

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN1020 - Introduksjon til datateknologi

Forelesning – 30.09.2026

Tjenester i Internett

Håkon Kvale Stensland



simula



Plan for ”nettverksdelen” av IN1020

- 28. august – Introduksjon til datakommunikasjon

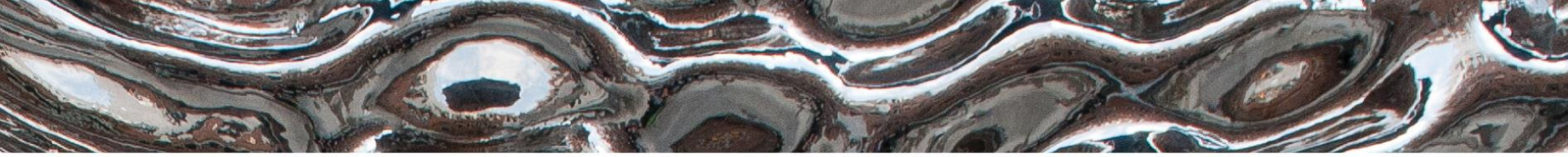
- 2. september – Lagdeling i datakommunikasjon
- 3. september – Hvordan fungerer din hjemmeruter?

- 9. september – Kryptografi: Nøkkelen til digital samhandling
- 11. september – Introduksjon til operativsystemer

- **30. september – Tjenester i internett**

Tjenester i Internett

- Transportlaget
- Applikasjonslaget:
  - Aksessmodeller & forbindelsesstrategier
  - DNS – Domain Name System
  - World Wide Web (www): HTTP-protokollen
  - Strømming av video
  - Internet of Things (IoT)



UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Transportlaget

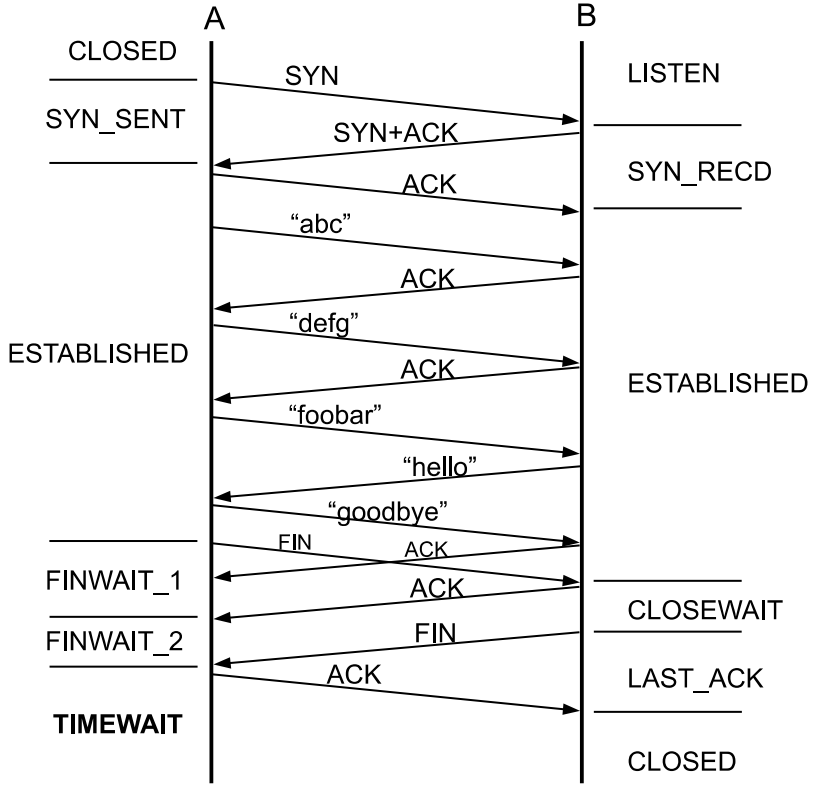


simula



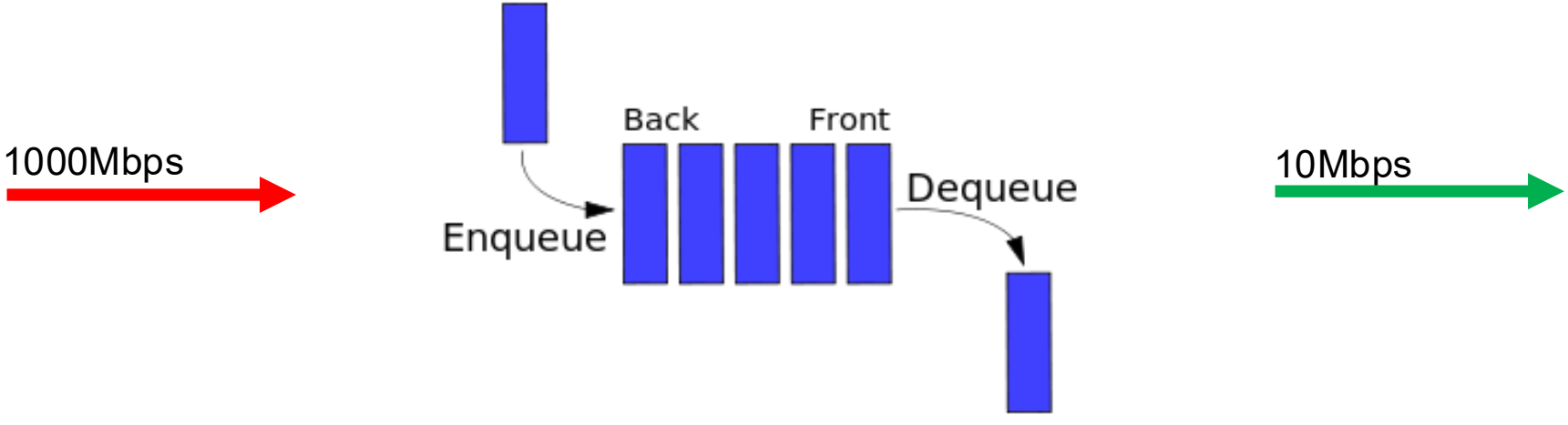
Transmission Control Protocol (TCP)

- Forbindelsesorientert
  - Settes opp ved et 3-veis-håndtrykk
    - SYN-SYN+ACK-ACK (se figur)
- Flytkontroll
  - Ikke sende fortere enn mottageren kan ta imot
- *Metningskontroll*
- Byte-strøm og levering i rekkefølge
- Pålitelighet
  - Implementert ved at bekreftelser på hver pakke sendes tilbake fra mottakeren
- Feilsjekking av nyttelasten (sjekksum)

