Objekt-interaksjon

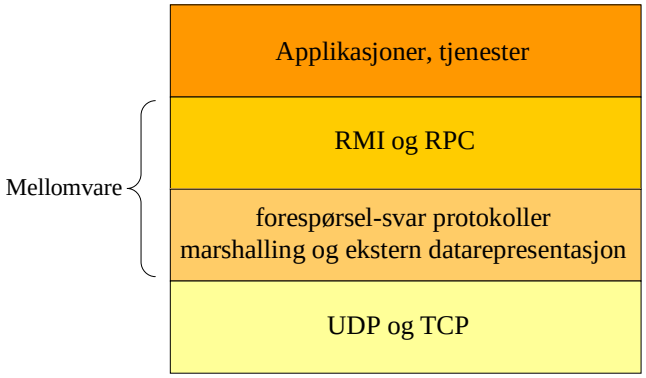
INF 5040 høst 2005

Foreleser: Frank Eliassen

Plan

- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

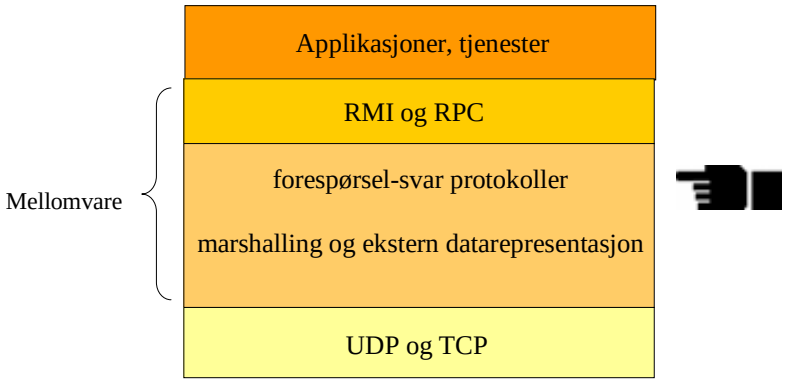
Lagdeling av mellomvare



Plan

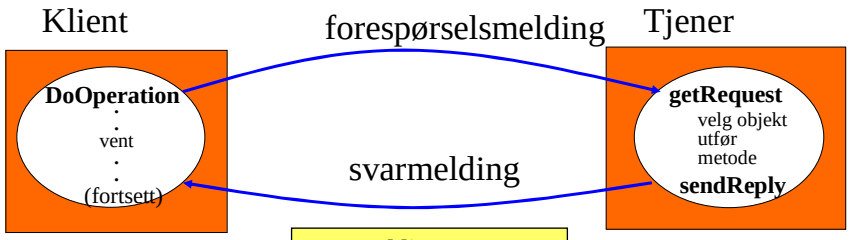
- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

Lagdeling av mellomvare



Klient-tjener kommunikasjon
Forespørsel-Svar (FS) protokoller

- Basert på UDP eller TCP



- **Meldingsstruktur**

meldingsType
forespørselId
ROR
metode
argumenter

Feilhandtering for FS protokoller over UDP (I)

- Protokollen kan utsettes for
  - omission failure
  - process crash failure
  - meldingsrekkefølge ikke garantert
- Feil oppdages som *timeout* i primitivet DoOperation:
  - tiltak avhenger av leveringsgarantiene som tilbys

Feilhandtering for FS protokoller over UDP (II)

- Timeout DoOperation
  - Send forespørsel-meldingen gjentatte ganger inntil
    - svar foreligger, eller
    - antar tjeneren har feilet (maks. antall retrans.)
- Duplikat forespørsel-meldinger
  - inntreffer når forespørsel-meldinger sendes mer enn en gang
  - kan føre til at operasjoner utføres mer enn en gang for samme forespørsel
  - => må kunne filtrere ut duplikat-forespørsler (jfr. forespørselId)
- Tapte svar-meldinger
  - tjeneren har allerede sendt svar-meldingen når den mottar en duplikat forespørselsmelding
  - => må kanskje utføre operasjonen på nytt for å få tak i svaret
    - OK for operasjoner som er "idempotent"

Feilhandtering for FS protokoller over UDP (III)

- Logger (historier):
  - benyttes av tjenere med operasjoner som *ikke* er "idempotent"
  - inneholder svarmeldinger som allerede er sendt
- Ulempe med logger:
  - lagerbehov
- dersom en klient kun kan gjøre et anrop av gangen til samme tjener, kan loggen begrenses i størrelse
- ved mottak av neste anrop fra samme klient, kan tjeneren slette siste svarmelding til denne klienten fra loggen

