

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN1020 - Introduksjon til datateknologi

Forelesning – 02.11.2021

Tjenester i Internett

Håkon Kvale Stensland

simula

Plan for ”nettverksdelen” av IN1020

- 31. august – Datanett – En liten introduksjon
- 23. september - Introduksjon til operativsystemer
- 21. oktober – Historie, mer om lagdeling og protokoller
- 26. oktober – Hvordan fungerer din trådløse ruter?
- 28. oktober – Kryptering & Nettverkssikkerhet
- **2. november – Tjenester i Internett**
- *Uke 46* – Oppsummering i IN1020

Tjenester i Internett

- Repetisjon: Transportlaget
- Forbindelsesstrategier
- Aksessmodeller
  - Klient-tjener
  - Peer-to-Peer (P2P)
- World Wide Web (www): HTTP-protokollen
- Strømming av video
- E-post
- Internet of Things

Repetisjon:

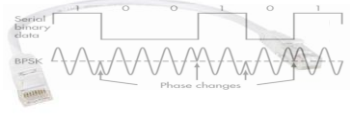
Lagene i Internett (TCP/IP referansemodellen)





Applikasjonslag	**http://www.uio.no**		Applikasjonslag
Transportlag	**192.168.1.5:80**		Transportlag
Nettverkslag	**192.168.1.5**	Nettverkslag	Nettverkslag
Linklag	**A1:B2:C3:D4:E5:F6**	Linklag	Linklag
Fysiske lag		Fysiske lag	Fysiske lag







Repetisjon: Lag 4 - Transportlaget

- TCP: HTTP, E-post, filoverføring, etc.
- UDP: Strømming av video og lyd

- Bruker «port» som en unik identifikator.
  - Representeres med et 16-bit heltall (65535 unike adresser)

TCP Segment Header Format

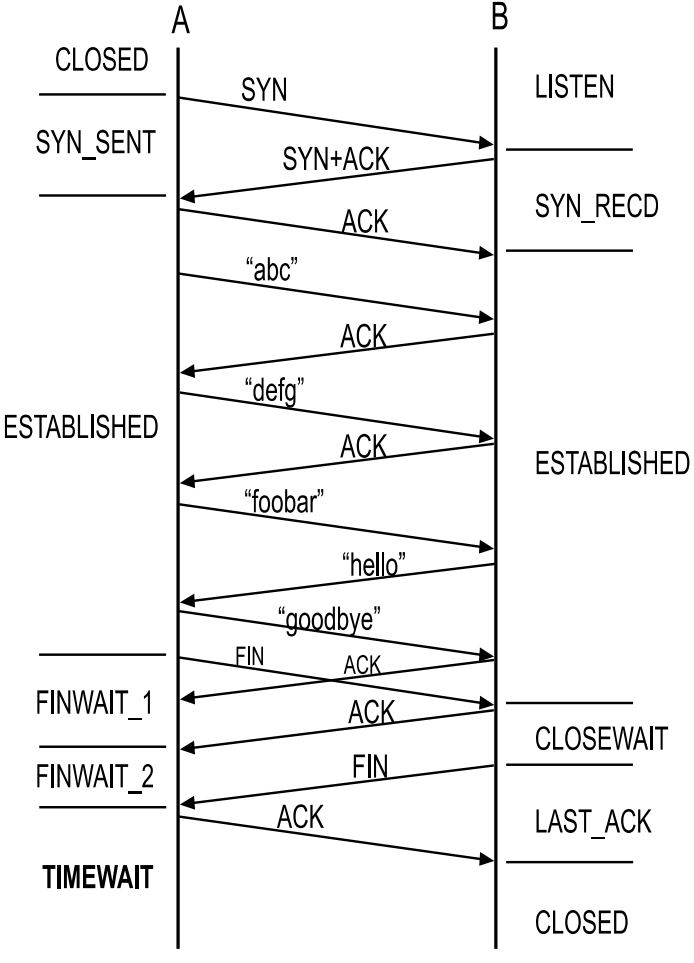
Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset	Res	Flags	Window Size
128	Header and Data Checksum		Urgent Pointer	
160…	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

Repetisjon: Lag 4 - Transportlaget

- **TCP:**
  - Oppsett av forbindelse (3-way handshake)
  - Pakkene leveres i riktig rekkefølge
  - *Pålitelighet* – Pakker sendes på nytt hvis kvitteringen (ACK) ikke kommer frem
  - Flytkontroll og metningskontroll
  - Feilsjekking av nyttelasten (sjekksum)
- **UDP:**
  - Tilkoblingsløs forbindelse
  - Ingen garantier, «Best-effort»
  - Feilsjekking av nyttelasten (sjekksum)



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.

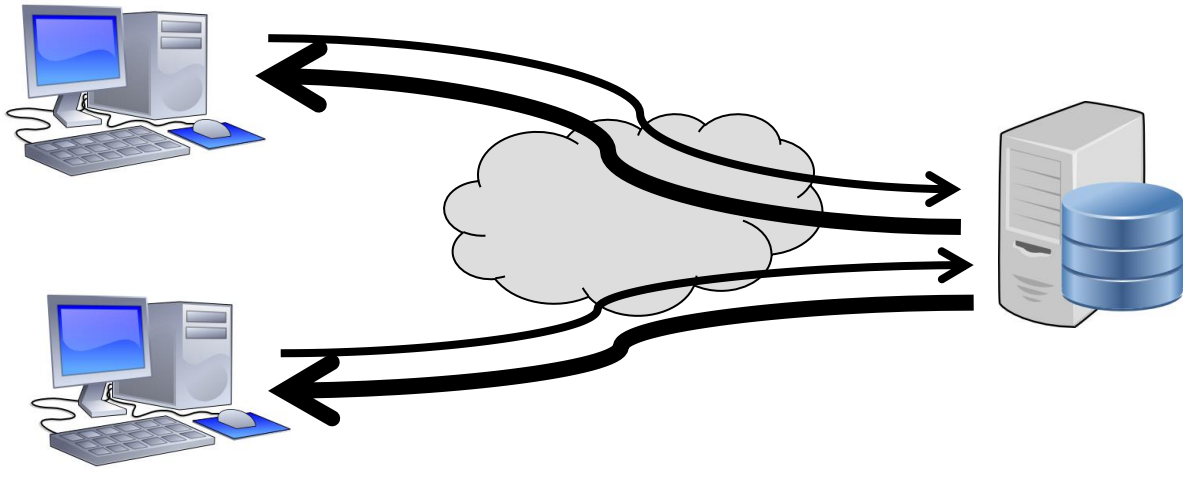
Publish-subscribe

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

Aksessmodeller: Klient-tjener

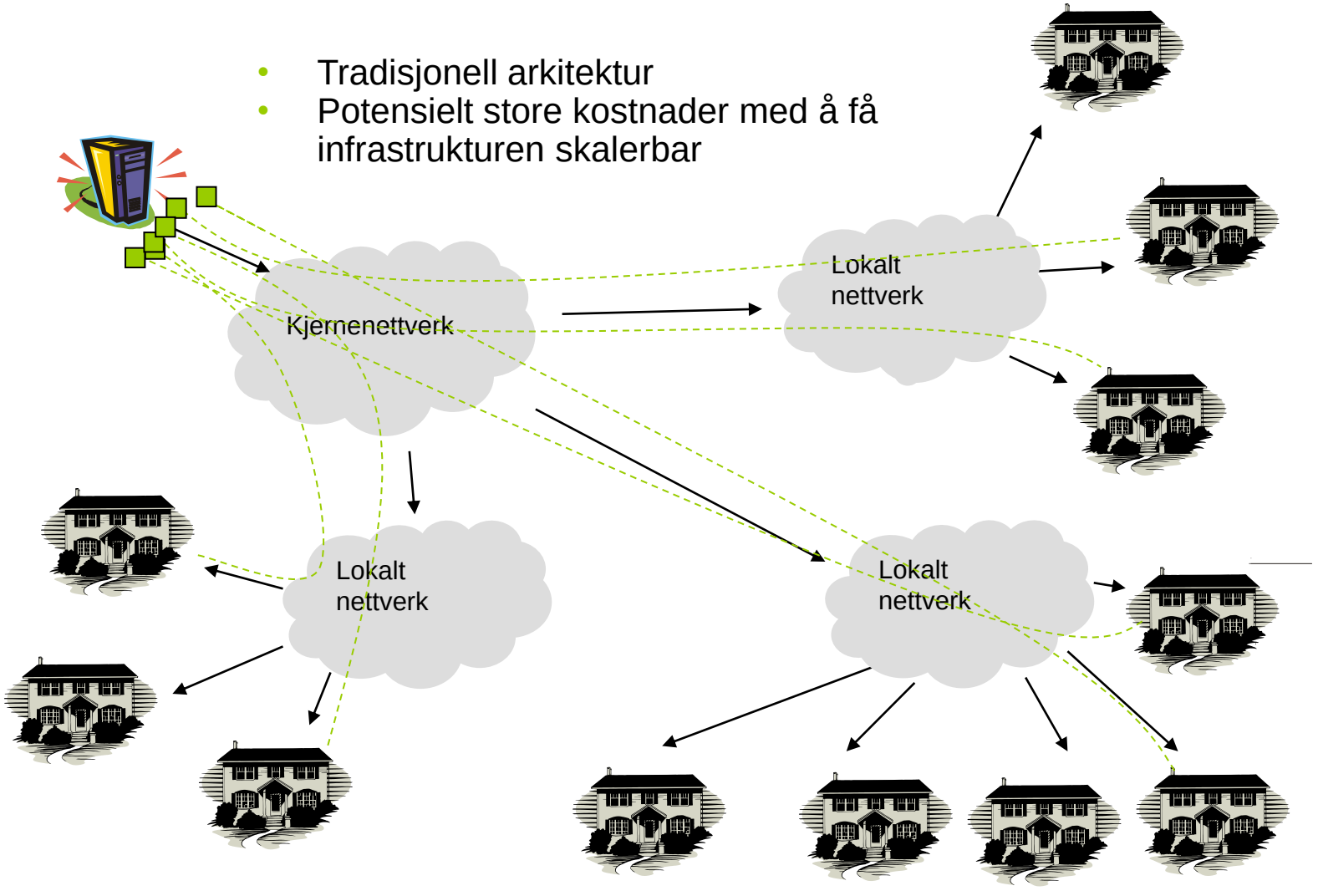
- Tradisjonell kommunikasjonsmodell, lettfattelig abstraksjon
  - Klienter ber om en tjeneste (oppretter en forbindelse)
  - Tjenere leverer tjenesten (svarer på forespørselen)