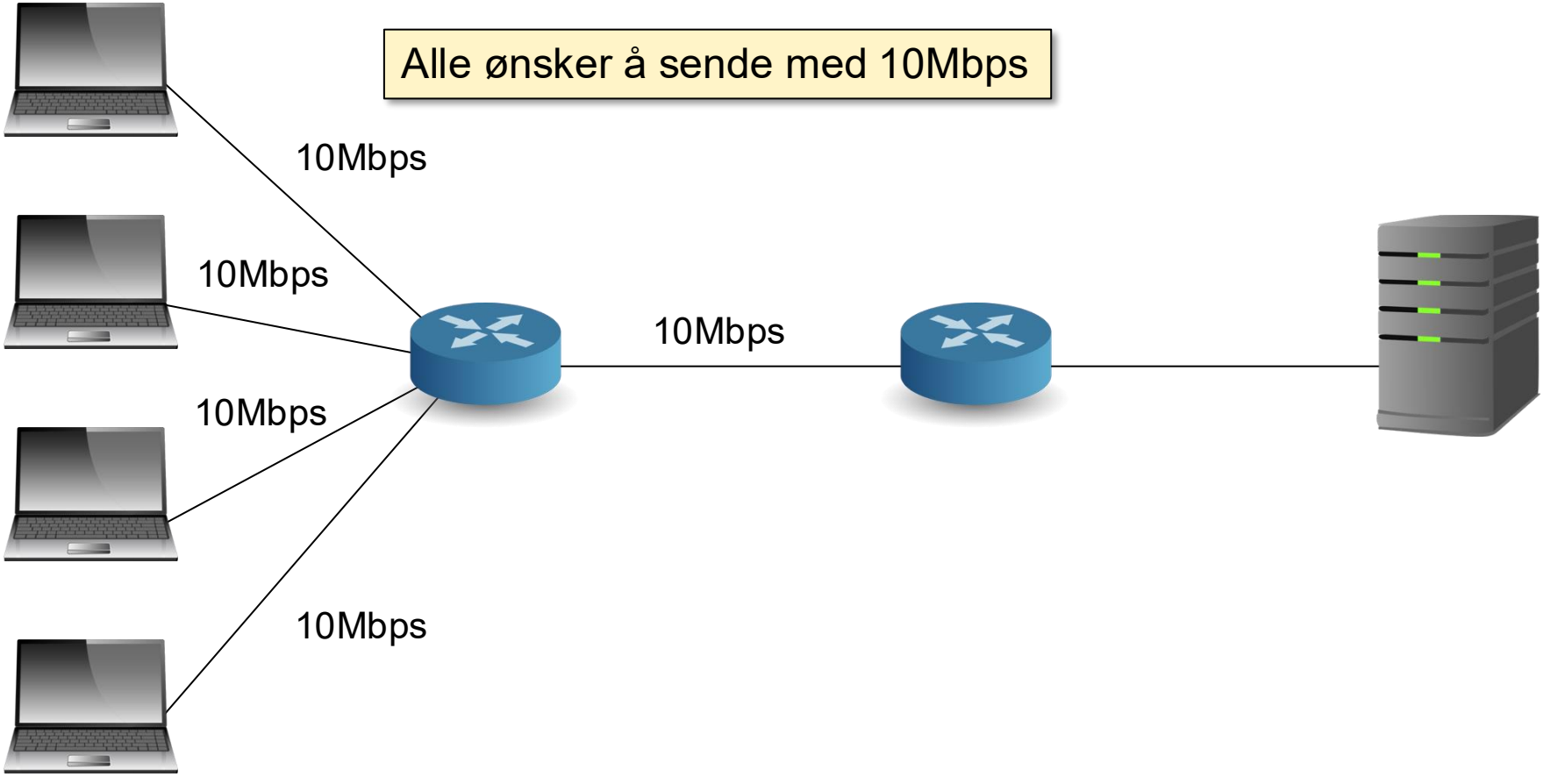


Metningskontroll

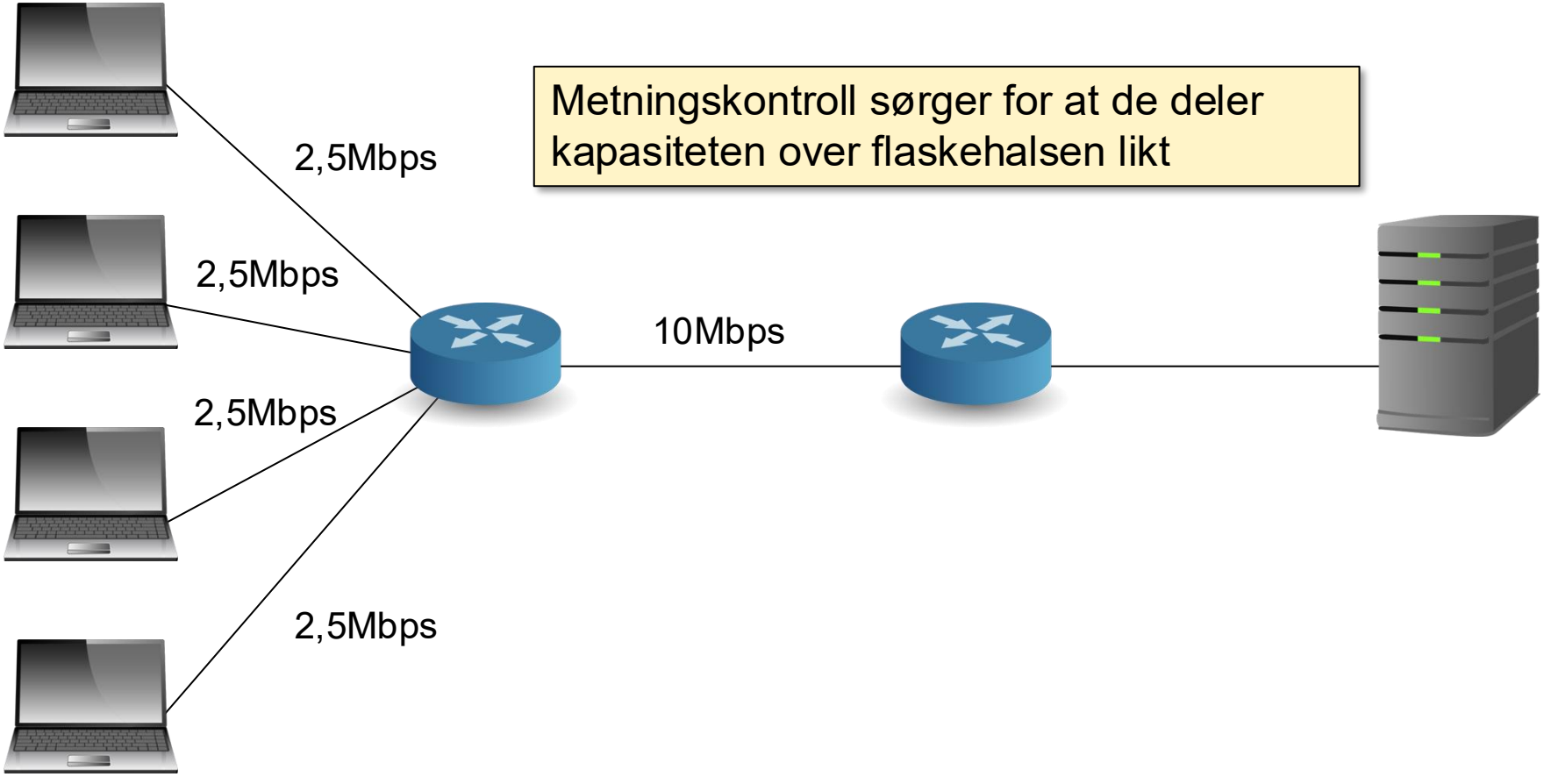
- En ruter er i utgangspunktet bare en FIFO (first-in-first-out) kø.
- Om det blir for mye trafikk over en gitt kø, vil det føre til stopp i trafikken. Dette kalles ”congestion”. Om det er så mye traffikk at det blir full stopp for alle, kalles det “congestion collapse”.
- For å unngå dette bygde man inn mekanismer i TCP for å tilpasse senderaten til nettverksforholdene.



fortsatt metningskontroll…



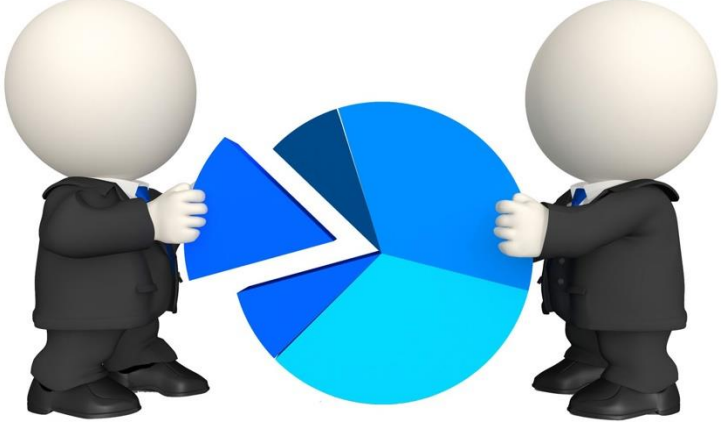
… ennå ikke ferdig med metningskontroll



Metningskontroll sørger for at de deler kapasiteten over flaskehalsen likt

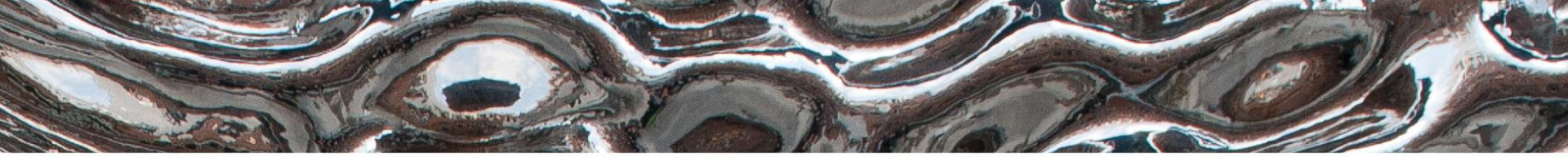
… stadig metningskontroll!

- Ressursene deles teoretisk sett likt mellom **forbindelser**
- For å kapre mer av kapasiteten: åpne flere forbindelser i parallell.
- Ingen god metode for å fordele ressurser i Internett rettferdig pr. maskin eller pr. bruker i dag.



Nyttige nettverksverktøy

Verktøy	
ping	Måler forsinkelsen mellom din maskin og en annen på internett
traceroute	Gir informasjon om hvert hopp en datapakke er innom på vei til målet.
tracepath	Samme som traceroute, men trenger ikke superusertilgang
netstat	Gir informasjon om nettverksforbindelser på din maskin
dig	Gjøre DNS-oppslag og se hva som returneres.
whois	Slå opp hvem som eier/disponerer en IP-adresse
tcpdump / wireshark	Logge pakker som går inn og ut av maskinen, samt utføre analyse



UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Aksessmodeller i Applikasjonslaget



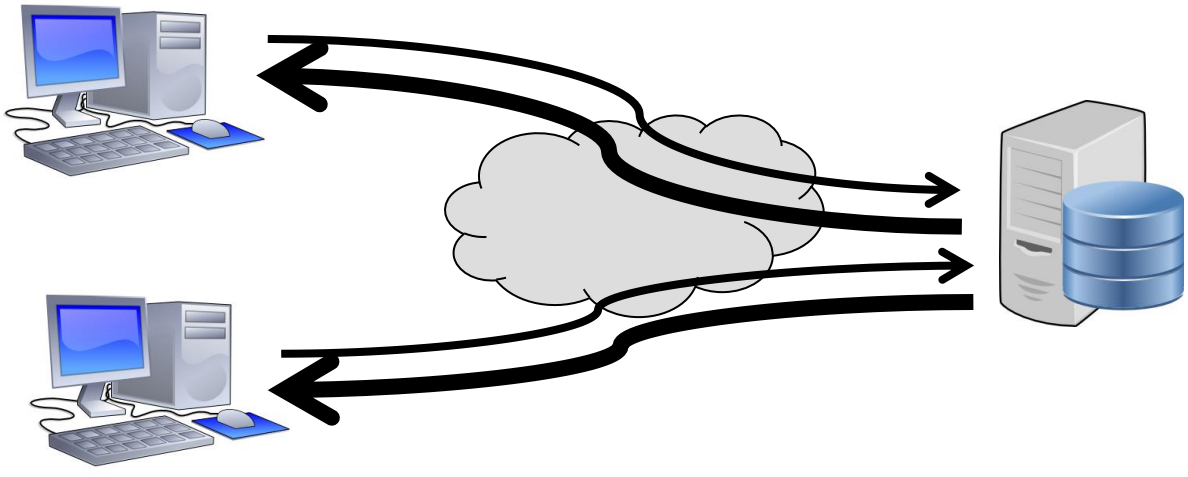
simula

tfj

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



Forbindelsesstrategier (push og pull)

Pull

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

Push

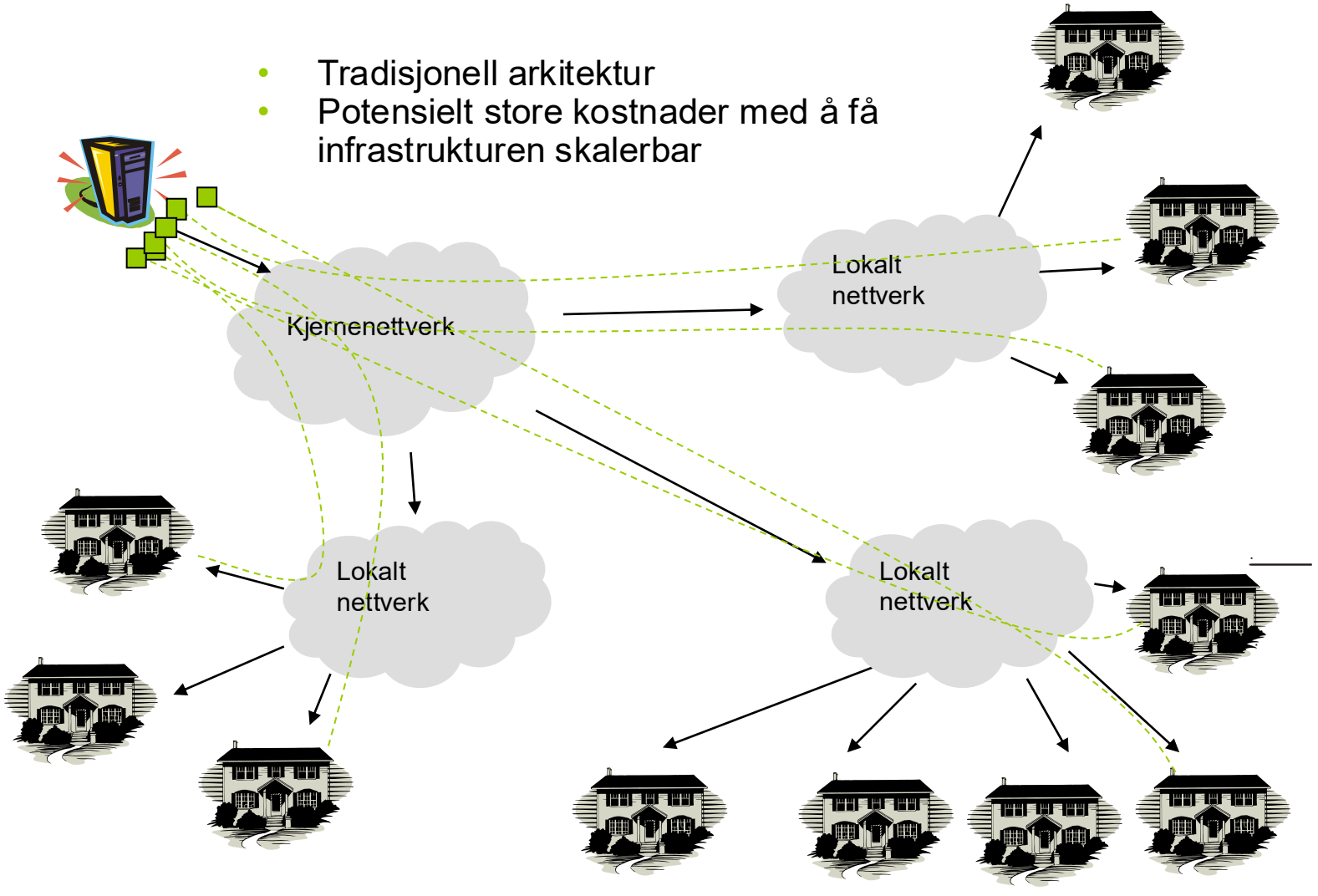
– Tjeneren ”dytter” en tjeneste (f.eks en beskjed) til klienten.
– Krever at det er en forbindelse fra før, eller at klienten lytter.

