Systemmodeller for distribuerte system

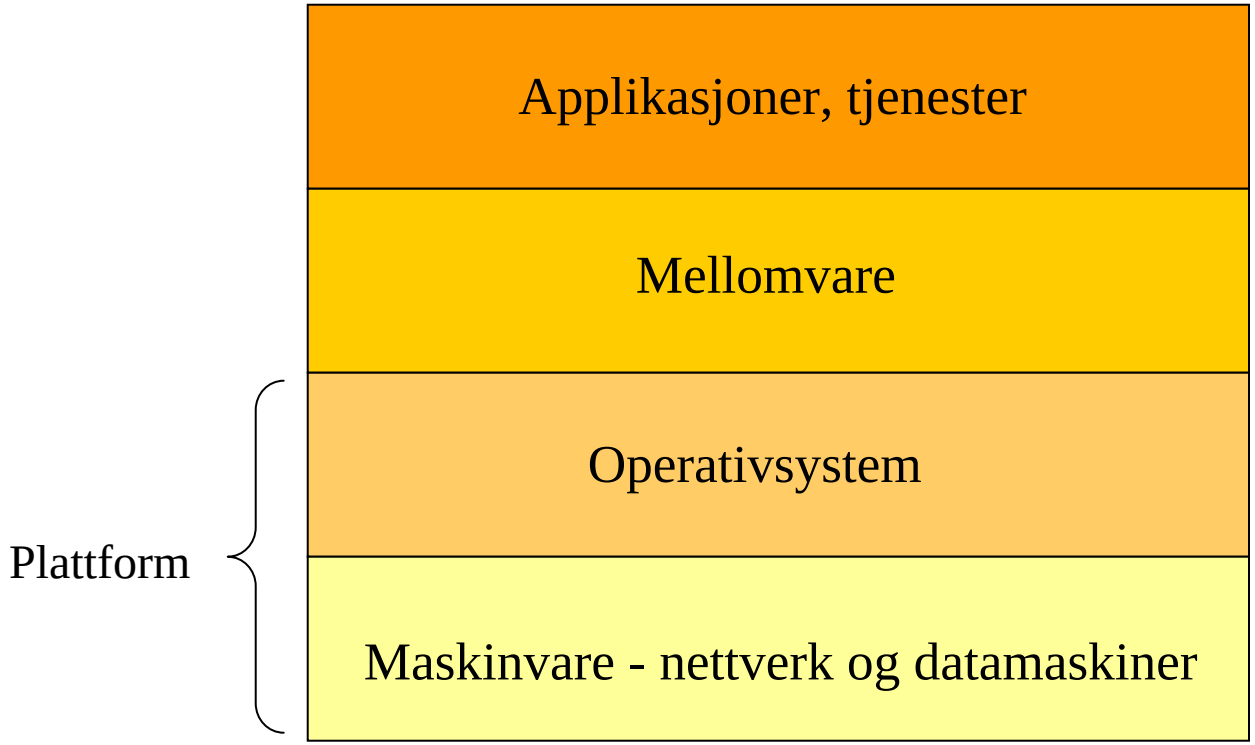
INF5040 høst 2005

foreleser: Olav Lysne

Systemmodeller

- Motivasjon
  - illustrere felles egenskaper og designvalg for distribuerte system i én beskrivende modell
- To typer modeller
  - Arkitekturmodeller: definerer komponentene til systemet, den måten de interagerer på, og på hvilken måte de utplasseres i et underliggende nettverk av datamaskiner
  - Fundamentale modeller: formell beskrivelse av egenskapene som er felles i arkitekturmodellene. Tre fundamentale modeller:
    - interaksjonsmodeller
    - feilmodeller
    - sikkerhetsmodeller

Lagdeling av DS



Distr. tjeneste = antall tjenerprosesser

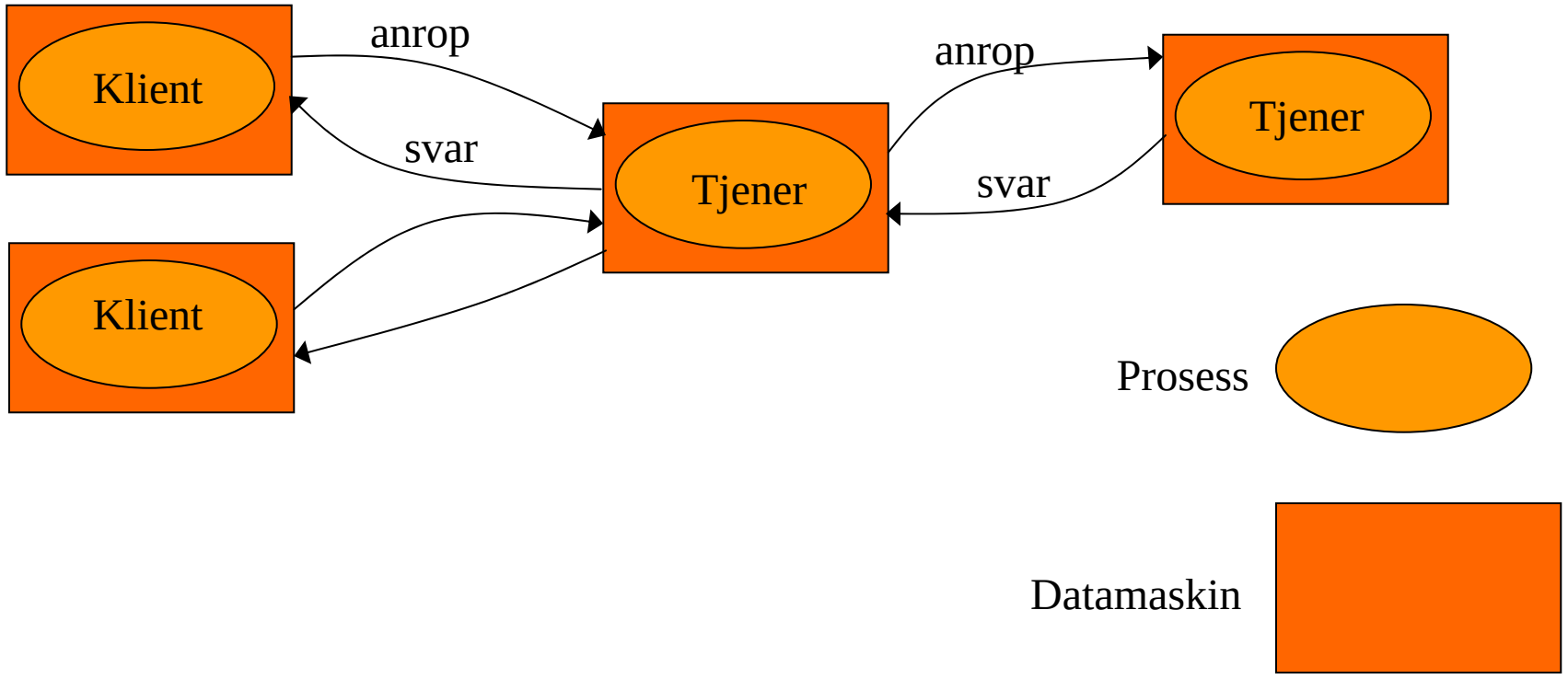
Skjule heterogenitet
Uniform prog. modell
MV tjenester