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



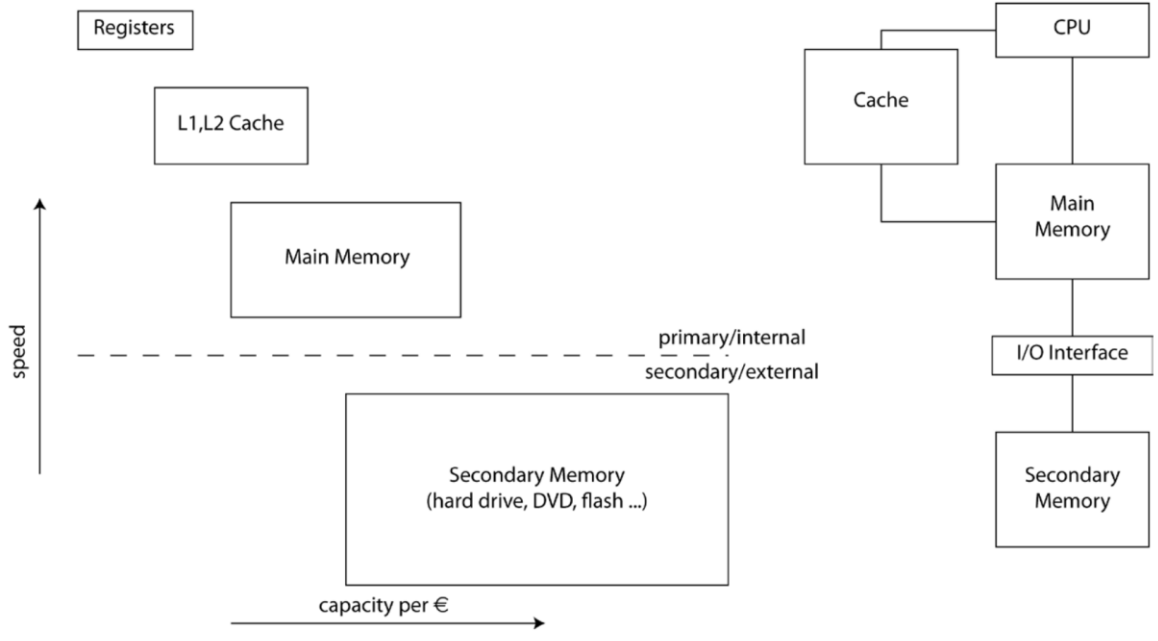
Klient-Tjener

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



Fysisk plassering av innholdet

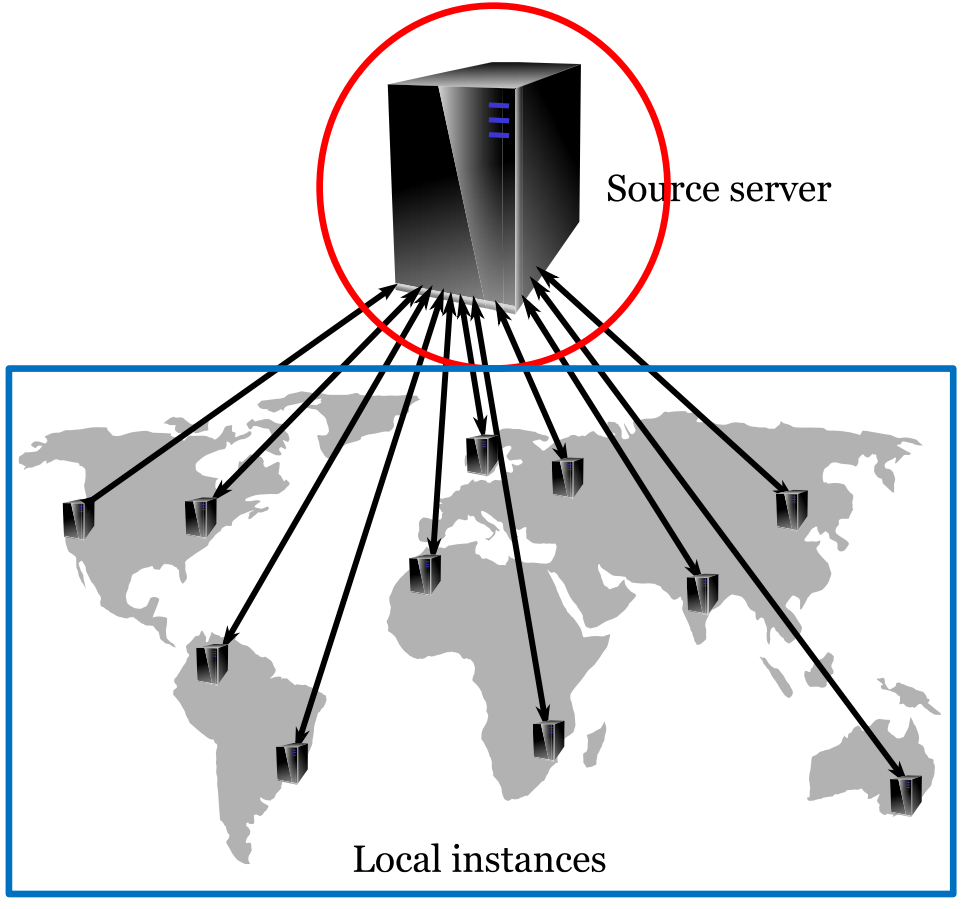
- Dere husker cachehierarkiet fra Omids forelesning?



Content Delivery Network (CDN)

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

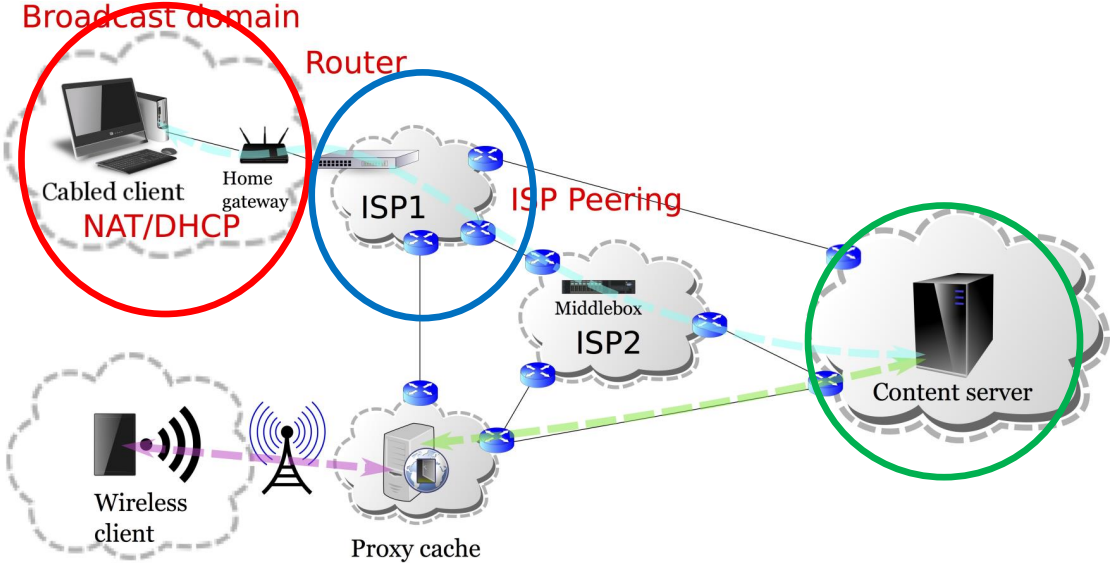
- Koster ekstra maskinvare og lagringsplass
+ Sparer kapasitet i backbone-nettet
+ Gir lavere forsinkelse på forespørsler
+ Hindrer overbelastning av tjeneren



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

Fysisk plassering av innholdet

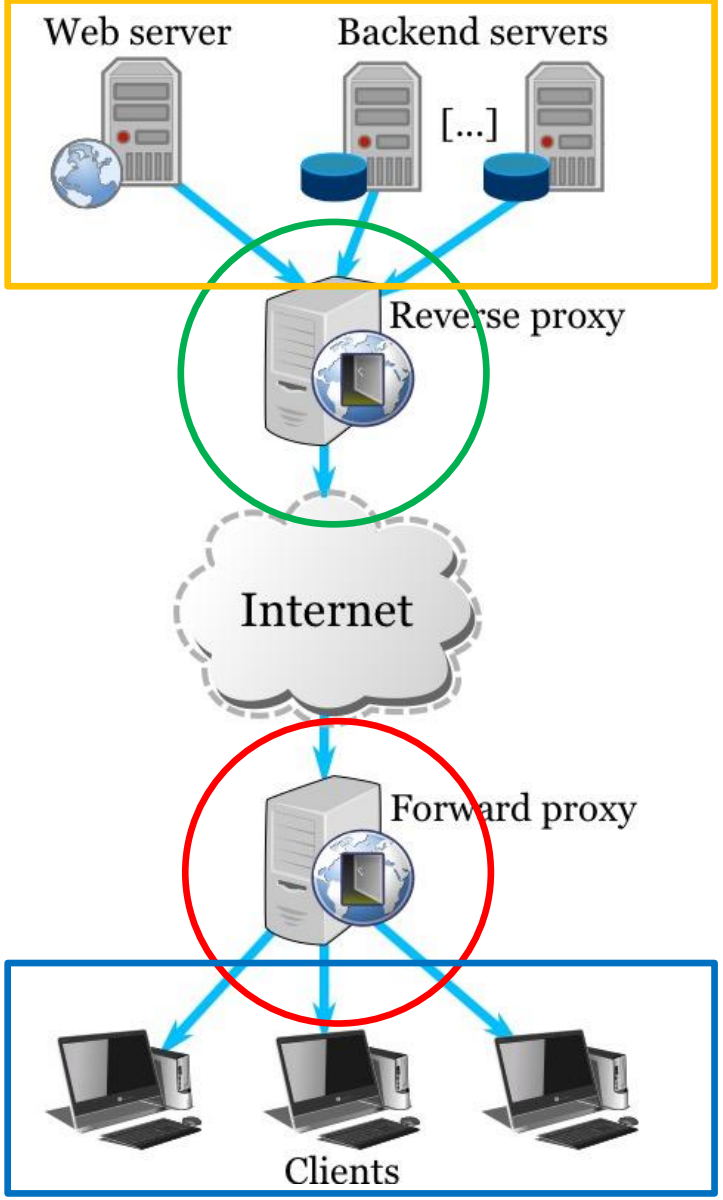
Cachenivå	Fysisk beliggenhet	Est. RTT
Lokal Proxy	Organisasjon / LAN	< 10ms
Content Delivery Network	ISP	< 50ms
Original datakilde	Internett	~10ms - ~250ms



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).