Publish-subscribe

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

Klient-Tjener

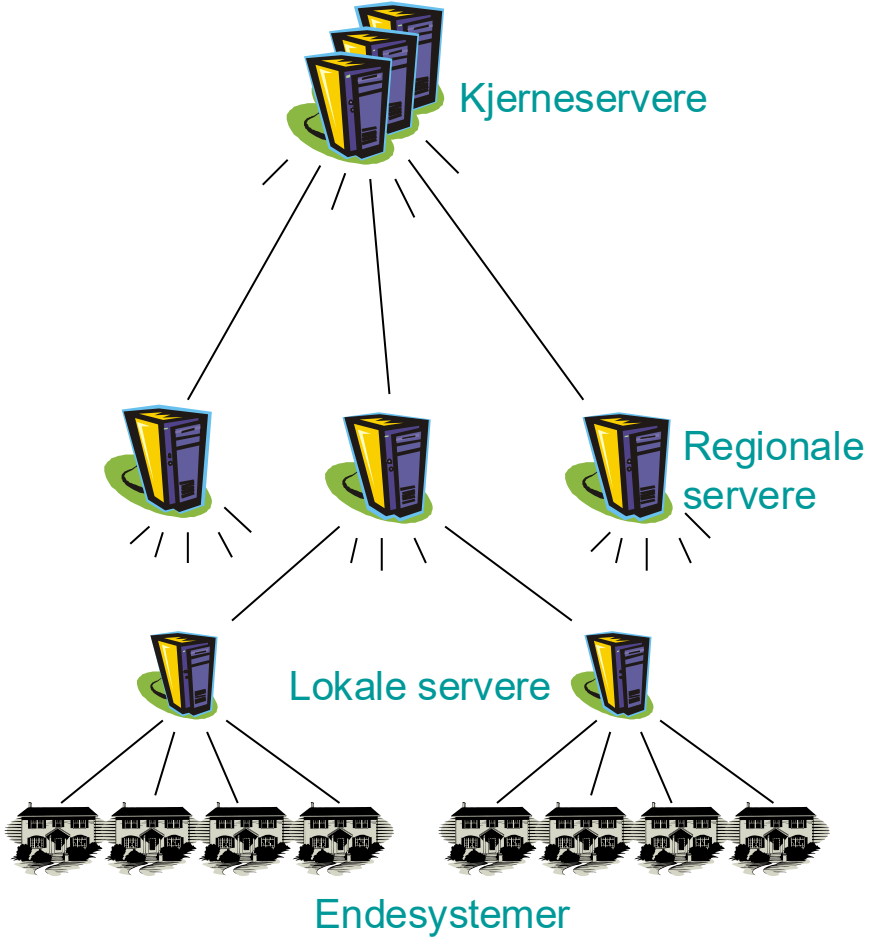
- Tradisjonell arkitektur
- Potensielt store kostnader med å få infrastrukturen skalerbar



Distribusjon med CDN

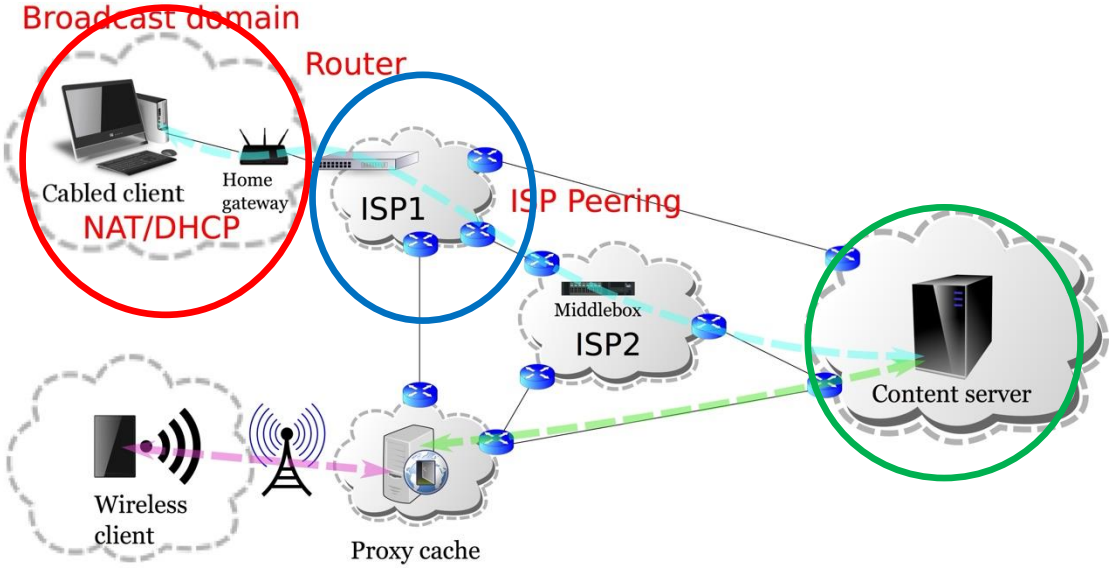
- **Content Delivery Network**
- **Hierarkiske systemer for å distribuere innhold**
  - Vanlig å ”cache” populært innhold.

- Populære videoer blir lagret nærme sluttbrukerne
- Videoer som ikke er så populære lagres enten på eller nærme kjerneserveren.
- Spesielt viktig hvis båndbredden er begrenset.
  - For eksempel: WiFi på fly



Fysisk plassering av innholdet

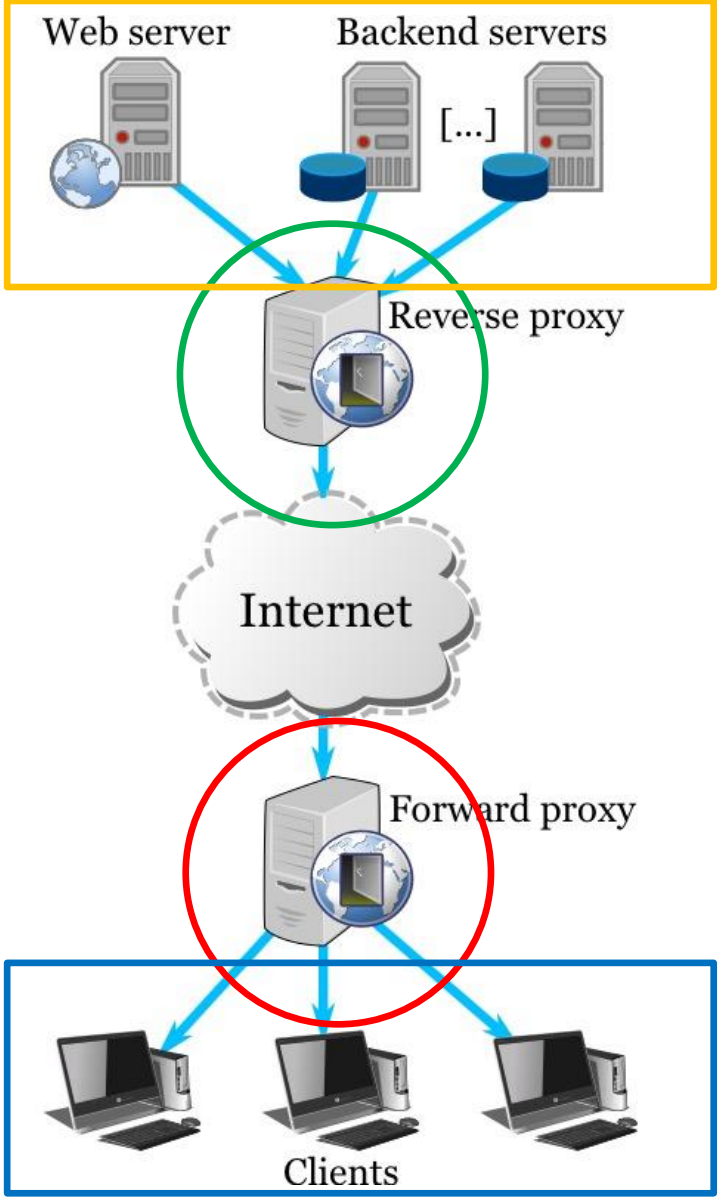
Cachenivå	Fysisk beliggenhet	Est. RTT
Lokal Proxy	Organisasjon / LAN	< 10ms
Content Delivery Network	ISP	< 50ms
Original datakilde	Internett	~10ms - ~250ms



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).