Forvaltning av data-maskinressurser

Systemarkitekturer

Klient/tjener modell:

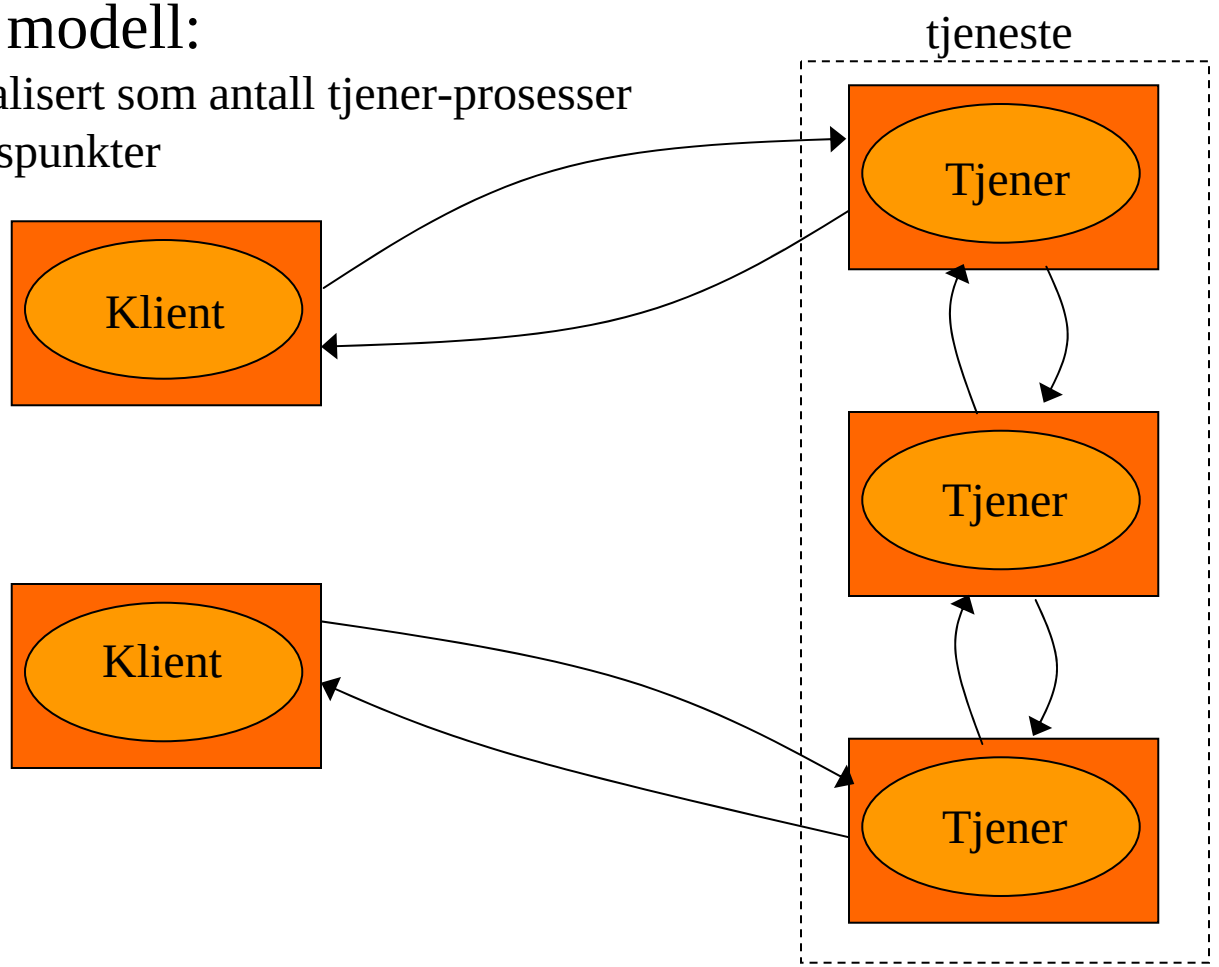
kjent i mer enn 20 år, svært mye anvendt i DS design