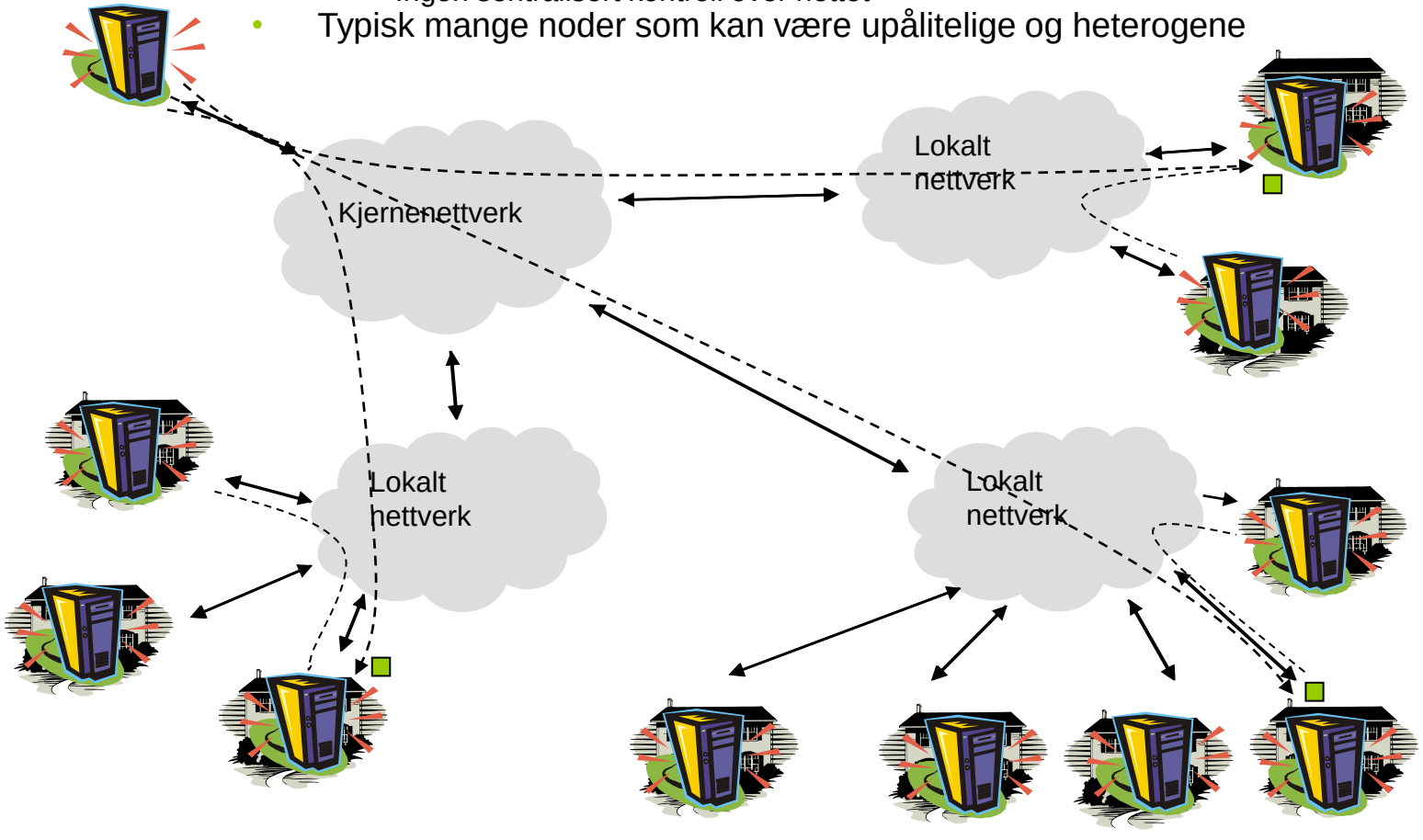
+ Lastbalansering
+ Sparer nettverkskapasitet
+ Lavere forsinkelse



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

Aksessmodeller: Peer-to-Peer (P2P)

- *Ikke en ny ide…- en distribuert systemarkitektur*
  - Ingen sentralisert kontroll over nettet
- Typisk mange noder som kan være upålitelige og heterogene

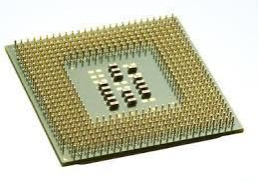


Peer-to-Peer

- **På flere måter likt CDN:**
  - Endesystemer kan være både klient og tjener
  - Distribuert lagring
  - Innholdet nærmere sluttbruker
  - Begrenset lagring per endesystem
- **Men ulikt på noen punkter:**
  - Ingen klare roller på endesystemene
  - Ikke noe hierarkisk forhold
  - Klienter vet nødvendigvis ikke hvor innholdet er
    - Trenger protokoller for å «oppdage innhold»
  - Endesystemer i et P2P nettverk kommer og går

P2P – Styrker og svakheter

- Ideelt til å dele ressurser mellom enheter (peers)
- Ressurser som dele kan være:
  - CPU (SETI@home, Folding@home osv.)
  - Lagring/harddiskplass (distribuerte databaser, distribuerte hashtabeller)
  - Nettverk/lagring (bittorrent)
  - CPU/lagring/nettverk (Bitcoin)
  - Eller nesten hva som helst…
- Tilgangen/fordelingen av ressurser styres av algoritmer.
- Robust mot endringer i nettverket av peers





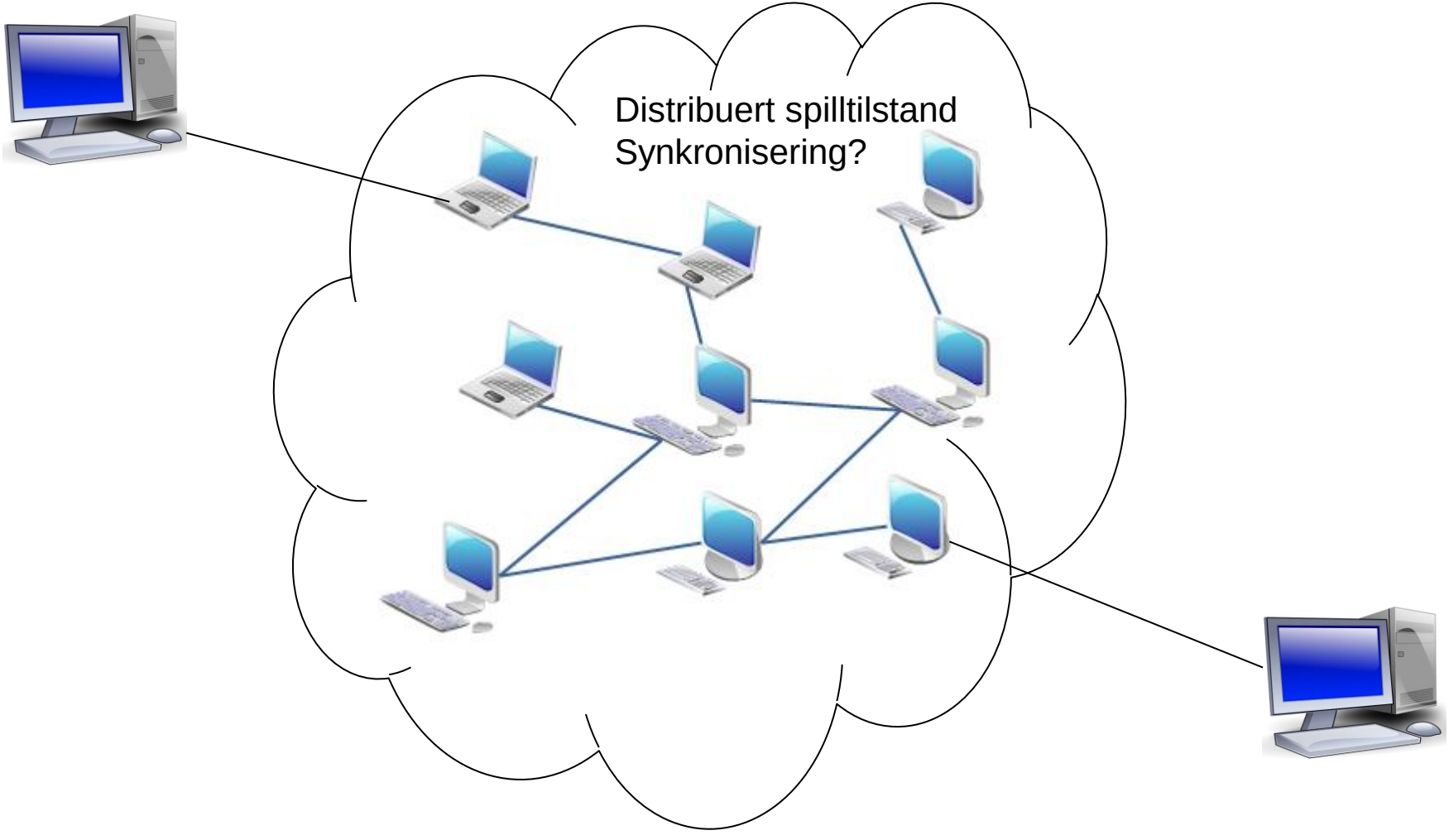
P2P – Styrker og svakheter

- Egner seg dårlig til applikasjoner som er avhengig av lav forsinkelse.
  - Onlinespill med høyt tempo (som First Person Shooters, f.eks. ”Battlefield”)
- Om man er avhengig av en viss grad av sentral kontroll.
  - P2P er vanskelig å temme når ”katta først er sluppet ut av sekken”
- Peers kommer og går
  - Mye redundans og ressurser kreves for å sikre kontinuitet
- Kan være vanskelig å effektivt indeksere og finne igjen elementer



