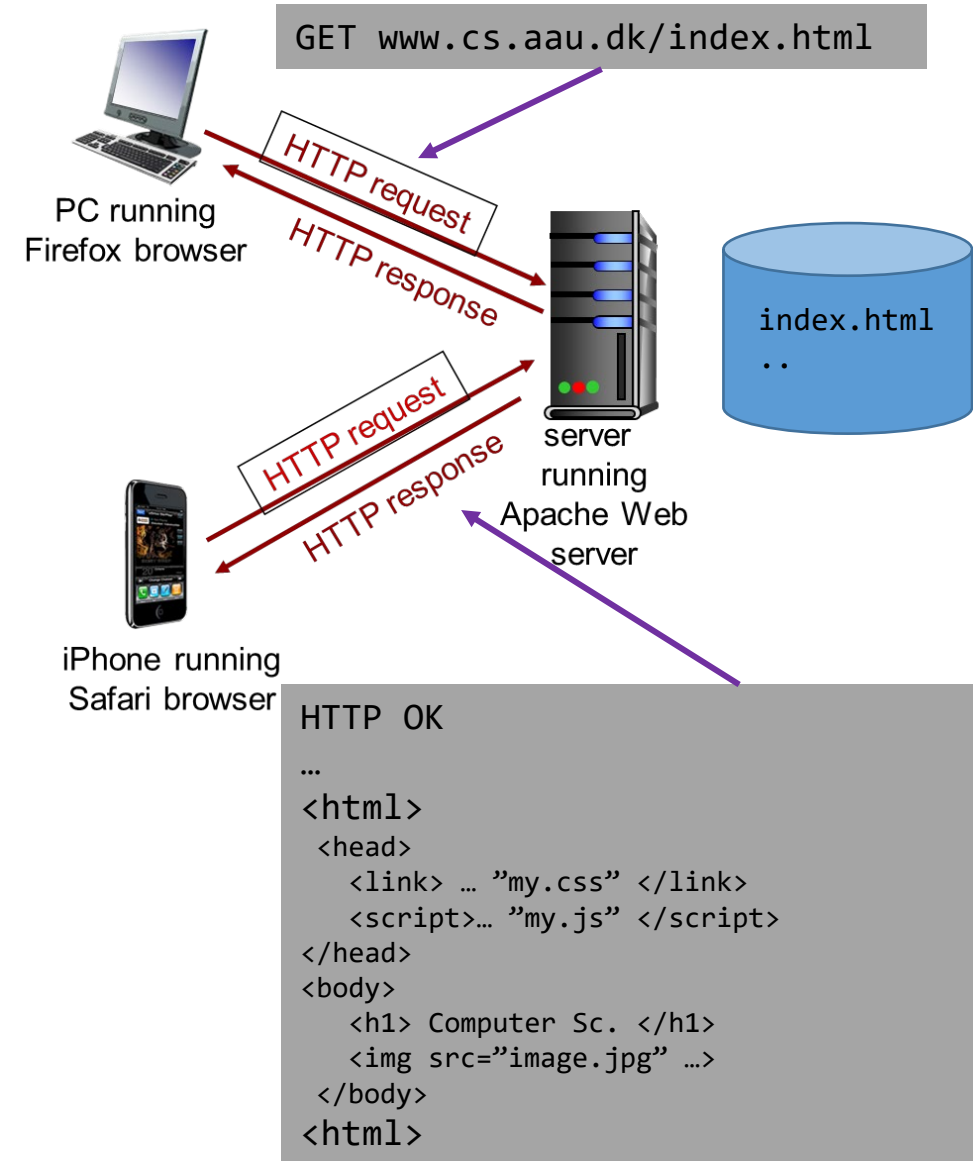
- Hvorfor og hvordan "cacher" HTTP dokumenter?

Simple HTTP Scenario: static html file

HTTP: hypertext transfer protocol

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

Det eksakte forløb afhænger af HTTP protokol version (jfv. Forelæsning om applikationslagsprotokoller)



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 Meddelelser (syntax)

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

request line
(GET, POST,
HEAD commands)

header
lines

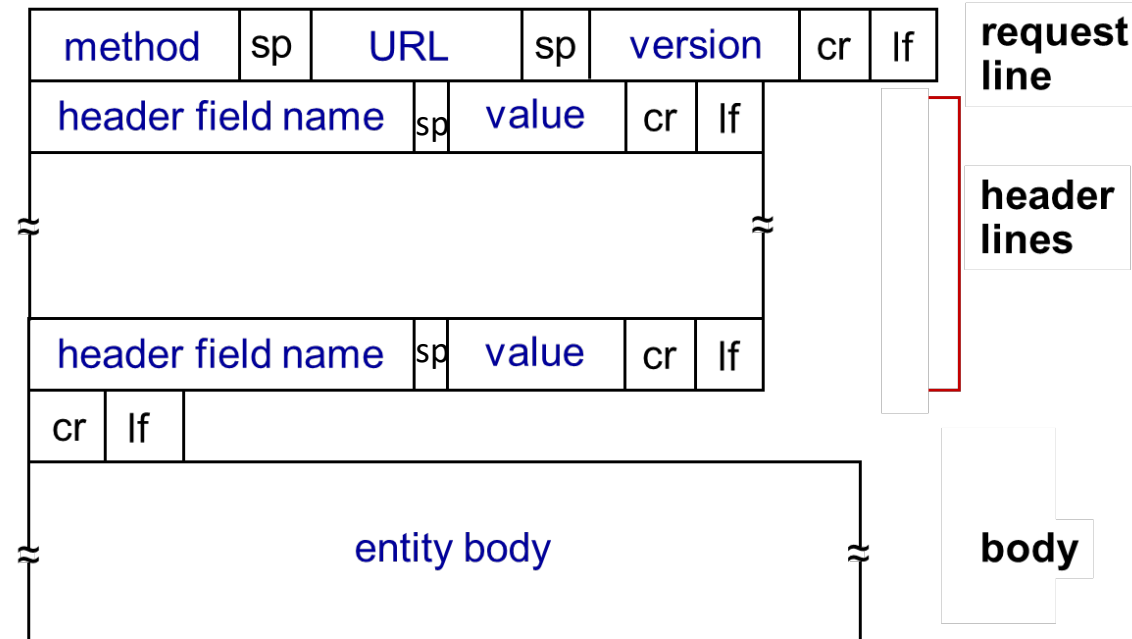
carriage return,
line feed at start
of line indicates
end of header lines

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

space

carriage return character
line-feed character

Generelt format for HTTP forespørgsel



HTTP respons har en tilsvarende format

HTTP Meddelelser: Respons

- HTTP Respons
 - Et lignende format

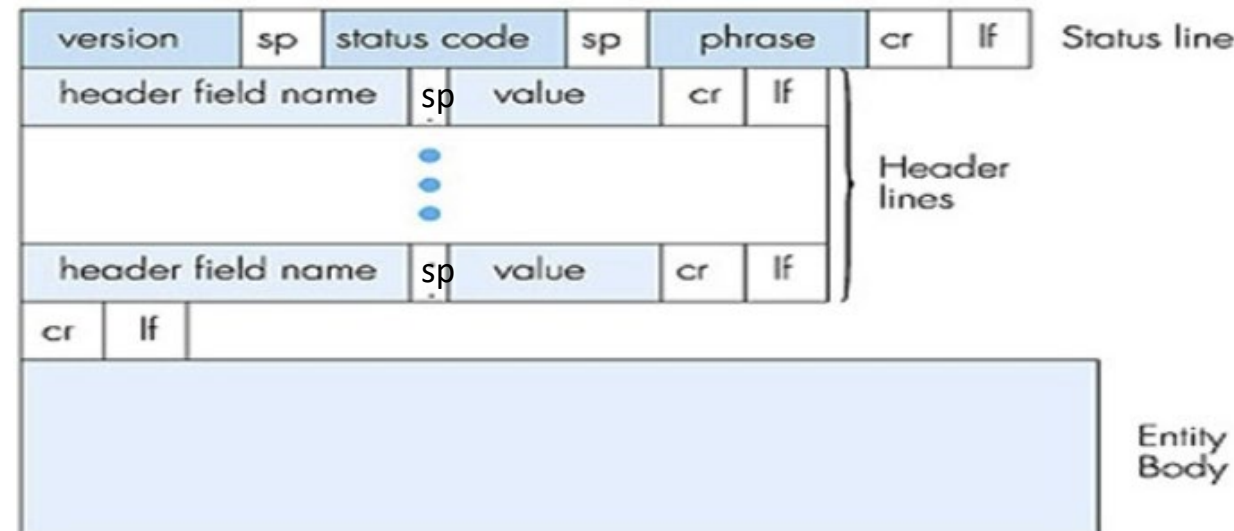
status line
(protocol
status code
status phrase)