Systemarkitekturer

Klient/tjener modell:

- tjeneste realisert som antall tjener-prosesser
- flere aksesspunkter

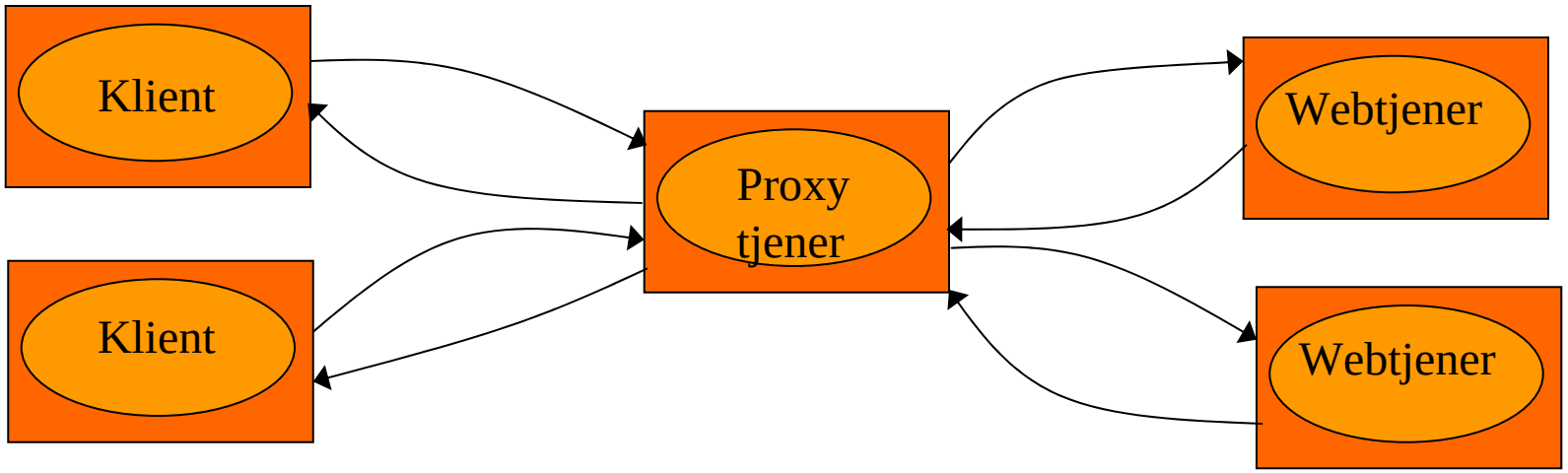


Systemarkitekturer

Klient/tjener modell med proxy-tjener:

Cache: lager av nylig-brukte dataobjekter som er nærmere klienten enn de opprinnelige objektene selv.

Proxytjener: cache som er delt mellom flere klienter



Systemarkitekturer

Likeverdige ("peer") prosesser:

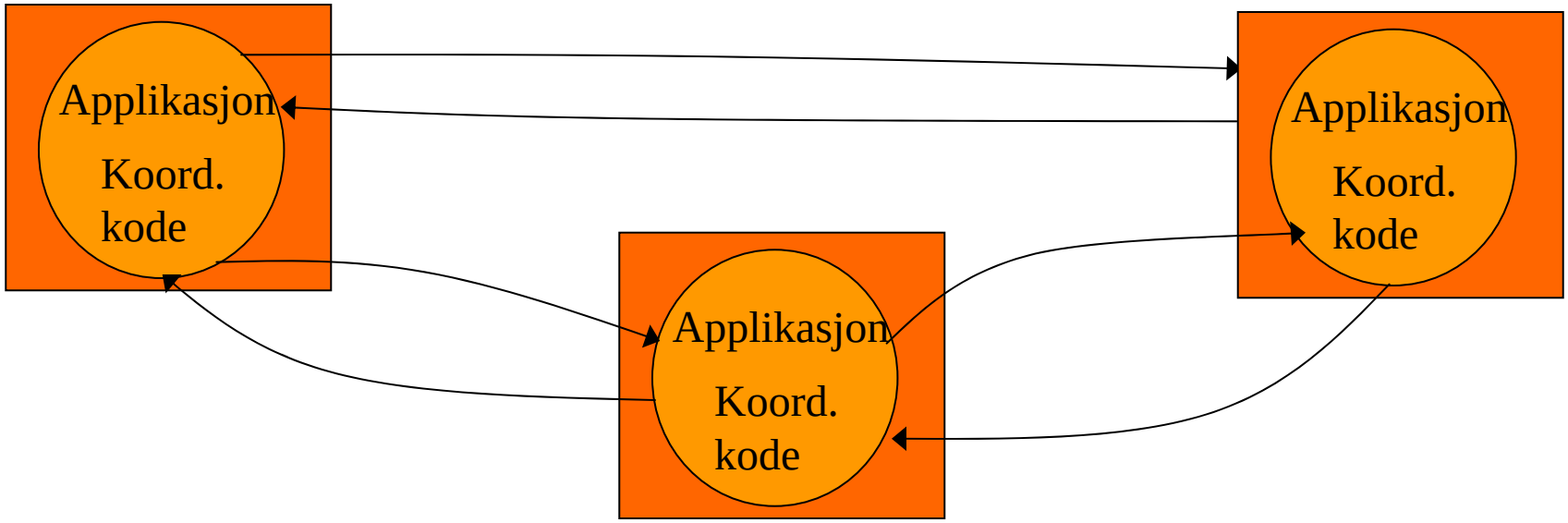
Prosessene har like roller, ingen forskjell mellom klienter og tjenere .