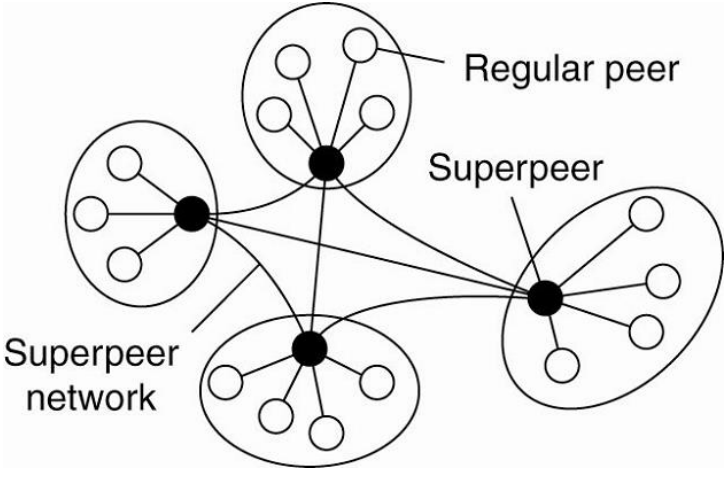


P2P – Styrker og svakheter



Varianter av P2P-baserte topologier

- Ren P2P:
  - Alle peers likeverdige
- Hybrid P2P:
  - Noen sentrale noder som har mer funksjonalitet enn andre
    - sentrale tjenere eller
    - ”superpeers”
  - Kan være valgt fordi:
    - De ikke står bak brannmur og/eller NAT
    - De har mye av en ressurs, f.eks. båndbredde, harddiskplass, CPU-regnekraft
  - Slike løsninger kan hjelpe til f.eks. å få ned forsinkelsen i en P2P-løsning



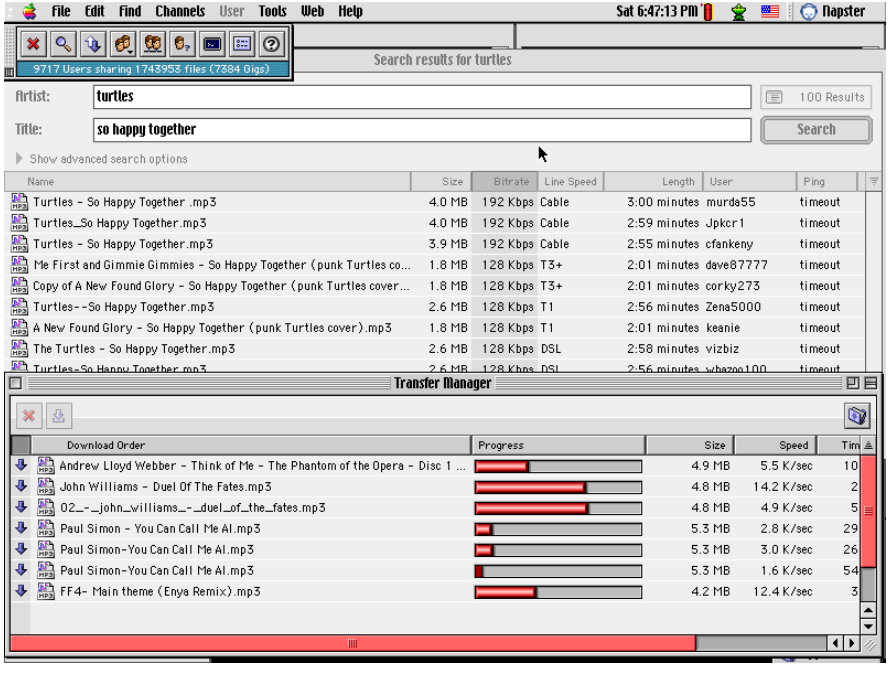


Eksempel: Napster (ikke pensum)

- Program for deling av filer (musikk) over Internett

- Arkitektur
  - Sentralisert index (katalog)
  - Distribuert lagring og nedlastning
  - Alle filer lastet ned blir delt
  - Lansert i 1999

- P2P aspekter:
  - Endesystemene var også filtjenere



Napster

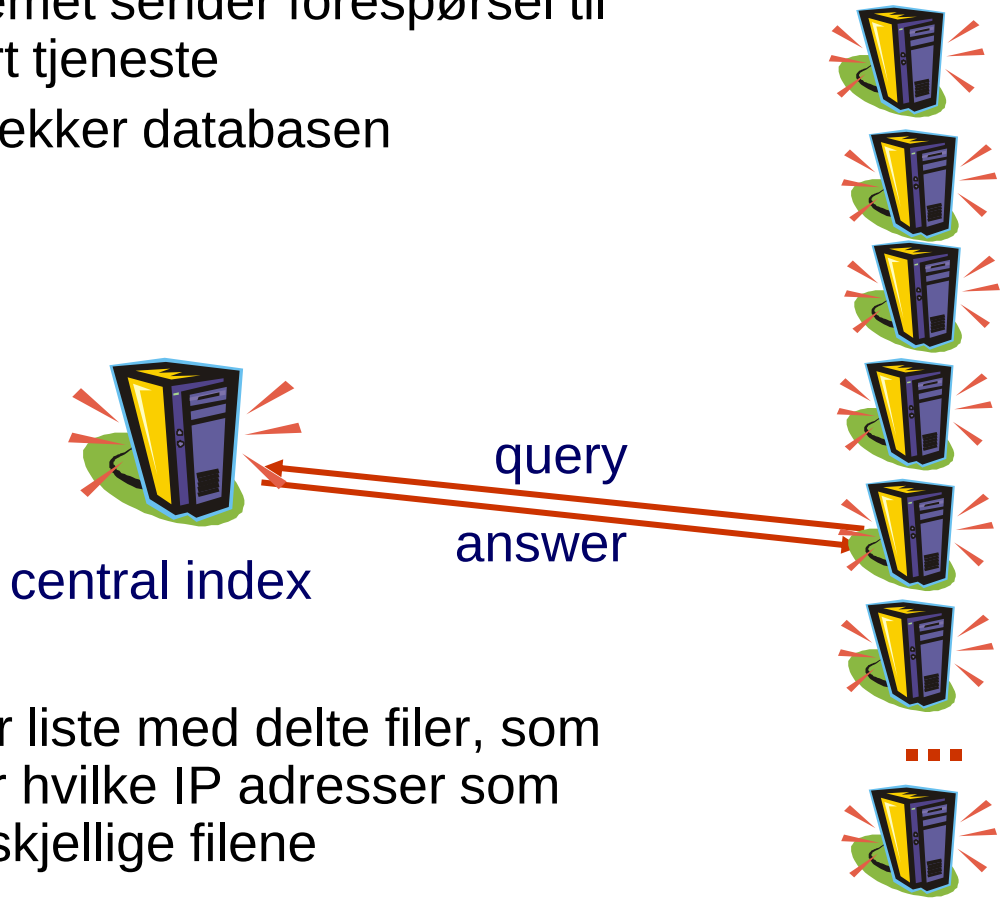
- Endesystemet kobler til Napster sin sentrale tjener.
- Sender over liste med delte filer



- Napster registrerer brukerinfo, knytter det til en IP adresse og lagrer listen med delte filer

Napster

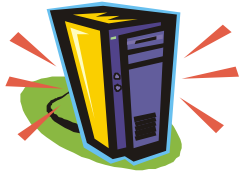
- Endesystemet sender forespørsel til sentralisert tjeneste
- Napster sjekker databasen



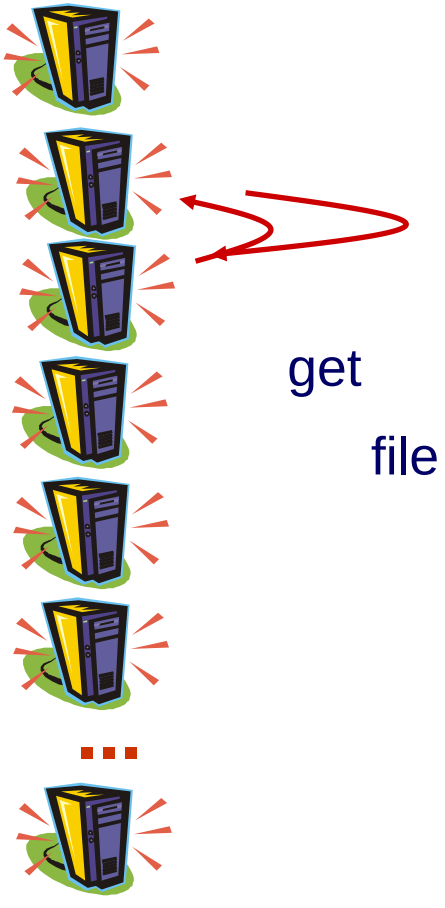
- Returnerer liste med delte filer, som inneholder hvilke IP adresser som har de forskjellige filene

Napster

- Endesystemet velger filen, og begynner å ta kontakt direkte for å lase ned filen
- Maskinene kontaktet hvis tilgjengelige.



central index



Napster: Evaluering

- Skalerbart:
  - Replikasjon av filer:
    - Antall kopier av populære filer stiger.
  - Liten tilgjengelighet for filer som ikke er populære.
  - Ignorerer fullstendig nettverkstopologien.
- Innholdslokasjon
  - Enkel og sentral søkemekanisme
- Pålitelighet
  - Ingen avhengighet mellom normale brukere.
  - Sentralisert server er «single-point-of-failure»
    - Ikke «advokat-sikkert», Napster ble stengt i 2001…

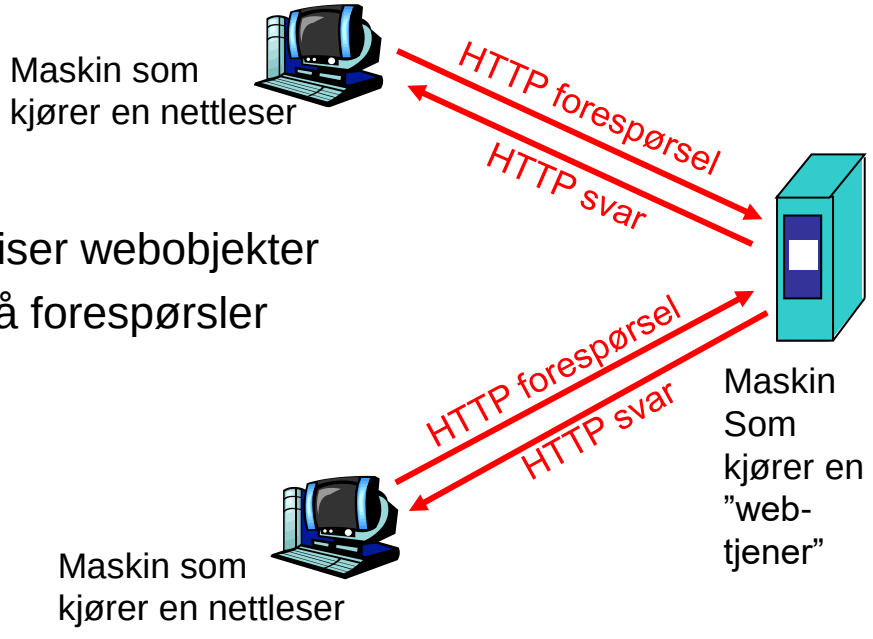


World Wide Web (www): HTTP-protokollen

HTTP: hypertext transfer protocol

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

- Fire hovedversjoner:
- HTTP/1.0 (1990)
- HTTP/1.1 (1999)
- HTTP/2 (2015)
- HTTP/3 (2018)



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.