header
lines

data, e.g.,
requested
HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
  GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
  1\r\n
\r\n
data data data data data ...
```

Generelt format for HTTP respons



HTTP

Forbindelseshåndtering

Hvordan effektiviseres HTTP client-server kommunikation?

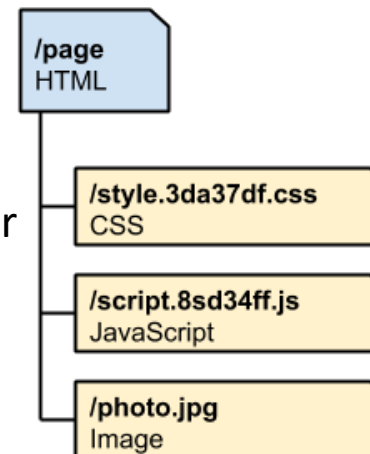
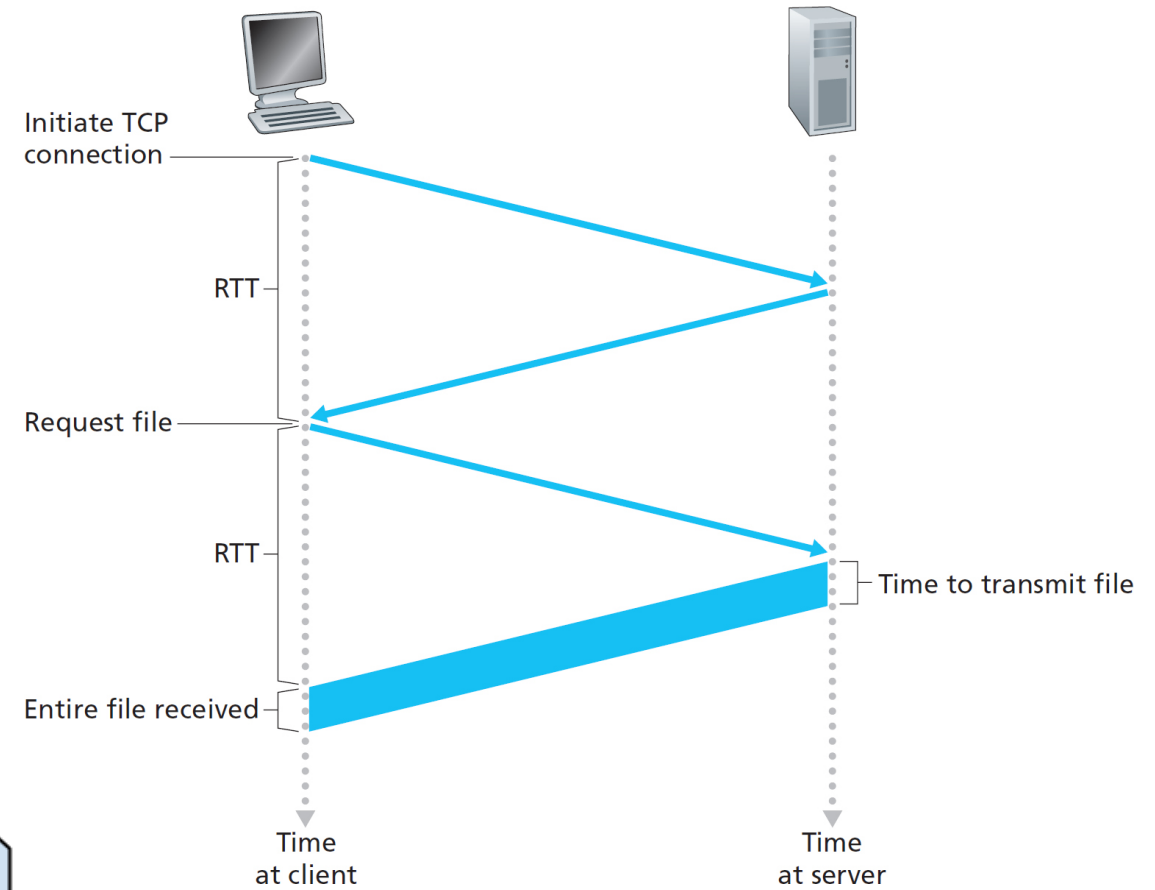
Hvordan bruges TCP bedst?

HTTP Respons Tid

RTT (definition): tid det tager for 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 and første få bytes af HTTP responset
- Fil transmissionstid
- HTTP respons tid = $2RTT + \text{file transmission time}$
- Simple browsere henter linkede objekter sekventielt
- Hvis proceduren gentages sekventielt for hvert objekt bliver det **langsomt** at indlæse en side
- I eksemplet: $4 * \text{HTTP Responstid}$
 - $1 * \text{HTTP Responstid}$ til at hente siden /page.html
 - $+ 3 * \text{HTTP Responstid}$ til at hente linkede objekter
 - $> 8 \text{ RTT}$

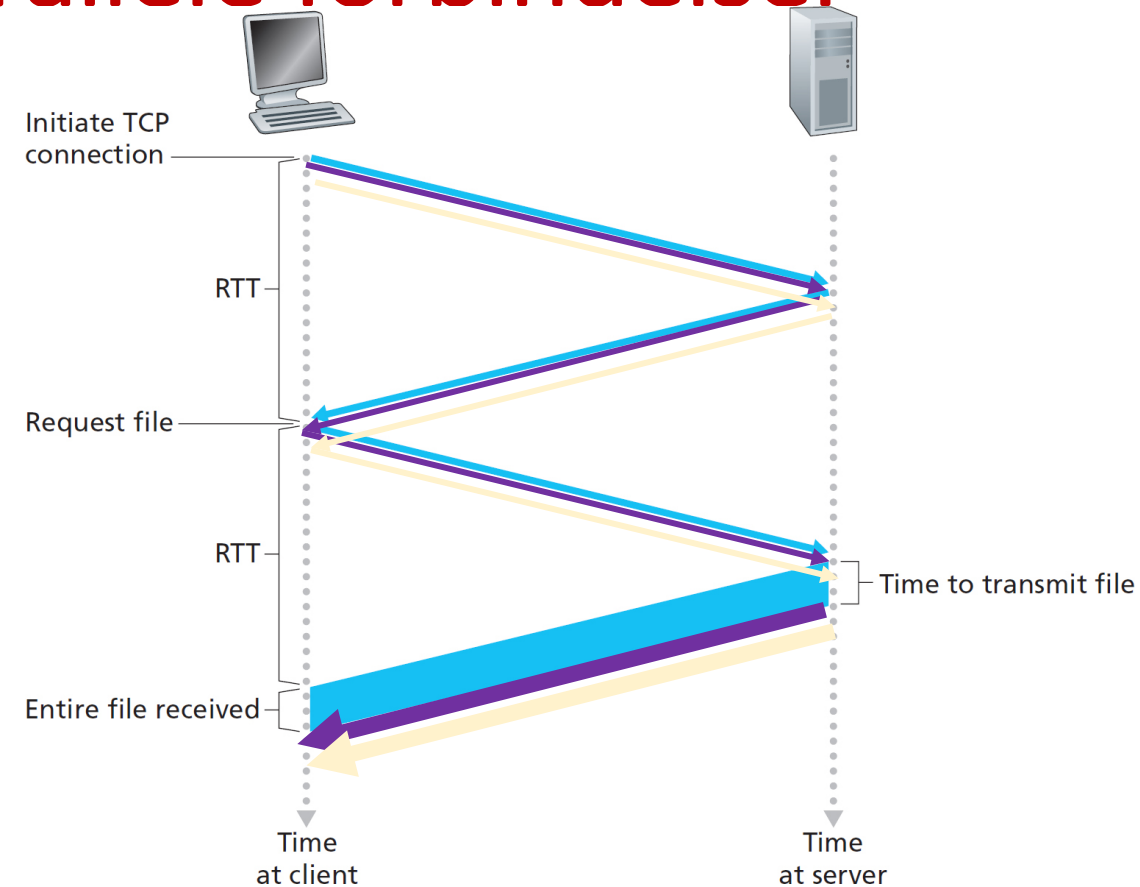
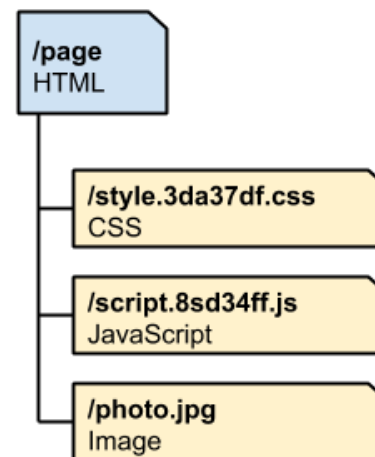


HTTP Responstid med parrallele forbindelser

Klient kan oprette flere “**parallelle**” forbindelser til server

HTTP respons tid:

- 2RTT+ file transmission time+server overhead og evt. Kapacitetsbegrænsninger downlink til klient
- Meget hurtigere for klienten
- Krævende for server, da den skal reservere buffer kapacitet til hver forbindelse (og servicere mange kliner)
- I eksemplet: 2* HTTP Responstid
 - 1*HTTP Responstid til at hente siden (/page.html)
 - + 1*HTTP Responstid til at hente linkede objekter



3 parallelle forbindelser

Løsning: Persistent HTTP

Persistent HTTP (default i HTTP/1.1):

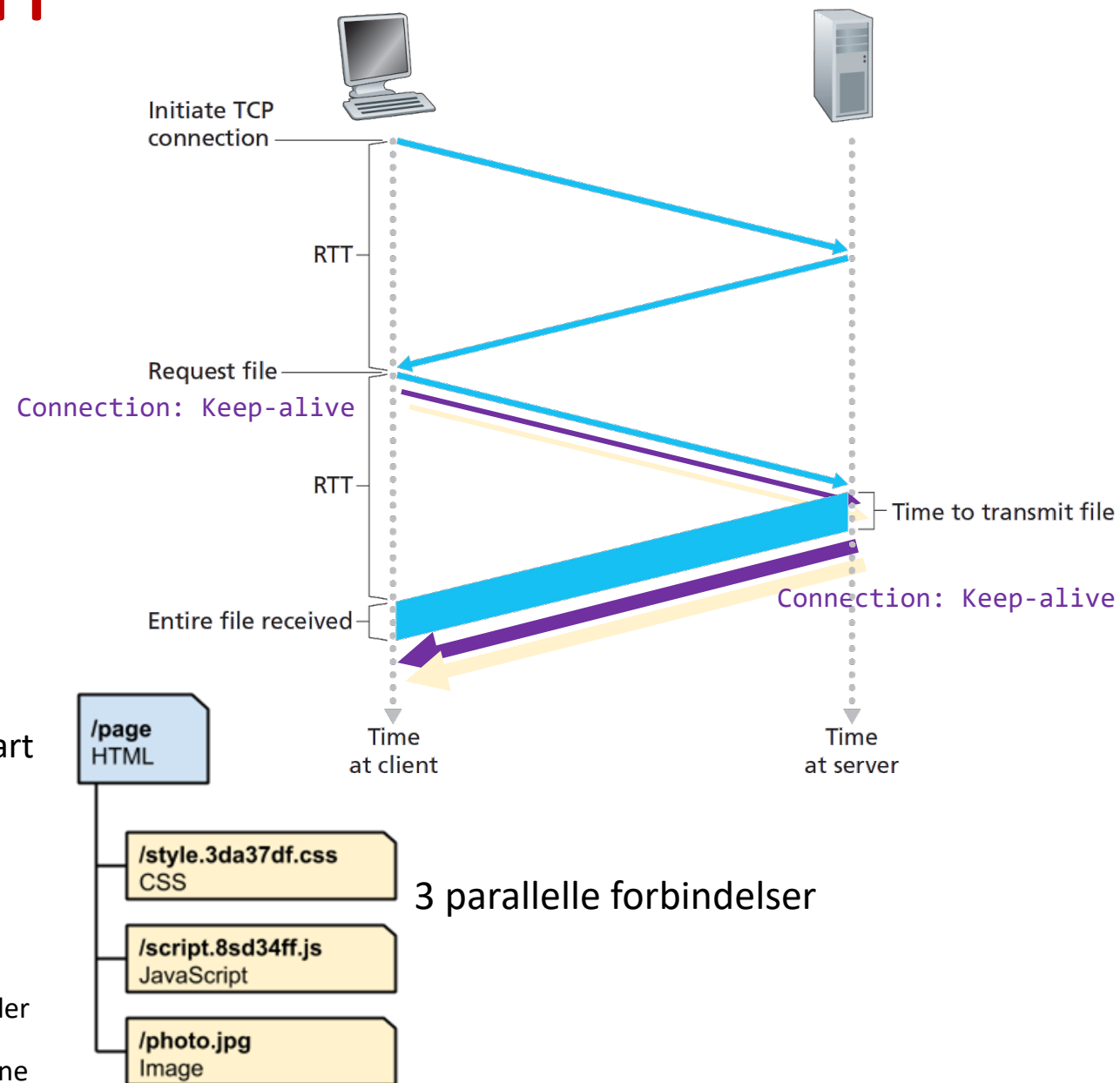
- Server holder forbindelsen åben efter afsendelse af respons
- Efterfølgende HTTP forespørgsler og svar sendes på samme åbne forbindelse:
 - Sparer overhead ved oprettelse af TCP forbindelser

HTTP Pipelining

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

HTTP response tid med pipelining:

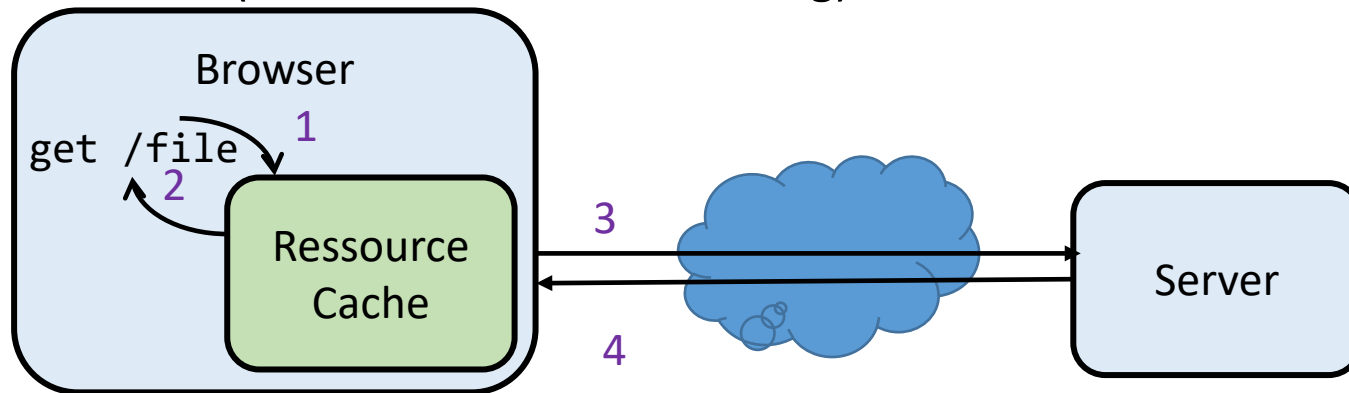
- $2RTT + \text{file transmission time} + \text{server overhead}$ og evt. kapacitetsbegrænsninger downlink til klient
- Meget hurtigere for klienten
- Mindre krævende for server, da den kun skal vedligeholde én forbindelse pr klient, dog skal klienten lukke forbindelsen så snart den er færdig.
- I eksemplet:
 - $1 * \text{HTTP Response tid} (/page.html)$
 - $1 * RTT + 3 * \text{fil transmission tid}$
- HTTP/2
 - Server kan “skubbe” indhold til klient uden forespørgsel; fx den formoder at klienten kan få brug for “photo.jpg”
 - Respons sendes ikke nødvendig i samme rækkefølge som forespørgslerne
 - $\text{Req}_1 \rightarrow \text{req}_2 \rightarrow \text{respons}_2 \rightarrow \text{respons}_1$, hvis respons_2 allerede er klar!