FS protokoller over TCP

- UDP har begrenset pakkestørrelse
  - => behov for multi-pakke-protokoller
- FS protokoller over TCP unngår dette problemet
  - TCP sørger for pålitelig levering av meldinger
- Problem:
  - mye overhead dersom forbindelse må opprettes ved hver forespørsel => behov for optimalisering (la forbindelsen stå)
  - begrensning på maks antall samtidige TCP-forbindelser kan skape problem
- Dersom begrenset meldingsstørrelse kan UDP med fordel brukes
  - enklere protokoll

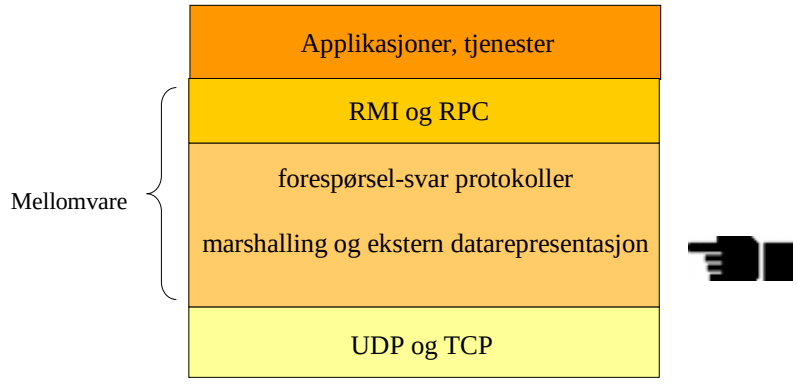
HTTP: en FS protokoll

- Brukes av web-browser klienter til å sende forespørsler til web-servere og motta svar
- Spesifiserer metodene, argumenter, og resultater samt regler for representasjon (marshalling)
  - Ikke objekt-basert
  - Fastsatt mengde med metoder (GET, PUT, POST, etc, etc)

method	*URL or pathname*	*HTTP version*	*headers*	*message body*
GET	//www.dcs.qmw.ac.uk/index.html	HTTP/ 1.1		

HTTP version	*status code*	*reason*	*headers*	*message body*
HTTP/1.1	200	OK		resource data

Lagdeling av mellomvare

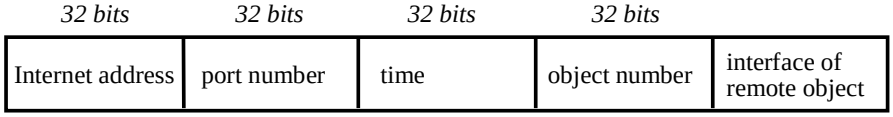


Marshalling
Ekstern datarepresentasjon

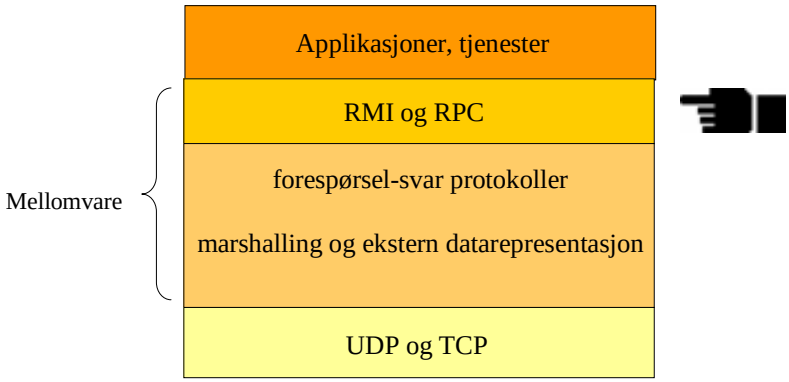
- "marshalling"
  - avbilde datastrukturer til meldinger (sekvens av dataverdier)
  - oversette sekvensen av dataverdier til en ekstern representasjon
- "unmarshalling"
  - inverse av "marshalling"
- Ekstern datarepresentasjon
  - en representasjon av dataene under overføringen av meldingen
  - Sun XDR (representasjon av mest brukte datatyper)
  - ASN.1/BER (ISO standard, basert på "type-tags", åpen)
  - NDR (benyttes i DCE RPC)
  - CDR (benyttes i CORBA RMI, binær layout for IDL typer)
  - Java Object Serialization (JOS)
  - XML (benyttes i SOAP)