Eksempler:

samarbeidssystemer (CSCW) som "whiteboard" applikasjoner o.l.

interaktive nett-baserte spill

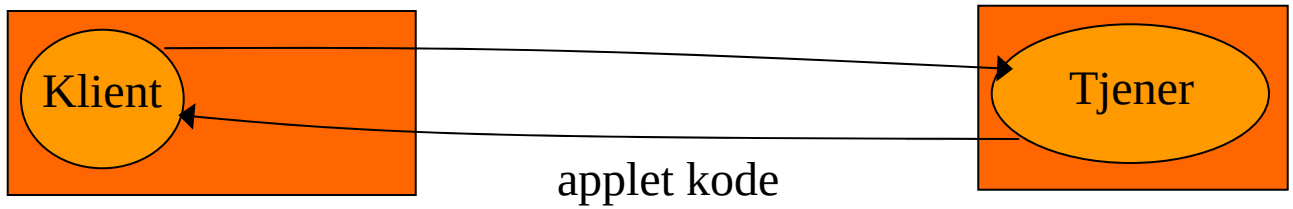
Mulig å lage som samling prosesser som hver er kombinert klient/tjener?



Systemarkitekturer

Variasjoner over klient/tjener modellen:

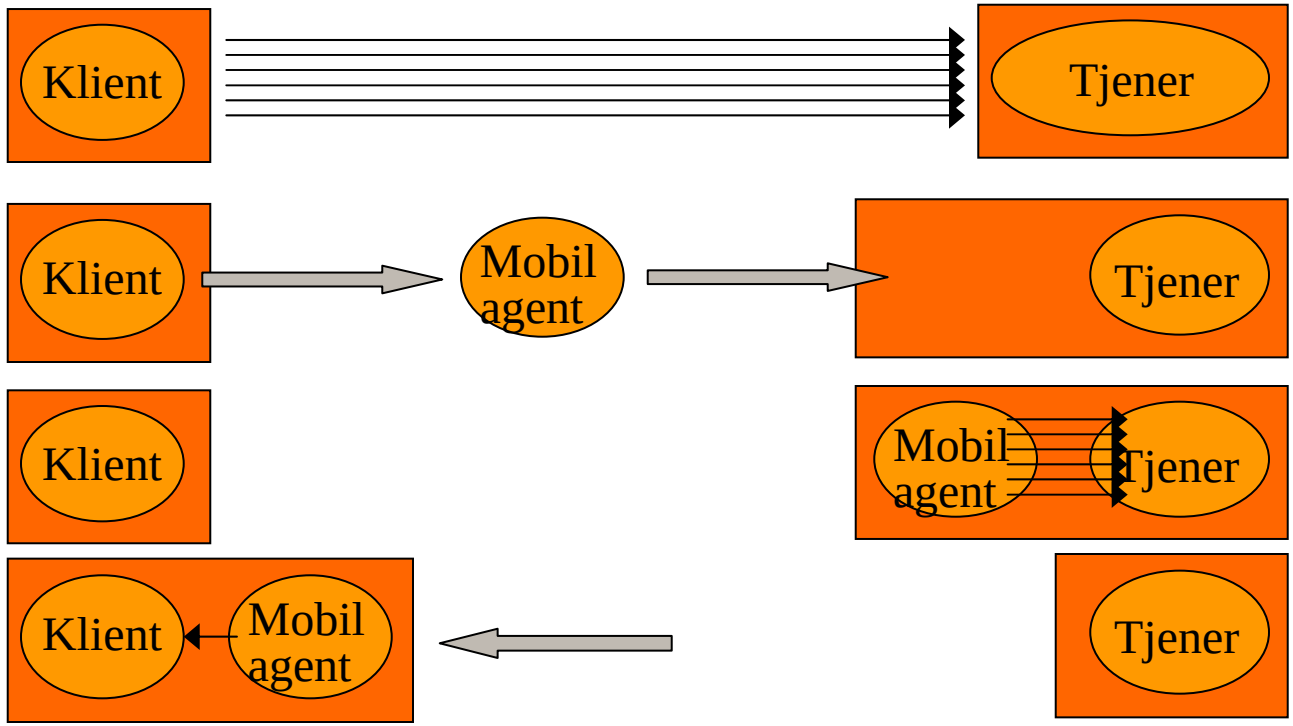
Mobil kode (applets) . Muliggjør f.eks. “push-modell”: tjeneren anroper klienten



Systemarkitekturer

Variasjoner over klient/tjener modellen:

Mobile agenter . Program (kode + data) som flytter seg fra maskin til maskin i nettet og utfører en oppgave på vegne av noen.