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 nyligt beregnede resultater
 - L1-L3 cache i computer arkitektur, disk/fil-cache,...
 - Web-cache
- I HTTP Web sammenhæng: netværket er "langsomt",
 - Hurtigere svar tid / visning af siden
 - Mindsker trafikken til server (penge, strøm)
 - (Mindsker server belastning)

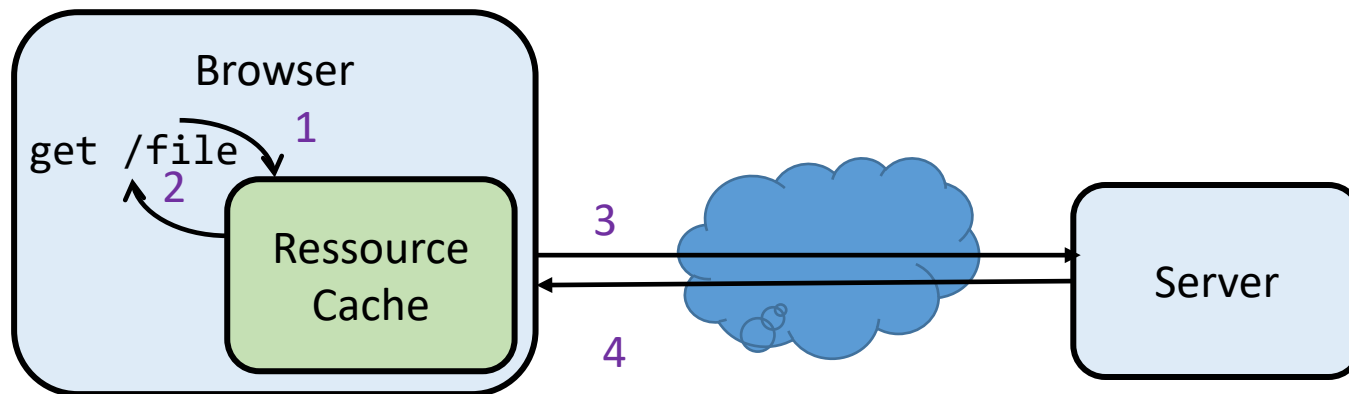


1. Browser kigger først i cache om det efterspurgte objekt ("file") findes der
2. Hvis ja, hent objektet derfra
3. Hvis nej, send forespørgsel til web-server
4. Gem objektet i cache, og lever objektet

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

HTTP Caching

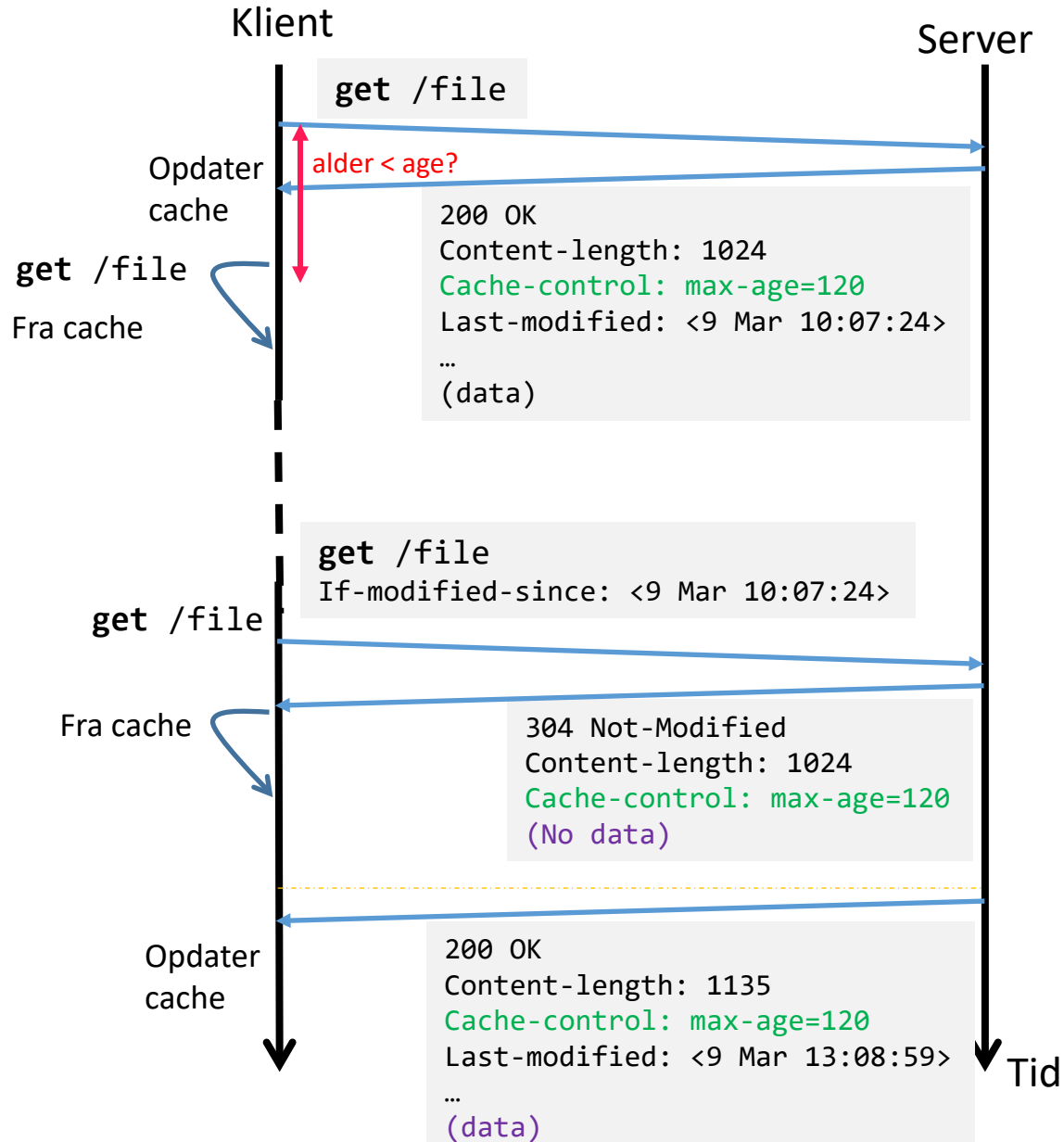
- HTTP headers muliggør at *serveren* kan angive
 - Hvorvidt et objekt kan caches (om det er genbrugeligt)
 - Hvor længe kopien er gyldig
- Kun serveren (og web-app programmør) kender til "foranderligheden" af objektet
 - Sjældent ændres: CSS filer, fleste billeder, ikoner, varemærker, statiske html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



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 lever objektet

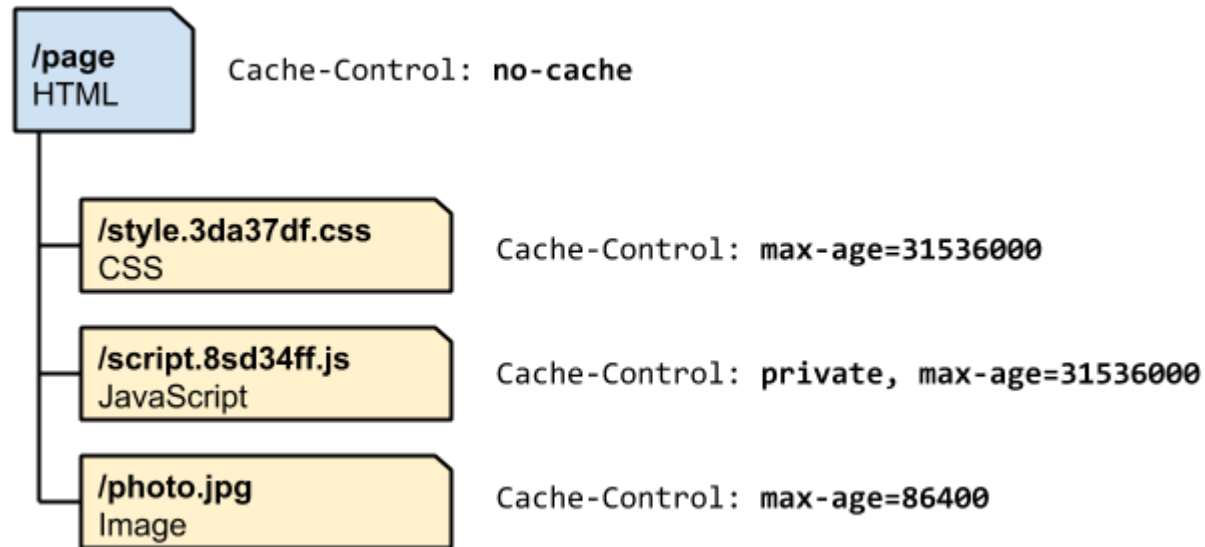
- Hvis objektet ændres, hvordan finder klienten så ud af den ikke bruger forældet info?

HTTP Cache kontrol: Betinget forespørgsel



- **Max-age:** angiver i sekunder hvor lang tid objektet kan betrages som "frisk" og dermed om klienten må bruge det cachede objekt
- **Conditional Get:** checker 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>
 - Objektet er ændret siden <tidsstempel>

HTTP Cache kontrol

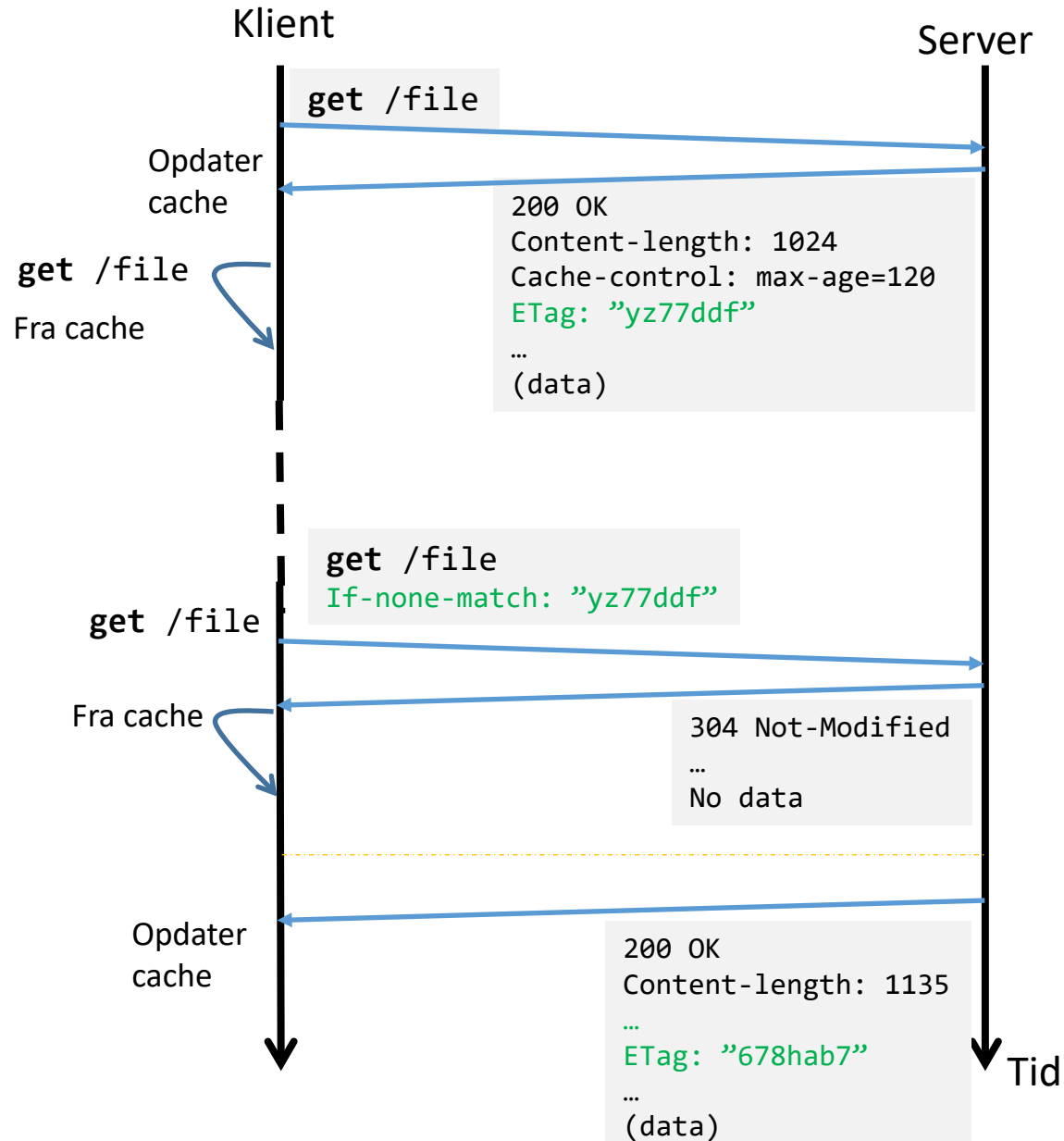


- No-cache: lokalt indhold skal altid valideres hos server først (sparer stadig data, hvis objektet må genbruges)
- No-store: Objektet må ikke gemmes i cache; skal hentes på ny for hver forespørgsel.
- Private vs. public. Privat på kun caches i klientens cache
- Max-age , If-modified-since
- ...
- Hvad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

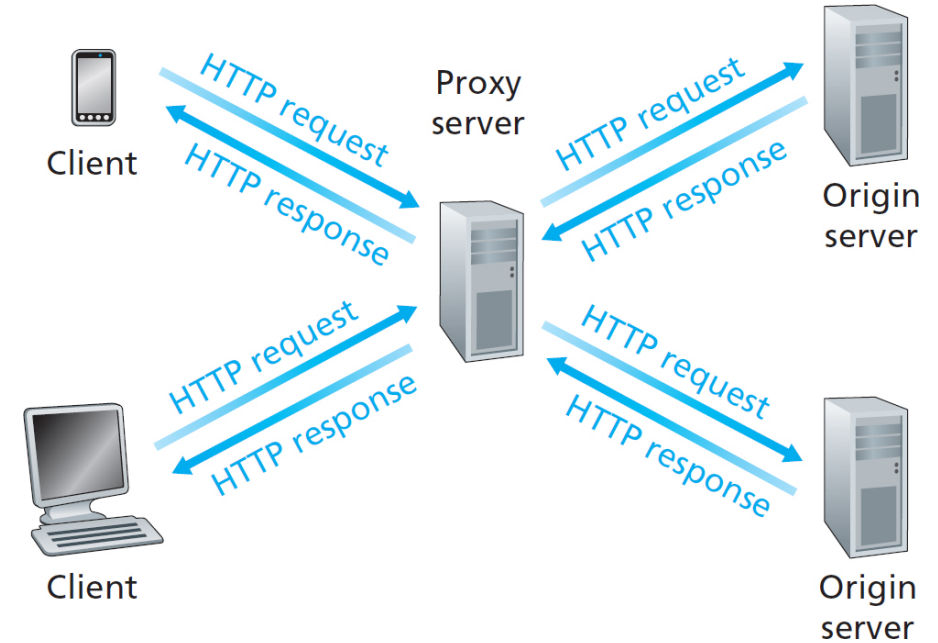
- Web-cache, delt med alle dens klienter
- Load-balancer til en server farm
- Indholds filter
- Anonymizer
- Compression/Encryption

Web-cache (placeret hos ISP)

- Reducerer svar tid til klienter
- 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?

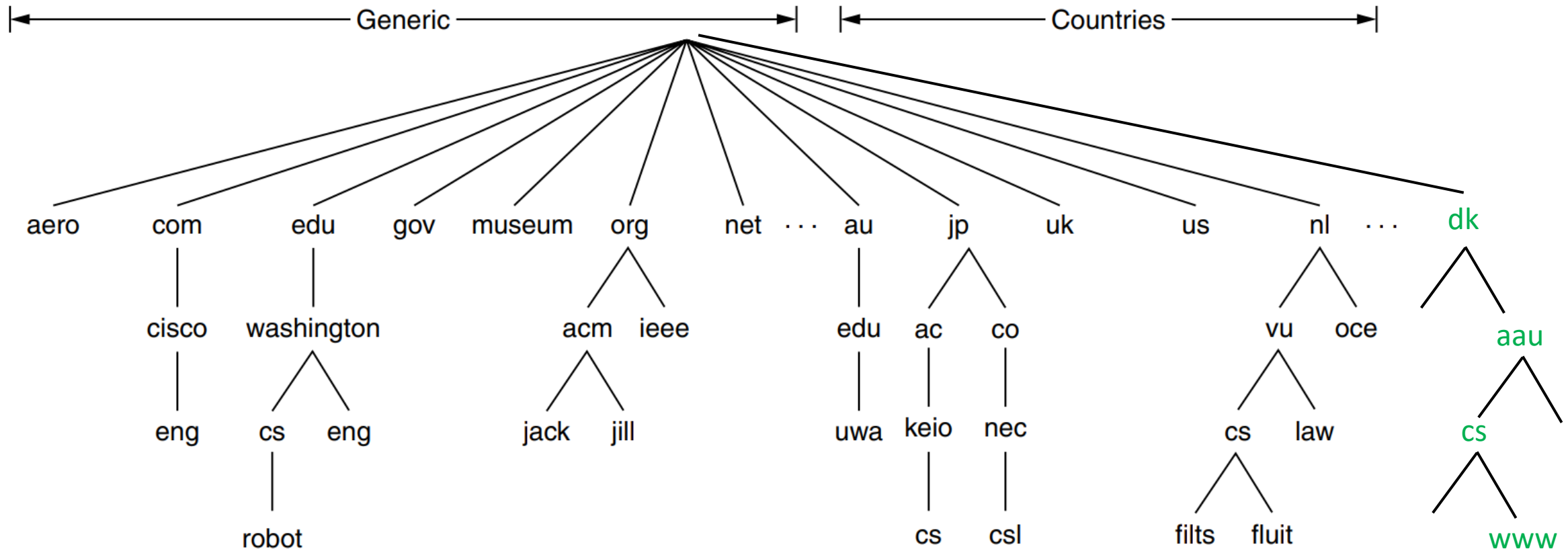
Hvilke (slutbruger) værktøjer findes til DNS?

Domain Name System

- Navngivning
 - Mennesker foretrækker meningsfulde navne
 - Hosts og routere fortrækker unikke numre (fx 32 bit IP adresse)
- Oversætte 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
- **DNS: Distribueret Hierarkisk Database**
 - Fordele belastning på mange servere, organiseret i et hierarki
 - Undgå "single-point-of-failure"
- Applikationslagsprotokol: UDP port 53

