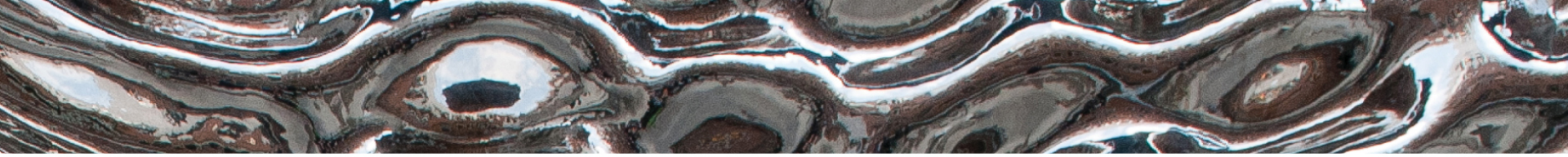


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN1020 - Introduksjon til datateknologi

Forelesning – 2.11.2018

Tjenester i Internett

Håkon Kvale Stensland & Andreas Petlund

simula

tfs

Plan for ”nettverksdelen” av IN1020

- *21. september – Kryptering til hverdags og fest*
- 19. oktober – *Historien til datanettverk*
  Lagdeling i Internettarkitekturen
- 24. oktober – Lagene i Internett spiller sammen
- 26. oktober – Lagene i Internett spiller sammen (forts.)
  - *(Presentasjon av Oblig 3)*
  - *+ Tjenester i Internett*
- 31. oktober – Pustepause: Spørsmål & repetisjon om datanettverk
- **2. november – Tjenester i Internett**

Forbindelsesstrategier (push og pull)

Pull

– Klienten ber tjeneren om en tjeneste (f.eks et dokument)
– Tradisjonelt den vanligste metoden

Push

– Tjeneren ”dytter” en tjeneste (f.eks en beskjed) til klienten.
– Krever at det er en forbindelse fra før, eller at klienten lytter.

Publish-subscribe

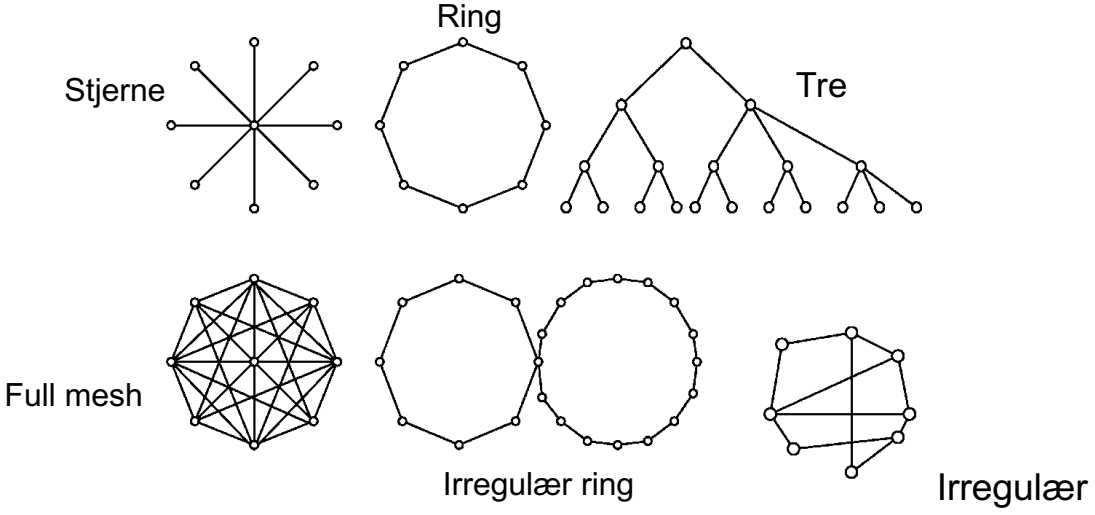
– Variant av ”Push” der tjeneren dytter ut beskjeder til en gruppe av abonnenter (subscribers)

Repetisjon: topologi i LAN

Punkt til punkt:

- Flere forskjellige kabler, kabeltyper eller radiolinker som kommuniserer fra punkt til punkt.
- Kabel eller radiolink kobler alltid sammen to noder.
- En-til-en overføring.

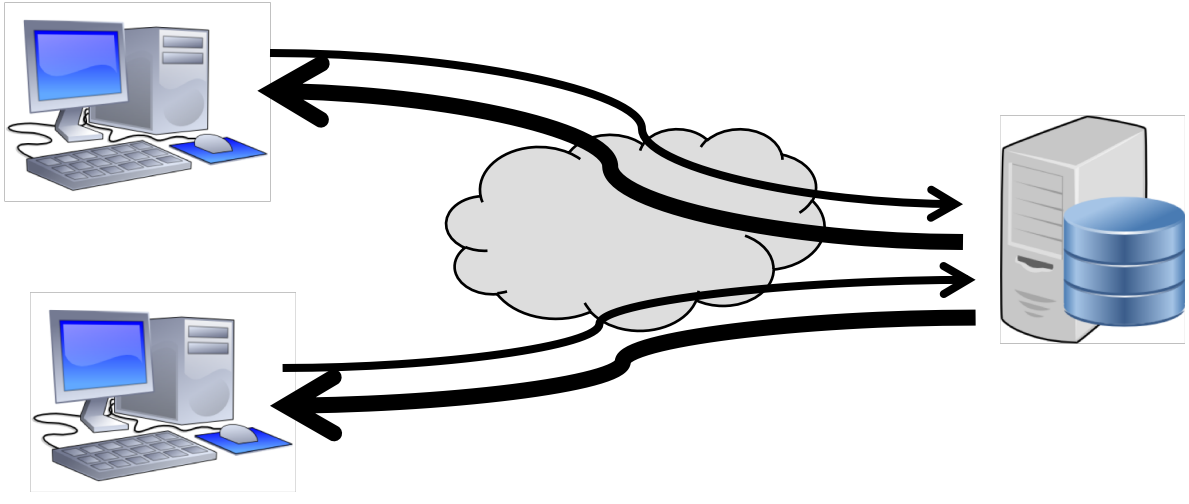
Topologieksempler:



Aksessmodeller: Klient-tjener

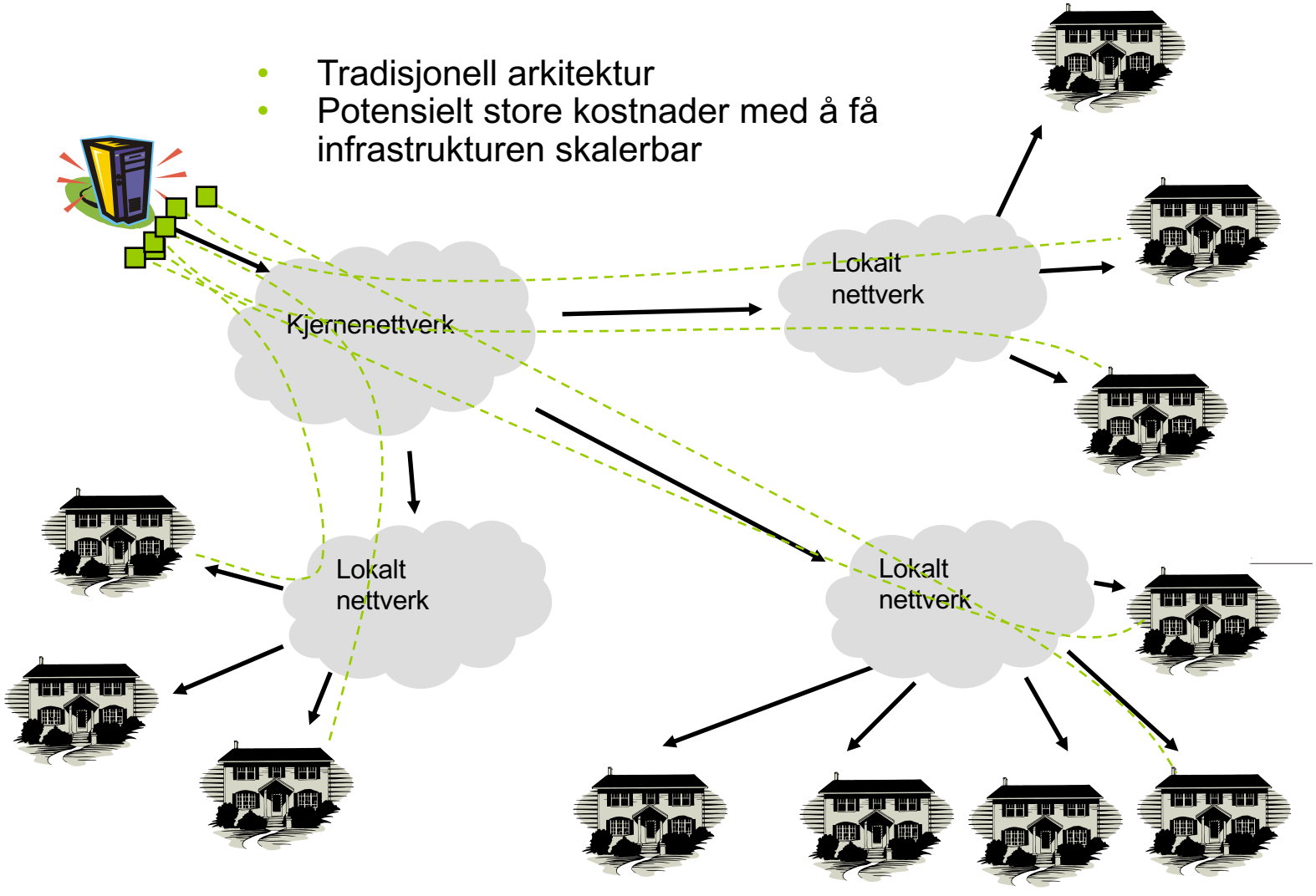
- Tradisjonell kommunikasjonsmodell, lettfattelig abstraksjon
  - Klienter ber om en tjeneste (oppretter en forbindelse)
  - Tjenere leverer tjenesten (svarer på forespørselen)

- Eksempler: Webklient (nettleser)/Webtjener, Mailklient/tjener, FTP (filoverføring)



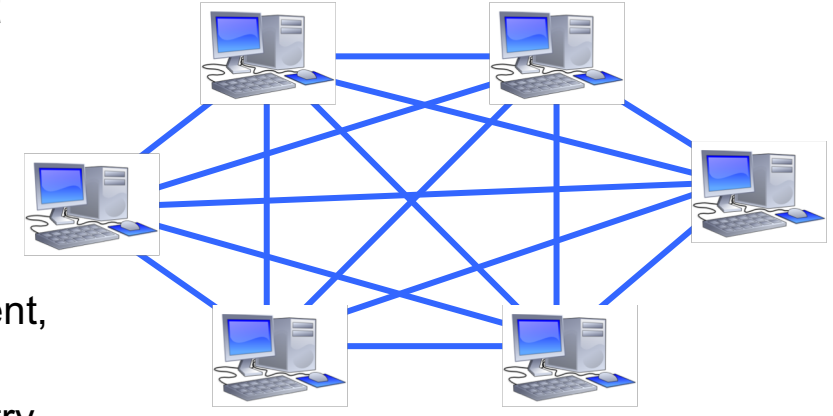
Distribusjon av data klient-tjener

- Tradisjonell arkitektur
- Potensielt store kostnader med å få infrastrukturen skalerbar



Aksessmodeller:Peer-to-Peer (P2P)

Applikasjonslagsparadigme kjent siden 2000-tallet



Det første velkjente programmet: Napster

- fildeling (mest brukt til musikk)
- erklært ulovlig
- Etterfulgt av mange andre: Gnutella, Kazaa, BitTorrent, Freenet
- senere plukket opp av forskere: CAN, Chord, Tapestry, Kademlia, Pastry
- Idé: unngå kontroll og/eller sensur

Kjente tjenester

- video streaming: PPTV, P2PTV
- Distribuert regnekraft: SETI@home
- Bitcoin (blockchain)
- TOR (The Onion Router)

Gammel teknologi som ligner P2P:

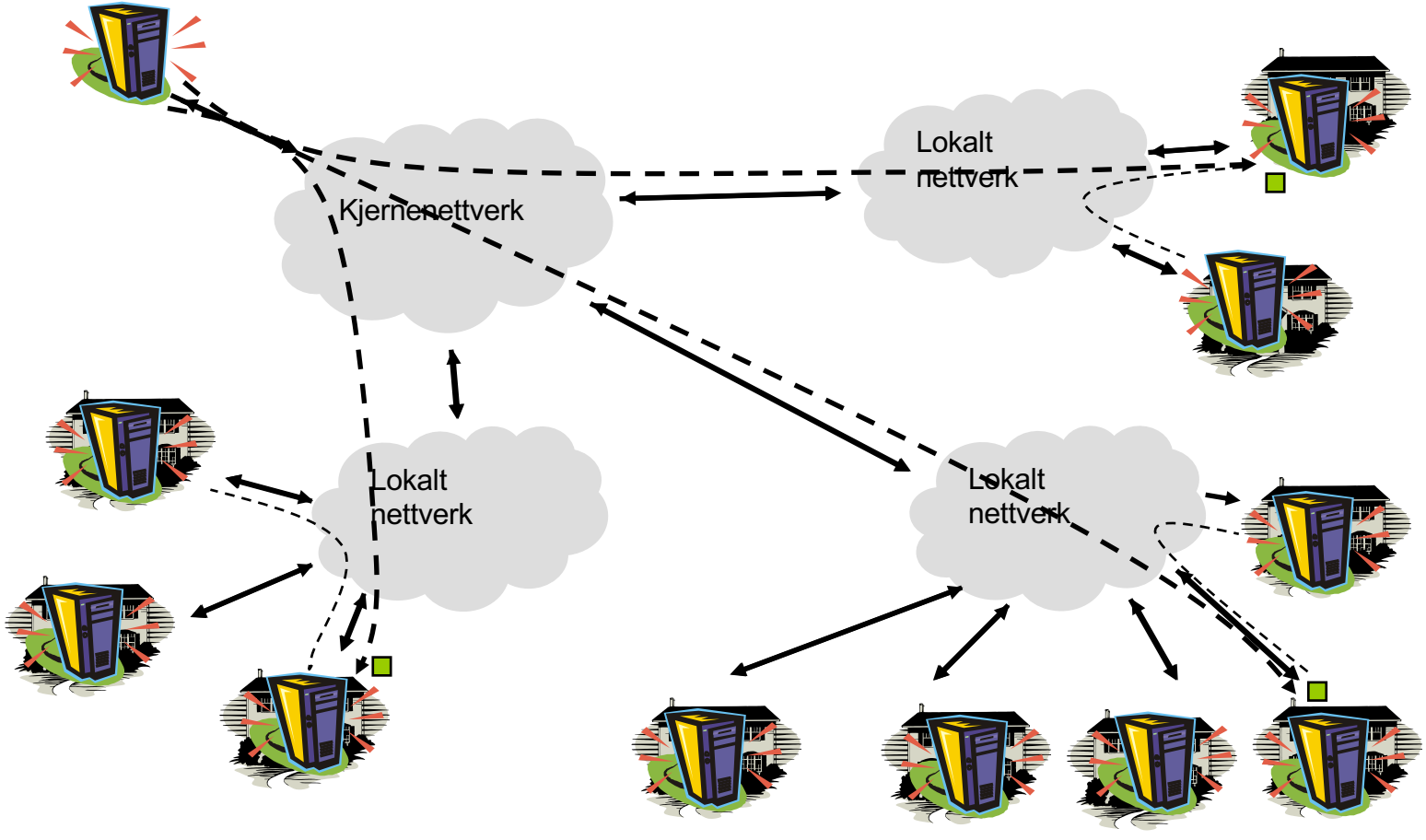
- Telefoni
- Usenet news
- IP Routing

Faktisk er P2P = den originale modellen for Internett

- alle noder er likeverdige
- alle noder kan nå hverandre
- eierskapet er distribuert

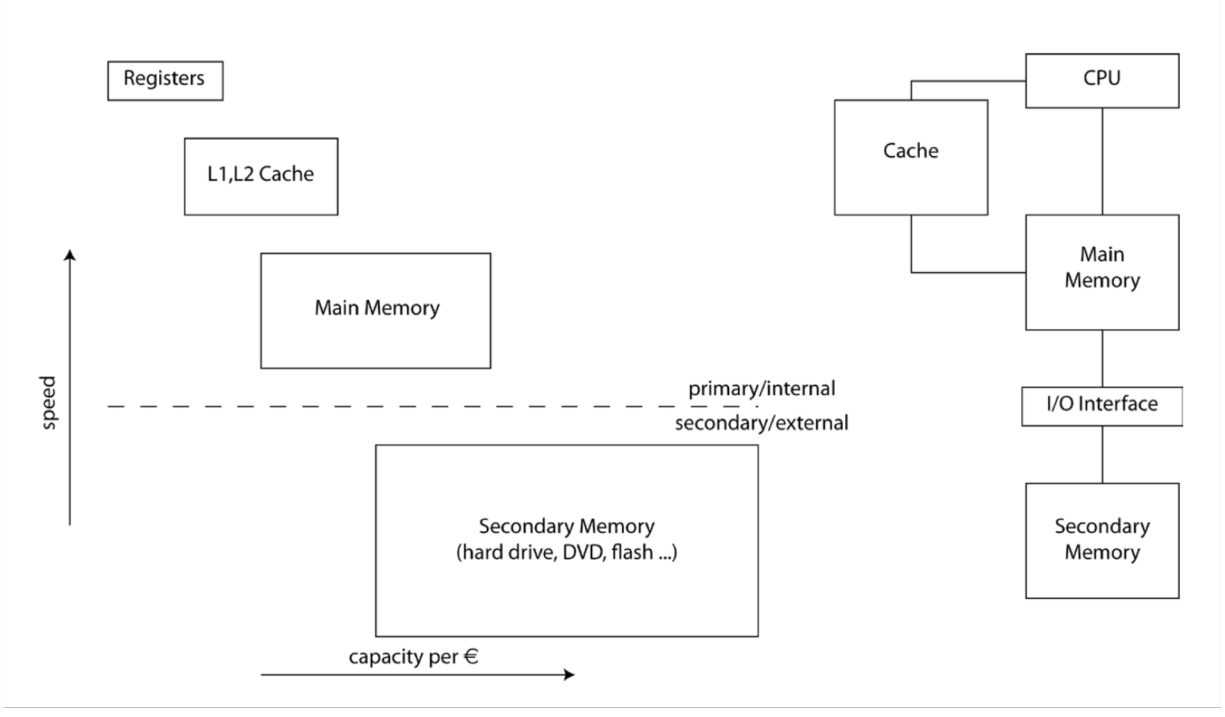
Distribusjon av data med P2P

- Typisk mange noder som kan være upålitelige og/eller heterogene



Fysisk plassering av innholdet

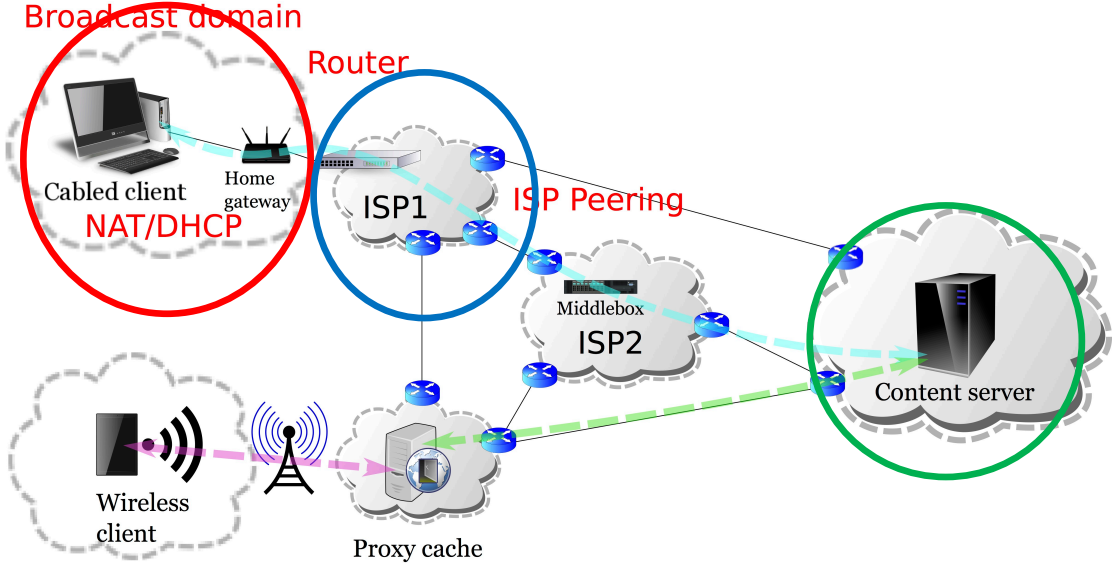
- Dere husker cachehierarkiet fra Omids forelesning?



Fysisk plassering av innholdet

- Denne figuren kan utvides så den gjelder også utenfor datamaskinen

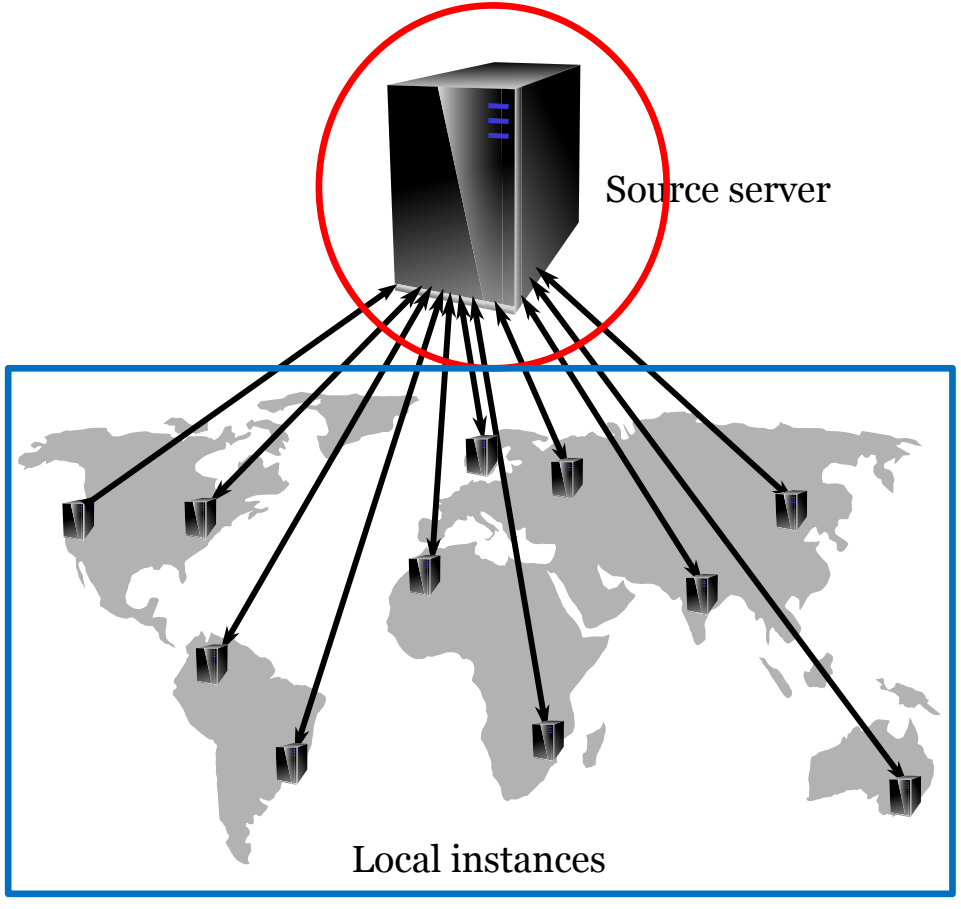
Cachenivå	Fysisk beliggenhet	Est. RTT
Lokal Proxy	Organisasjon / LAN	< 10ms
Content Delivery Network	ISP	< 50ms
Original datakilde	Internett	~10ms - ~250ms



Content Delivery Network (CDN)

- Innhold som skal leveres til klienter over hele verden…
- …kan replikeres til maskiner som ligger fysisk nær brukerne.

- Koster ekstra maskinvare og lagringsplass
+ Sparer kapasitet i backbone-nettet
+ Gir lavere forsinkelse på forespørsler
+ Hindrer overbelastning av tjeneren

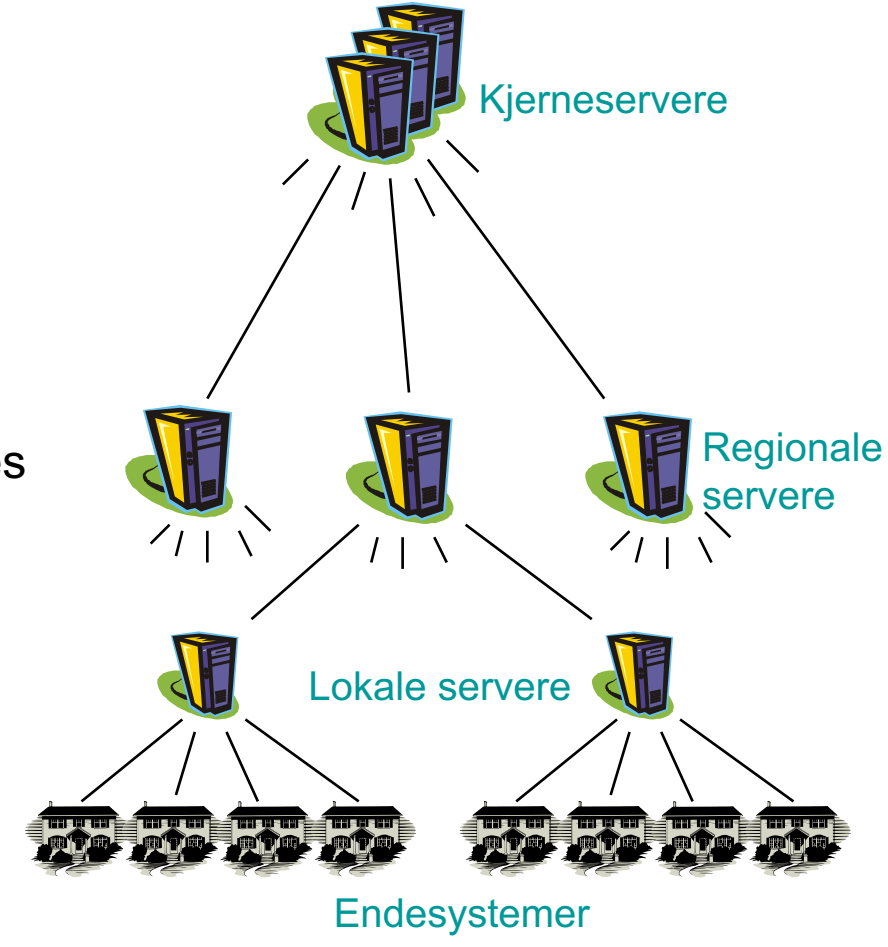


Figur fra: ”B. Briscoe *et al.*, "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials*

Hierarkisk innholdsdistribusjon med CDN

- **Hierarkiske systemer for å distribuere innhold**
  - Vanlig å ”cache” populært innhold.