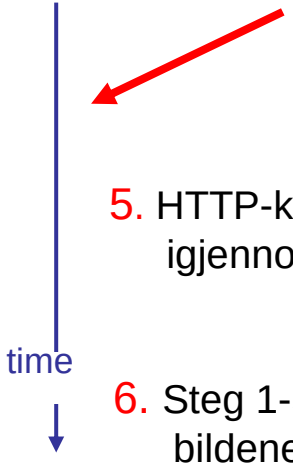
2. HTTP-klienten sender HTTP *request message* (med URL) til TCP-forbindelsen

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.)

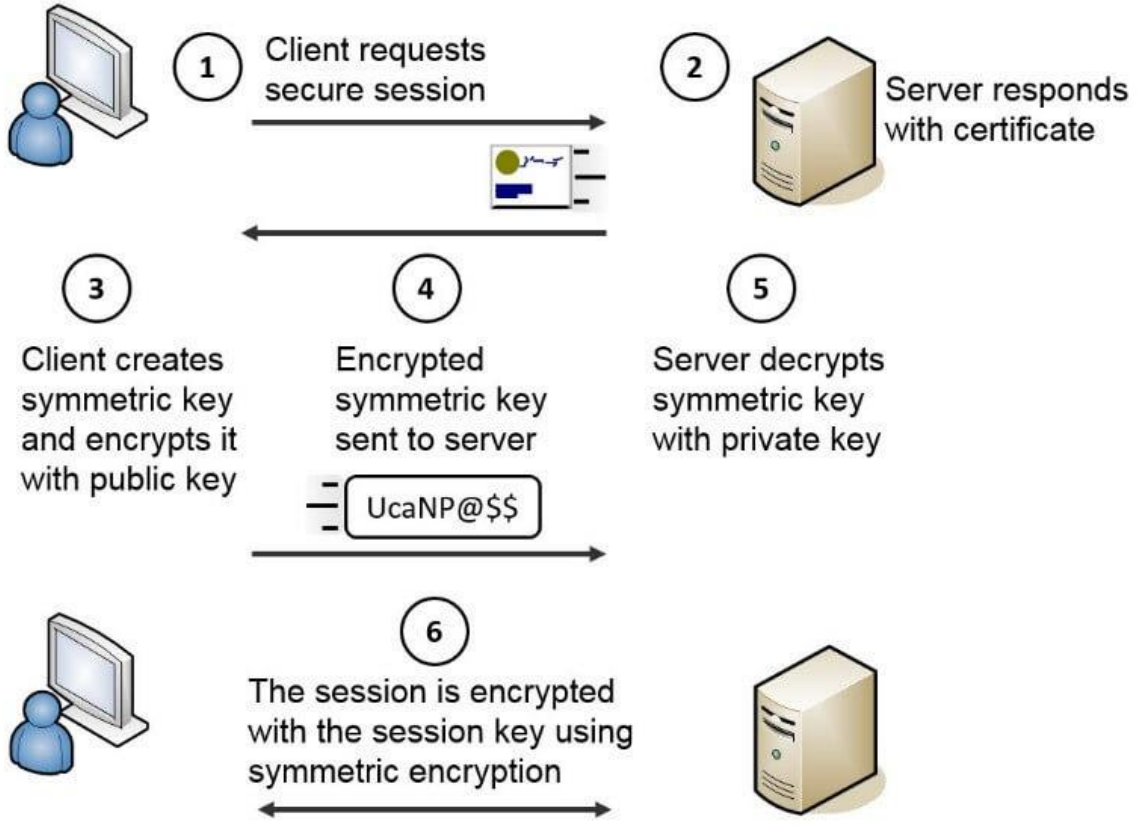


4. HTTP-tjeneren lukker TCP-forbindelsen.

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 (kryptering)



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)