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

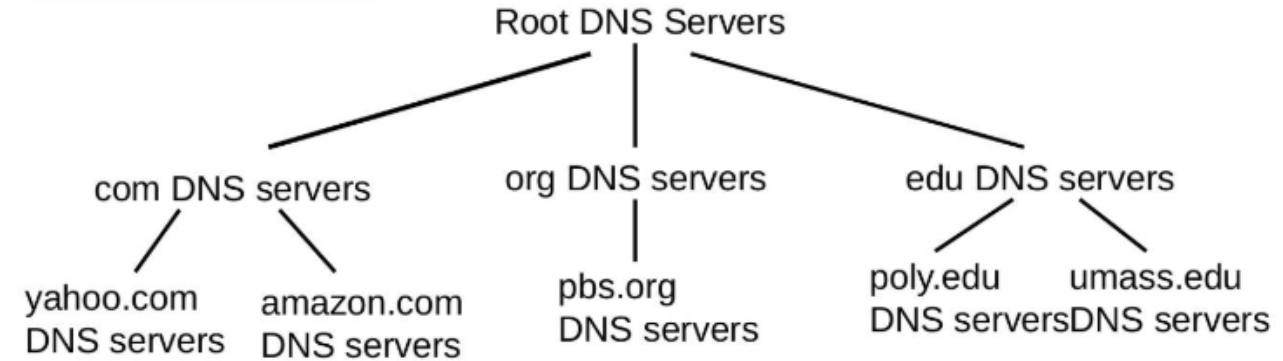
Eksempler

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



Princip skitse:

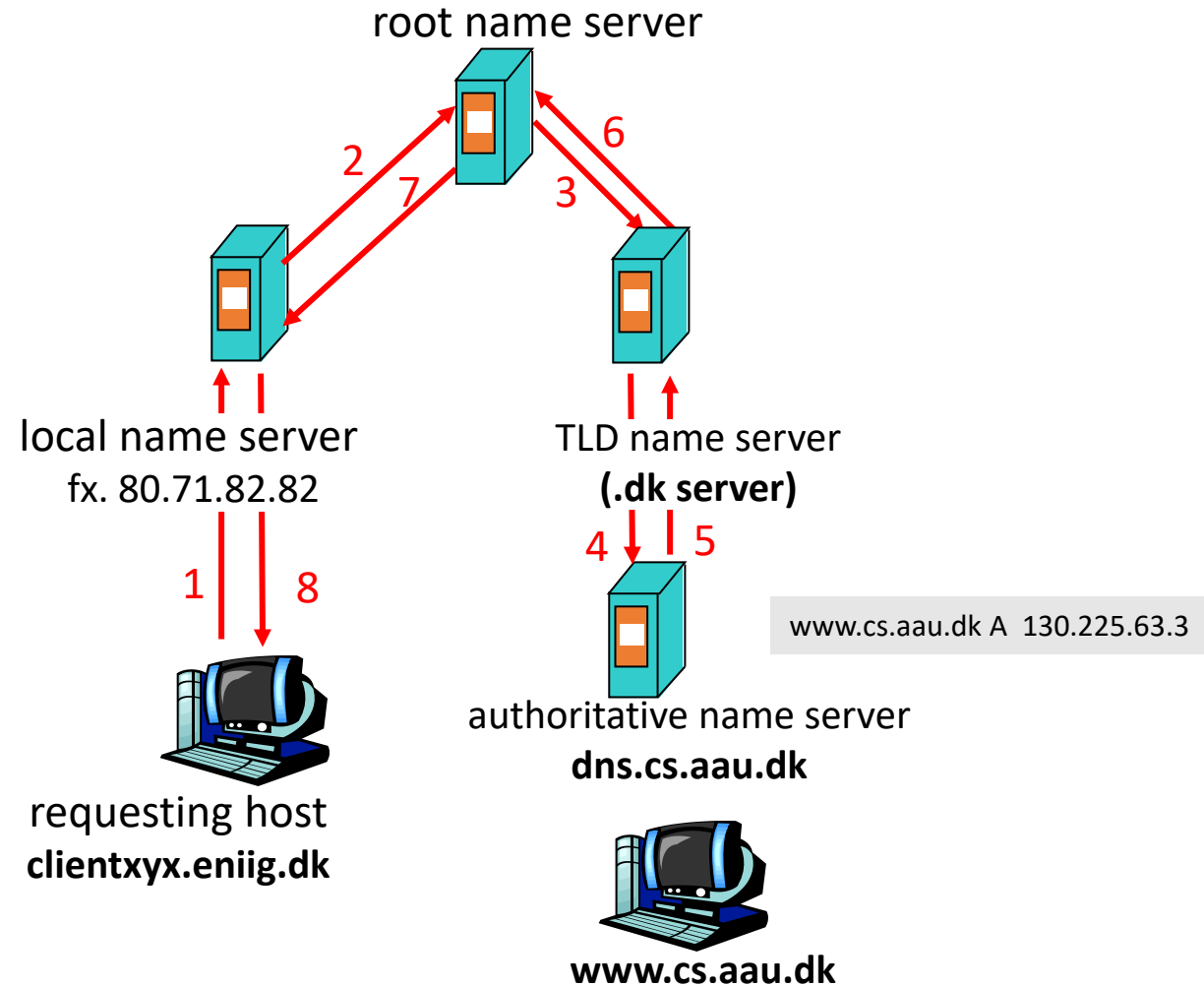
1. Klient kontakter lokal DNS server: IP for www.cs.aau.dk?
2. Den lokal kontakter en Root DNS server
3. RootDNS videreformidler til en den ønskede TLD-server
4. TLD videreformidler til den autoritative server
5. Den autoritative server sender svaret tilbage mod klienten
130.225.63.3

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

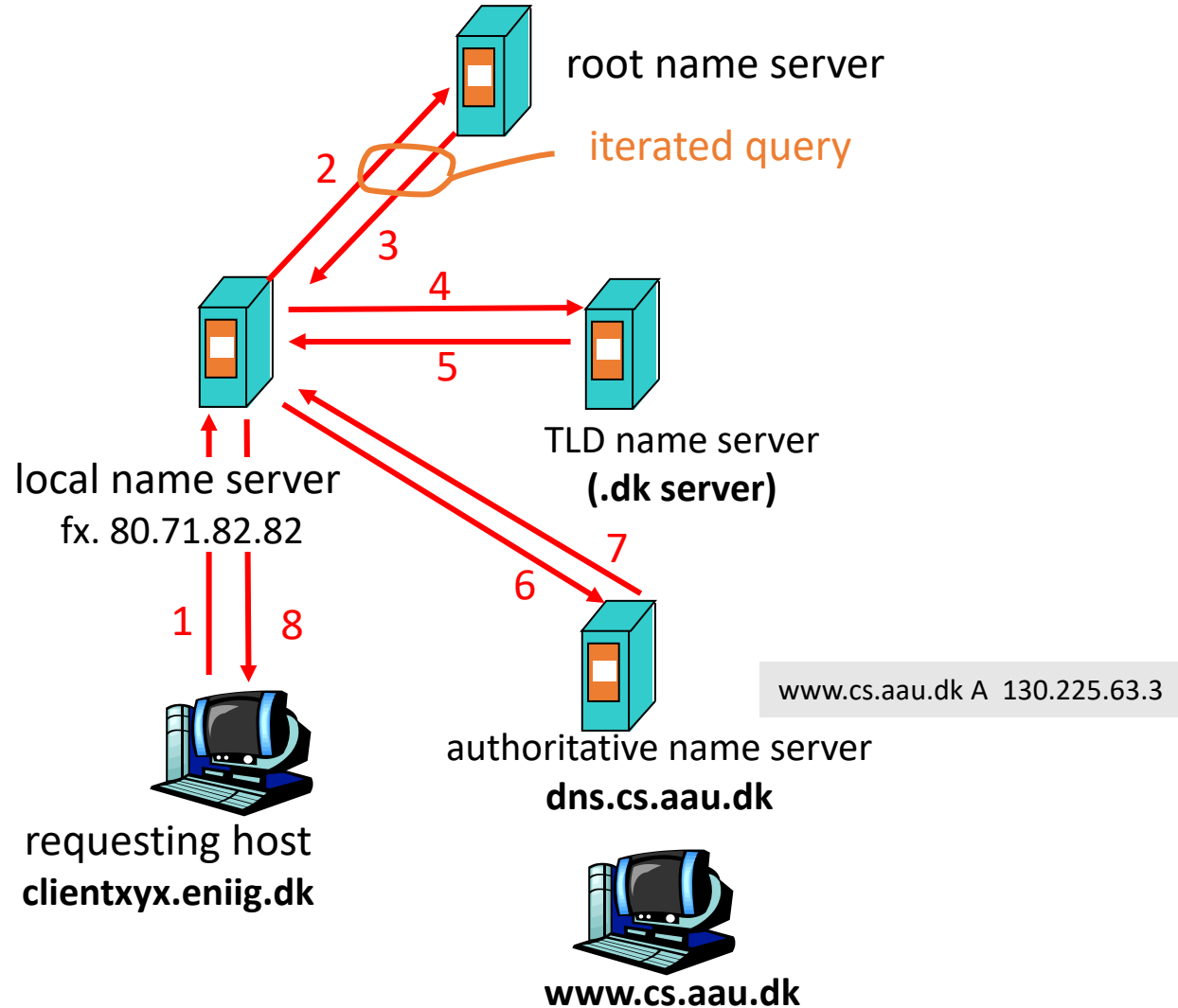
- **Metafor: rekursivt procedurekald**
- **Root name server:**
 - Kender ikke nødvendigvis den autoritative, men henvender sig til en på lavere niveau (fx TLD) som den beder om at løse opgaven



Iterativ DNS Forespørgsel

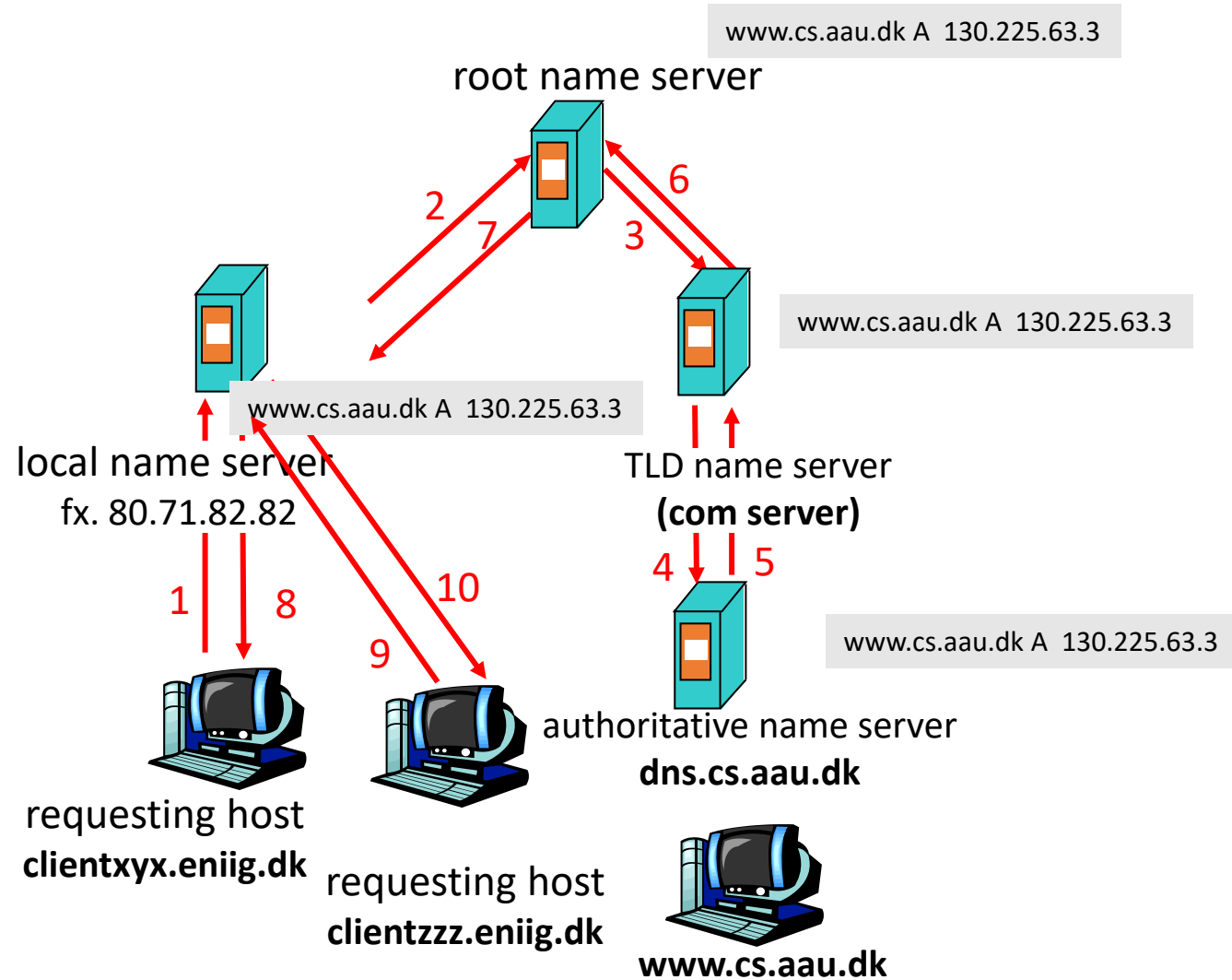
Iterativ DNS forespørgsel (query):

1. Klienten forespørger dens lokale NS
 2. Den lokale forespørger root
 3. Root svare tilbage med navn på TLD
 4. Den lokale forespørger 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
- Metafor: "while" løkke på den lokale
 - Root name server:
 - Kender ikke nødvendigvis den autoritative, men henviser til en på lavere niveau (fx TLD) som kan hjælpe "



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 for 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) ??
 - Kliner 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

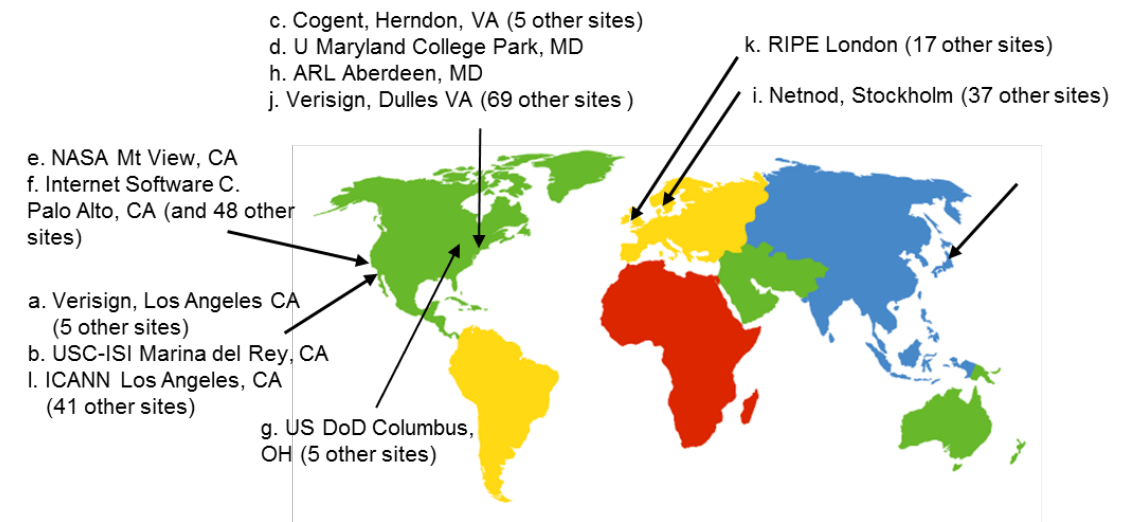


Root Servers

- 13 globale logiske servere
 - a.root-server.net
 - En logisk server dækker over mange (pt >1000) fysiske replikerede servere
- Kontaktes altid når den lokale ikke kan oversætte navnet
- Centraliseret funktion \Rightarrow Spørgsmål:
 - Flaskehals?
 - Enkelt kilde til fejl (single point of failure)?
 - Undgås ved replikering og "caching"

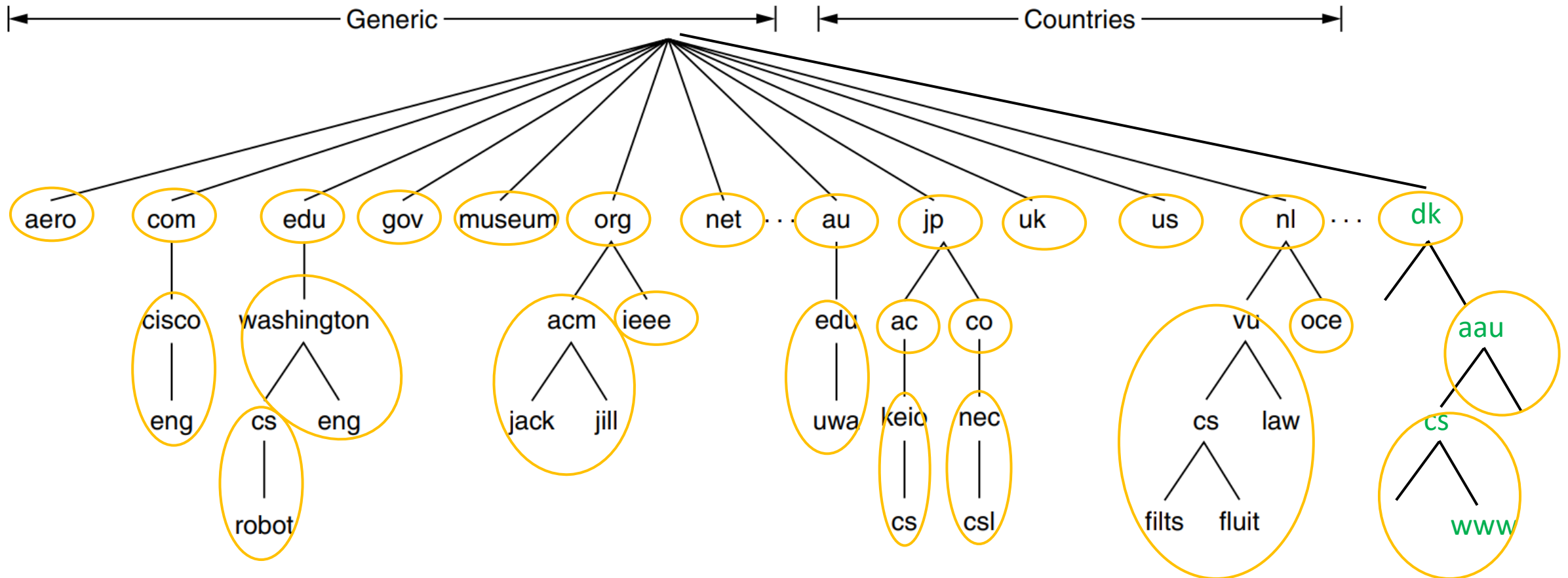
HOSTNAME	IP ADDRESSES	MANAGER
a.root-servers.net	198.41.0.4, 2001:503:ba3e::2:30	VeriSign, Inc.
b.root-servers.net	199.9.14.201, 2001:500:200::b	University of Southern California (ISI)
c.root-servers.net	192.33.4.12, 2001:500:2::c	Cogent Communications
d.root-servers.net	199.7.91.13, 2001:500:2d::d	University 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

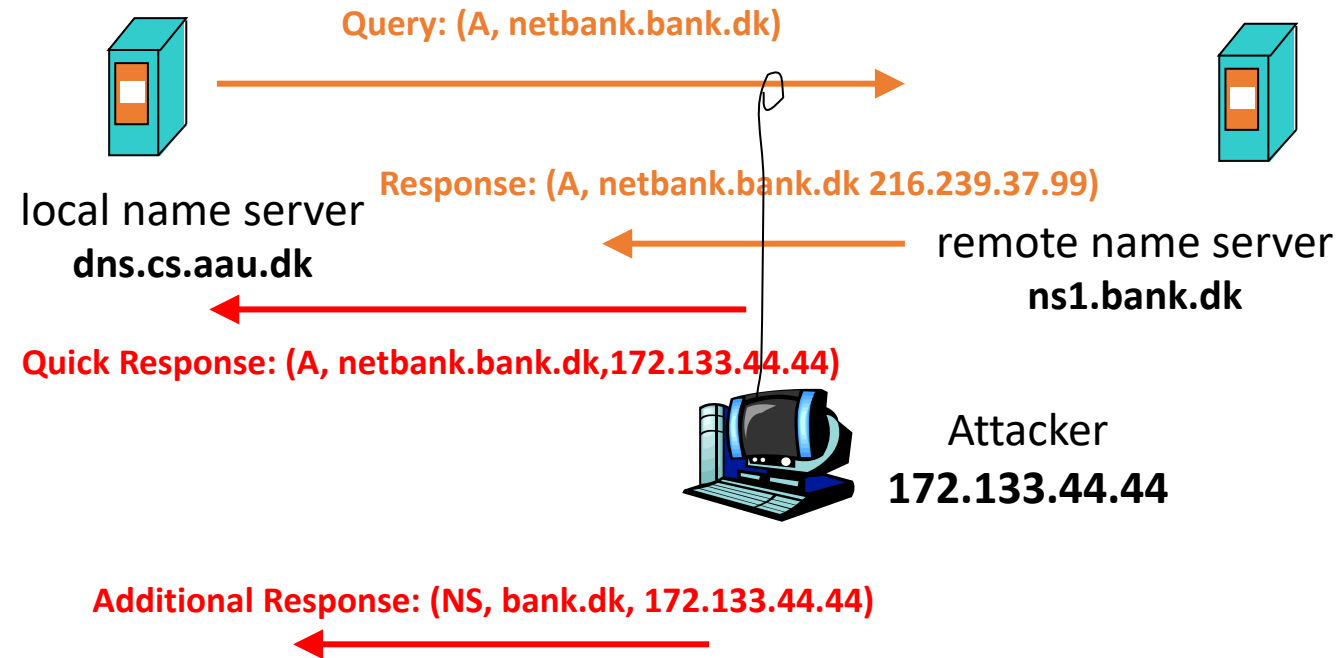


Opdatering af DNS

- Registrering af nye domæne
 - nsmd.dk (Nørresundby Micro Devices)
 - Køb og registrer domænet hos registrator (<https://www.dk-hostmaster.dk/> , minmums 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
 - Management interface hos primær authoritative server
 - Dynamic-DNS: programmeringsmæssig / automatisk opdatering (fx. Ved opretelse af ny virtuel maskine med en server)

DNS Sikkerhed

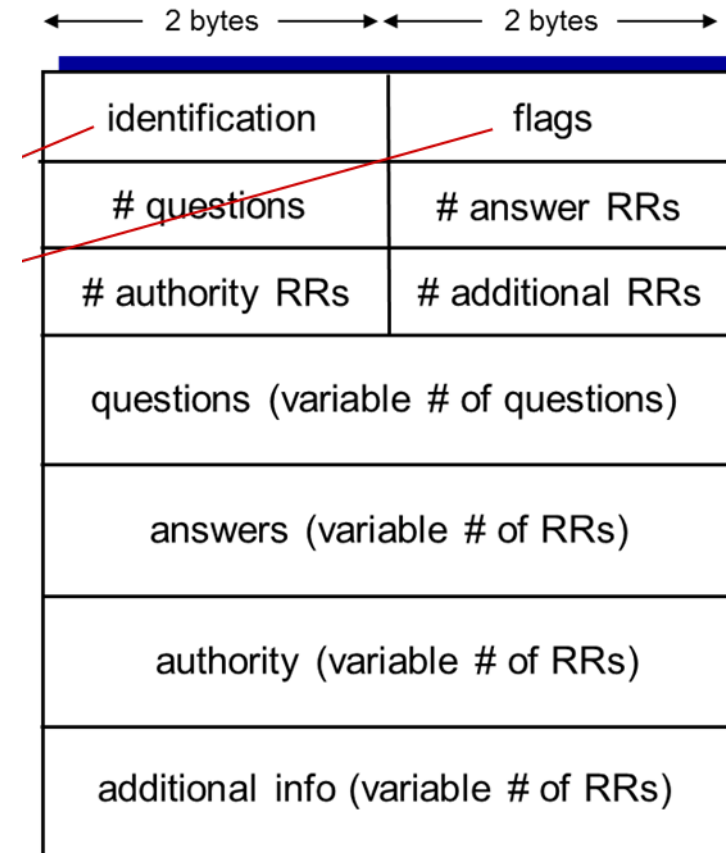
- Spoofing og Cache-forgiftning