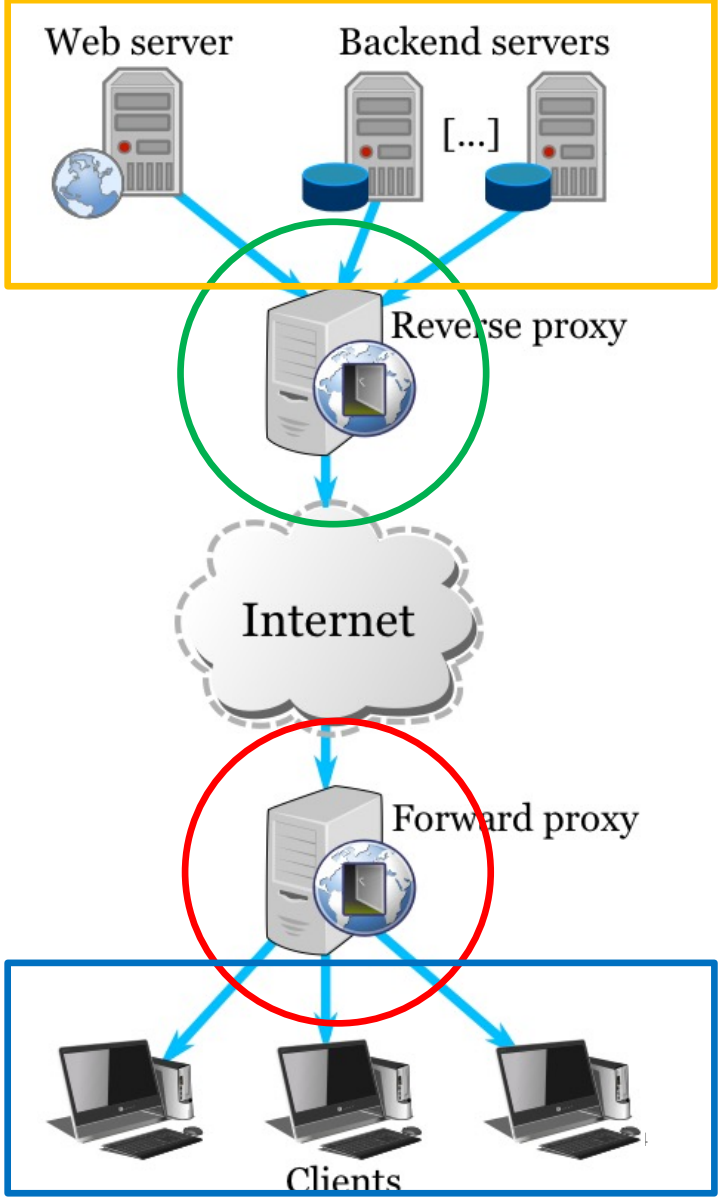
- Populært innhold blir lagret nær sluttbrukerne
- Innhold som ikke er så populært lagres enten på eller nær kjerneserveren
- Spesielt viktig hvis båndbredden er begrenset.
  - For eksempel: WiFi på fly



Proxy-cache

- En ”Forward Proxy” står nær klienten
  - og mellomlagrer data for klientene, slik at de ikke behøver å gå helt til kilden.
- En ”Reverse Proxy” står nær tjeneren
  - og mellomlagrer data fra én eller flere tjenere, slik at klienten slipper å gå helt til kilden(e).

+ Lastbalansering
+ Sparer nettverkskapasitet
+ Lavere forsinkelse



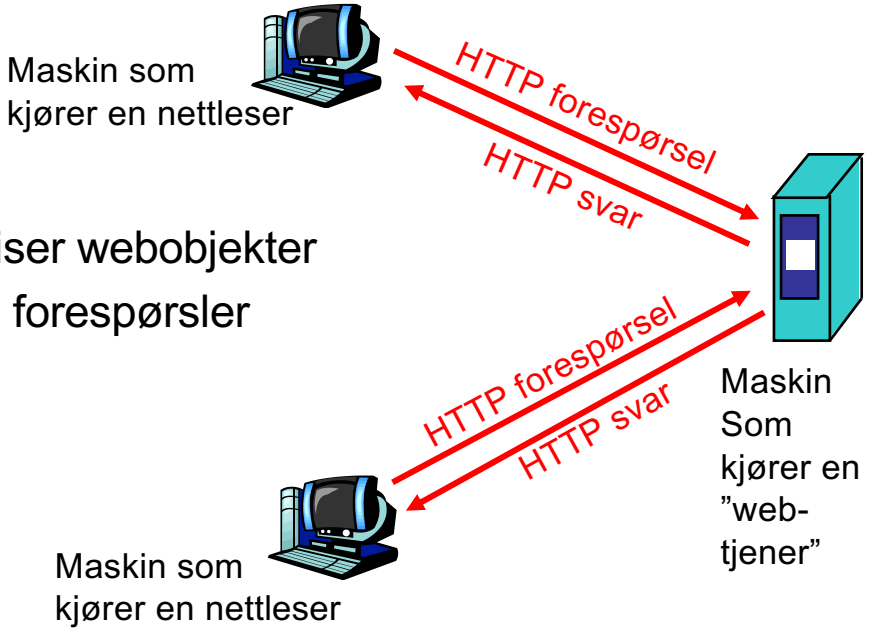
Figur fra ”B. Briscoe *et al.*, "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials*

World Wide Web (www): HTTP-protokollen

HTTP: hypertext transfer protocol

- Applikasjonslagsprotokollen for Web
- Klient-/tjenermodell
  - *klient:* nettleser som spør etter, får og viser webobjekter
  - *tjener:* sender webobjekter som svar på forespørsler

- Tre hovedversjoner:
- HTTP/1.0 (1990)
- HTTP/1.1 (1999)
- HTTP/2 (2015)



HTTP-protokollen

HTTP: bruker TCP som transport:

- Klienten oppretter en TCP-forbindelse (socket) til tjeneren, port 80
- Tjeneren godtar TCP-forbindelsen fra klienten
- HTTP-meldinger (protokollmeldinger på applikasjonslaget) utveksles mellom nettleseren (HTTP-klient) og Webtjeneren (HTTP-tjener)
- TCP-forbindelsen lukkes

HTTP er “stateless”

- Tjeneren sparer ikke på tilstandsinformasjon om tidligere forespørsler

Protokoller som sparer på ”tilstand” er komplekse!

- Tilstanden må vedlikeholdes
- Om en tjener eller klient ”kræsjer”, kan tilstanden bli ulik mellom dem. Da må den gjennoprettes.

HTTP-eksempel

Anta at en bruker skriver URLen www.mn.uio.no/ifi/index.html

1a. HTTP-klienten starter en TCP-forbindelse HTTP-tjeneren (prosess) på www.mn.uio.no. Port 80 er standard for HTTP-tjenere.

1b. HTTP-tjeneren på www.mn.uio.no lytter etter TCP-forbindelser på port 80. "godtar" forbindelsen, gir beskjed tilbake til klienten.

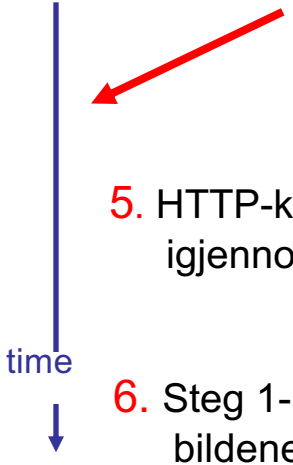
2. HTTP-klienten sender HTTP *request message* (med URL) til TCP-forbindelsen

3. HTTP-tjeneren mottar forespørselen, lager en *response message* med objektet det spørres etter (ifi/index.html), sender meldingen til TCP-forbindelsen.

tid

(la oss nå anta at index.html inneholder tekst og referanser til 10 JPEG-bilder)

HTTP example (cont.)

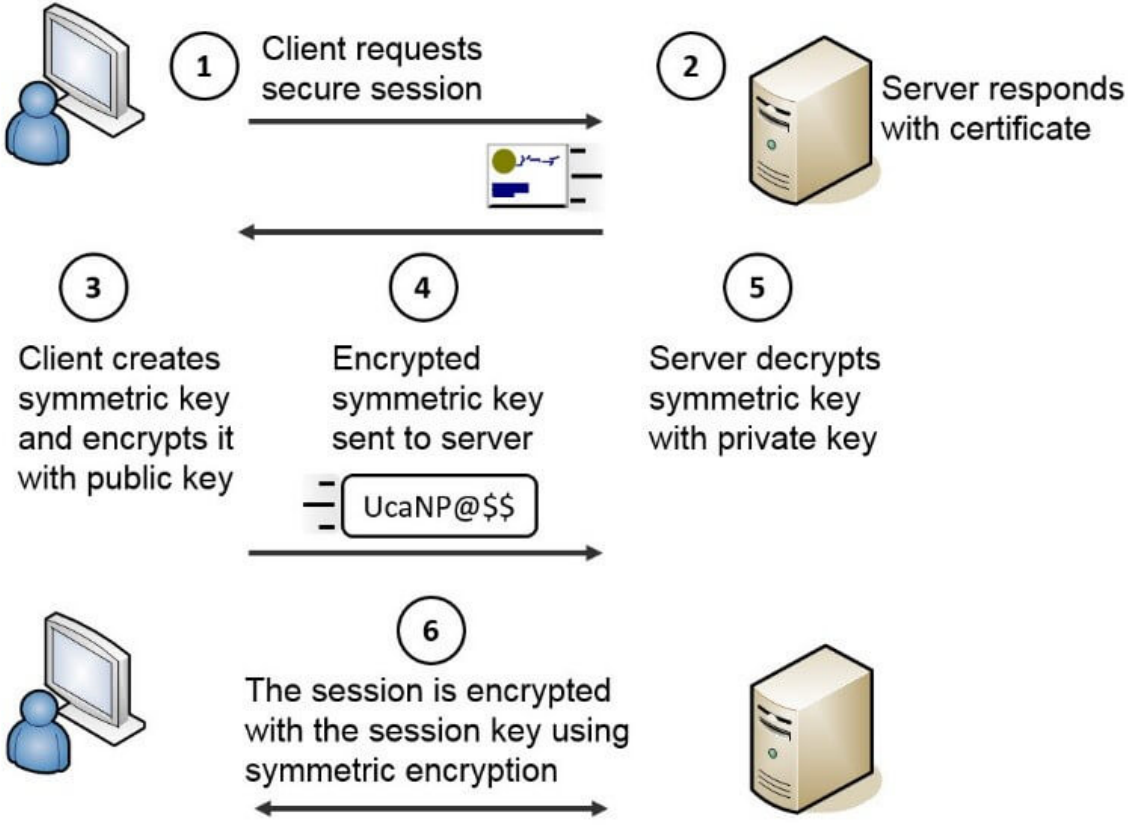


4. HTTP-tjeneren lukker TCP-forbindelsen.

5. HTTP-klienten mottar svarmeldingen med HTML-fil, viser HTML. Mens den går igjennom HTML-filen, finner den referansene til de 10 JPEG-bildene

6. Steg 1-5 gjentas for hvert av de 10 JPEG-bildene

HTTP med SSL


1
Client requests secure session
2
Server responds with certificate
3
Client creates symmetric key and encrypts it with public key
4
Encrypted symmetric key sent to server
UcaNP@$$
5
Server decrypts symmetric key with private key
6
The session is encrypted with the session key using symmetric encryption

Persistente og ikke-persistente forbindelser

Ikke-persistent

- HTTP/1.0: tjeneren leser forespørselen, svarer, lukker TCP-forbindelsen
- 2 RTTer for å hente objektet
  - TCP-forbindelse
  - forespørsel og overføring
- Hver overføring lider av TCP sin gradvise økning i senderate (slow start)
- Mange nettlesere åpner flere parallelle forbindelser

Persistent

- Standard for HTTP/1.1
- over samme TCP-forbindelse: tjeneren leser forespørselen, svarer, leser ny forespørsel…
- Klienten sender forespørsler for alle de objektene den trenger så fort den mottar hoveddokumentet (HTML)
- Færre RTTer, mindre slow start

Persistent med ”pipelining”

- Spør etter mange objekter på én gang (enda færre RTTer)
- Svaret kommer i serie etter hverandre i rekkefølgen forespørslene ankom tjeneren.