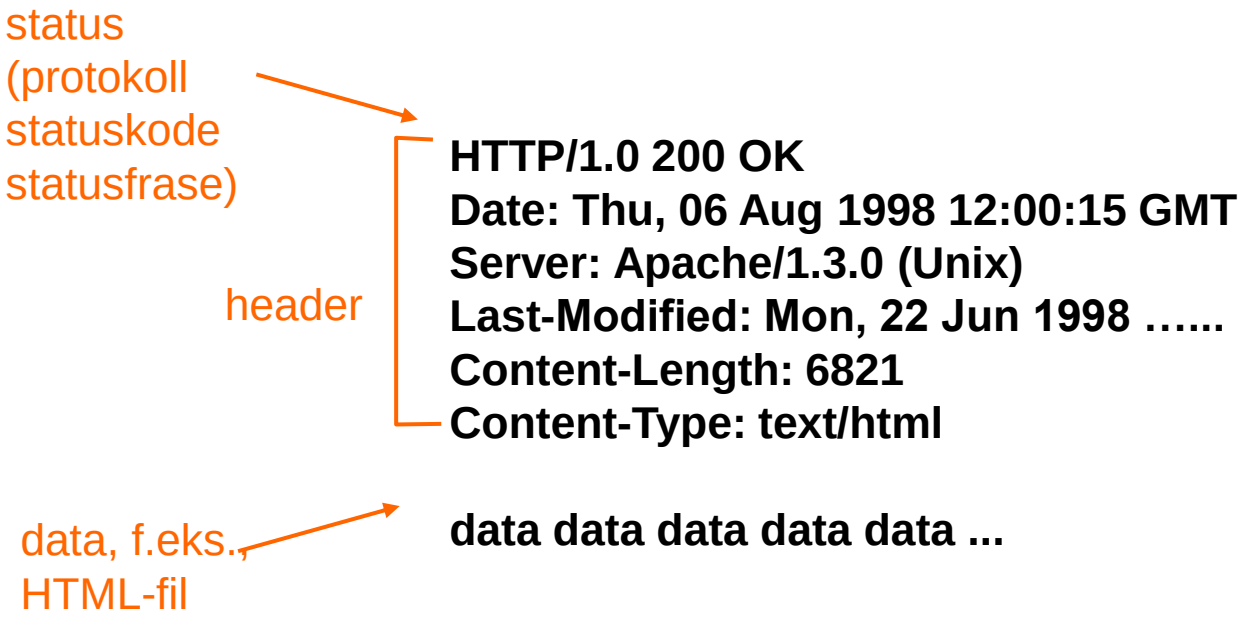
header

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

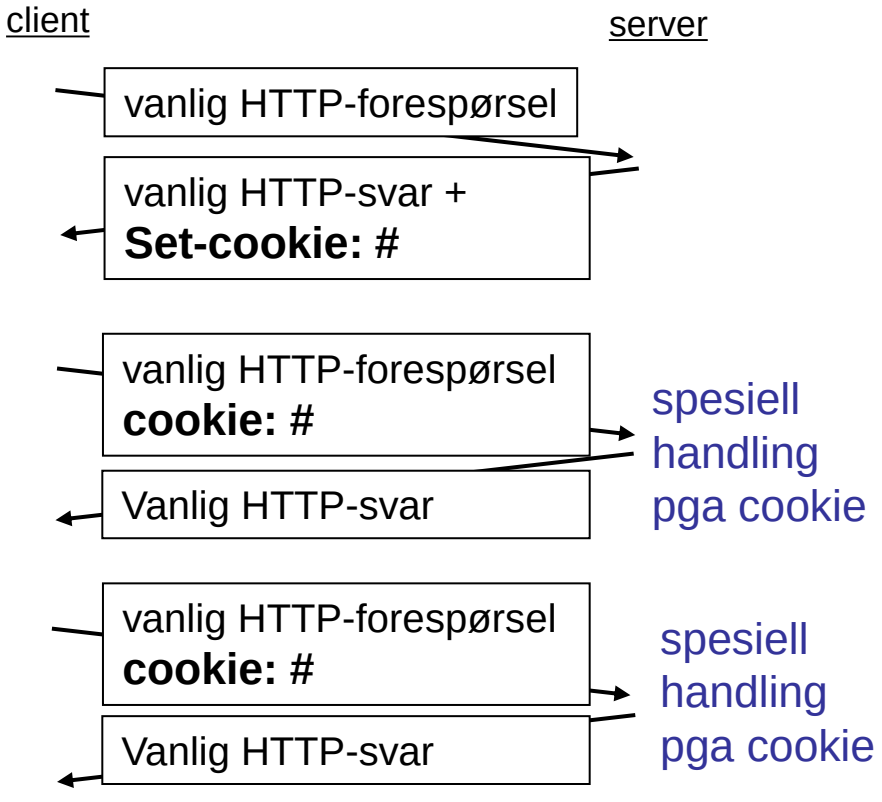
505 HTTP Version Not Supported

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

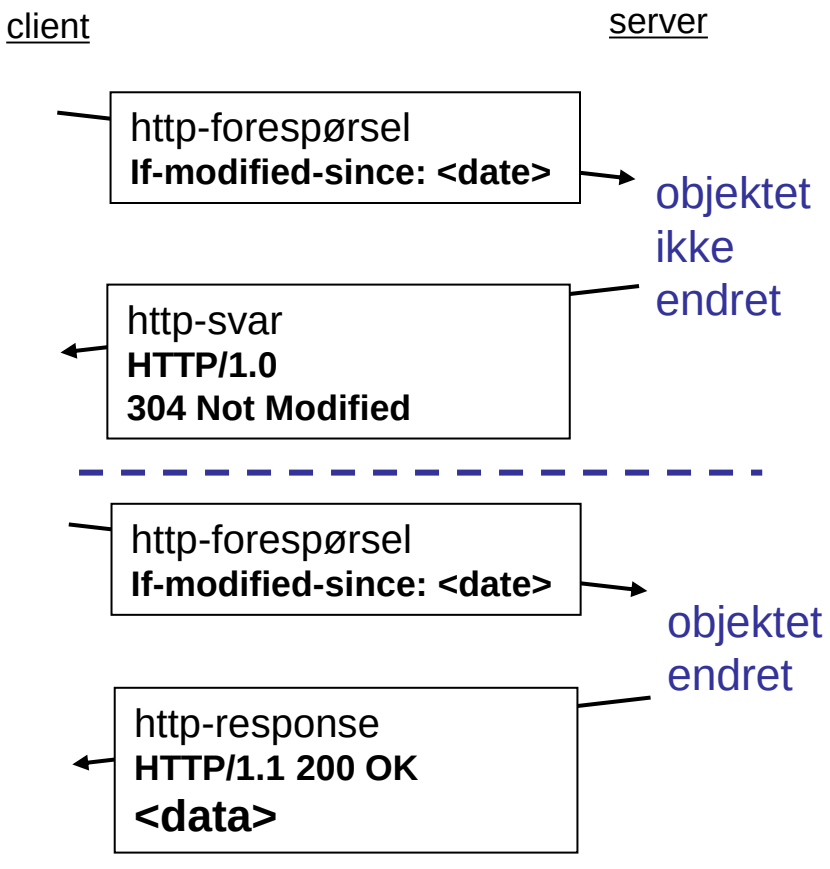
  Set-cookie: 1678453
- Klienten legger ved cookien med etterfølgende forespørsler

  cookie: 1678453

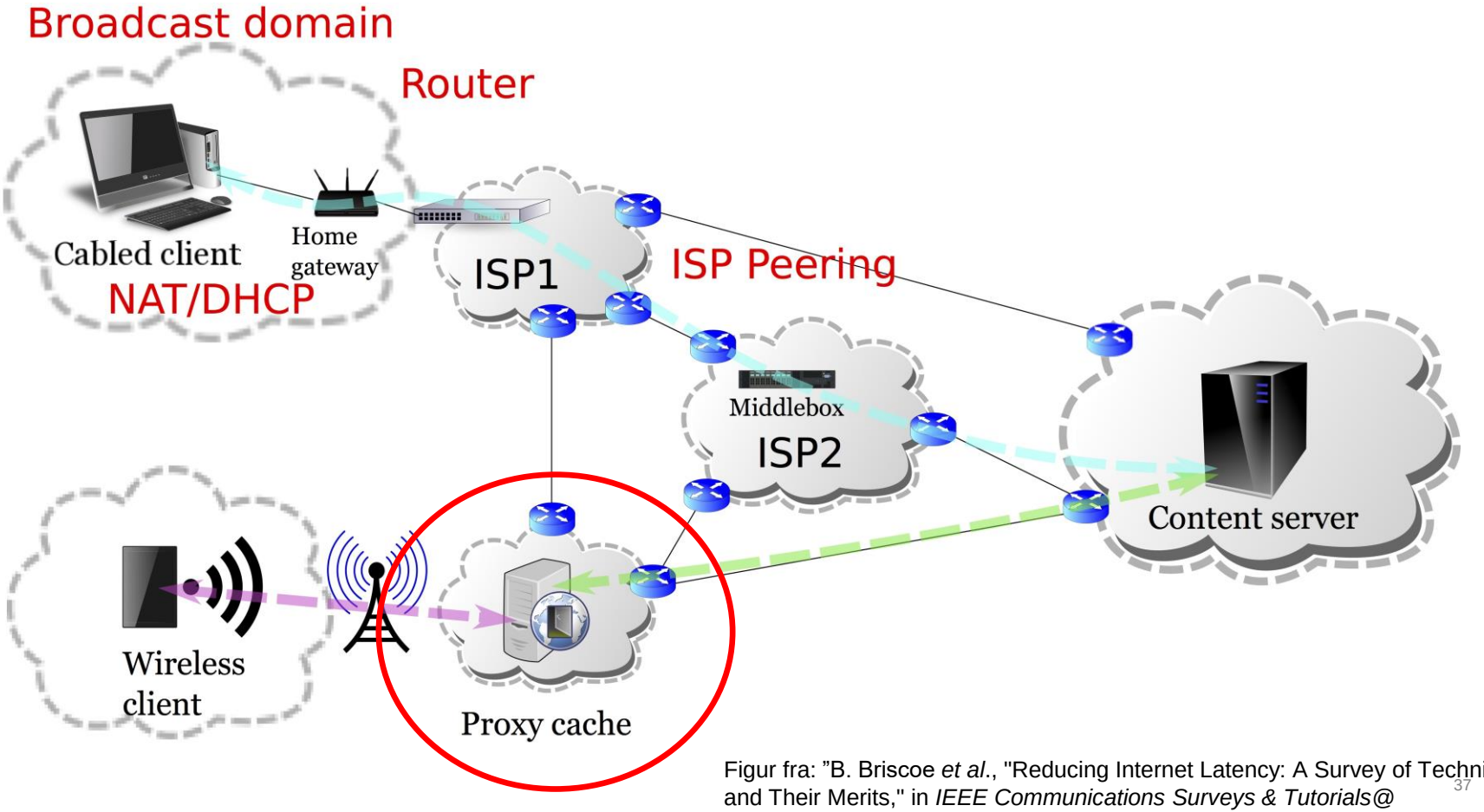


Betinget GET: klient-side caching

- 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



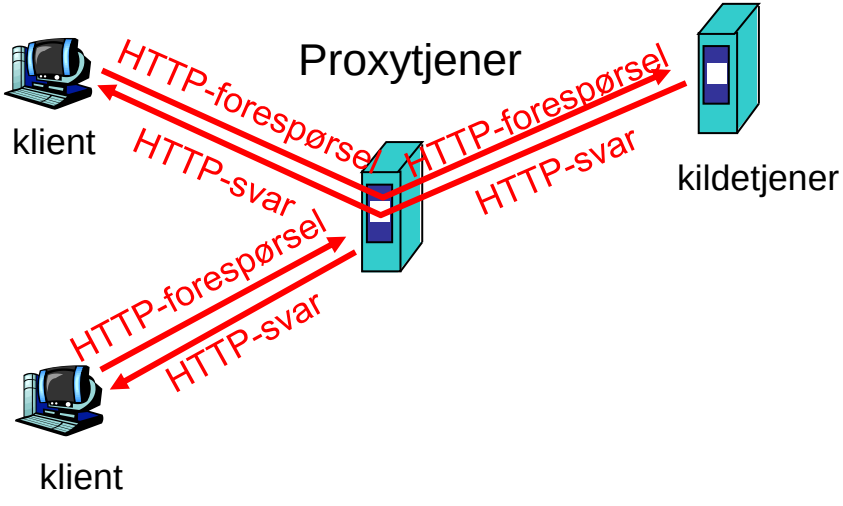
Figur fra: ”B. Briscoe *et al*., "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials@*

HTTP Proxytjener

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

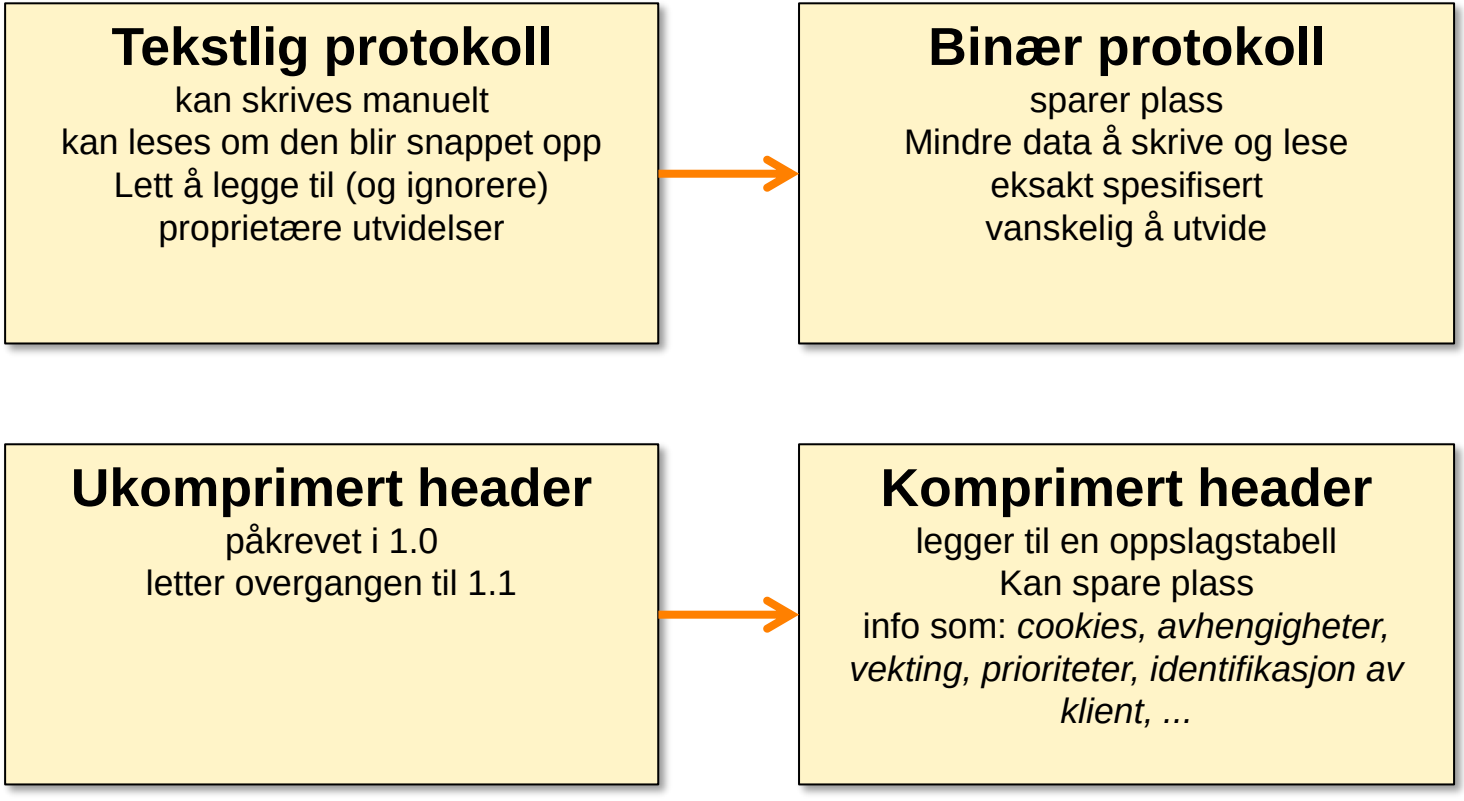
- Brukeren konfigurerer nettleseren: Web skal gå via proxytjener