 - Forfalske et svar og omdirigere klient til ønsket site
 - Lægge forfalskede oplysninger i en DNS cache
 - Teknisk svært (hastighed)
- Denial-of-service
 - Kan vi nedlægge en name-server ved at bombardere med forespørgsler
 - Nej, caching, og mange replikerede root server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



DNS Værktøjer

DNS besked format:

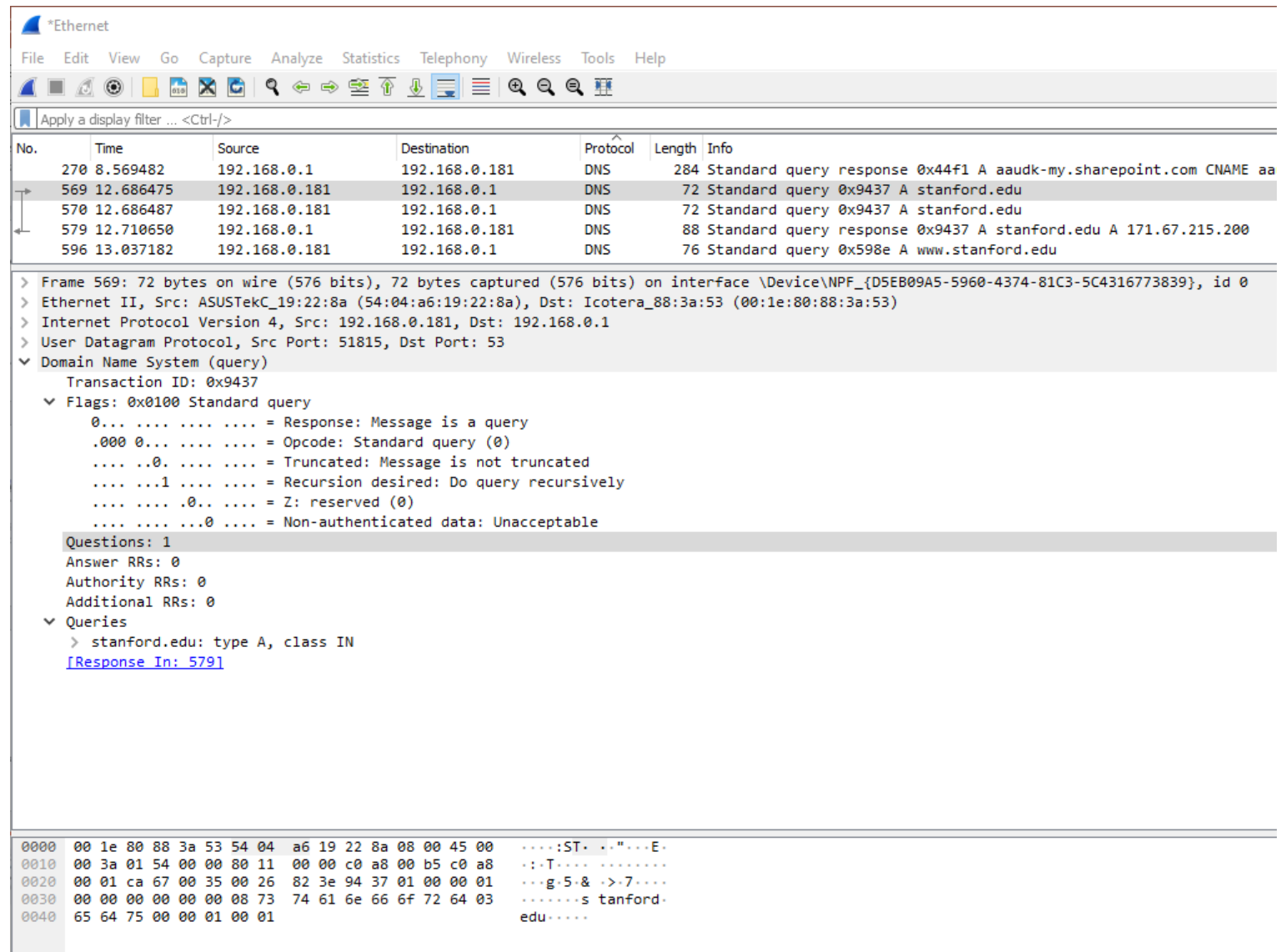
- Samme format for spørgsmål og svar
 - Et flag (bit) bestemmer hvilket
- Identifikation: 16 bit tal til at matche forespørgsel med svar
- Flag, fx:
 - Forespørgsel eller svar
 - Rekursivt foretrukket
 - Rekursivt tilgængeligt
 - Svaret er autoritativt
- Antal forespørgsler / svar i meddelelsen
- Navn,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



DNS besked format: eksempel forespørgsel

Liste med opsnappede pakker,
sorteret efter protokoltype

Wireshark dekoderet
info fra UDP payload



The image shows a Wireshark network packet capture. The top pane displays a list of captured packets, sorted by protocol type. Packet 569 is selected, showing it is a DNS Standard query from 192.168.0.181 to 192.168.0.1. The middle pane shows the packet details, including the Ethernet II header, Internet Protocol Version 4 header, User Datagram Protocol header, and the Domain Name System (query) section. The DNS query is for 'stanford.edu' type A, class IN. The bottom pane shows the raw packet data in hexadecimal and ASCII.

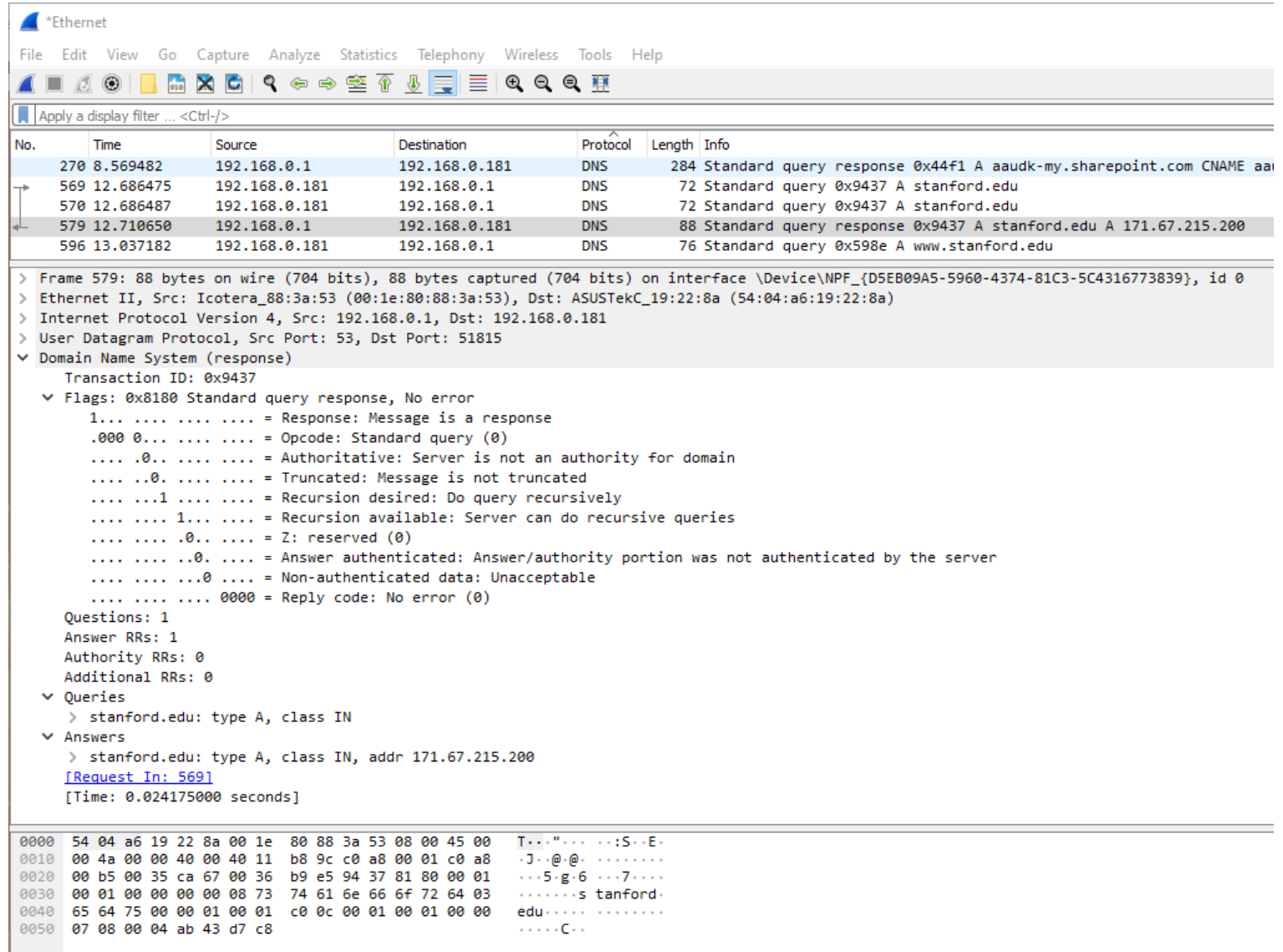
No.	Time	Source	Destination	Protocol	Length	Info
270	8.569482	192.168.0.1	192.168.0.181	DNS	284	Standard query response 0x44f1 A aaudk-my.sharepoint.com CNAME aa
569	12.686475	192.168.0.181	192.168.0.1	DNS	72	Standard query 0x9437 A stanford.edu
570	12.686487	192.168.0.181	192.168.0.1	DNS	72	Standard query 0x9437 A stanford.edu
579	12.710650	192.168.0.1	192.168.0.181	DNS	88	Standard query response 0x9437 A stanford.edu A 171.67.215.200
596	13.037182	192.168.0.181	192.168.0.1	DNS	76	Standard query 0x598e A www.stanford.edu

> Frame 569: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface \Device\NPF_{D5EB09A5-5960-4374-81C3-5C4316773839}, id 0
> Ethernet II, Src: ASUSTekC_19:22:8a (54:04:a6:19:22:8a), Dst: Icotera_88:3a:53 (00:1e:80:88:3a:53)
> Internet Protocol Version 4, Src: 192.168.0.181, Dst: 192.168.0.1
> User Datagram Protocol, Src Port: 51815, Dst Port: 53
v Domain Name System (query)
Transaction ID: 0x9437
v Flags: 0x0100 Standard query
0... .. = Response: Message is a query
.000 0... .. = Opcode: Standard query (0)
.... .. = Truncated: Message is not truncated
.... ..1 = Recursion desired: Do query recursively
.... ..0.. = Z: reserved (0)
.... ..0 = Non-authenticated data: Unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
v Queries
> stanford.edu: type A, class IN
[\[Response In: 579\]](#)