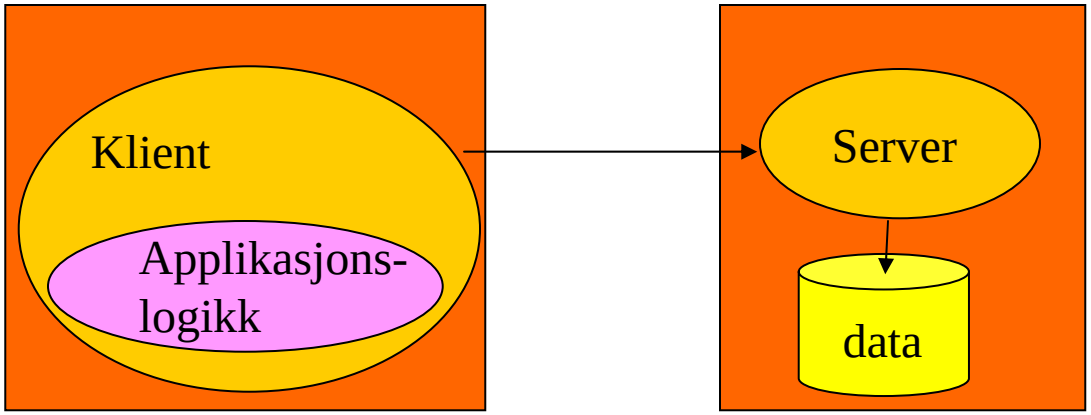


Tynne vs tykke klienter

- "Historisk" har trenden for applikasjonsarkitekturer pendlet mellom tykke og tynne klienter
- En variant er nettPCer
- Trenden i dag?
  - synes å være tynne klienter
  - små håndholdte klienter (WAP, PDA, ...)
  - "allestedsværende databehandling"

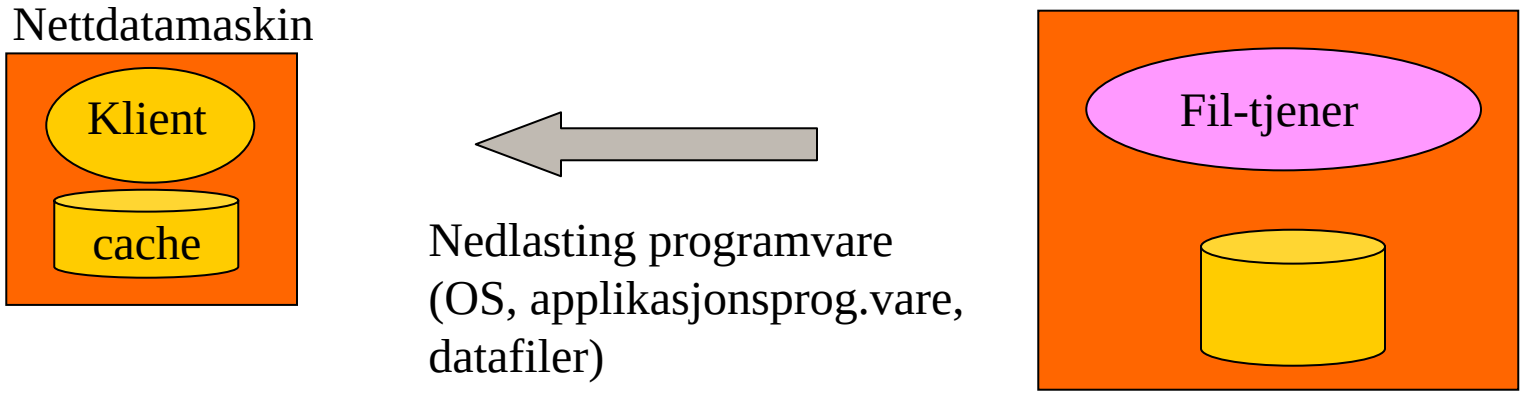
Tykke klienter

- Identisk programvare installert i alle klienter
- Tillater individuell installasjon og konfigurering
- Alltid tilgjengelig, enkel lisensiering
- Problemer
  - brukere kan ødelegge installasjonen
  - vanskelig å holde programvare “up to date”



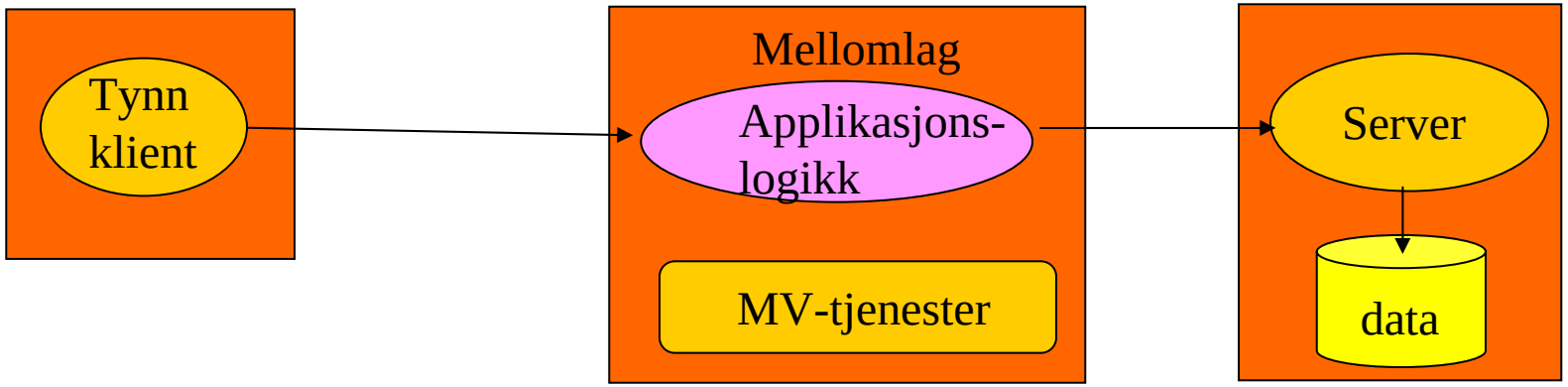
Nett-datamaskin/PC

- Nettdatamaskiner forsøker å unngå installasjonsproblemene i klienten
  - Programvare lastes ned i klienten etter behov
  - Konfigurasjon kan bestemmes på forhånd (vedlikeholdes på ett sted)
- Problemer
  - tilgjengelighet (tjeneren feiler)
  - lisensproblemer (lisenspool i stedet for lisens til hver maskin)