- 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
antakelse: tjenern 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

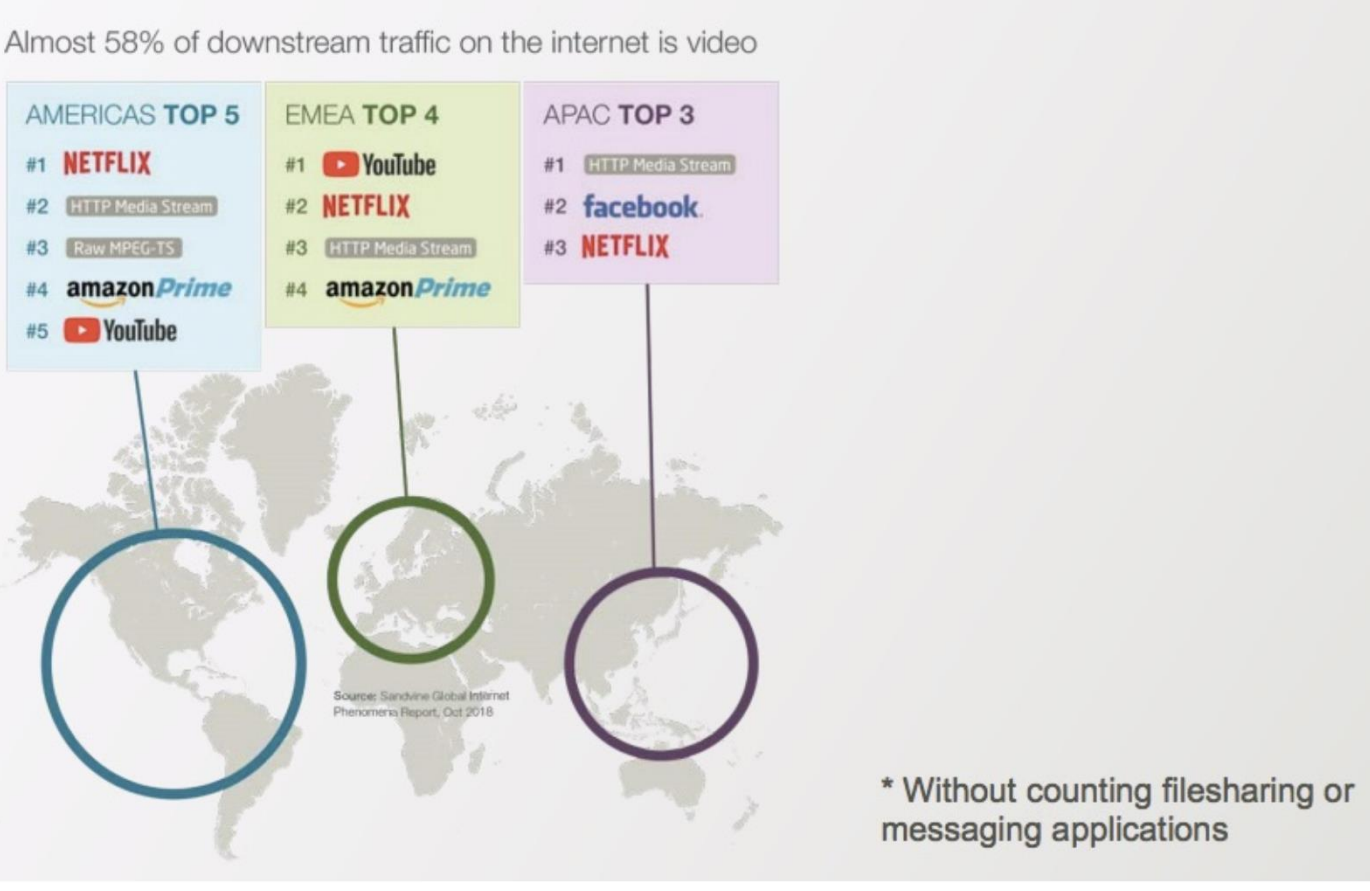


DNS Cache

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

Videostreaming



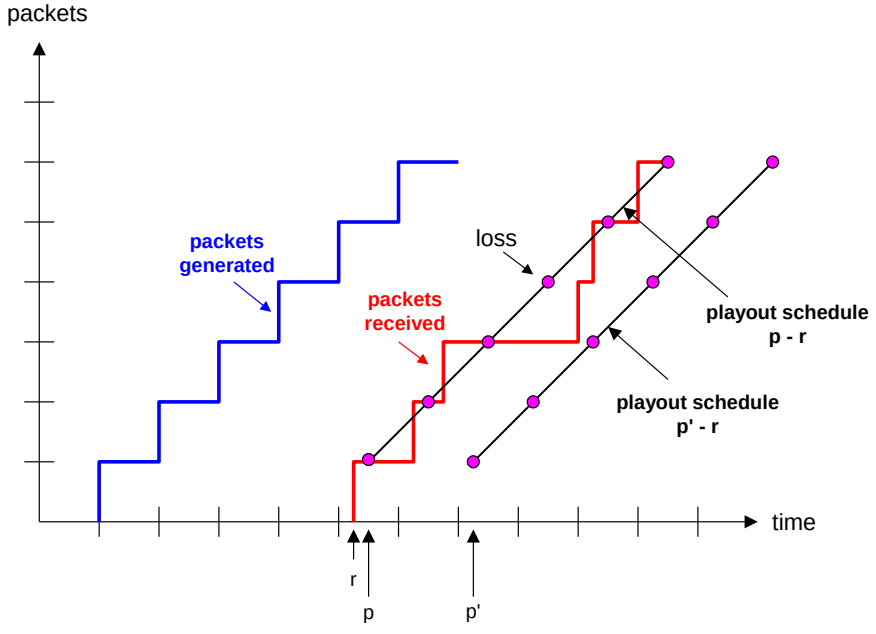


Kilde: Sandvine - http://digg.com/2018/streaming-video-worldwide - 4.10.2019

Videostreaming - utfordringer

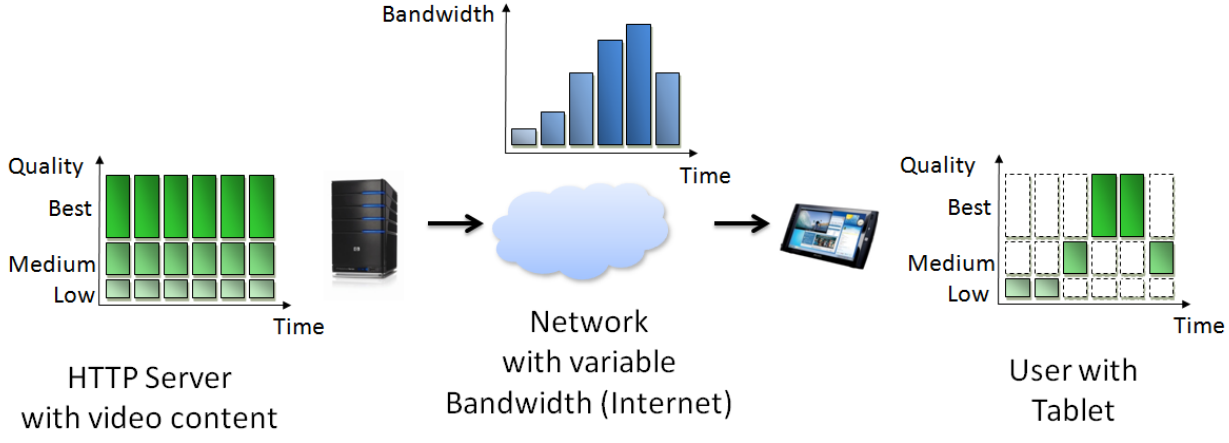
Multimedia

- 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

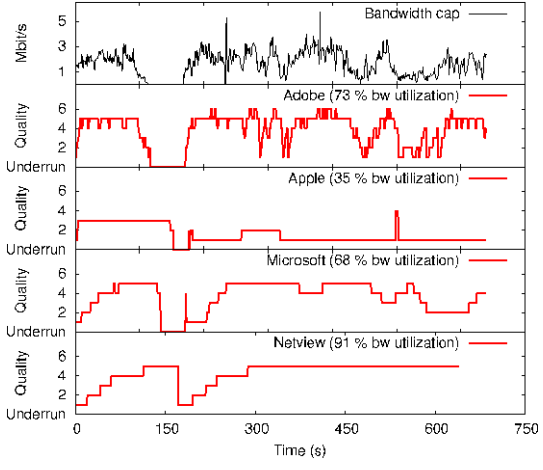


HTTP-streaming

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 tilpasningsstrategi
  - 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