Marshalling
Objekt-referanser

- Remote-object-reference (ROR)
  - Identifikator for fjernt objekt som er gyldig utover i et distribuert system
  - Må genereres på en måte som sikrer entydighet over tid og rom (en ROR kan ikke gjenbrukes)
  - Eksempel:



Lagdeling av mellomvare



Klassifikasjon av FS protokoller

- Klassifikasjon etter (Spector, 1982):
  - benyttes til å implementere ulike typer RMI (og RPC)
- Request (R) protokoll
  - Kun Request-melding. Ingen svarmelding fra tjener.
  - Ingen bekreftelse på at operasjonen er utført.
- Request-Reply (RR) protokoll
  - Reply-meldingen bekrefter at Request-meldingen er utført
  - Et nytt kall fra klienten bekrefter mottak av Reply-meldingen
- Request-Reply-Acknowledge (RRA) protokoll
  - egen melding fra klient for å bekrefte mottak av Reply-melding
  - tillater tap av Ack-melding
    - Ack med gitt *forespørselId* bekrefter alle lavere forespørsel*Id*

RMI forespørselsemantikk

- Påliteligheten til ulike RMI typer under partielle feilsituasjoner:

	Feilhandteringstiltak		RMI anropssemantikk
Retransmisjon av Request melding	Dupliseringsfilter	Utføre metoden på nytt *eller* sende Reply meldingen på nytt	
Nei (R)	____	____	Kanskje
Ja (RR)	Nei	Utfør metoden på nytt	Minst-en-gang
Ja (RR)	Ja	Send Reply meldingen på nytt	Høyst-en-gang

RMI forespørselsemantikk i CORBA og Java

- RMI i CORBA og Java har "høyst en gang" forespørselsemantikk/pålitelighet under partielle feilsituasjoner.
  - kalles synkrone forespørsler (synchronous requests)
- Andre former for synkronisering gir annen feilsemantikk
  - One-way operations: kanskje-semantikk

Implementasjon av RMI

- Tre hovedoppgaver:
- Grensesnittprosessering
  - Integrasjon av RMI mekanismen i et programmeringsspråk
  - Grunnlaget for å realisere aksesstransparens
- Kommunikasjon
  - meldingsutveksling (forespørsel/svar protokoll)
- Binding, lokalisering, og aktivisering
  - Lokalisere en passe tjenerprosess som innkapsler fjernobjektet
  - Aktivisering av objekt-implementasjon
  - Grunnlaget for å realisere lokasjonstransparens

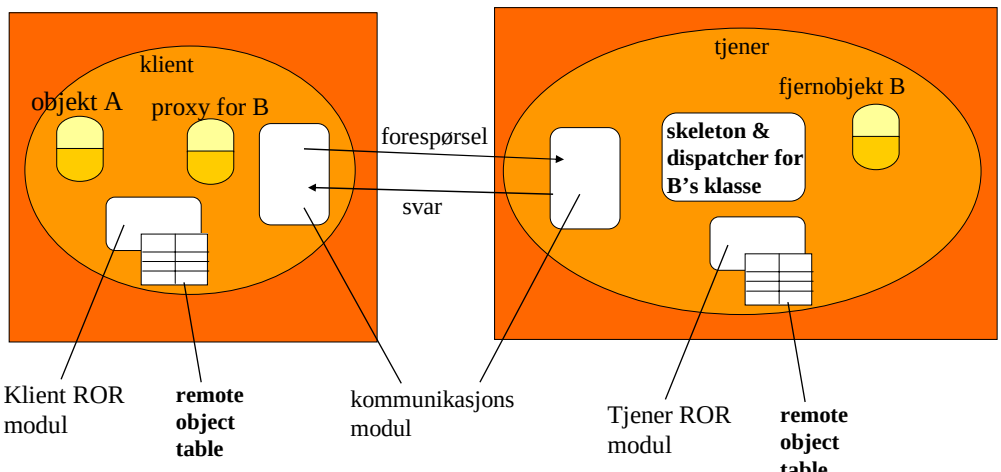
RMI grensesnittprosessering

- Klient proxy
  - Lokalt “proxy” objekt for hvert fjernobjekt en klient kan aksessere (“stand-in” for fjernobjekt).
  - Opprettes automatisk av mellomvaren.
  - Klassen til proxy-objektet har samme grensesnitt som klassen til det fjerne objektet. Kan utføre typesjekking
  - Utfører marshalling av forespørsel og unmarshalling av svar
  - Sender forespørsel-melding til tjener
  - Proxy-klassen genereres av stub-kompilator basert på IDL beskrivelser