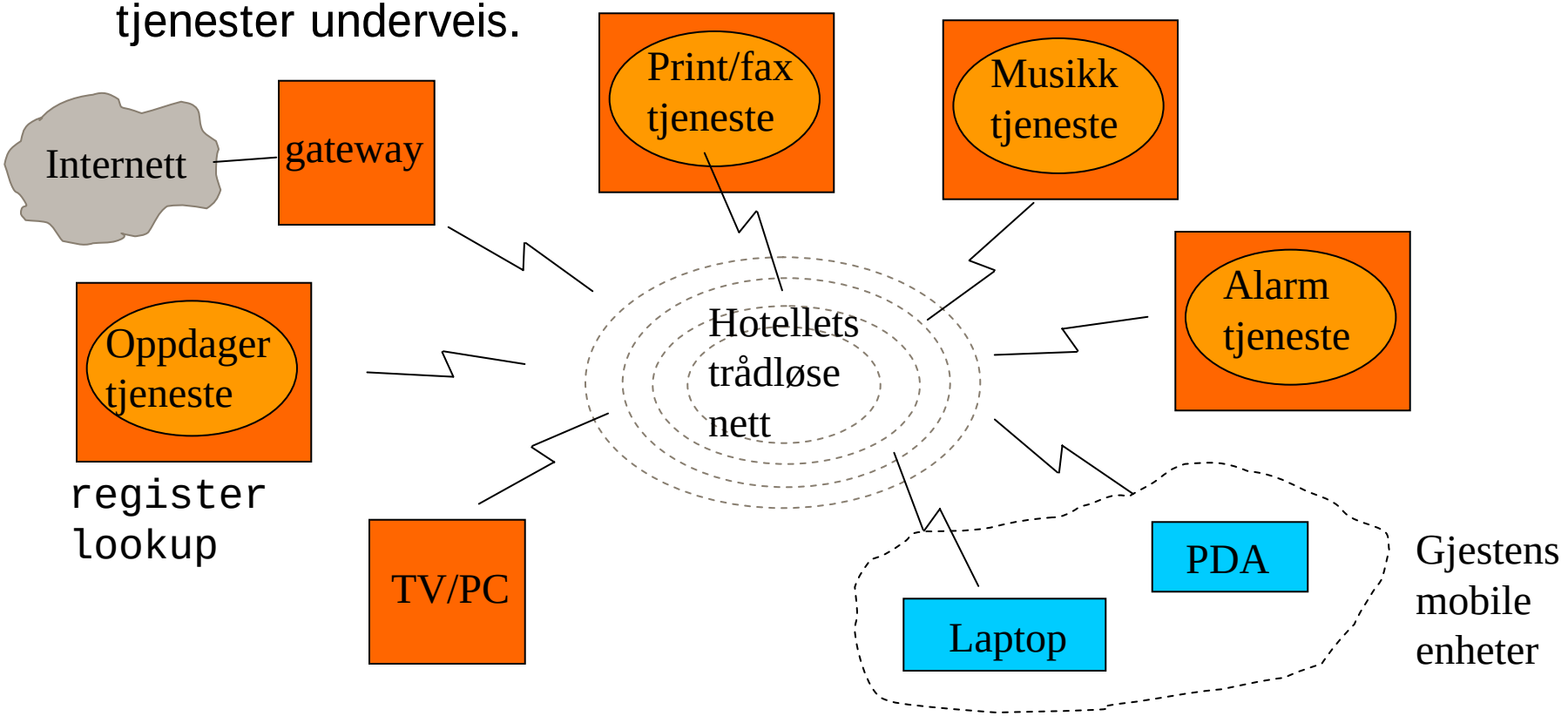
Tynne klienter

- Tynne klienter forsøker også å unngå installasjonsproblemene i klienten
  - Tynn klient: programvarelag som understøtter vindusbasert brukergrensesnitt (X.11 server, Web-browser, ...)
  - applikasjonsprogram eksekverer på fjern tjenermaskin
- Problemer
  - tilgjengelighet (tjeneren feiler)
  - lisensproblemer (lisenspool i stedet for lisens til hver maskin)
  - applikasjoner med høy grad av interaktivitet



Spontane nettverk

- Brukere bærer med seg mobile enheter (laptop, PDA, ...) mellom forskjellige nettverksomgivelser (hotellnettverk, flyplassnettverk, jernbanestasjonsnettverk, ...) og kan utnytte lokale og fjerne tjenester underveis.



Fundamentale modeller

- Egenskaper delt av alle arkitekturmodeller
  - kommuniserer ved å sende meldinger over et nettverk
  - krav om ytelse, pålitelighet, og sikkerhet
- Fundamental modell
  - abstraherer bort alle unødvendige detaljer
  - benyttes til å adressere spørsmål som
    - hva er de viktigste entiteter i systemet?
    - hvordan interagerer de?
    - hva er det karakteristiske som påvirker deres individuelle og samlede adferd?
- Hensikten med fundamentale modeller
  - gjøre eksplisitt alle relevante antagelser om systemet vi modellerer
  - finne ut hva som generelt er mulig og umulig under de gitte antagelser