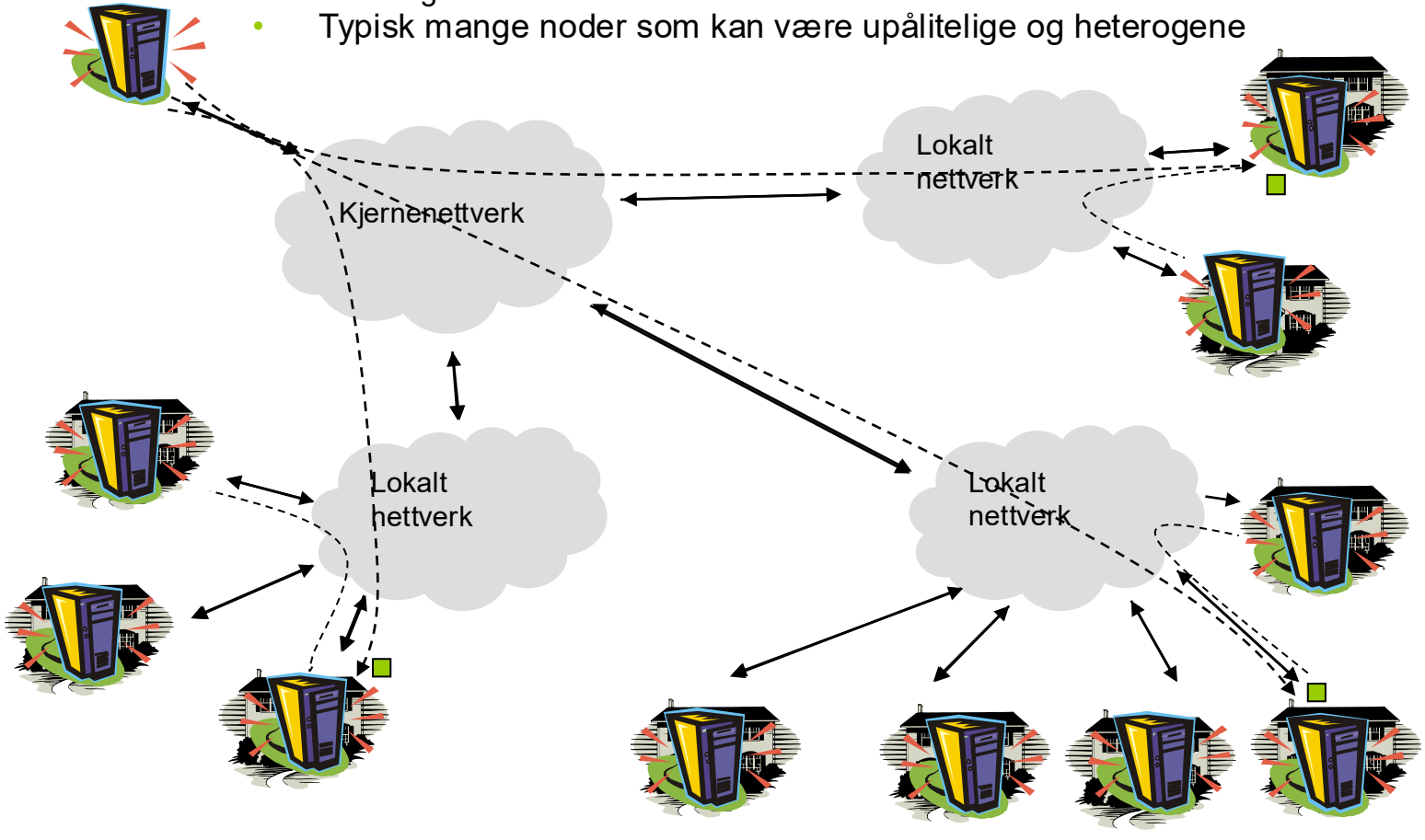
+ Lastbalansering
+ Sparer nettverkskapasitet
+ Lavere forsinkelse



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

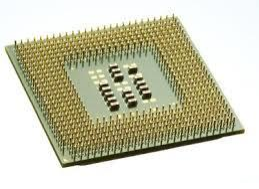


Peer-to-Peer

- **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



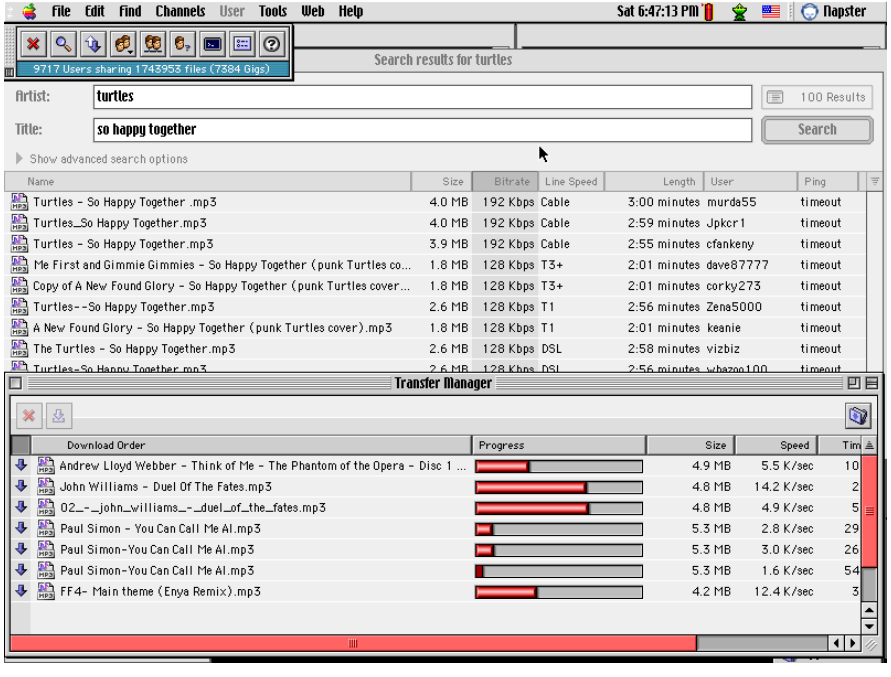


Eksempel: Napster

- 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

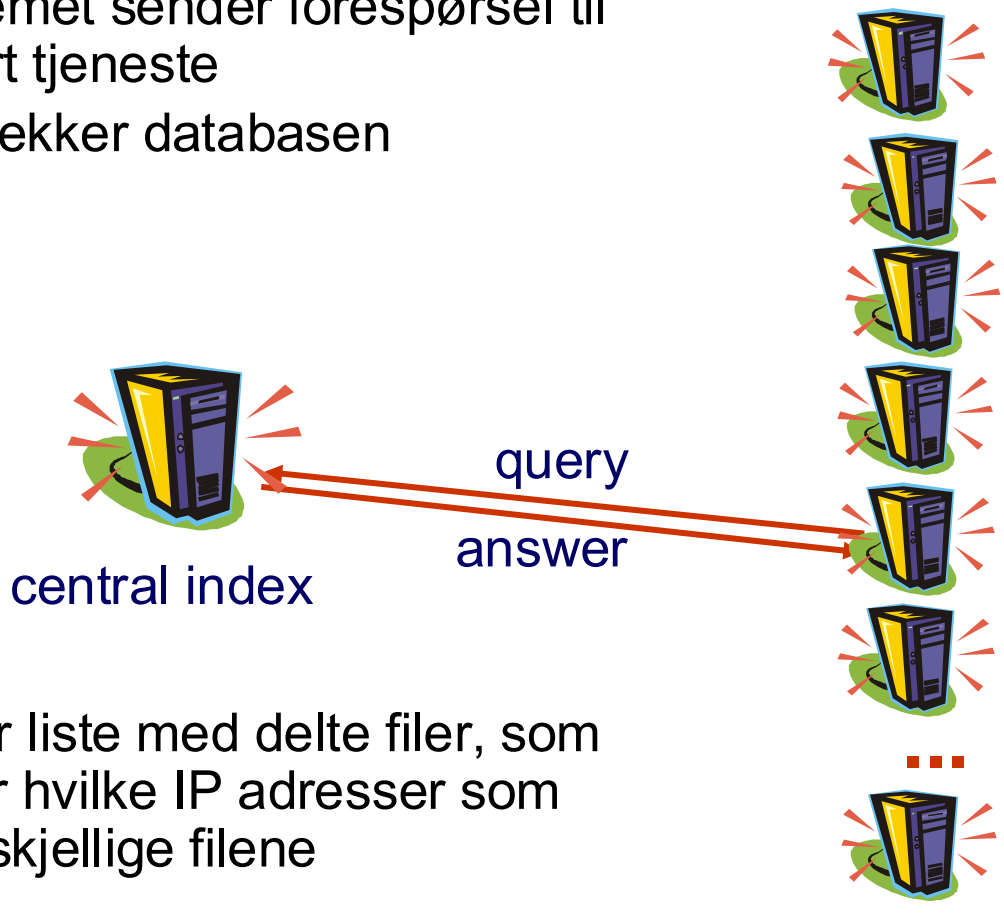
- Endesystemet kobler til Napster sin sentrale tjener.
- Sender over liste med delte filer



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

Napster

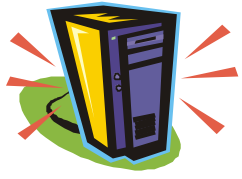
- Endesystemet sender forespørsel til sentralisert tjeneste
- Napster sjekker databasen



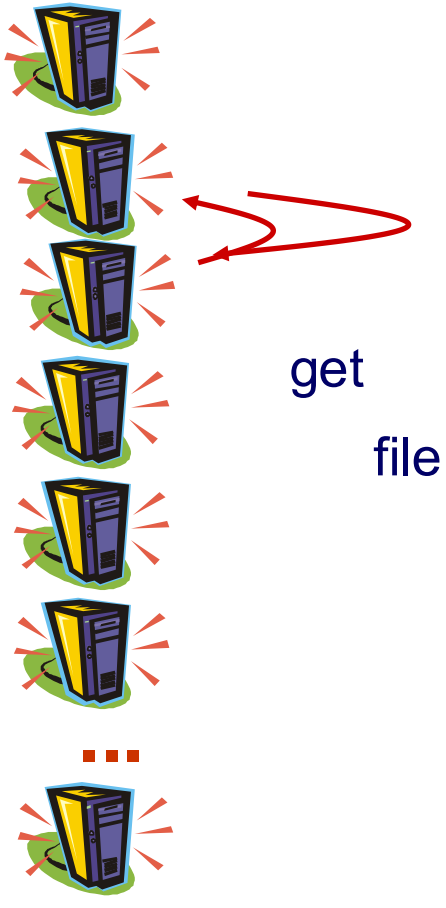
- Returnerer liste med delte filer, som inneholder hvilke IP adresser som har de forskjellige filene