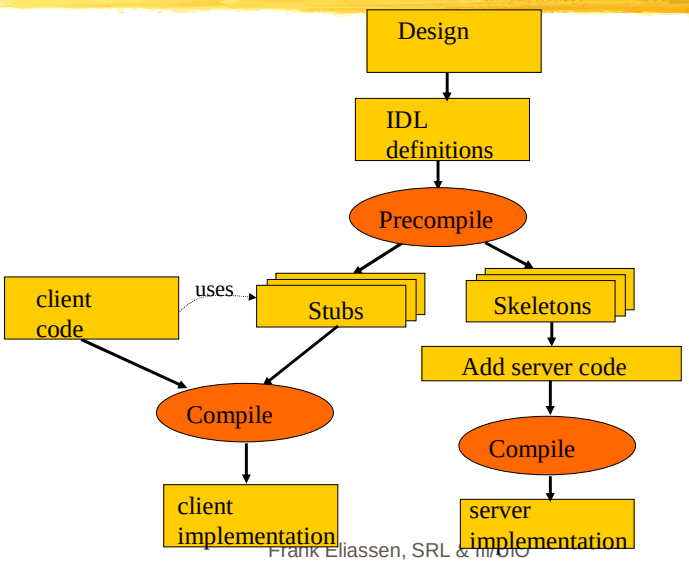
RMI grensesnittprosessering

- Tjener skeleton
  - Et fjernobjekt som kan aksesseres via RMI, har et “skeleton”-objekt.
  - Mottar forespørsler og omgjør de til lokale metodekall vha. en *dispatcher* som velger skeleton og metode basert på *ROR* informasjon og metodenavn
  - Utfører unmarshalling av forespørsel og marshalling av svar
  - Sender svarmeldinger tilbake til klient
  - Skeleton-klassen genereres av stub-kompilator basert på IDL beskrivelser

Proxy og skeleton i RMI



Generering av proxy og skeleton i RMI



RMI binding, lokalisering, og aktivisering

- Binding i RMI
  - tilsvarer å avbilde et symbolsk objektnavn til ROR
  - Utføres av en “binder” (navnetjeneste el.lign.)
  - (tema for seinere forelesning: navngiving og trading)
- Lokalisering i RMI
  - tilsvarer å avbilde en ROR til en kommunikasjonsidentifikator
- Aktivisering i RMI
  - tilsvarer å opprette et aktivt objekt fra et tilsvarende passivt objekt (f.eks. på forespørsel). Utføres av en *activator*
    - registrere passive objekt som er tilgjengelig for aktivisering
    - aktivisere tjenerprosesser (og aktivisere fjernobjekt i dem)
  - Eksempler seinere (CORBA og Java)

Lokalisering i RMI

- Lokalisering i RMI : tilsvarer å avbilde en ROR til en kommunikasjonsidentifikator
  - integrert i ROR
    - Adressen kan ekstraheres direkte fra objekt-referansen
  - Lokaliseringstjeneste
    - En navnetjeneste benyttes av klient proxy ved hvert anrop
  - cache/broadcast
    - Hver klient har cache av bindinger (ROR, komm.identifikator)
    - Dersom ROR ikke finnes i cache, utføres broadcast med ROR
    - Tjenere som har objektet svarer med komm.identifikator
  - foroverpekere eller adressehint (til f.eks lokaliseringstjeneste)
    - Benyttes ved objekt-migrering

Plan

- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

Objekt-tjenere: Tjenere som er skreddersydd til å understøtte distribuerte objekter