HTTP/1.x meldingsformat: request

- To typer HTTP-meldinger: *request*, *response*
- HTTP request:
  - ASCII (lesbart av mennesker)

Request (kommandoer: GET, POST, HEAD)

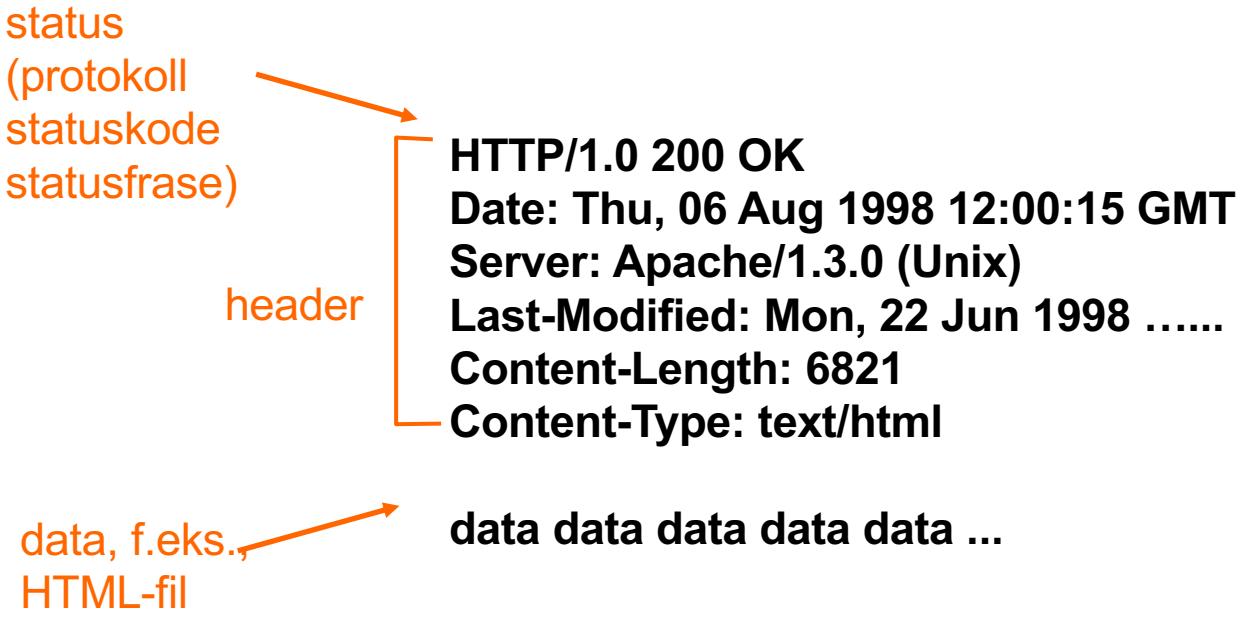
header

GET /ifi/index.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:no

Blanke linjer (return), indikerer slutten på Meldingen.

(ekstra return, ny linje)

HTTP/1.x meldingsformat: response



HTTP/1.x response statuskoder (eksempler)

200 OK

- Forespørselen var vellykket, objektet kommer senere i meldingen

301 Moved Permanently

- Objektet har byttet plassering. Referanse til ny plassering kommer senere i meldingen.

400 Bad Request

- Forespørselen var uforståelig for tjeneren

404 Not Found

- Dokumentet ble ikke funnet på tjeneren

505 HTTP Version Not Supported

Prøve HTTP/1.x (klient) selv

1. Telnet til din favoritthjemmeside:

telnet www.uio.no 80

Åpner en TCP-forbindelse til port 80 (standardport for HTTP) hos www.uio.no
Alt du skriver inn blir sendt over denne forbindelsen

2. Skriv inn en GET-forespørsel:

GET / HTTP/1.1

Når du skriver inn dette (trykker return én gang), blir denne minimale (men fullstendige) GET-forespørselen etter rotdokumentet sendt til HTTP-tjeneren

3. Fort, skriv inn ”Host”-feltet:

Host: www.uio.no

En tjener kan være vert for mange hjemmesider, så klienten må spesifisere hvilken vert den ønsker i HTTP-headeren. Om det ikke gjøres vil den ofte gi en feilmelding tilbake.

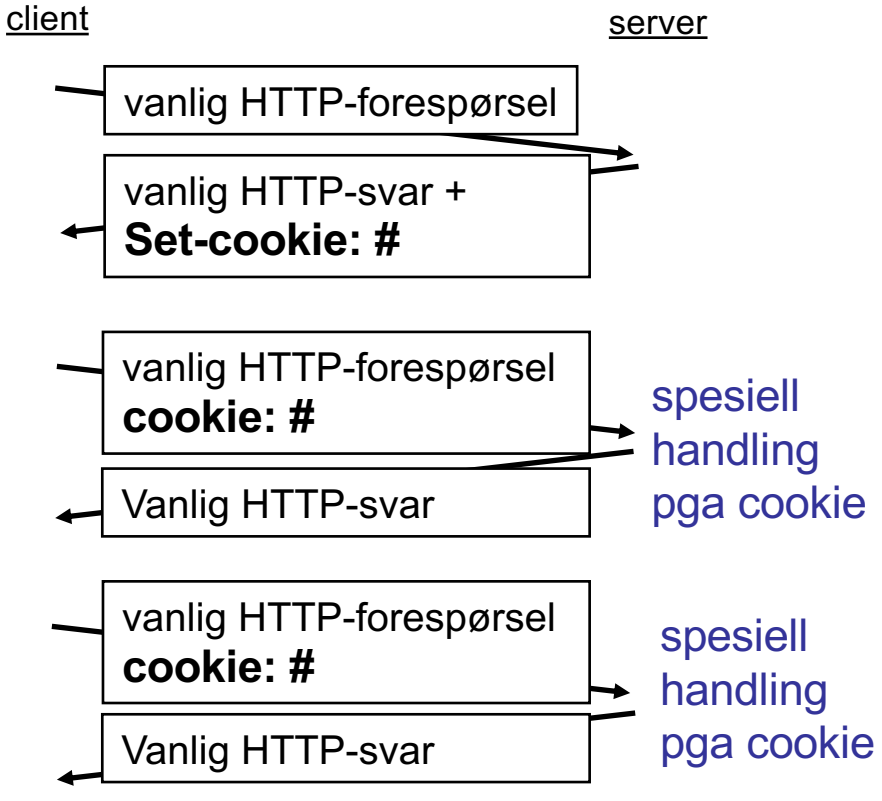
4. Trykk return to ganger og se resultatet

Cookies: ta vare på ”tilstand”

- Tjeneren lager et cookie # , tjeneren husker #, senere brukt til:
  - autentisering
  - Huske brukerpreferanser, tidligere valg
    - produkter brukeren har sett på o.l.
- tjeneren sender cookien til klienten i svarmeldingen

  Set-cookie: 1678453
- Klienten legger ved cookien med etterfølgende forespørsler

  cookie: 1678453

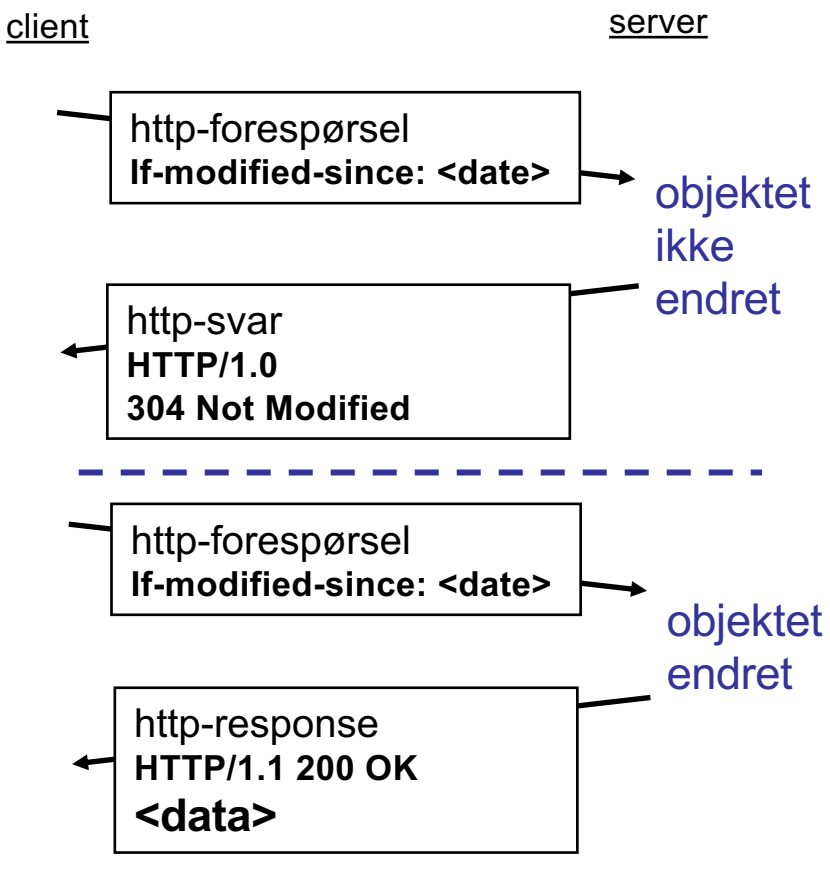


Betinget GET: klient-side caching

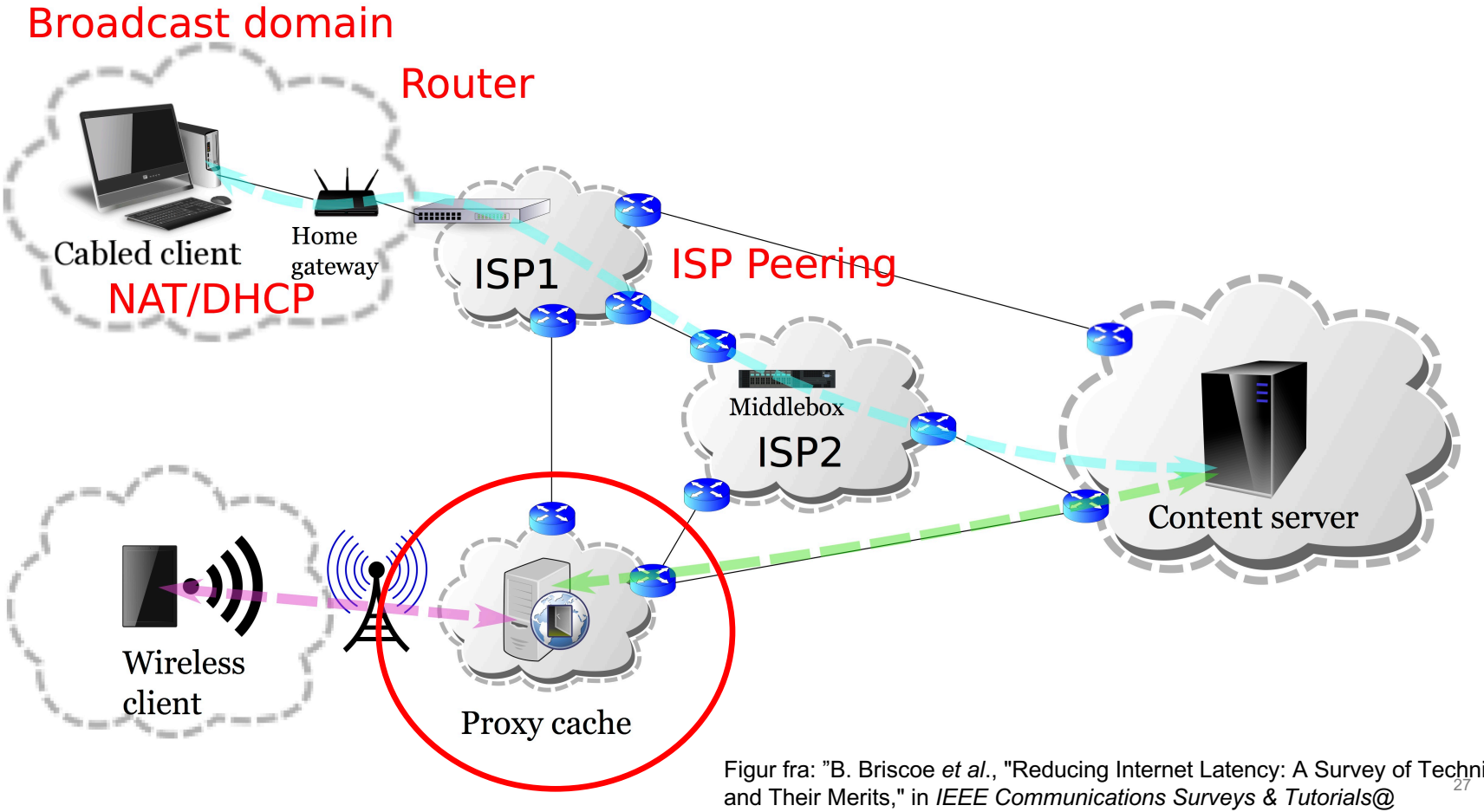
- Mål: ikke sende objektet om brukeren har en oppdatert versjon i lokal cache
- klient: oppgi tid/dato for cachet kopi i HTTP-forespørselen

  If-modified-since: <date>
- tjener: svaret inneholder ikke objektet dersom den cachede kopien er oppdatert:

  HTTP/1.0 304 Not Modified



HTTP Proxytjener

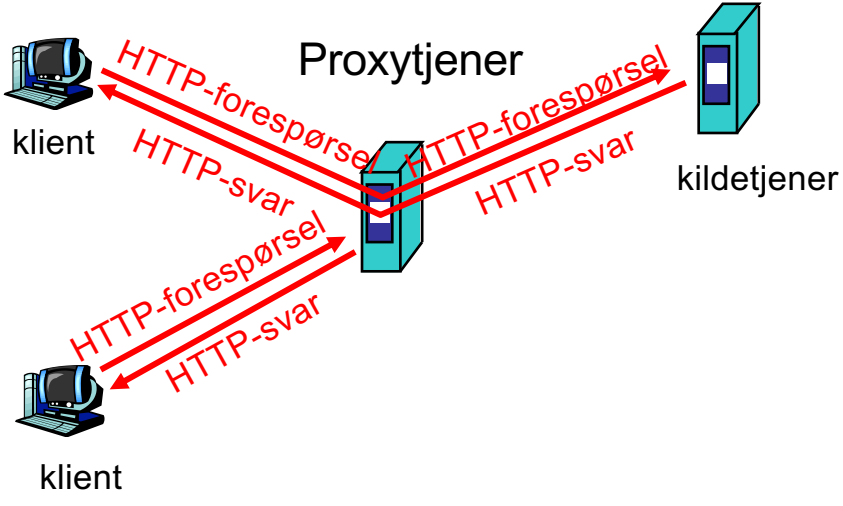


Figur fra: ”B. Briscoe *et al*., "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials*@