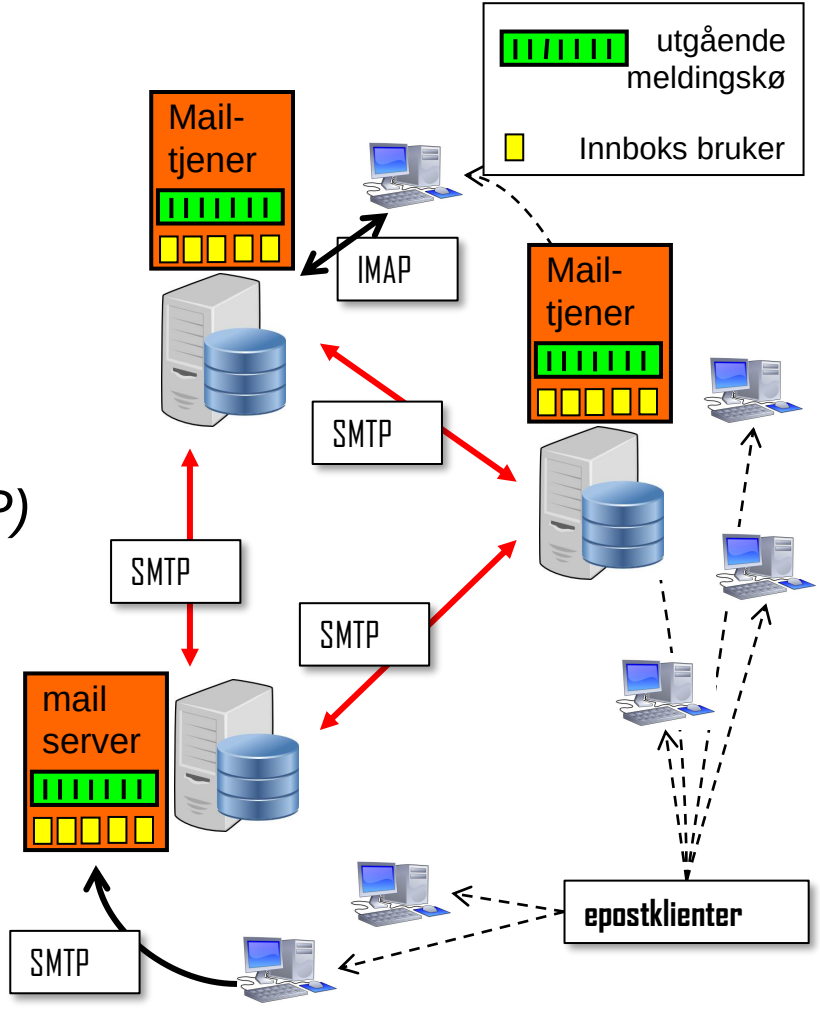
Epost: tjenere

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 pålitelig overføring av 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: 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.

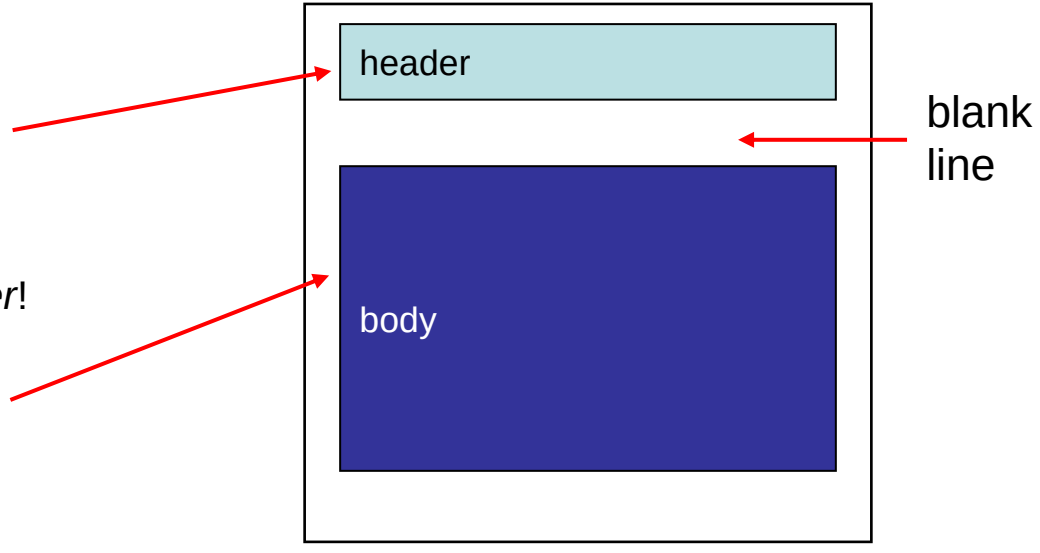
Meldingsformat

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

MIME-versjon

kodemetode

multimediadata type, undertype, deklarasjon av parametre

kodede data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

"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)

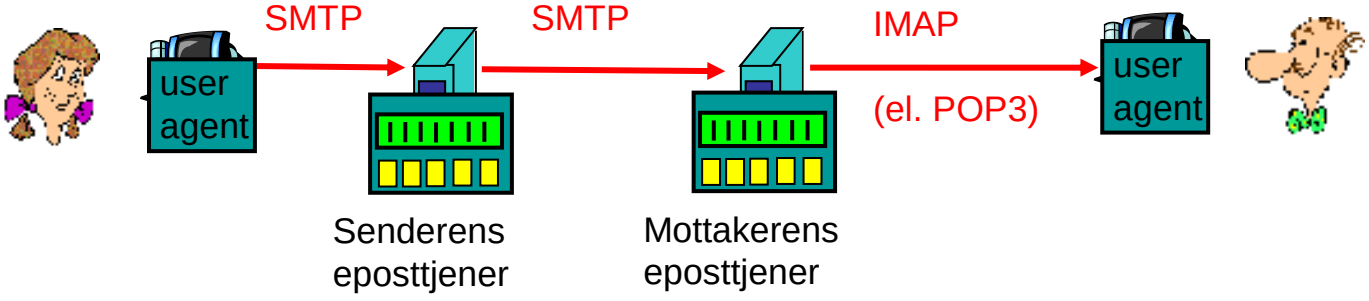
Video

- 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**

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: Hotmail , Yahoo! Gmail, etc.

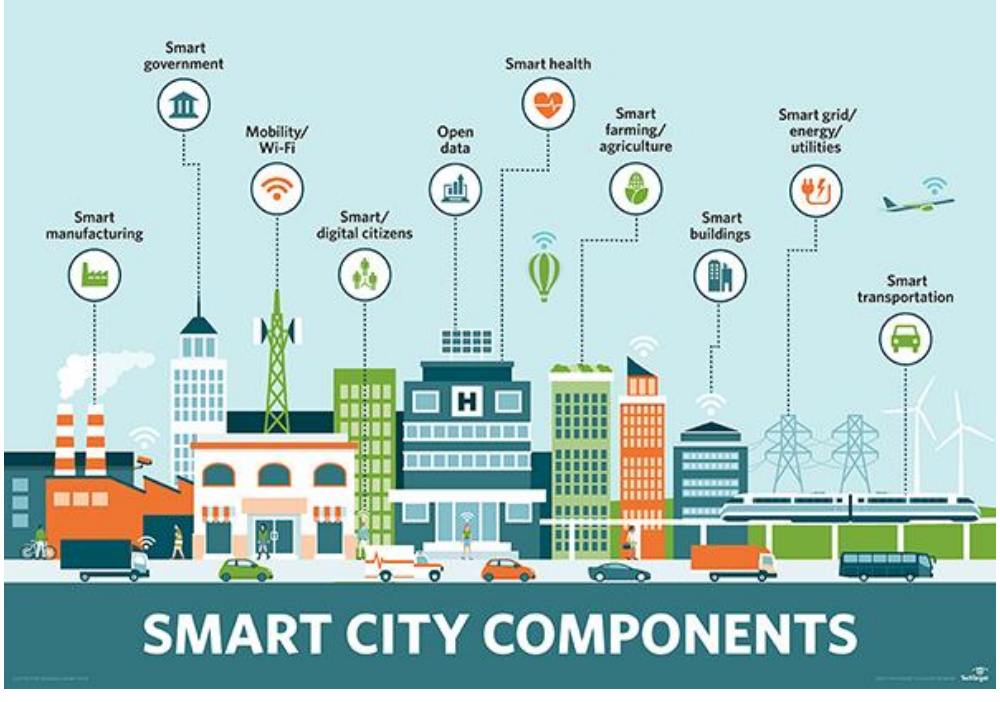
IMAP kan mer

- 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

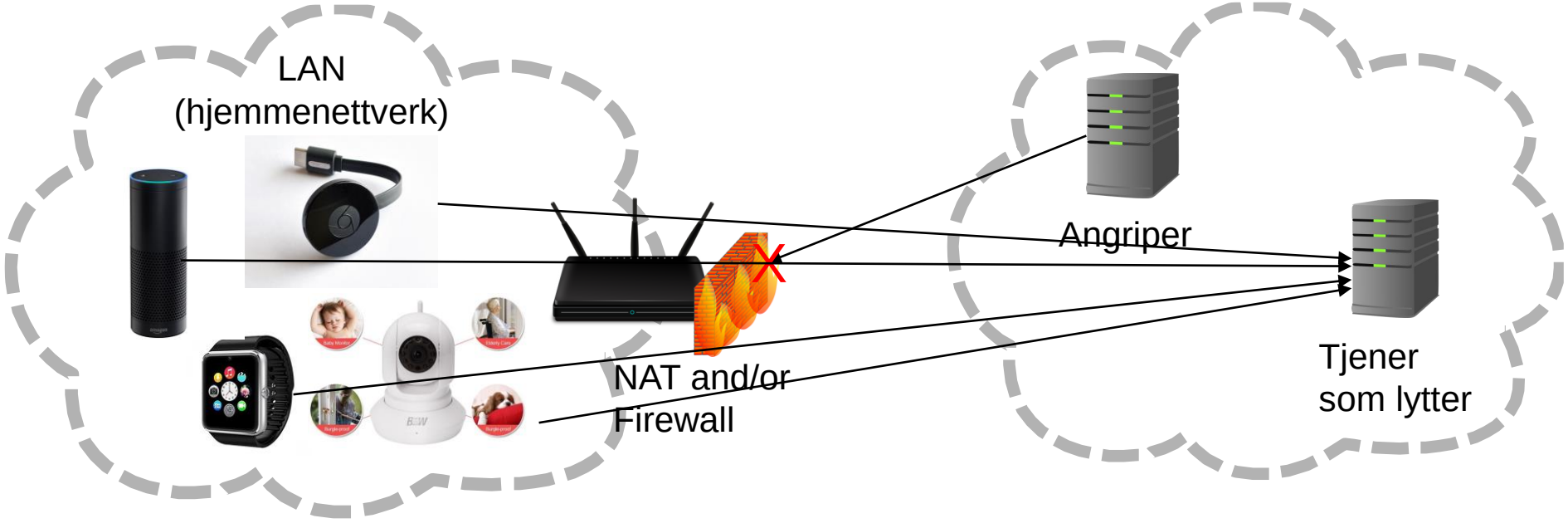
Internet of Things (IoT)

- 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



Viktig å huske at S'en i IoT står for sikkerhet… ;)

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”
- 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/

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”
- Underholdende lesing: https://twitter.com/internetofshit