```
0000 00 1e 80 88 3a 53 54 04 a6 19 22 8a 08 00 45 00 .....ST. .".E.
0010 00 3a 01 54 00 00 00 11 00 00 c0 a8 00 b5 c0 a8 .T.....
0020 00 01 ca 67 00 35 00 26 82 3e 94 37 01 00 00 01 .g.5.& .>.7...
0030 00 00 00 00 00 08 73 74 61 6e 66 6f 72 64 03 .....s tanford.
0040 65 64 75 00 00 01 00 01 edu.....
```

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

DNS besked format: eksempel svar



Wireshark packet capture showing a DNS response from 192.168.0.181 to 192.168.0.1. The response contains a CNAME record for aadk-my.sharepoint.com pointing to aadk-my.sharepoint.com and an A record for stanford.edu pointing to 171.67.215.200.

No.	Time	Source	Destination	Protocol	Length	Info
270	8.569482	192.168.0.1	192.168.0.181	DNS	284	Standard query response 0x44f1 A aadk-my.sharepoint.com CNAME aadk-my.sharepoint.com
569	12.686475	192.168.0.181	192.168.0.1	DNS	72	Standard query 0x9437 A stanford.edu
570	12.686487	192.168.0.181	192.168.0.1	DNS	72	Standard query 0x9437 A stanford.edu
579	12.710650	192.168.0.1	192.168.0.181	DNS	88	Standard query response 0x9437 A stanford.edu A 171.67.215.200
596	13.037182	192.168.0.181	192.168.0.1	DNS	76	Standard query 0x598e A www.stanford.edu

> Frame 579: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface \Device\NPF_{D5EB09A5-5960-4374-81C3-5C4316773839}, id 0
> Ethernet II, Src: Icotera_88:3a:53 (00:1e:80:88:3a:53), Dst: ASUSTekC_19:22:8a (54:04:a6:19:22:8a)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.181
> User Datagram Protocol, Src Port: 53, Dst Port: 51815
▼ Domain Name System (response)
Transaction ID: 0x9437
▼ Flags: 0x8180 Standard query response, No error
1... .. = Response: Message is a response
.000 0... .. = Opcode: Standard query (0)
.... .0.. .. = Authoritative: Server is not an authority for domain
.... ..0. = Truncated: Message is not truncated
.... ...1 = Recursion desired: Do query recursively
.... 1... .. = Recursion available: Server can do recursive queries
....0.. = Z: reserved (0)
....0. = Answer authenticated: Answer/authority portion was not authenticated by the server
....0 = Non-authenticated data: Unacceptable
.... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
▼ Queries
> stanford.edu: type A, class IN
▼ Answers
> stanford.edu: type A, class IN, addr 171.67.215.200
[\[Request In: 569\]](#)
[Time: 0.024175000 seconds]