Fundamentale modeller

- Aspekter av distribuerte system vi ønsker å uttrykke
  - Interaksjonsmodell
    - prosesser, meldinger, koordinering (synkronisering)
    - må uttrykke at meldinger har vilkårlig forsinkelse, og at forsinkelse begrenser nøyaktig koordinering og vedlikehold av global tid
  - Feilmodell
    - definerer og klassifiserer feil som kan forekomme i DS
    - grunnlag for å analysere effekter av feil og for design av system som kan tolerere feil
  - Sikkerhetsmodell
    - definerer og klassifiserer sikkerhetsangrep som kan forekomme i DS
    - grunnlag for å analysere trusler mot et system og for design av system som kan motstå truslene

To varianter av interaksjonsmodellen

- Synkrone distribuerte system
  - tiden det tar å utføre et steg i en prosess har en nedre og øvre grense
  - tiden det tar å overføre en melding over en kanal har en kjent øvre grense
  - hver prosess har en lokal klokke hvis avviksrate fra sann tid har en kjent øvre grense
- Asynkrone distribuerte system
  - tiden det tar å utføre et steg i en prosess har ingen øvre grense
  - tiden det tar å overføre en melding over en kanal har ingen nedre eller øvre grense
  - hver prosess har en lokal klokke hvis avviksrate fra sann tid kan være vilkårlig stort

Signifikans synkrone vs asynkrone DS

- Mange koordineringsproblemer har en løsning i synkrone distribuerte system, men ikke i asynkrone
  - f.eks. “The two army problem” eller “Agreement in Pepperland” (se [Coulouris])
- Ofte antar vi synkronitet selv om det underliggende distribuerte system i essens er asynkront
  - eksempel: Internett er i essens asynkront men vi benytter timeout i protokoller over Internett for å detektere feil
  - basert på estimat av grenseverdier
  - men: design basert på grenseverdier som ikke kan garanteres, vil ikke være pålitelige

Ordning av hendelser

- distribuerte koordineringsprotokoller har behov for ordne hendelser i tid ("hente før"-relasjon)
  - hendelser: sending og mottak av meldinger
  - eksempel: oppdatering på replikerte data må skje i samme rekkefølge i alle replika
  - vanskelig å benytte *fysiske klokker* i datamaskiner til koordinering (f.eks. klokkeverdi i meldinger)
    - har begrenset tidsoppløsning og går med ulik hastighet (klokkedrift)
    - egenskapene til meldingsutveksling begrenser nøyaktigheten vi kan synkronisere klokkene med i et DS [Lamport 78]
  - mulig å beskrive logisk ordning av hendelser selv uten nøyaktige klokker vha av *logiske klokker* [Lamport78]