Napster

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



central index



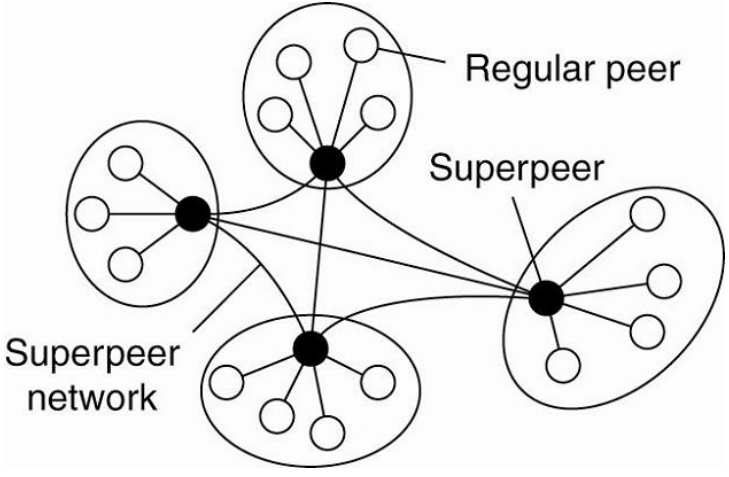
Napster: Evaluering

- 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…

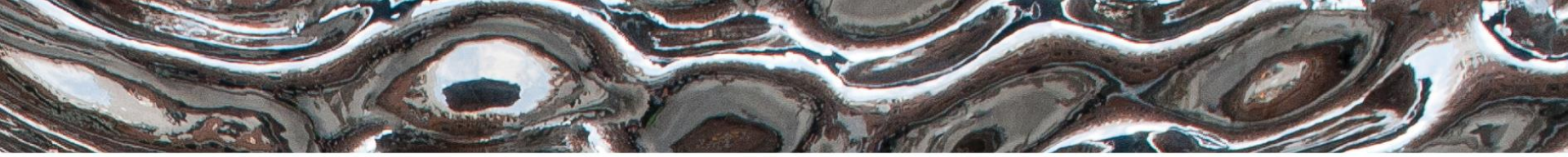


Varianter av P2P-baserte topologier

- 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







UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

DNS

Domain Name System



simula

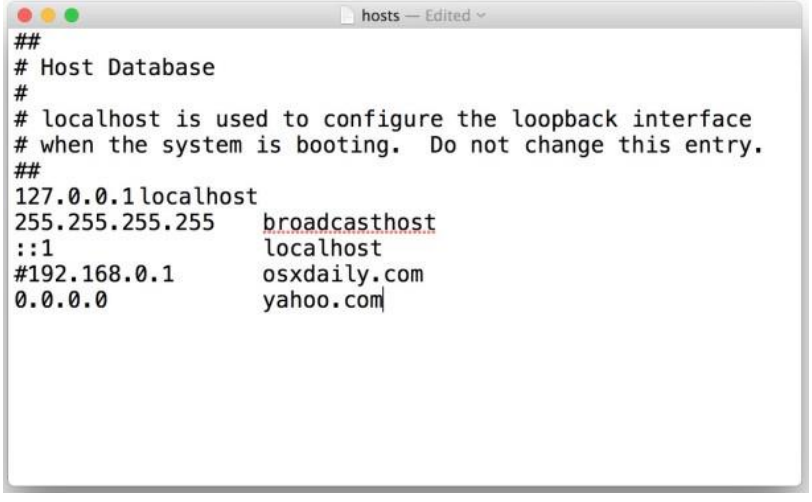


Hvordan koble til en annen maskin?

Med IP-adresse?

- `telnet 127.0.0.1 23`
  snakker med min egen maskin
  - Brukes ofte da det er en veldefinert adresse

- `wget http://173.194.39.31:80/`
  snakker med en av Google sine maskiner
  mulig å huske, men ikke praktisk.



- `ssh 9.228.93.3`
  forsøke å kontakte en maskin du visste hadde denne adressen i 2001
  umulig å huske om du ikke bruker den daglig

- det er mulig å lage alias for en IP-adresse i filen /etc/hosts (Unix variants)
- Opprinnelig administrert av Stanford Research Institute. Endringer ble distribuert via epost ☺

Hvordan koble til en annen maskin?



Løsning: bruk “fornuftige” navn

- som f.eks.

```
ssh login.ifi.uio.no
wget www.google.com
```

- Ikke bare lettere å huske
- Har også en hierarkisk struktur (gjenspeiler organisasjonen)

Møt **Domain Name System (DNS)**

DNS - overblikk

Domain Name System

Hierarkisk navnetilordning

I motsetning til den originale flate strukturen i /etc/hosts

f.eks.: .com → google.com → mail.google.com

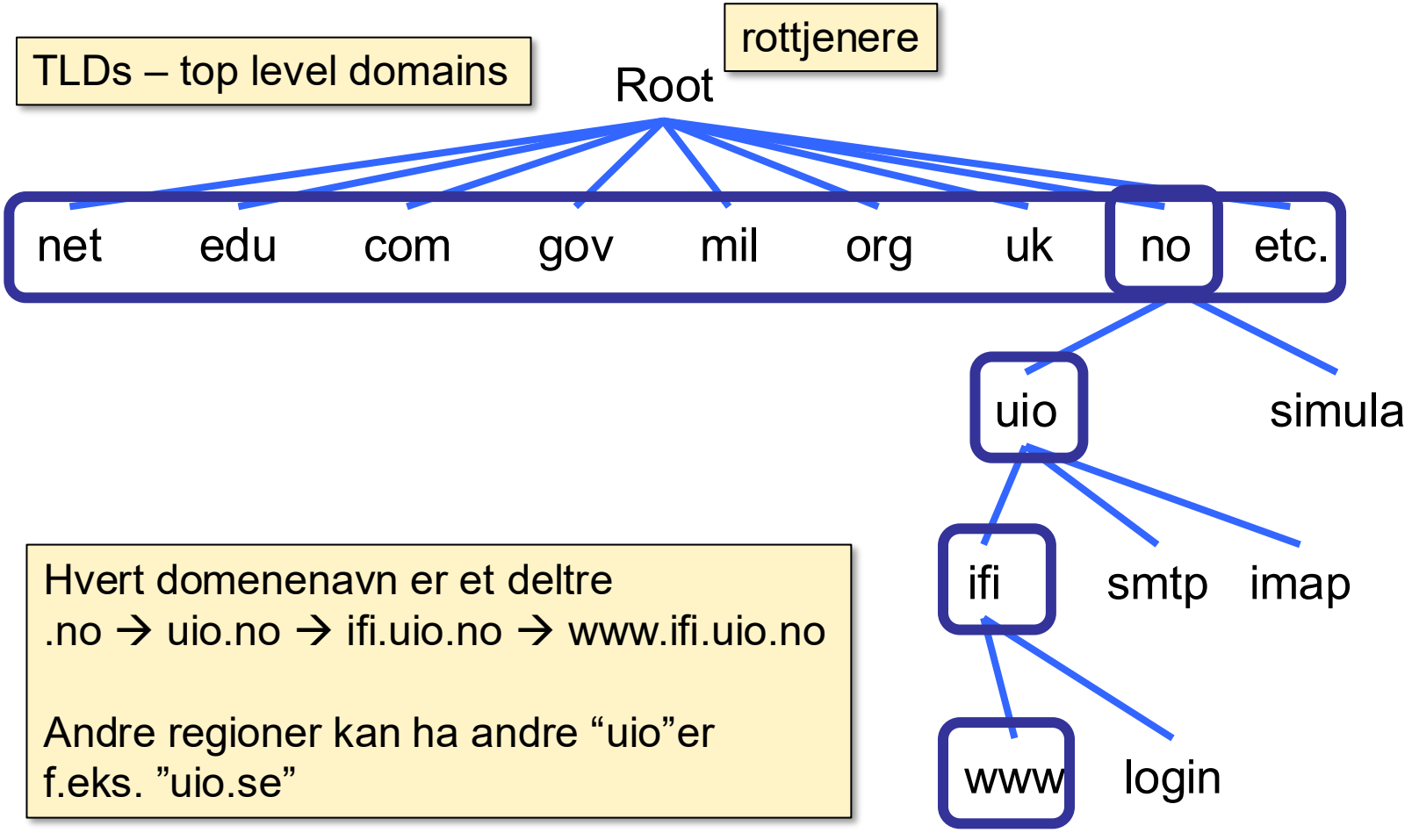
Distribuert database



Enkel klient/tjener arkitektur

- UDP eller TCP port 53
- tjenere må bruke TCP seg i mellom (fra nylig)
- klienter som bruker TCP avvises ofte
  - reduserer lasten på DNS-tjeneren

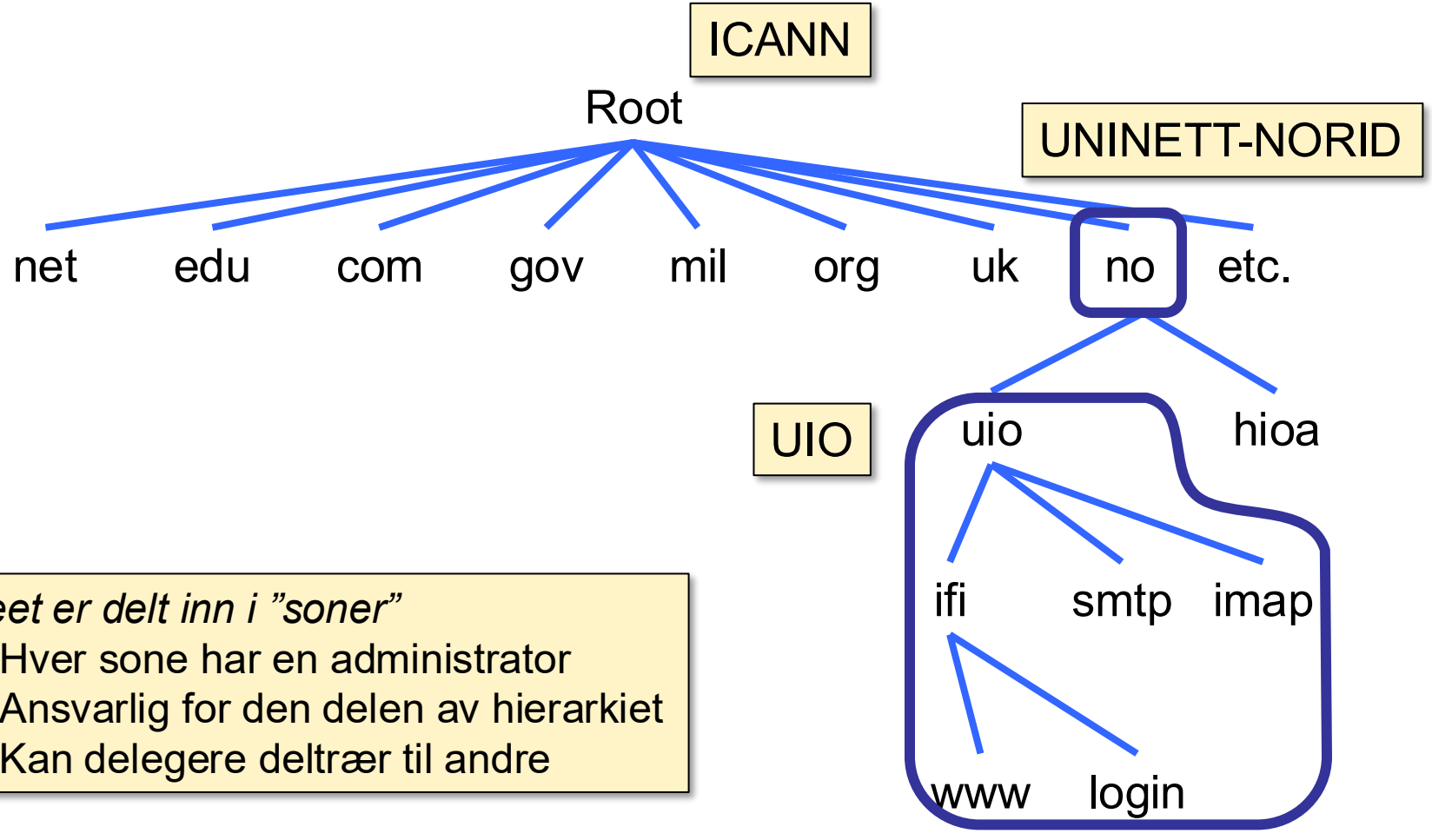
Navnehierarki



Hvert domenenavn er et deltre
.no → uio.no → ifi.uio.no → www.ifi.uio.no

Andre regioner kan ha andre “uio”er
f.eks. ”uio.se”