HTTP Proxytjener

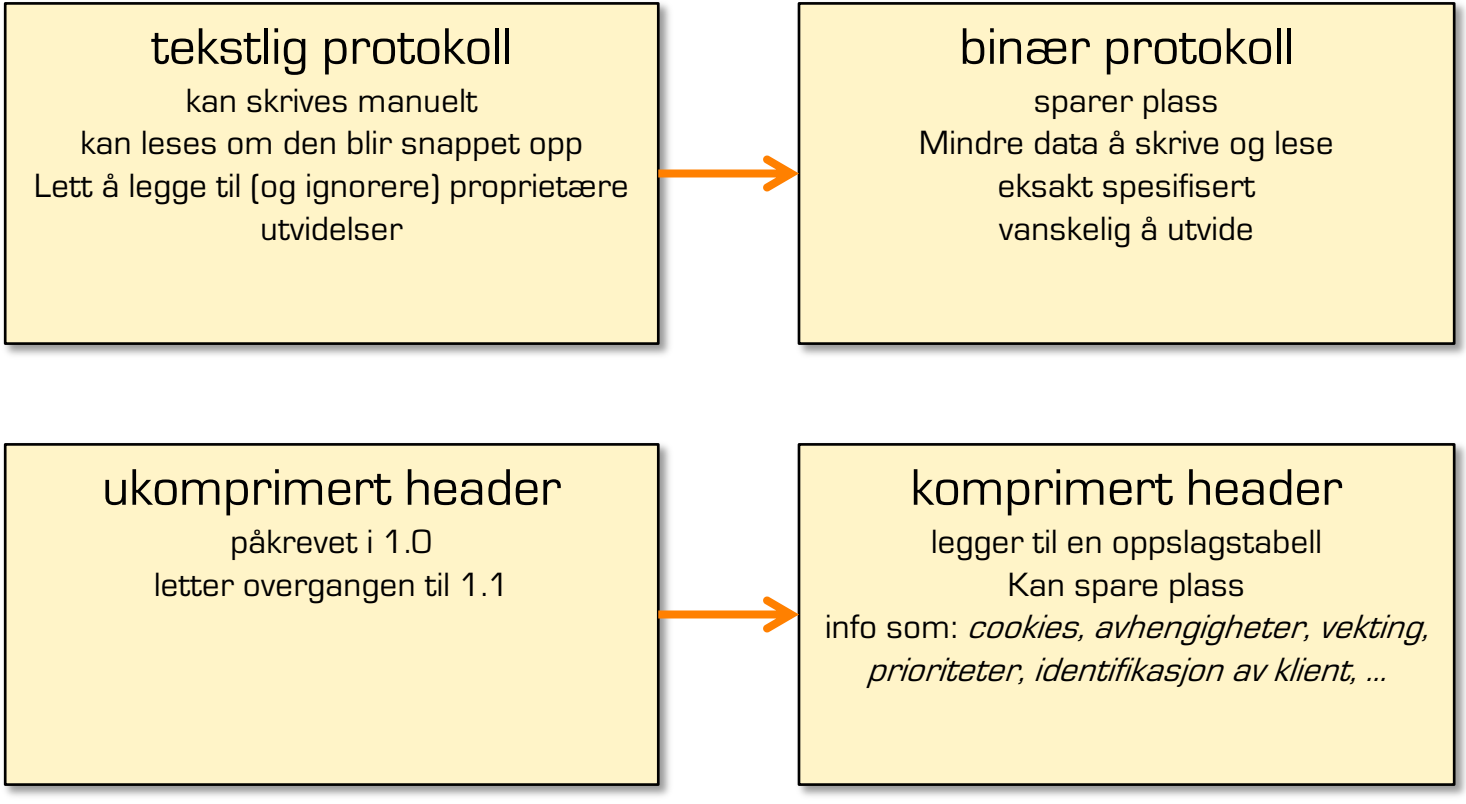
Mål: levere svar til klienten uten å gå helt til kilden

- Brukeren konfigurerer nettleseren: Web skal gå via proxytjener
- Klienten sender alle HTTP-forespørslene til proxytjeneren
  - Om objektet er i web cache: Proxytjeneren leverer objektet
  - Om objektet ikke er i web cache, sender Proxytjeneren en forespørsel til kilden og lagrer svaret før den sender svaret til klienten.



Antagelse: cachen er nærmere klienten (f.eks i samme nettverk) => raskere svar, mindre langdistansetrafikk

Endringer i HTTP/2



Endringer i HTTP/2

Motvirke trenden med å åpne mange parallelle forbindelser

sekvensiell og blokkerende
øker hastigheten ved å åpne flere parallelle TCP-forbindelser (1.x)
Øker hastigheten med pipelining (1.1)

→

multiplekset i én forbindelse
Sender alle forespørsler på en gang
Tjeneren velger rekkefølge på svarene (f.eks. reklame først) og kan blande beskjeder fra forespørsler.

Større muligheter for at tjeneren kan ta «egne» avgjørelser.

client pull
antagelse: tjeneren tar ikke vare på tilstand

→

client pull + server push
antagelse: tjeneren vet best

DNS prefetching for web

Klient

Link: www.example.org/doc1.htm
Link: www.daserva.com/img.jpg
Link:padawan.co.uk/jedimindtrick.css
…
…

Bakgrunnsprosess

DNS lookup:
www.example.org
www.daserva.com
Padawan.co.uk

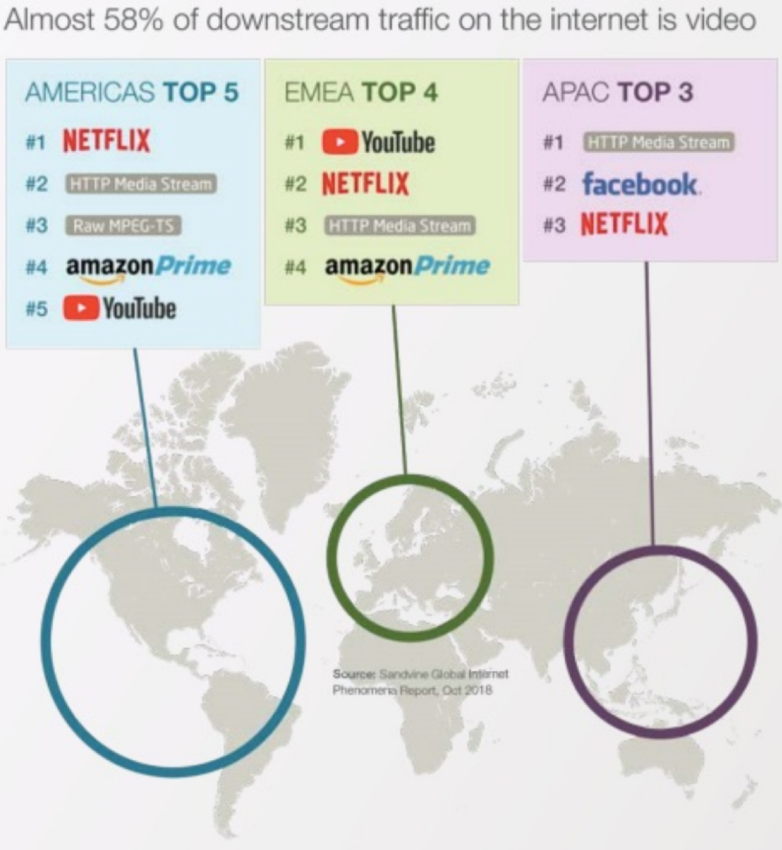
DNS cache

Når brukeren trykker en link er allerede DNS-oppslaget utført.

Videostreaming

GLOBAL APPLICATION CATEGORY TRAFFIC SHARE

1	VIDEO STREAMING	57.69% ⬇	22.43% ⬆
2	WEB	17.01% ⬇	20.98% ⬆
3	GAMING	7.78% ⬇	2.68% ⬆
4	SOCIAL	5.10% ⬇	3.73% ⬆
5	MARKETPLACE	4.61% ⬇	1.90% ⬆
6	FILE SHARING	2.84% ⬇	22.05% ⬆
7	MESSAGING	1.72% ⬇	8.12% ⬆
8	SECURITY	1.41% ⬇	7.48% ⬆
9	STORAGE	1.41% ⬇	9.37% ⬆
10	AUDIO STREAMING	1.05% ⬇	0.46% ⬆



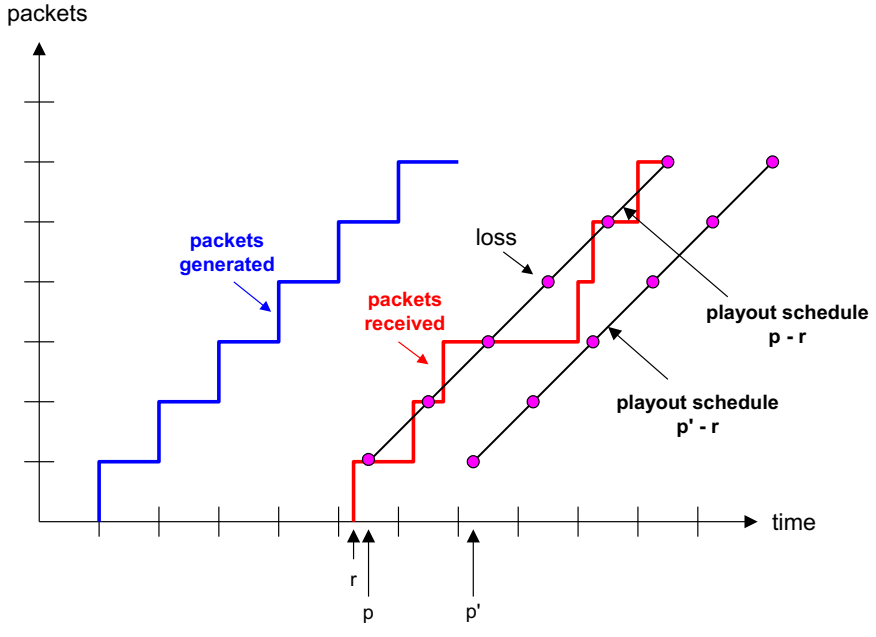
* Without counting filesharing or messaging applications

Kilde: Sandvine - http://digg.com/2018/streaming-video-worldwide - 4.10.2018

Videostreaming - utfordringer

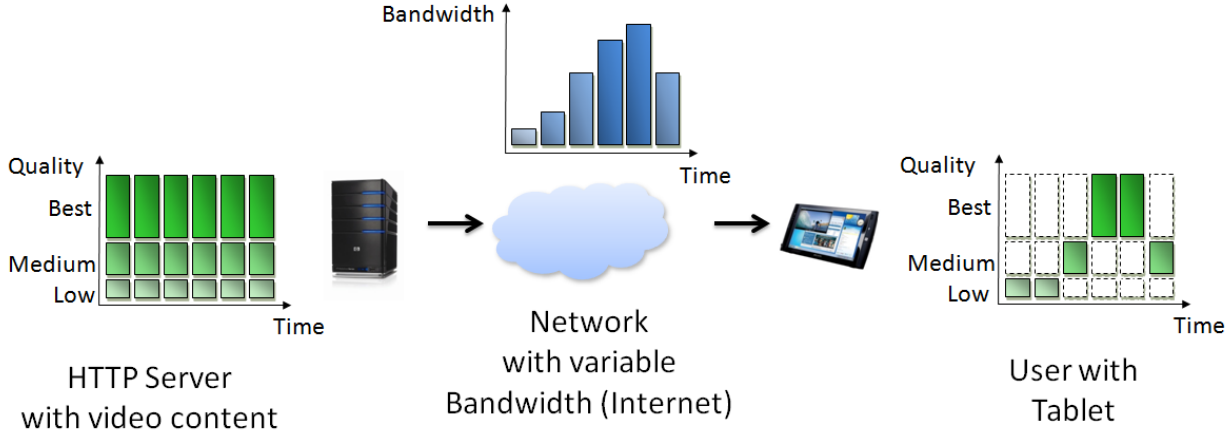
Multimedia

- Video 57% av trafikken på Internett
  - Netflix alene utgjør 30% i USA
- Nedlasting:
  - Hele innholdet lastet ned til lokal maskin.
- Streaming
  - En konstant strøm av data (til avspilling slutter)
  - Lettere å ta vare på opphavsrett
    - Lite lokal lagring
  - Sparer nettverksressurser
    - Brukes bare for det av innholdet som blir sett
- UDP eller TCP?
- Nettverksutfordringer
  - forsinkelse, tap, variasjoner i leveringstid (”jitter”)
  - jitter kompensasjon
  - Tapskompensasjon

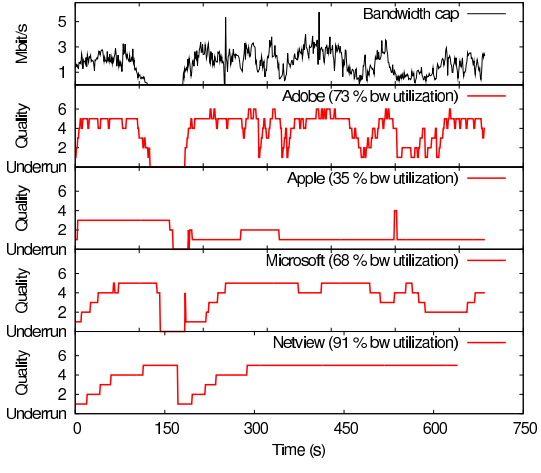


HTTP-streaming

Dynamisk, adaptiv streaming over HTTP (DASH)