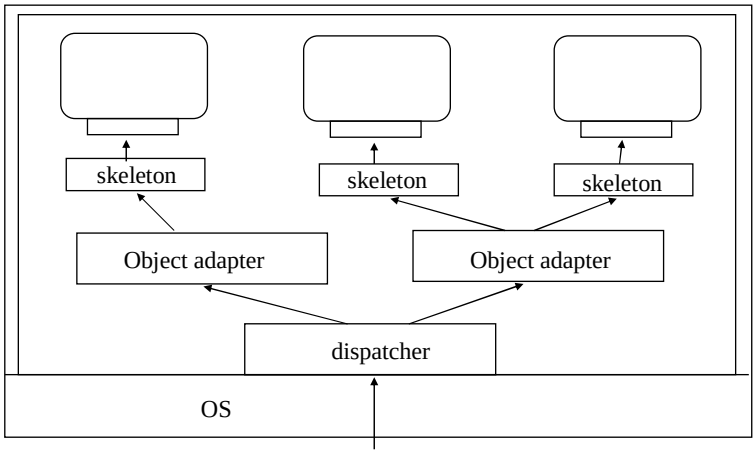
- Tjenester realiseres ved objekter som tjeneren innkapsler
  - Tjenester legges til eller fjernes ved legge til eller fjerne objekter
- Objekt-tjenere opptrer som steder der objekter kan leve
- Flere måter å aktivisere et objekt på

Objekt-tjenere må tilordne prosesserings-ressurser til objekter når de aktiviseres

- Når et objekt aktiviseres, hvilke prosesserings-ressurser skal implementasjonen av det tilordnes?
- Aktiviserings-policy
  - En bestemt måte å aktivisere et objekt på
  - Ulike dimensjoner
    - Hvordan oversette mellom ROR og lokal implementasjon?
    - Skal tjeneren være enkelt-trådet eller fler-trådet?
    - Dersom fler-trådet, hvordan tilordne tråder til objekter og forespørsler? En pr objekt? En pr forespørsel?
    - Transiente vs persistente objekter, etc
- Objekt-tjenere bør understøtte flere aktiviserings-policies
- Objektene kan grupperes etter hvilken aktiviserings-policy de er underlagt

Organisering av objekt-tjener som støtter forskjellige aktiviserings-policies

- **Objekt-adapter:** programvare som implementerer en spesifikk aktiviserings-policy



Plan

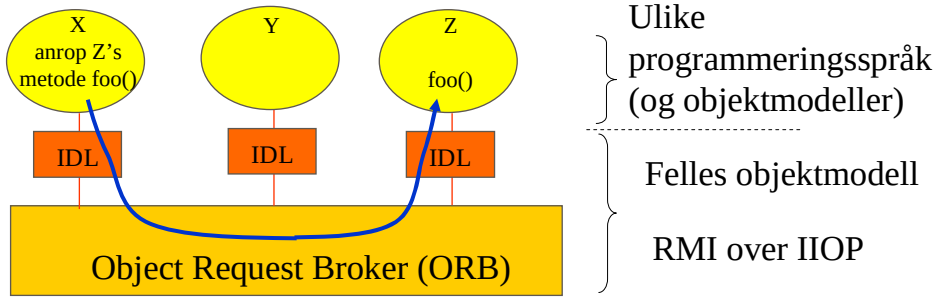
- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

CORBA mellomvare

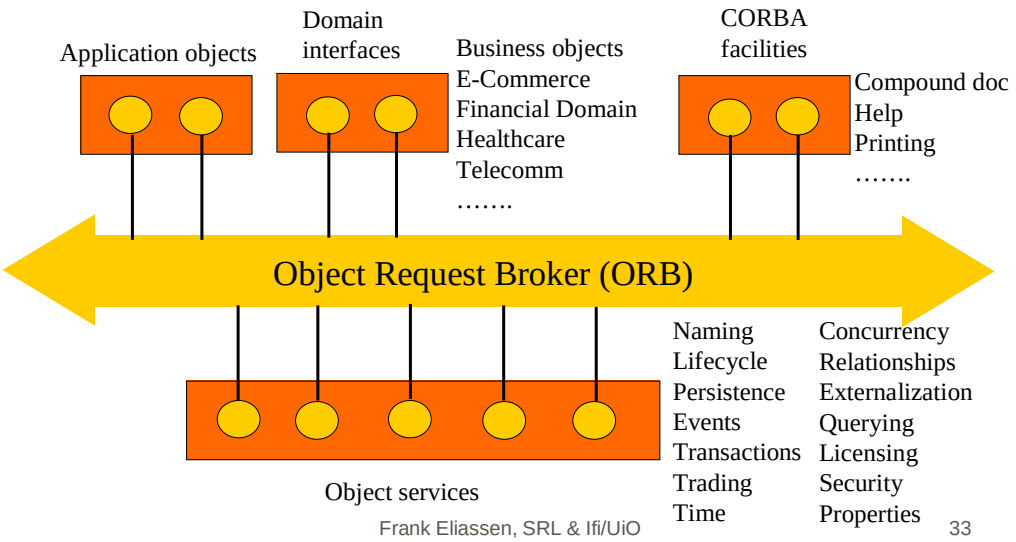
- Tilbyr mekanismer som gjør at objekter kan sende metodekall og motta svar på en transparent måte
  - lokasjontransparens
  - aksesstransparens
- kjernen i arkitekturen er en Object Request Broker (ORB)
- Spresifikasjon utviklet av medlemmene til Object Management Group