Administrasjon av navnehierarkiet



Treet er delt inn i "soner"

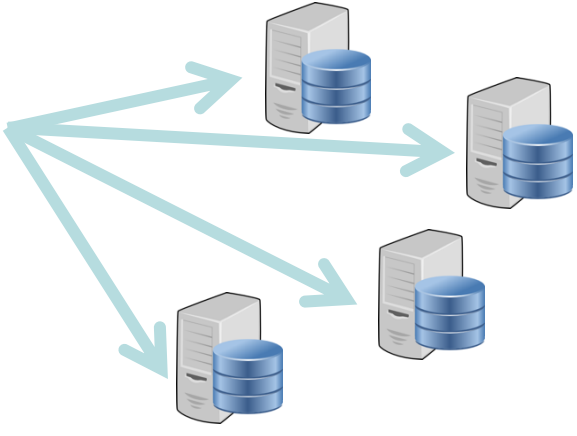
- Hver sone har en administrator
- Ansvarlig for den delen av hierarkiet
- Kan delegere deltrær til andre

Alias og lastbalansering

Én maskin kan ha mange alias



Ett domene kan kobles til mange maskiner

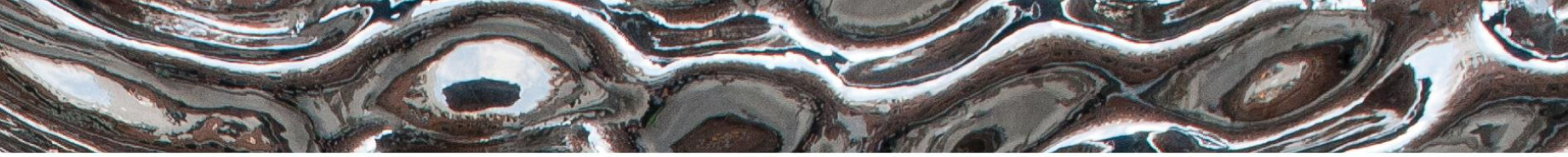


Eksempler:
k.root-server.net
og
login.ifi.uio.no

DNS og sikkerhet

- DNS: Navneoppslag, knytter maskinnavn til IP-adresse:
  - Kommunikasjonen foregår i klartekst(!)
- Farer:
  - DNS-forfalskning: Det introduseres uriktige opplysninger i navnetjener
  - DNS-modifisering: Man in the Middle-angrep, forfalsker meldinger
  - DNS-kapring: Endre hvilke navnetjener offerets maskin benytter
- Tiltak:
  - DNSSEC (*secure domain name system):* Signert DNS, basert på asymmetrisk kryptering og egen PKI
  - Navnetjenere må sikres godt
  - Se etter hengelåssymbol på websider (bekrefter autentisitet)

Sjekke domenenavn: https://www.norid.no/no/dine-domenenavn/hva-er-registrert/



UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Applikasjonslaget

HTTP

HyperText Transfer Protocol



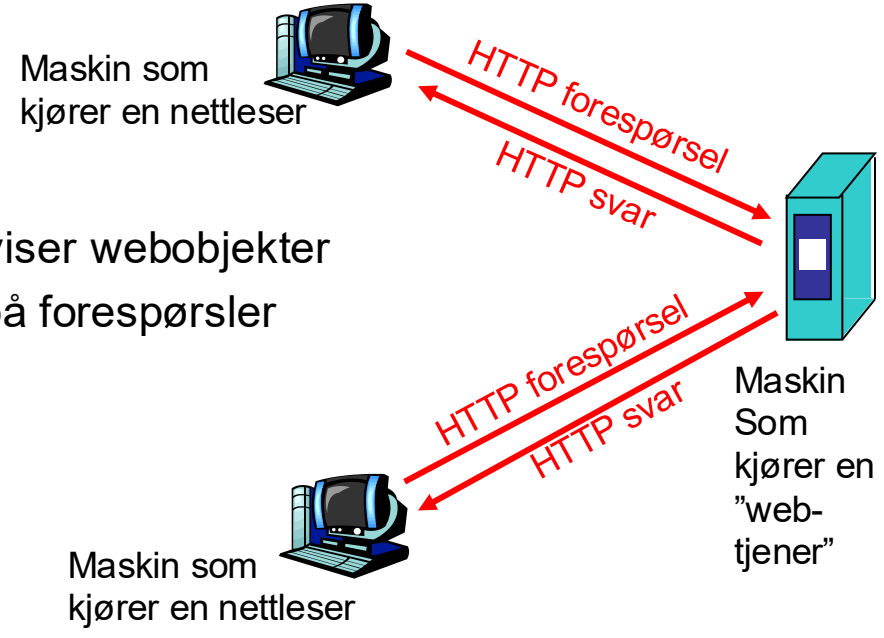
simula

tfj

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.

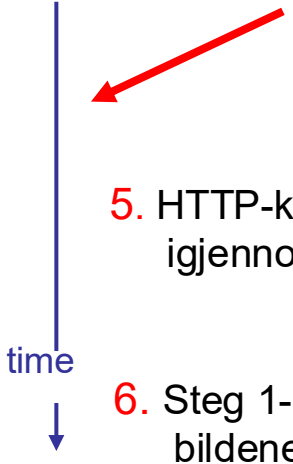
2. HTTP-klienten sender HTTP *request message* (med URL) til TCP-forbindelsen

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

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)