- Dele videoen i segmenter: fullstendig uavhengige små filmer
- Velge varighet på segmentene: 2-10 sekunder vanlig
- Velge antall kvalitetslag
- Velge tilpasninsgstrategi
  - Klienten velger, ikke tjeneren
  - Strategien hos avspilleren er det som utgjøre forskjellen



Epost

- Hovedkomponenter
  - "mailklienter"
    Message User Agent (MUA)
  - "mailtjenere"
    Mottak av meldinger / Videresending av meldinger
    - Mail submission agent (MSA)
    - Mail transfer agent (MTA)
    - Mail delivery agent (MDA)
    - Mail retrieval agent (MRA)
    - Ofte realisert som én komponent kalt Message Handling Service (MHS)
- MUA
  - eller "epostleser"
  - Skrive, redigere, organisere og lese eposter
  - utgående, innkommende lagres på eposttjener

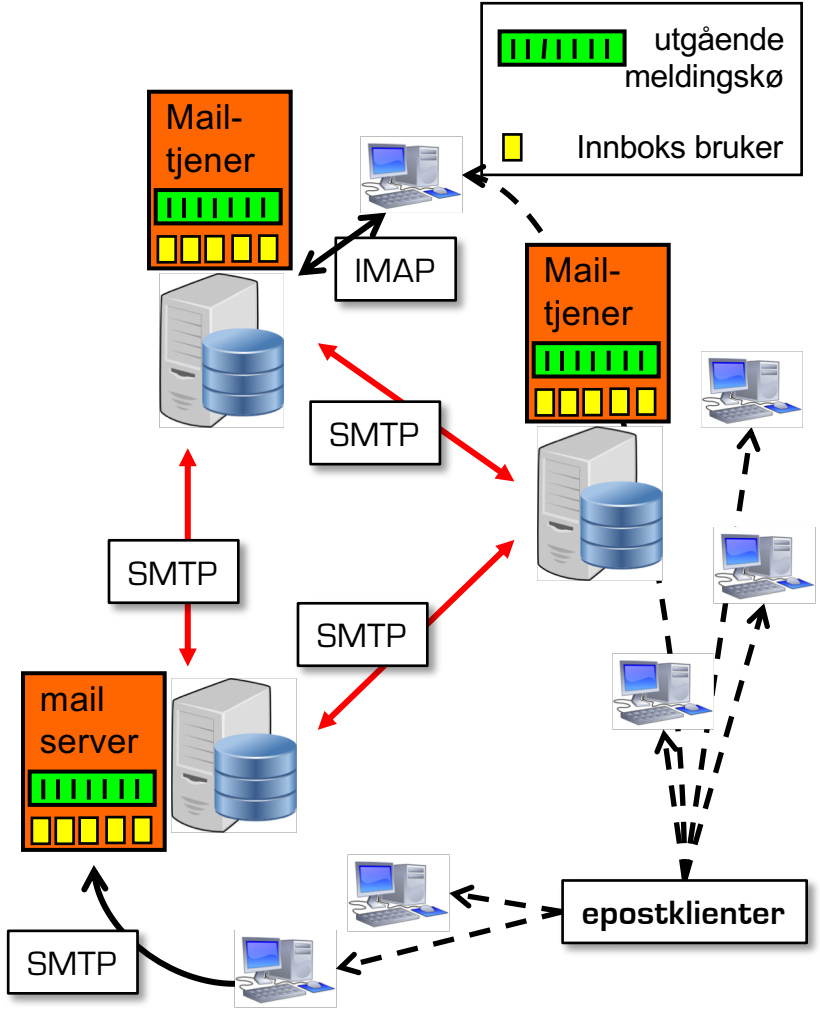
Epost: tjenere

Mailtjenere

- *mailbox* inneholder innkommende meldinger (hittil uleste) til brukeren
- *meldingskø* av utgående epostmeldinger (for sending)

Simple Mail Transfer Protocol (SMTP)

- Mellom eposttjenere for å sende epostmeldinger
- klient: sender av en epost
- tjener: den som mottar eposten



Epost: SMTP

- Bruker TCP til å, pålitelig, overføre epost fra klient til tjener.
- Standardisert port: 25
- Direkte overførsel: fra senderen til tjeneren som tar imot.
- Tre faser i overføringen
  - håndtrykk (greeting)
  - Overføring av beskjeder
  - avslutning
- Kommandoer/interaksjon
  - kommandoer: ASCII-tekst
  - svar: statuskode og frase
- Meldinger må være i 7-bit ASCII

Eksempel på interaksjon

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C:   How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

SMTP for hånd

```
telnet servername 25
```

Se 220-svaret fra tjeneren

Disse kommandoene lar deg sende en epost uten en epostklient:

HELO, MAIL FROM, RCPT TO, DATA, QUIT

SMTP: final words

SMTP bruker persistente forbindelser

SMTP krever (header & body) i 7-bit ASCII

Visse tegnkombinasjoner er ikke tillatt I meldingen (f.eks, `CRLF.CRLF`). Meldingen må derfor kodes (vanligvis i base-64 eller “quoted printable”)

SMTP-tjeneren bruker `CRLF.CRLF` for å avgjøre når meldingen slutter (ingen angivelse av lengde i header)

Sammenligning med HTTP/1.x:

- HTTP: pull
- SMTP: push
  - Frem til den siste tjeneren!
- Begge har ASCII kommando/svar interaksjon, statuskoder
- HTTP
  - Hvert objekt kapslet inn i sin egen svarbeskjed
- SMTP
  - Originalt som HTTP
  - nå: mange objekter sendt i meldinger med mange deler.

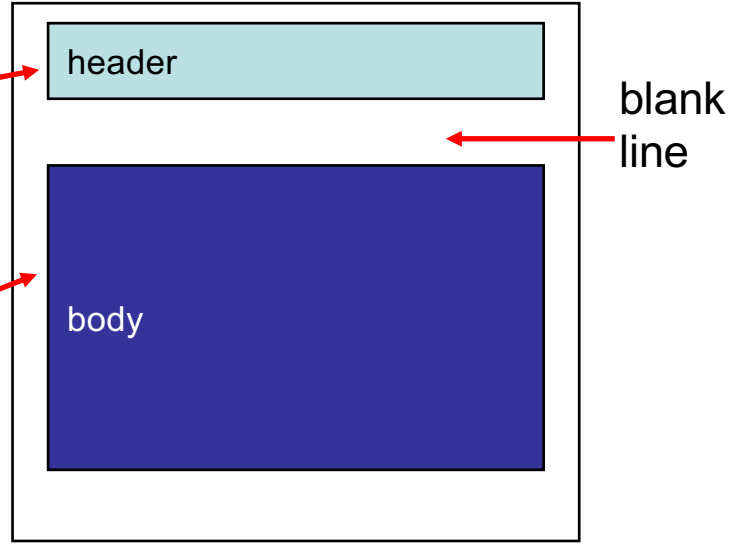
Meldingsformat

SMTP: protokoll for å utveksle epostmeldinger

Standard for tekstmeldinger:

- Headerlinjer f.eks.:
  - To:
  - From:
  - Subject:

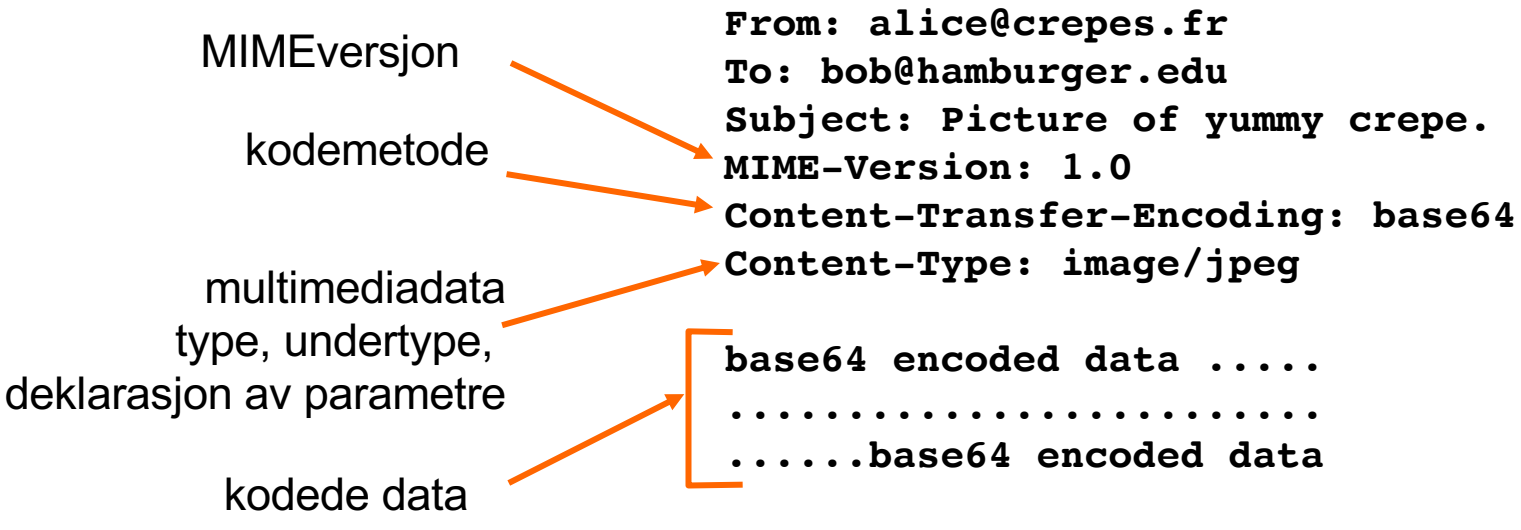
  forskjellig fra SMTP-kommandoer!
- body
  - ”meldingen”, bare ASCII-tegn



Meldingsformat: multimedia extensions

MIME: multipurpose Internet mail extension

Ekstra linjer i mailheaderen viser MIME-innholdstype



"klassisk" epost kan vise:
"Content-type: text/ascii", men
7-bit ASCII-tekst er fortsatt standard

MIME types

```
Content-Type: type/subtype; parameters
```

Text

- eksempler på undertyper: **plain, html**

Image

- eksempler på undertyper: **jpeg, gif**

Audio

- eksempler på undertyper: **basic** (8-bit mu-law encoded), **32kadpcm** (32 kbps coding)

Video

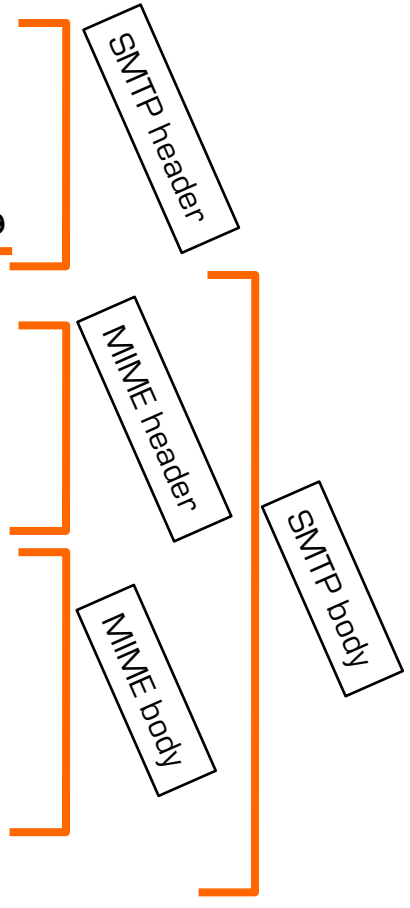
- eksempler på undertyper: **mpeg, quicktime**

Application

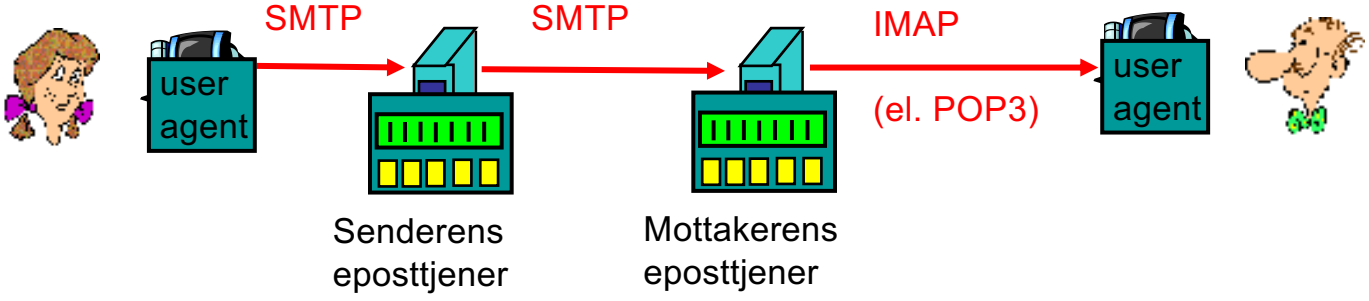
- Andre data som må leveres til et eksternt program før det kan vises
- eksempler på undertyper: **msword, octet-stream**