```
0000 54 04 a6 19 22 8a 00 1e 80 88 3a 53 08 00 45 00 T...". . .:S..E.
0010 00 4a 00 00 40 00 40 11 b8 9c c0 a8 00 01 c0 a8 .J..@.@. ....
0020 00 b5 00 35 ca 67 00 36 b9 e5 94 37 81 80 00 01 ...5.g.6 ...7...
0030 00 01 00 00 00 00 08 73 74 61 6e 66 6f 72 64 03 .....s tanford.
0040 65 64 75 00 00 01 00 01 c0 0c 00 01 00 01 00 00 edu.....
0050 07 08 00 04 ab 43 d7 c8 .....C..
```

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

```
C:\> 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
```

```
C:\> 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

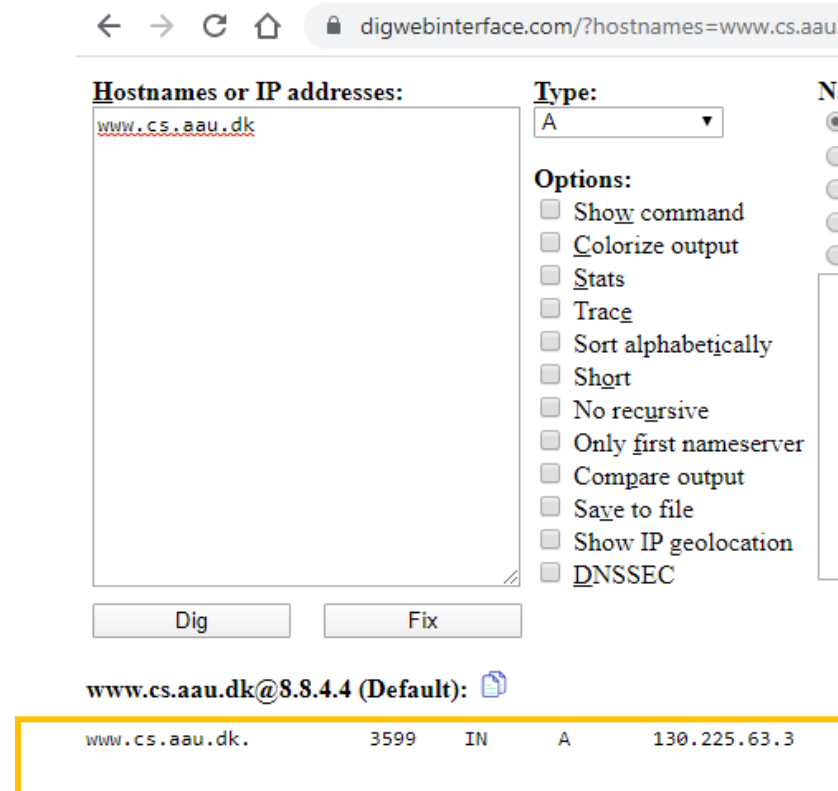
- Dig (domain information groper)

- Kommando linie

- Web-interface, fx

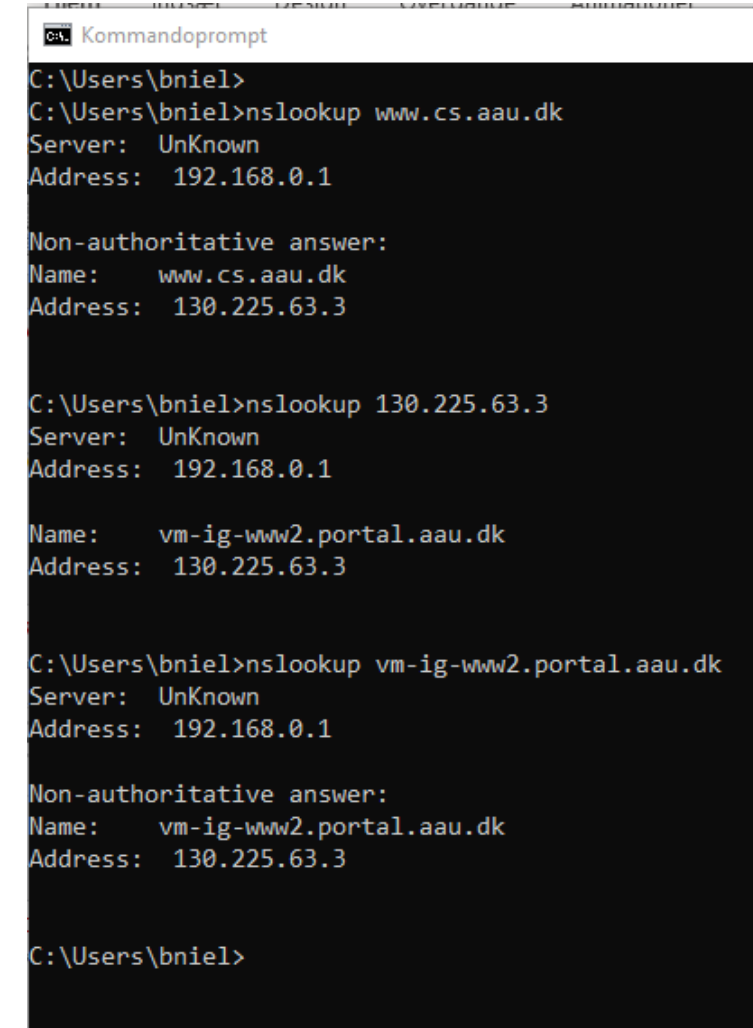
- <https://www.digwebinterface.com/>

- nslookup



The screenshot shows the digwebinterface.com website. The browser address bar displays "digwebinterface.com/?hostnames=www.cs.aau.dk". The main form has a text input field containing "www.cs.aau.dk" under the label "Hostnames or IP addresses:". To the right, the "Type:" dropdown is set to "A". Below this, there is a list of "Options:" with 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 of the form are "Dig" and "Fix" buttons. Below the form, the text "www.cs.aau.dk@8.8.4.4 (Default):" is followed by a table of DNS records. The first record is highlighted with a yellow box:

www.cs.aau.dk@8.8.4.4 (Default):				
www.cs.aau.dk.	3599	IN	A	130.225.63.3



```

C:\Users\bniel>
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-ww2.portal.aau.dk
Address: 130.225.63.3

C:\Users\bniel>nslookup vm-ig-ww2.portal.aau.dk
Server:      UnKnown
Address:     192.168.0.1

Non-authoritative answer:
Name:   vm-ig-ww2.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?

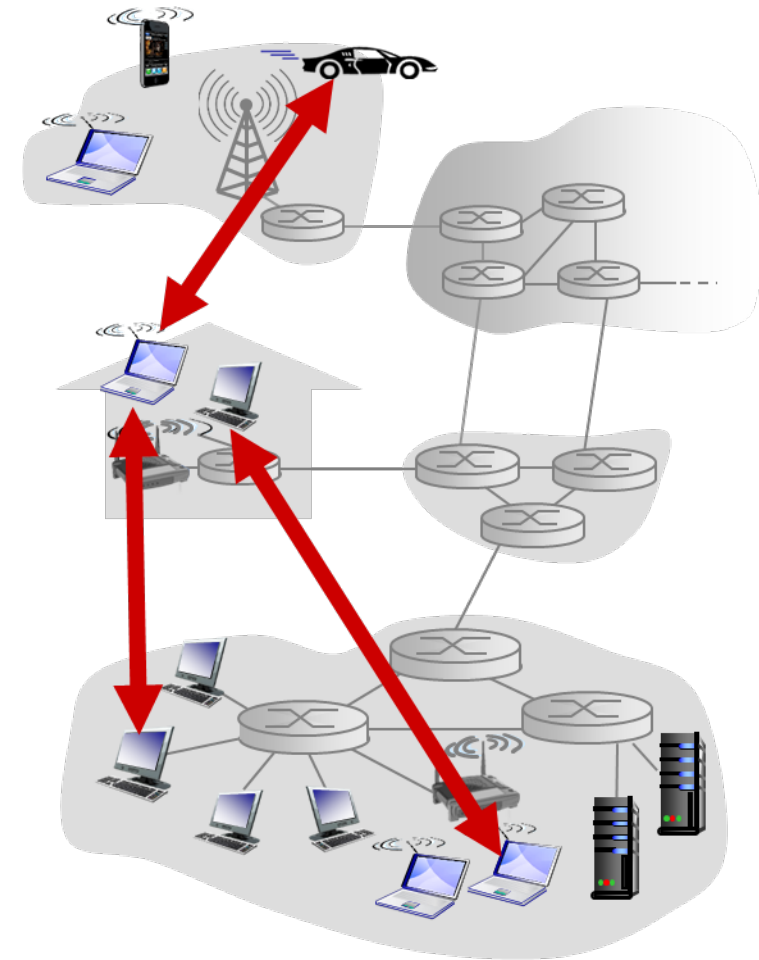
I modsætning til client-server

Hvilke fordele og ulemper er der ved den?

Hvordan virker BitTorrent i princippet?

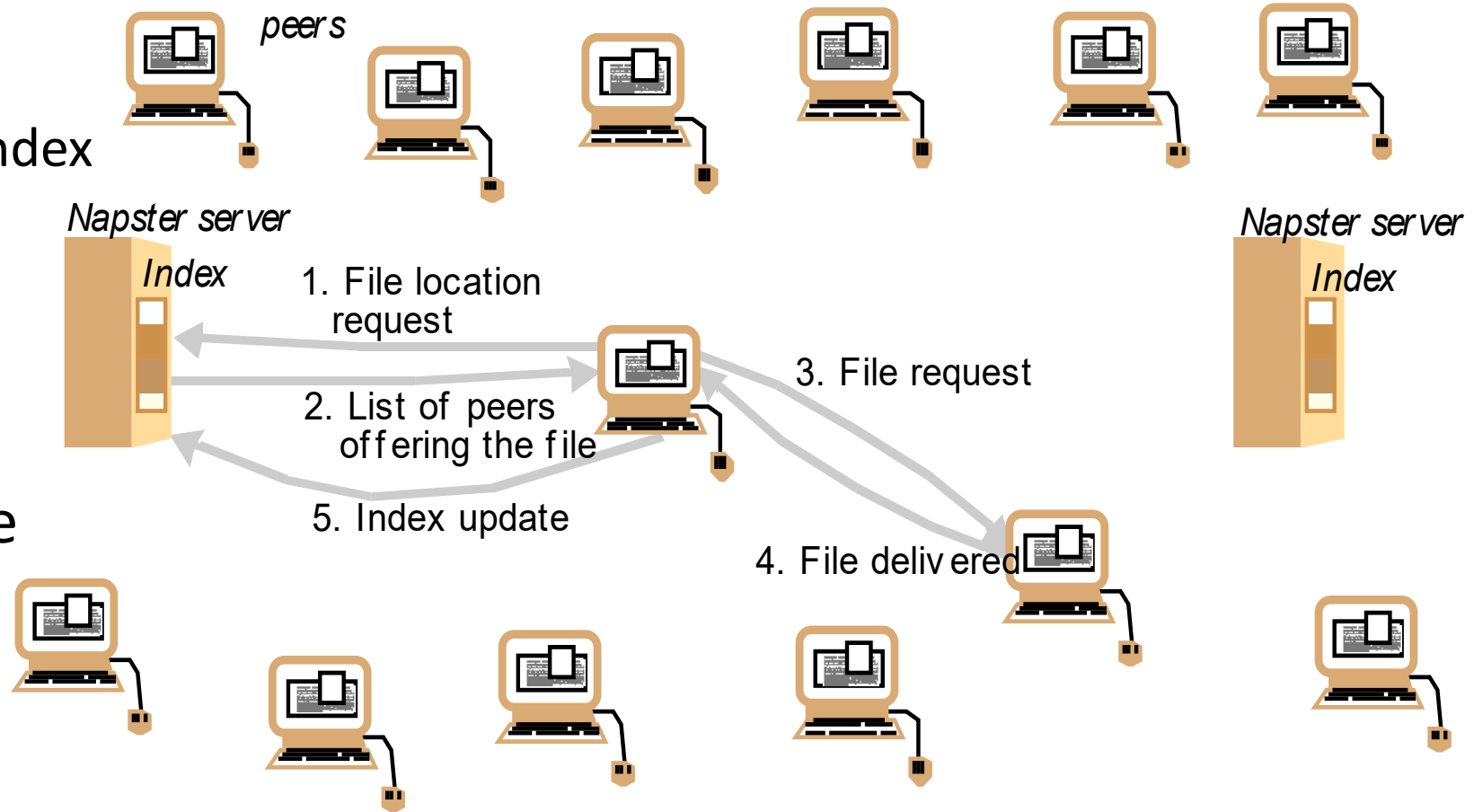
P2P Arkitektur

- Hosts (end-system) forbindes direkte sammen
 - Et overlejret netværk på applikationsniveau
 - Slut-brugernes maskiner, med ingen eller få faste servere
 - Slut-brugerne bidrager med resourcer (beregningskraft, lagringskapacitet, netværkskapacitet) efter noget-for-noget princip ⇒ **skalering!**
 - Ingen enkelt ejer ⇒ **Svært at lukke ned**
 - **Oppetid:** ingen centraliseret server
- Eksempler
 - Fil distribution (BitTorrent)
 - Streaming (KanKan)
 - VoIP (Skype),
 - TOR, ...
- Varmt forskningsområde i 0'erne
 - Distribuerede Hash Tabeller til hurtig søgning efter data
 - Anvendelser som
 - Fil-delning og distribution
 - Distribueret data-lagring, multi-cast, spil,...



Pionererne

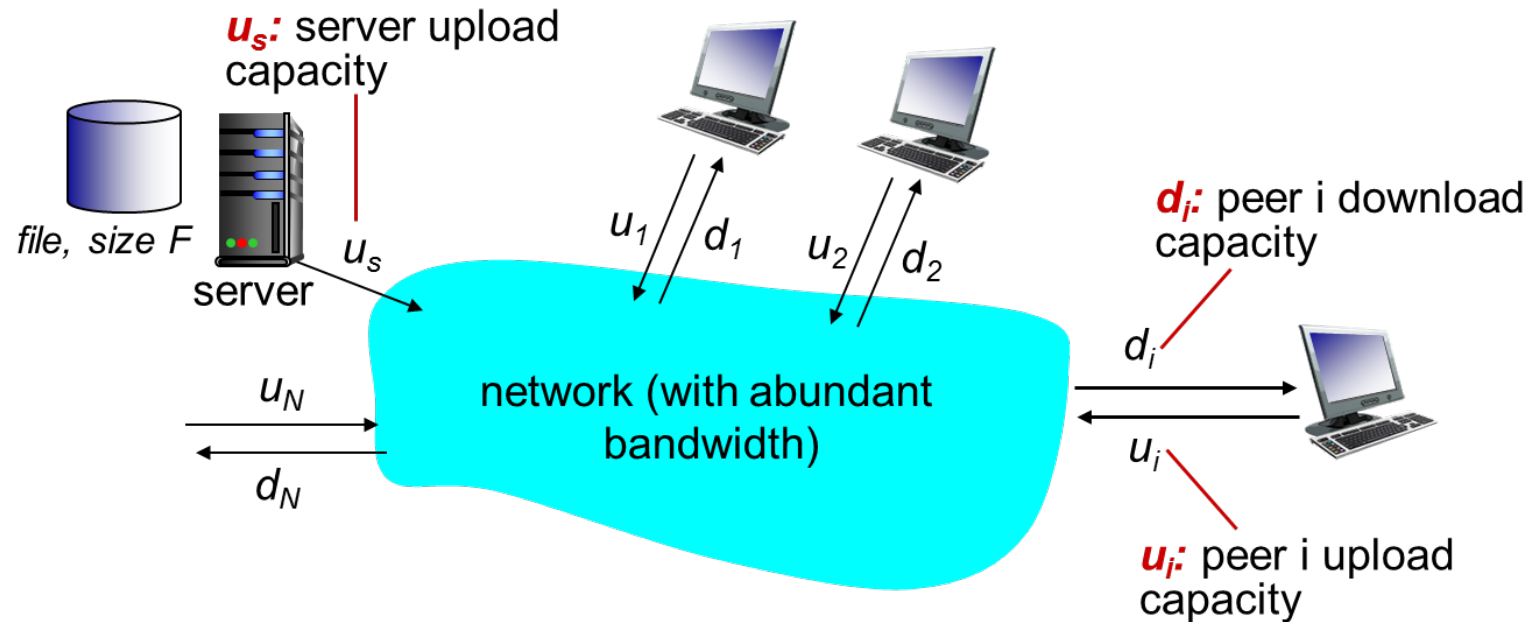
- Napster: 1999-2001
 - Deling af (.mp3) filer
 - Centraliseret, replikeret index
 - Lukket ned efter retskendelse,
 - 24 millioner brugere
- P2P blev synonym med ulovlig deling af materiale beskyttet af ophavsret
- Andre tidlige
 - Gnutelle, Freenet



File Distribution: Client-Server vs P2P

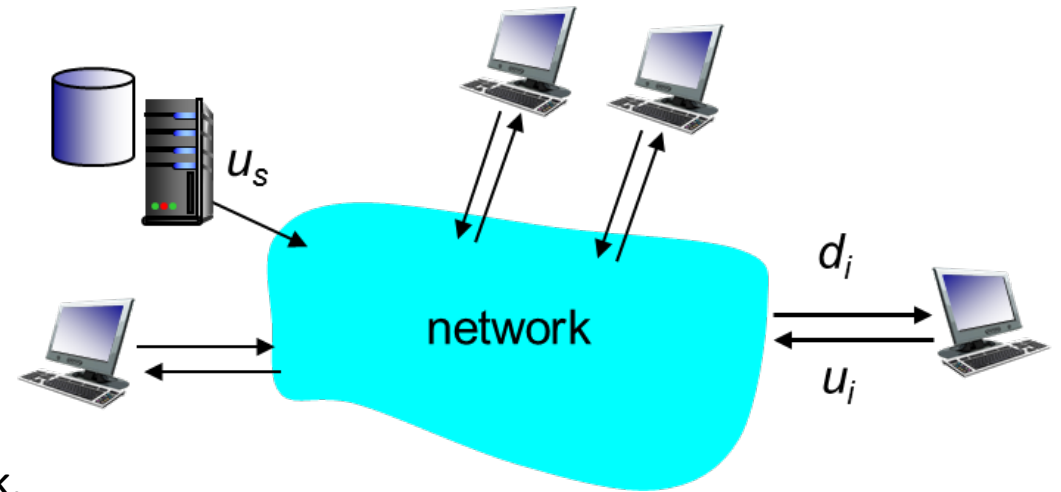
Spørgsmål: Hvor lang tid tager det at fordele en fil (af størrelsen F bits) fra én server til N peers?

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



Fil Fordelingsstid: Client-Server

- **Server transmission:** skal sende (uploade) 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 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



Fil Fordelingsstid : P2P

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

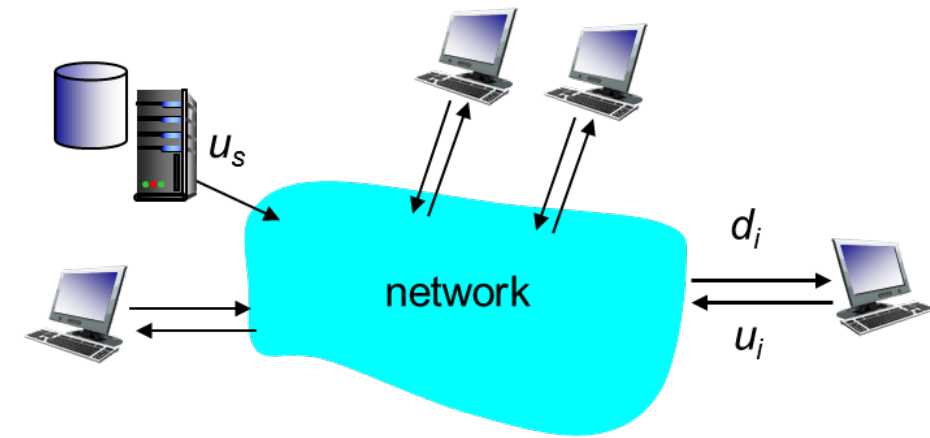
- Tid at sende en kopi: $\frac{F}{u_s}$

- Hver klient skal downloade en kopi

- Tid til at downloade en kopi: $\frac{F}{d_{min}}$

- 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?

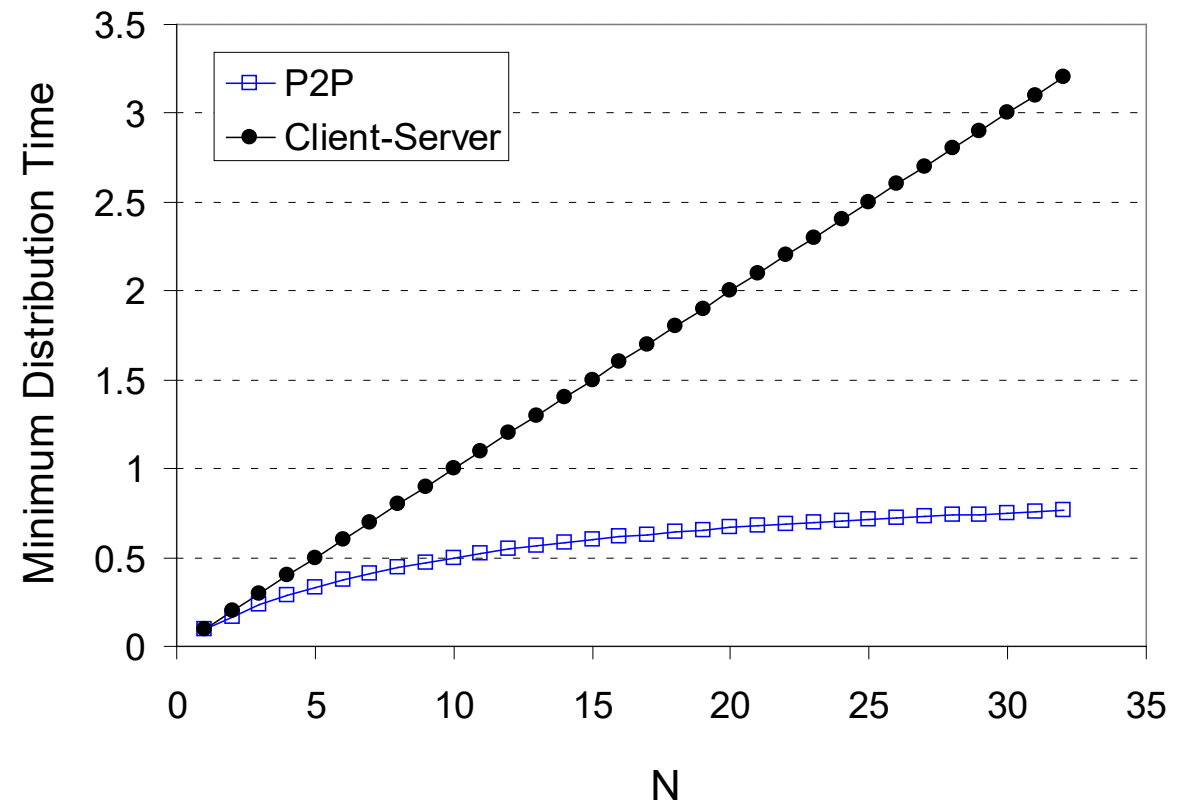
- Alle klienter har samme upload rate: u
- En fil som tager en time at uploade:
 $F/u = 1$ time
- Server har 10 gange en klients upload kapacitet
 $u_s = 10u$
- Langsomste klient har stor nok download rate
stor nok til ikke at bestemme, så
 $d_{min} \geq u_s$

- *Afgørende termer*

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

vs.

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

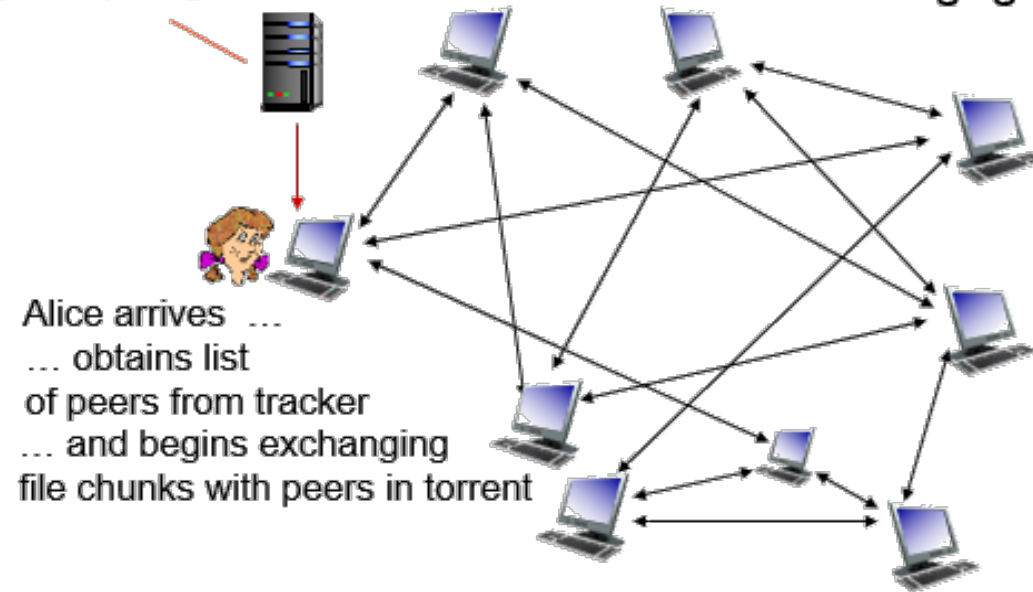
Netværkstekniske problemstilling da fleste klienter ikke har en offentlig fast IP (“NAT”)

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, uploades tidligere downloadede chunks
 - Vil gradvis (forhåbentligt) akkumulere alle chunks
- Mange lærerige aspekter, som vi ikke når

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



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
 - Tidsforskel mellem forskellige HTTP forbindelses-strategier
 - HTTP cache (conditional get)
 - P2P download tider
- Praktiske: Kan anvende netværkssværktøjerne
 - Wireshark: opsnappe og undersøge de udvekslede DNS meddelelser

SLUT