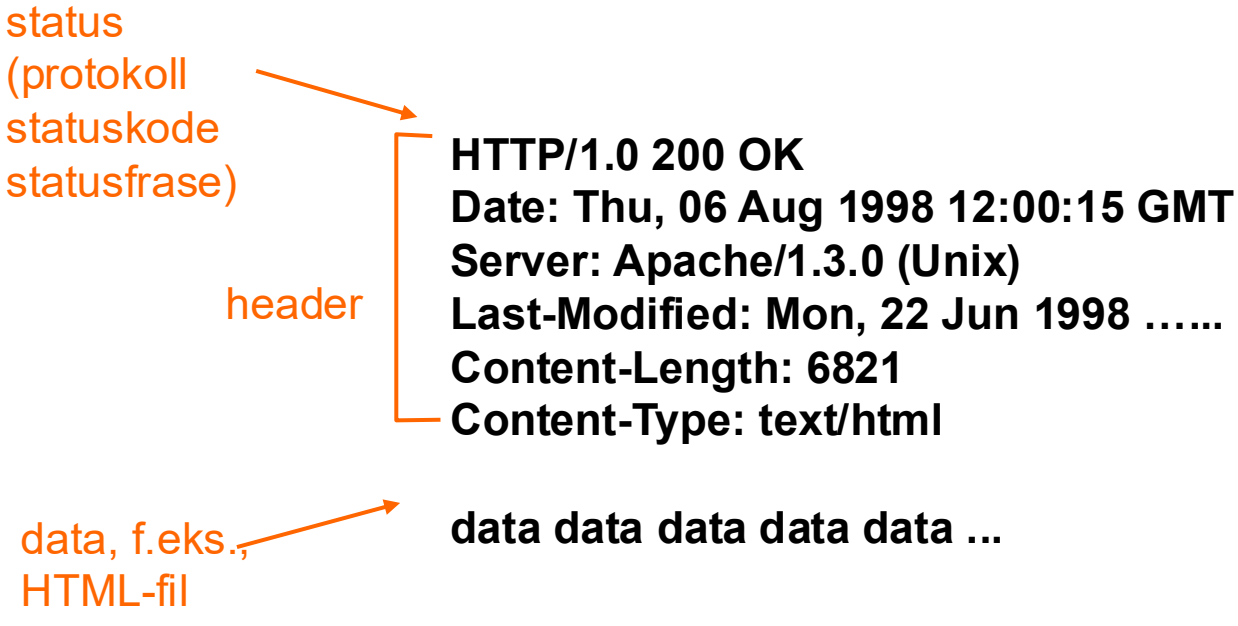
header

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

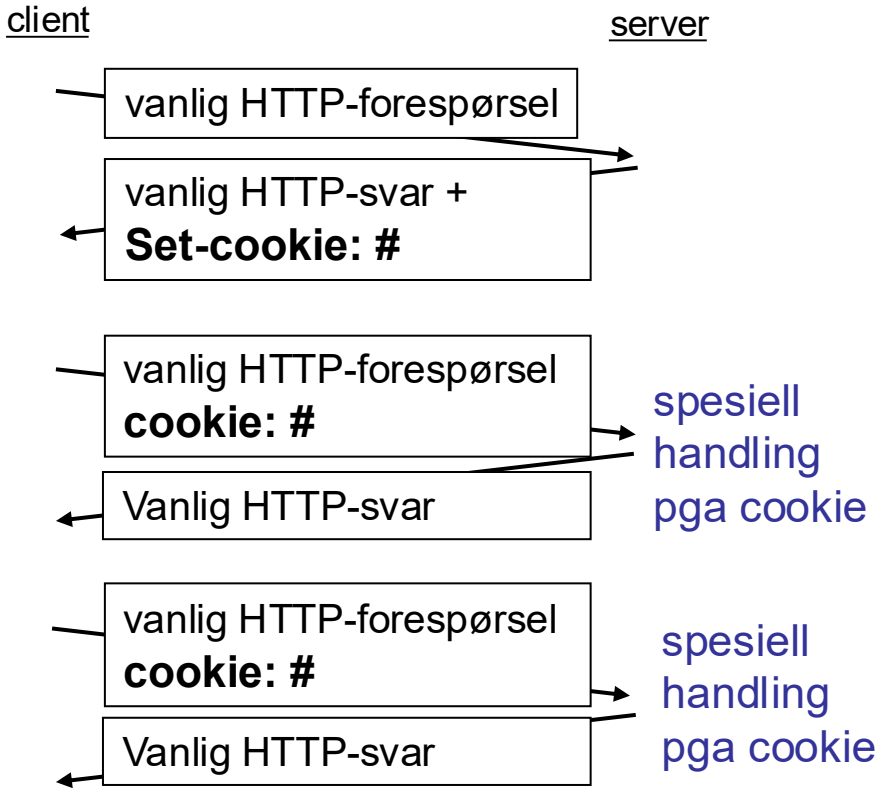
505 HTTP Version Not Supported

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

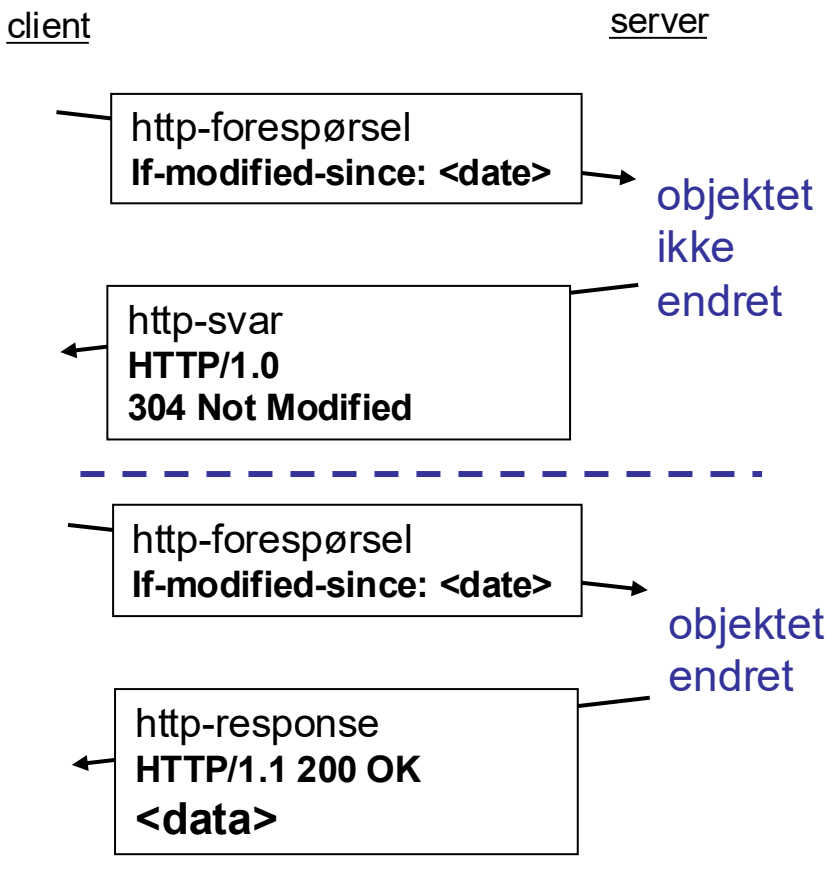


Betinget GET: klient-side caching

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

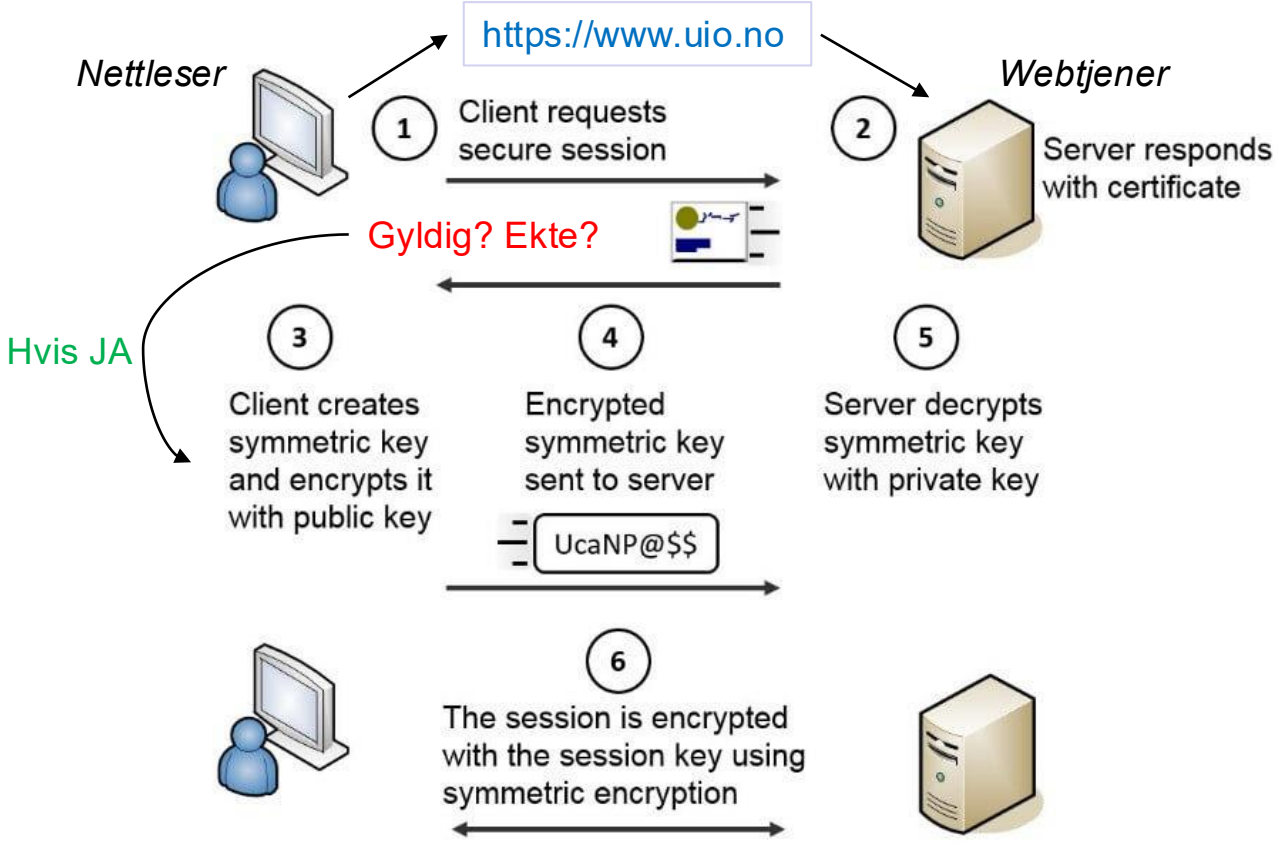
  If-modified-since: <date>
- tjener: svaret inneholder ikke objektet dersom den cachede kopien er oppdatert:

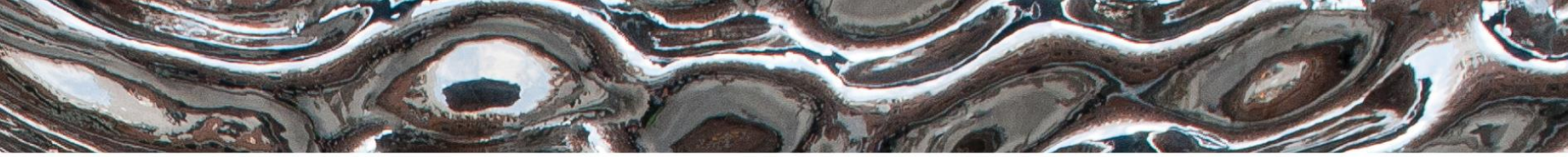
  HTTP/1.0 304 Not Modified



Sikrere HTTP: HTTPS (HyperText Transfer Protocol Secure)

- Ønske:
  - Sikre kryptert kommunikasjon mellom klient og tjener (konfidensialitet+integritet)
  - Sikre tillitt til at web-tjeneren er ekte (autentisitet)
- Løsning: HTTPS (HTTP over SSL/TLS)
  - Bruker en *offentlig-nøkkel infrastruktur* (Internett-PKI*)* med *offentlig-nøkkel sertifikater*
  - En webtjener har et kryptografisk nøkkelpar (offentlig + privat nøkkel), hvor den offentlige nøkkelen er distribuert i tilhørende sertifikat*,* utstedt og signert av en tiltrodd *sertifikatmyndighet* (CA):
    1. Nettlesere (klienter) kan *verifisere* at en nettside *(*webtjener) er autentisk ved å sjekke at webtjenerens sertifikat er ekte og gyldig, og at det er samsvar mellom sertifikatets domenenavn og nettsidens domenenavn
    2. Webtjeners offentlige nøkkel kan så benyttes til kryptert kommunikasjon fra klient til webtjener.


https://www.uio.no
Nettleser
Webtjener
1
Client requests secure session
2
Server responds with certificate
Gyldig? Ekte?
Hvis JA
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



UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Applikasjonslaget

Video Streaming

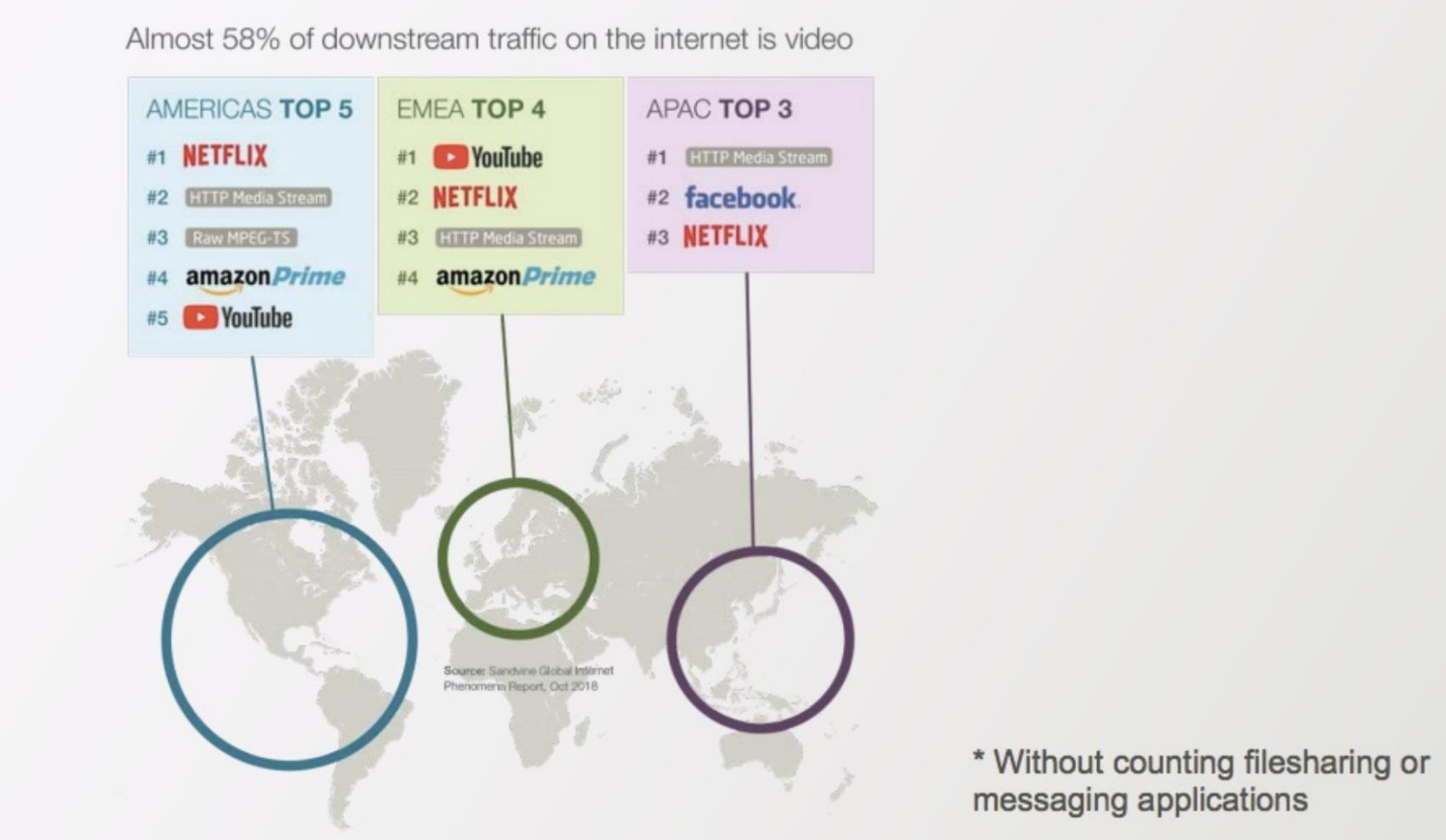


simula

tfj

Videostreaming



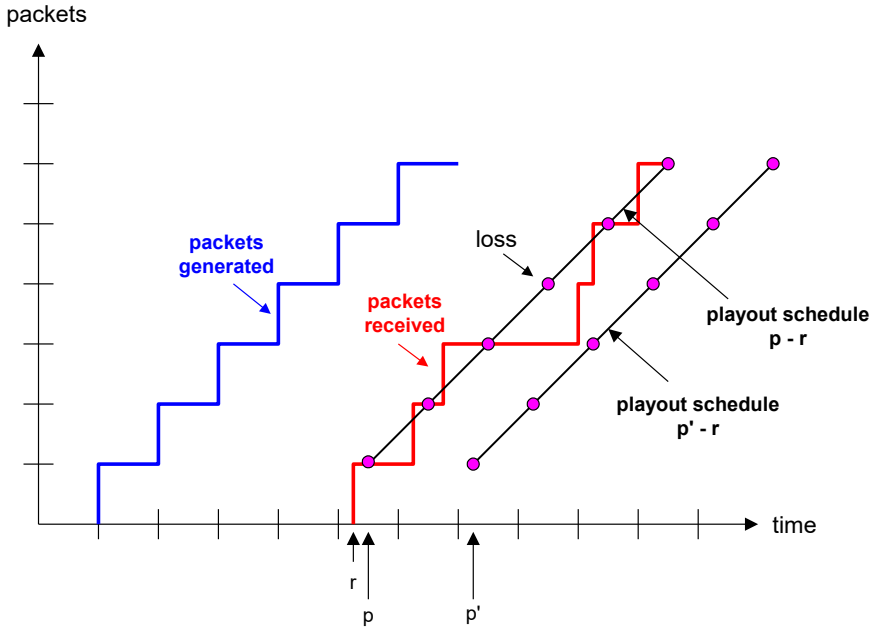


Kilde: Sandvine - http://digg.com/2018/streaming-video-worldwide - 4.10.2019

Videostreaming - utfordringer

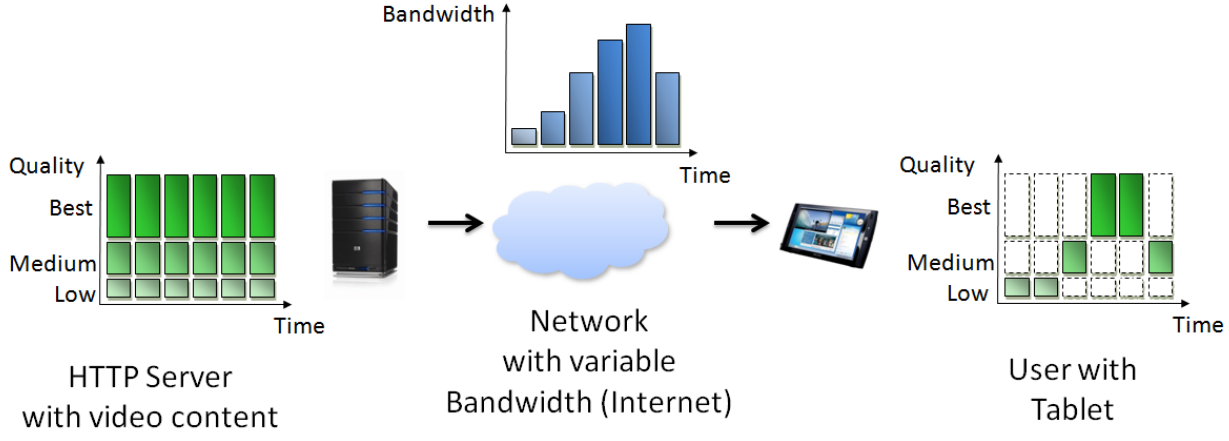
Multimedia

- 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

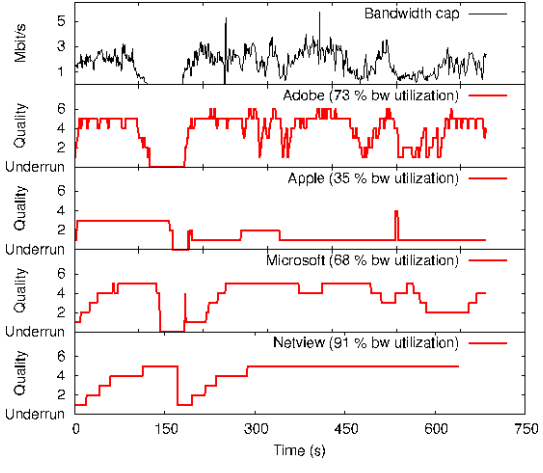


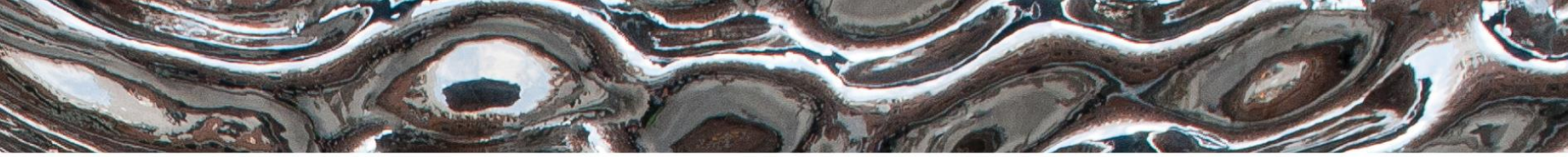
HTTP-streaming

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 tilpasningsgstrategi
  - Klienten velger, ikke tjeneren
  - Strategien hos avspilleren er det som utgjøre forskjellen





UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IoT

Internet of Things



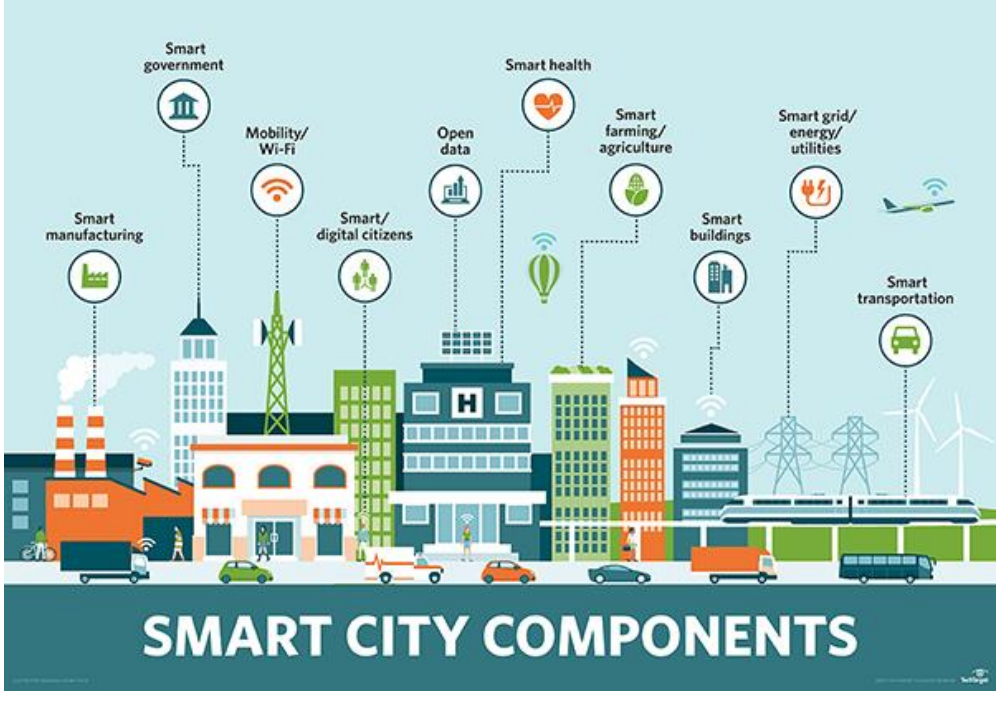
simula

tfj

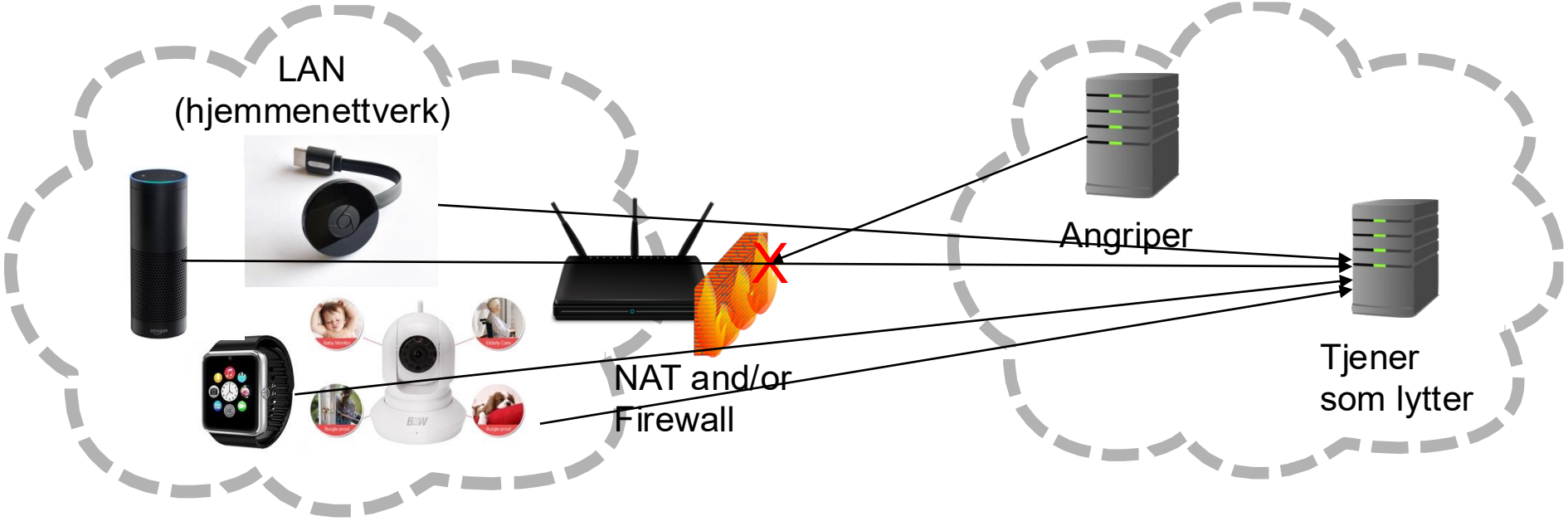
Internet of Things (IoT)

- 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".
  - 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”