Flerledds meldingstype

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
--98766789--
```



Protokoller for mailtilgang



- SMTP: leverer til mottagerens epposttjener
- Mail access protocol: henter eposten fra tjeneren
  - POP: Post Office Protocol
    - autorisering(agent <==> server) og nedlasting
  - IMAP: Internet Mail Access Protocol (*Interim* ➔ *Interactive* ➔ *Internet*)
    - flere funksjoner (mer kompleks)
    - manipulere meldinger lagret på tjeneren
  - HTTP: Gmail, Hotmail, Yahoo!, etc.

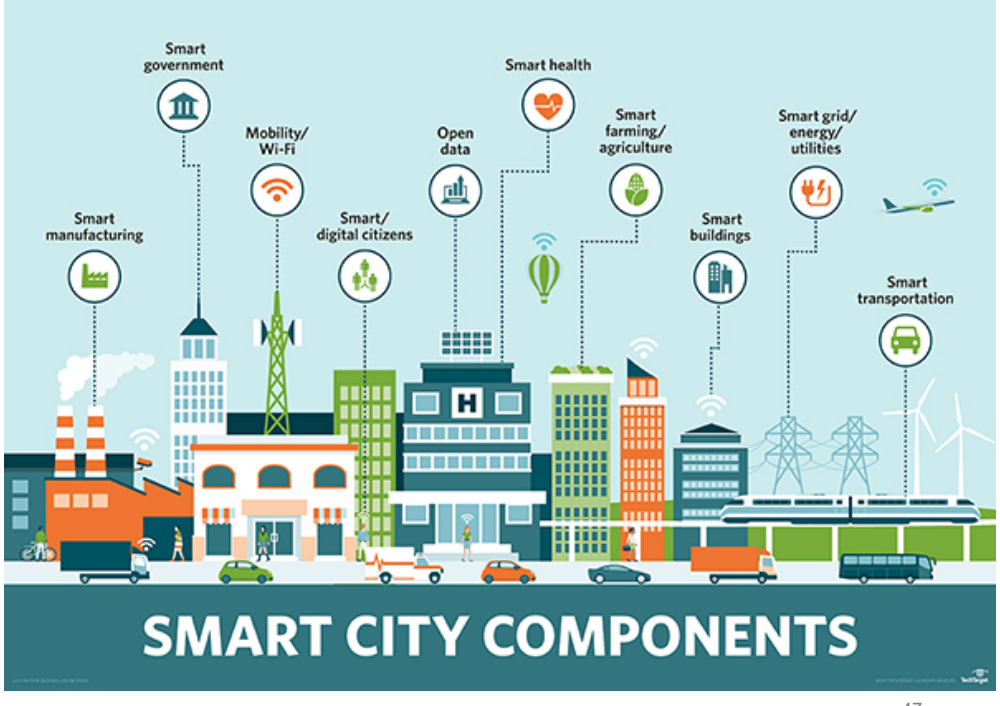
IMAP kan mer

- Egenskaper hos IMAP
  - Opprette, slette, bytte navn på epostmapper (på tjeneren)
  - Se etter nye meldinger, fjerne meldinger sette og fjerne flagg
  - Lese, søke og hente basert på søkeresultat
  - Søke i innholdet i meldinger
  - STORE and conditional STORE
  - CATENATE (to concatenate)
- Ofte brukt til :
  - TODO-lister
  - Notater med og uten Mime-elementer
- Erstatt “melding” med ”fil” og du har et ganske komplett filsystem

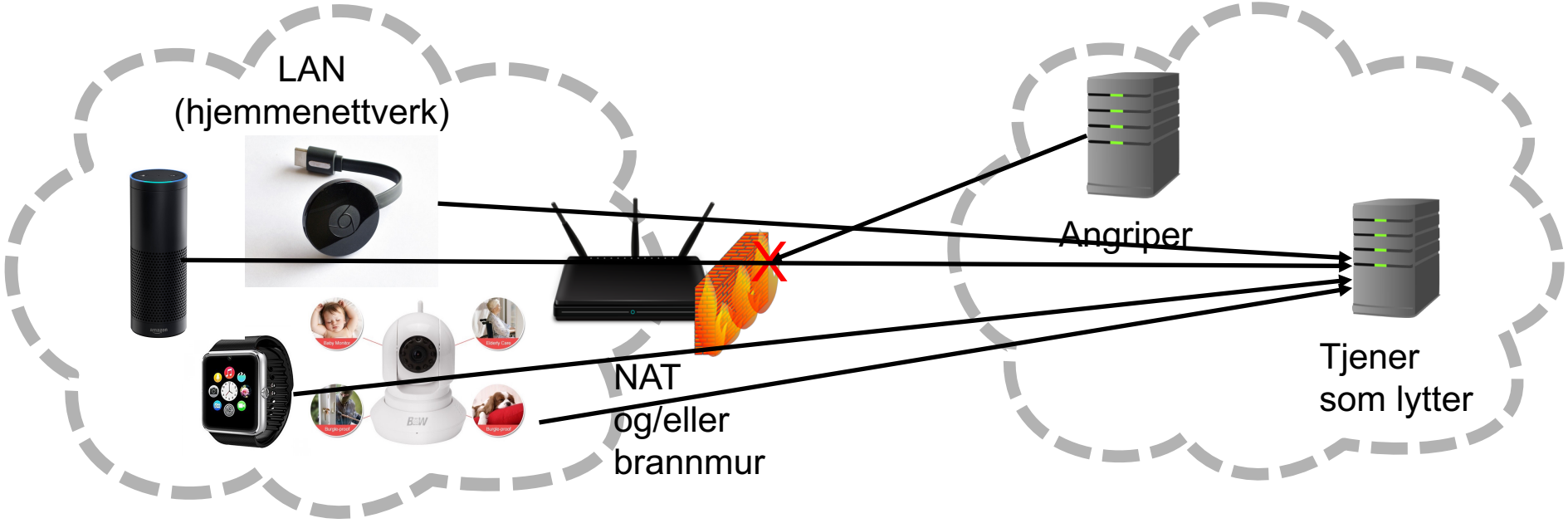
Internet of Things (IoT)

- Enheter i daglig bruk / husholdningen / arbeidsprosesser som leverer ekstra tjenester ved hjelp av Internett
- Kan være med på å revolusjonere hverdagen
  - Automatiske strømmålere
  - Helseapplikasjoner
  - Smarthus / Smartbyer
  - Logistikk

- Snakker med tjenere i ”skyen”
- Leverer data som brukes til analyse og tjenestelevering



Internet of Things (IoT) -sikkerhetsaspekter



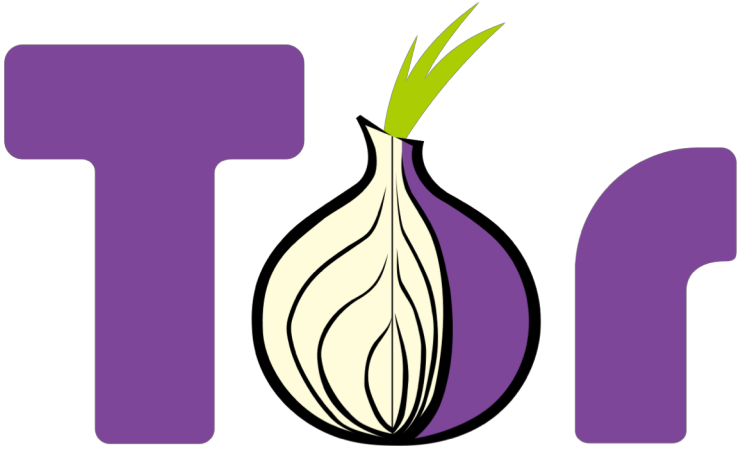
Internet of Shit (IoS ☺)

- Ofte er ting koblet på Internett bare fordi det høres kult ut
- Vi sitter igjen med en masse produkter på våre lokale nett som ”ringer hjem” (setter opp nettverksforbindelser med maskiner utenfor ditt LAN)
- Ofte kommer det ikke sikkerhetsoppdateringer til disse enhetene => sikkerhetshull / bakdør til ditt nettverk

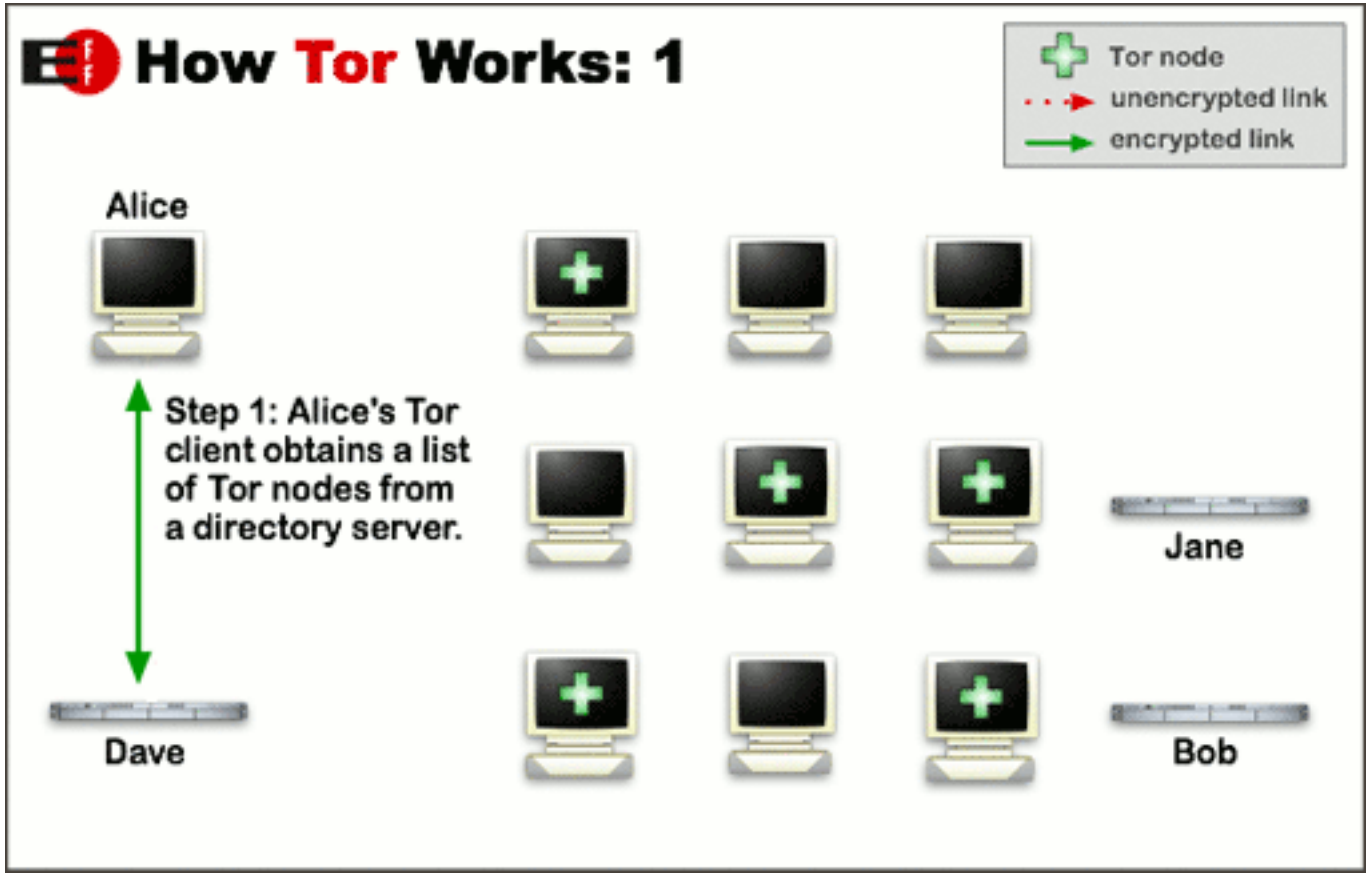
http://www.telegraph.co.uk/technology/2016/10/12/man-spends-11-hours-trying-to-make-cup-of-tea-with-wi-fi-kettle/

Eksempel på P2P applikasjon - The Onion Router (TOR)

- Nettverk av maskiner (peers) ”donert” som routere i TOR
- Trafikk krypteres og videresendes gjennom flere routere
- Krever en spesiell nettleser som kobler til nettverket.