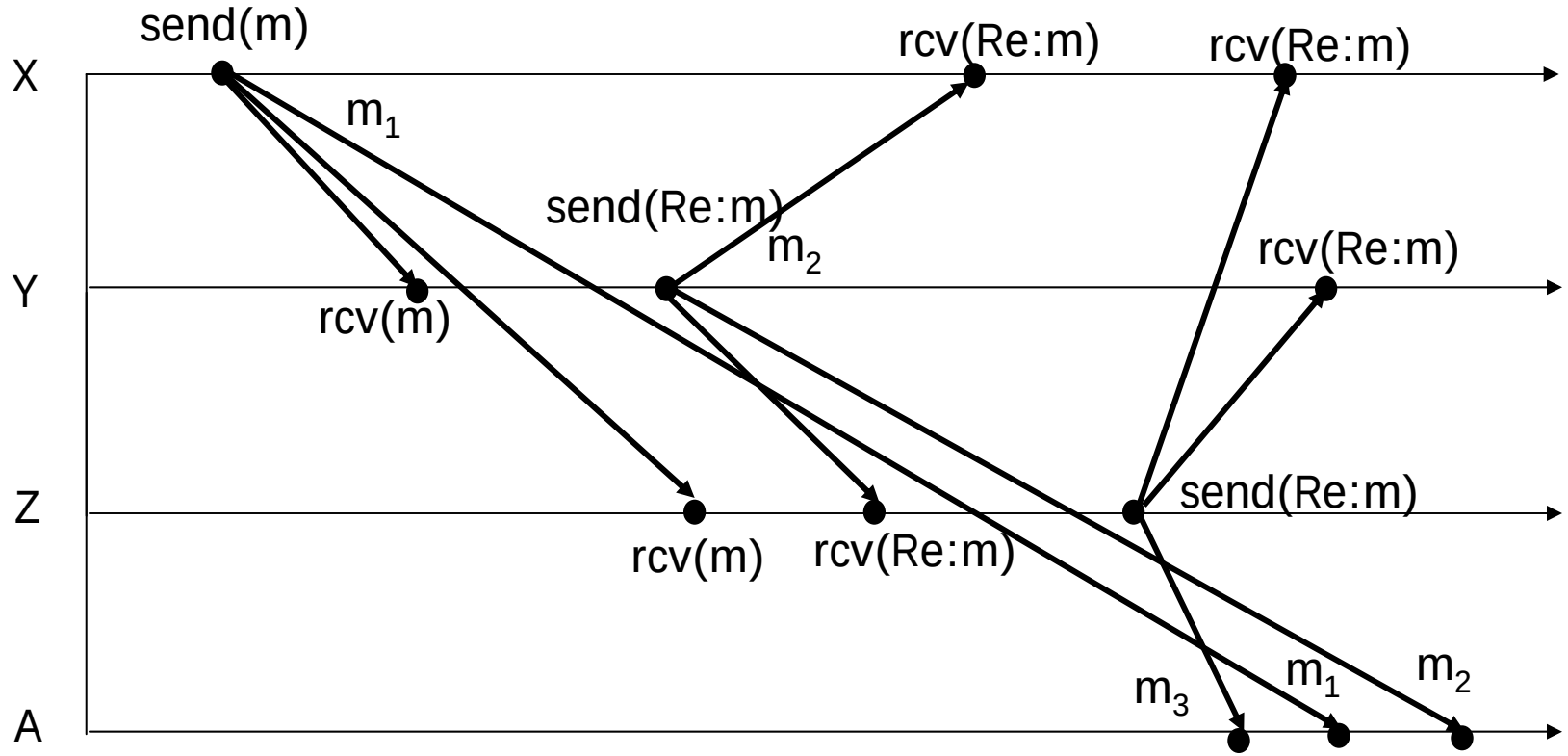
Logiske klokker

- Prinsipp
  - Hvis to hendelser skjer i samme prosess, da forekommer de i den rekkefølgen som prosessen observerte dem
  - Når en melding sendes mellom to prosesser, vil hendelsen "send meldingen" alltid skje før hendelsen "motta meldingen"
- *Hendte-før* relasjonen
  - fremkommer ved å generalisere de to relasjonene over slik at hvis x, y og z er hendelser og x "hendte før" y og y "hendte før" z, da har vi at x "hendte før" z
- logiske klokker utvider idéen over
  - mer seinere i kurset (kap. 10)

Eksempel: e-post utveksling

d

- Grafisk notasjon (diagram)



Feilmodell

- En beskrivelse av på hvilke måter feil kan forekomme i distribuerte system
- Beskrivelse av feilmodellen til en tjeneste muliggjør konstruksjon av en *ny* tjeneste som skjuler den feilaktige adferd til en tjeneste den bygger på
  - eksempel: TCP på toppen av IP
    - TCP: pålitelig byte-strøm
    - IP: upålitelig datagramtjeneste

Spesifikasjon av feilmodell

- Spesifikasjon av feilmodeller krever en måte å beskrive feil på.
- En tilnærming er å klassifisere feil-typer (Cristian, 1991) (Hadzilacos & Toueg, 1994)
  - Unnlatelsesfeil (omission failure)
  - Byzantisk feil (arbitrary failure)
  - Betimelighetsfeil (timing failure)

Omission failure

- En prosess eller kanal unnlater (pga feil) å utføre en aksjonen den er forventet å utføre

Feilklasse	Påvirker	Beskrivelse
Fail-stop	Prosess	En prosess stopper og forblir stoppet. Andre prosesser kan oppdage dens tilstand.
Crash	Prosess	En prosess stopper og forblir stoppet. Andre prosesser vil muligens ikke oppdage dens tilstand.
Omission	Kanal	En melding satt inn i et utgående meldings-buffer ankommer aldri mottakers inn-gående meldingsbuffer.

Omission failure (forts)

Feilklasse	Påvirker	Beskrivelse
Send-omission	Prosess	En prosess utfører en *send*-operasjon, men meldingen blir ikke lagt i det utgående meldingsbuffer.
Receive-omission	Prosess	En melding blir lagt i en prosess's inn-gående meldingsbuffer, men prosessen mottar den ikke.

Omission failure (forts)

- vanlig antagelse at en tjener har “**fail-stop**” feilmodell
  - tjeneren krasjer på en “pen” måte
    - den stopper helt
    - andre tjenere kan oppdage at den har feilet
  - dersom tjeneren likevel feiler på en annen måte, kan programvaren som bruker tjenesten, feile på uforutsigbar måte
- det er vanskelig å oppdage unnlatelsesfeil for prosesser i asynkrone system
  - “Two-army problem” eller “Agreement in Pepperland” i asynkron omgivelse der feil kan forekomme, har ingen løsning (se [Coulouris])

Byzantine failure

- Prosess eller kanal kan utvise vilkårlig adferd når den feiler,
  - sende/motta tilfeldige meldinger på vilkårlige tidspunkt
  - en prosess kan stoppe eller utføre “ulovlige” steg
  - en prosess kan unnlate å svare av og til
  - kan i tillegg være “ondsinnet”
- ved å ta utgangspunkt i byzantine failure, kan vi prøve å lage systemer som er “ultra-pålitelige” (handterer HW feil og gir garanterte svartider)
  - styresystemer i fly
  - pasientovervåkingssystemer
  - robot-kontroll
  - styresystemer for (kjerne)kraftverk

Betimelighetsfeil

- Betimelighetsfeil (Timing failure)
  - svar som ikke er tilgjengelig for en klient innen et spesifisert sanntidsintervall
  - betimelighetsgarantier krever garantert adgang til ressurser når det er behov for dem
- Eksempler:
  - kontroll- og overvåkingssystemer, multimedia systemer

Feilklasse	Påvirker	Beskrivelse
Clock	Prosess	Prosessens lokale klokke overskrider en grense for den avdriftsrate fra sann tid
Performance	Prosess	Prosessen overskrider maks tillatt tidsintervall mellom to prosesseringssteg
Performance	Kanal	Overføring av en melding tar lengere tid enn den erklærte øvre grense.

Oppsummering

- To typer systemmodeller
  - **Arkitekturmodeller**: definerer komponentene til systemet, den måten de interagerer på, og på hvilken måte de lokaliseres i et underliggende nettverk av datamaskiner
    - klient-tjener modeller (med varianter)
    - likeverdige prosesser
    - spontane nettverk
  - **Fundamentale modeller**: formell beskrivelse av egenskapene som er felles i alle arkitekturmodeller
    - interaksjonsmodeller
    - Feilmodeller
    - Sikkerhetsmodeller (ikke dekt i dette kurset, men se bl.a. INF3190)