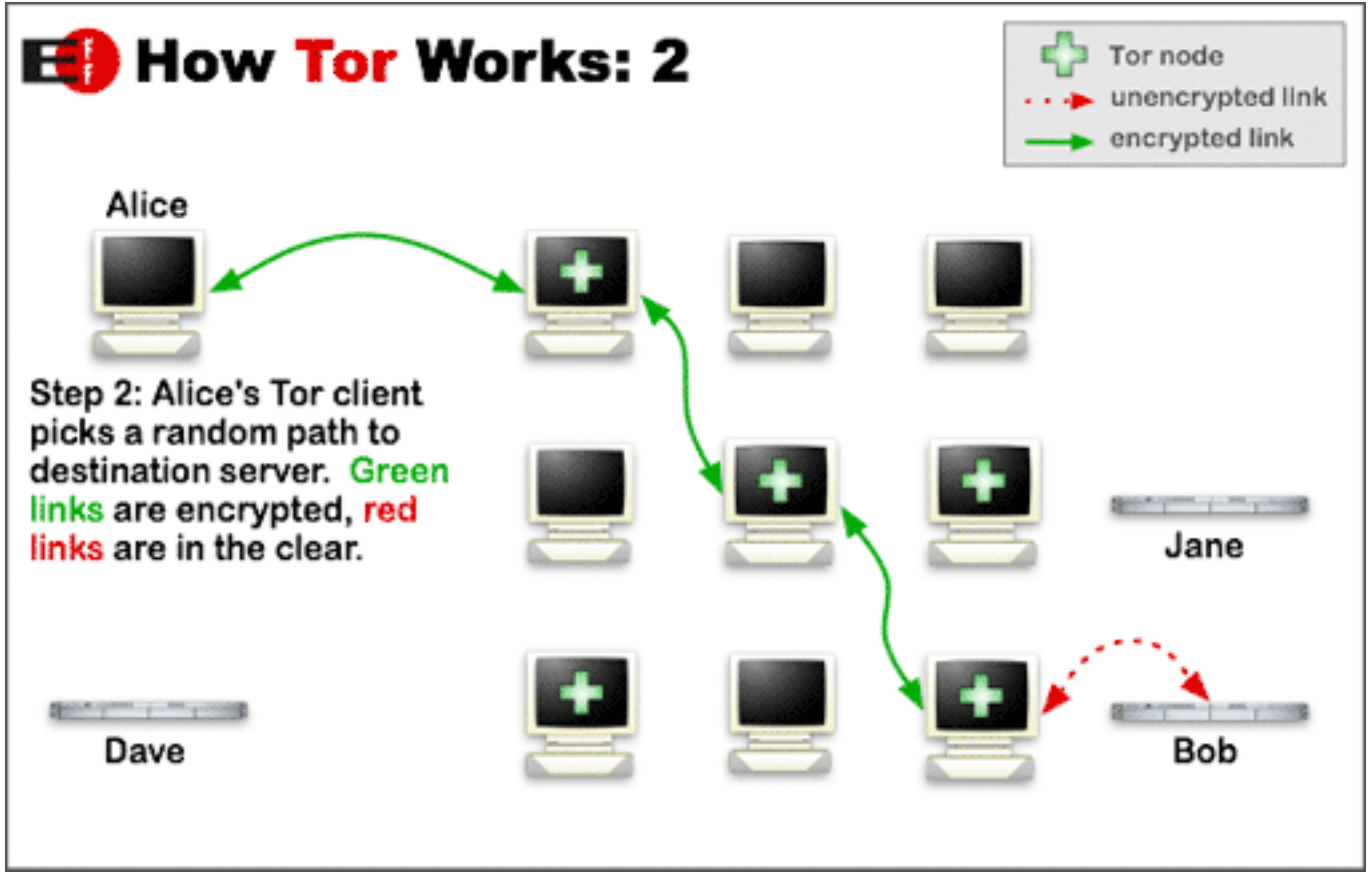


Hvordan virker TOR



Figur hentet fra https://www.torproject.org/about/overview.html.en

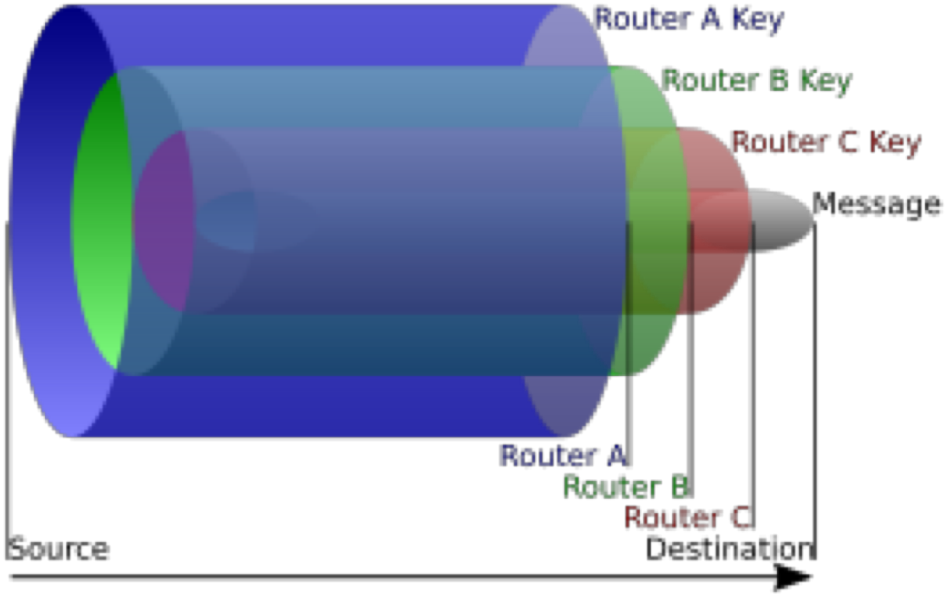
Hvordan virker TOR



Figur hentet fra https://www.torproject.org/about/overview.html.en

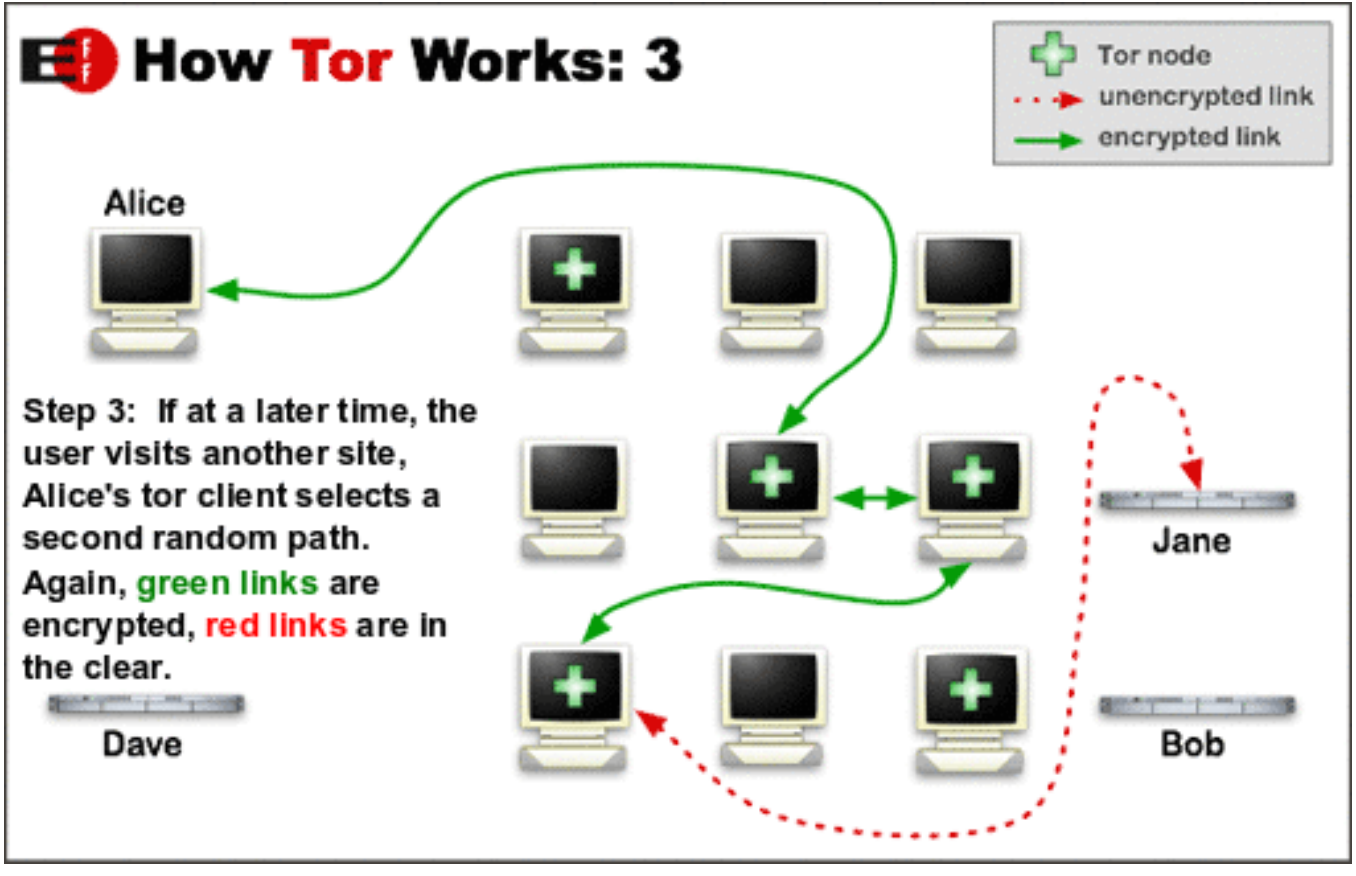
Hvordan virker TOR

- Flere lag med kryptering beskytter metadata (som avsender og mottakeradresse)
- Hver router vet bare hvem de har fått pakken fra, og neste router i kjeden.
- Siste hopp er ikke kryptert (for vanlige tjenere)
- Kan sette opp en tjener til å koble til kryptert, da blir innholdet beskyttet ende-til-ende.



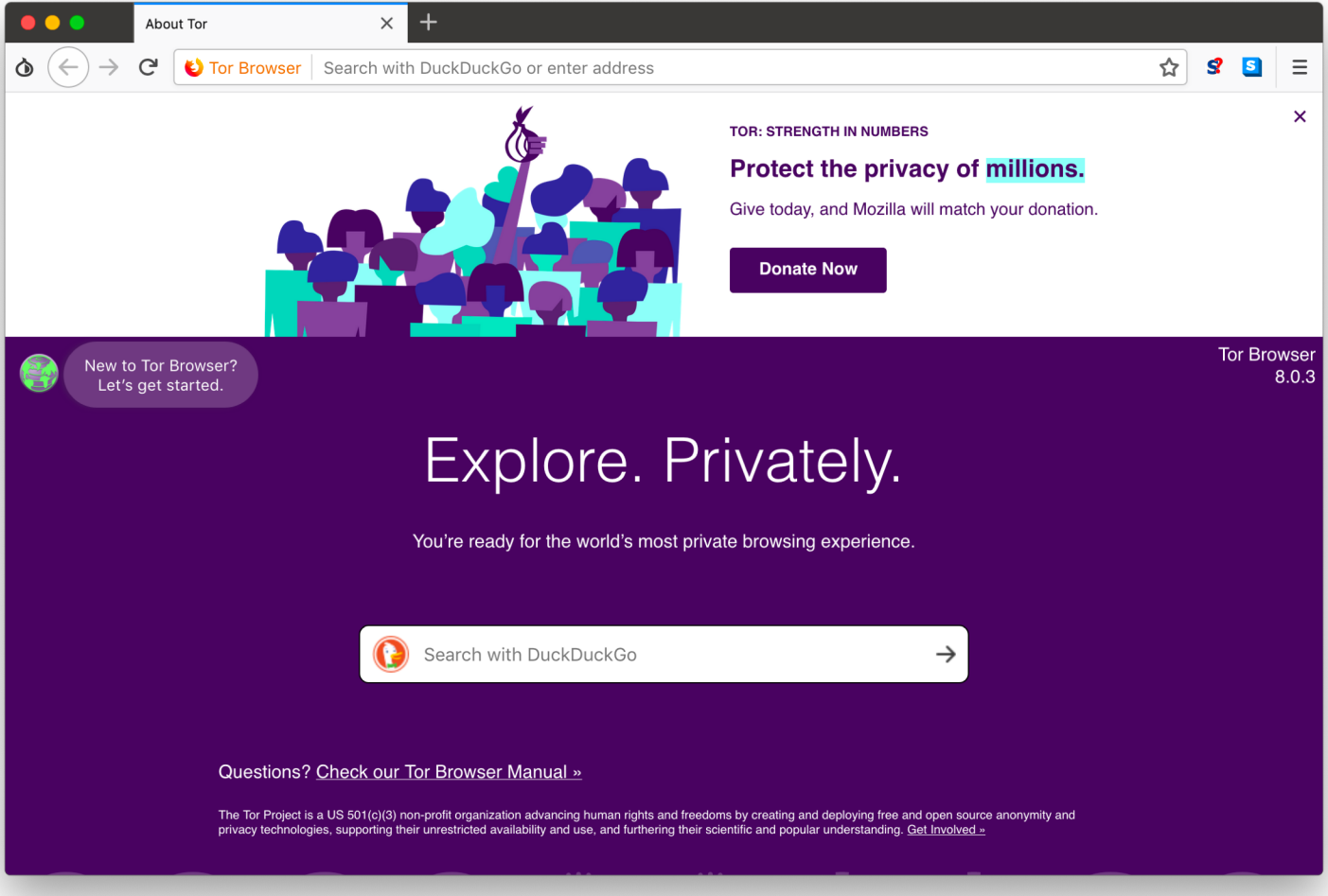
Figur hentet fra https://www.torproject.org/about/overview.html.en

Hvordan virker TOR



Figur hentet fra https://www.torproject.org/about/overview.html.en

En titt på TOR



The deep web / the dark web

Figur hentet fra https://medium.com/@smartrac/the-deep-web-the-dark-web-and-simple-things-2e601ec980ac

Hvordan avdekke kriminelle TOR-brukere (eksempler)

- Honeypots, phishing og ZeroDay
  - Prisen for et zerodayangrep? USD 50.000 – USD 1.500.000[1]
  - Nylig eksempel: Politiet satte opp falske kriminelle distribusjonssider i Australia[2]
  - Tjeneren avslørte seg da den tok kontakt for å laste et profilbilde uten å gå via TOR[2]
- Statistisk analyse av pakker som går inn og ut av nettverket.
  - Forutsetter kontroll over begge endepunkter
  - Pakkestørrelser, avstand mellom forespørsler, antall pakker pr. forespørsel osv.

1) https://www.cyberscoop.com/zero-day-vulns-are-rarer-and-more-expensive-than-ever/
2) https://www.vg.no/spesial/2017/undercover-darkweb/

Ressurser

- Bøker og artikler:
  - Tanenbaum, Andrew S. "Computer Networks". Prentice Hall PTR
  - James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach
  - Internet latency: “B. Briscoe *et al*., "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2149-2196”
  - P2P Networking and Applications (Morgan Kaufmann Series in Networking (Hardcover)): John Buford, Heather Yu, Eng Keong Lua: 9780123742148: Amazon.com: Books

- Underholdende lesing: https://twitter.com/internetofshit
- https://github.com/kgryte/awesome-peer-to-peer
- https://en.wikipedia.org/wiki/BitTorrent