CORBA RMI

Klienter kan kalle metoder til fjerne objekter uten bekymring for:
objekt-lokasjon, programmeringsspråk,
operativsystem-plattform, kommunikasjons-
protokoller og maskinvare.



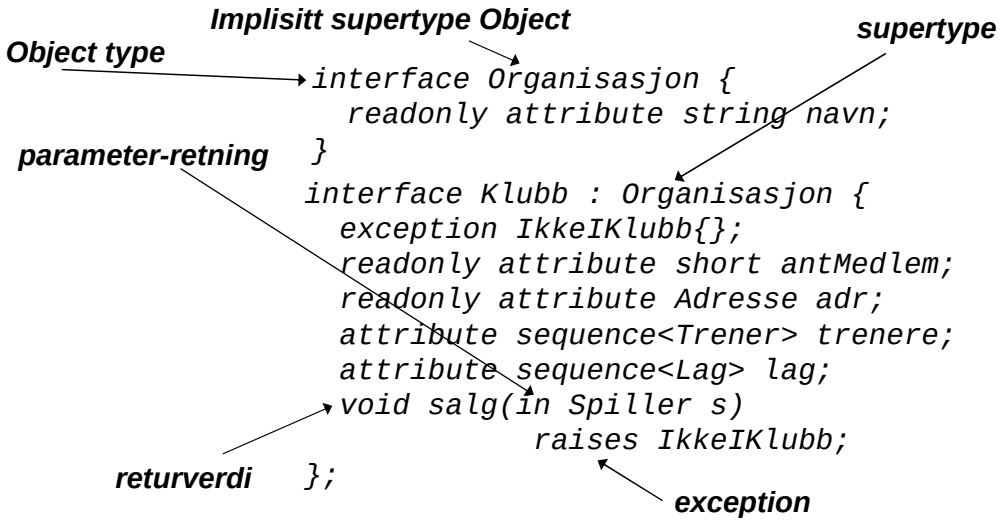
CORBA Tjenester



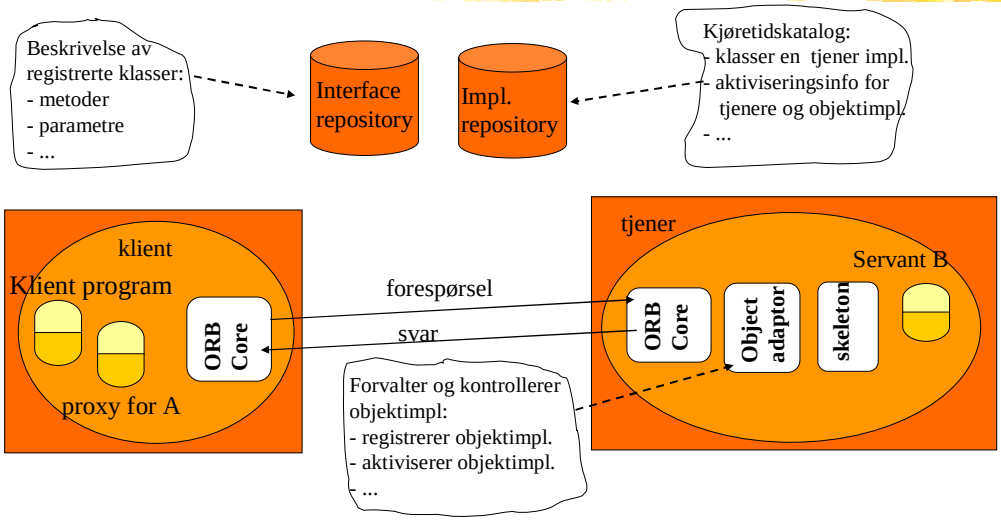
CORBA IDL

- Språk for å definere CORBA objekt-typer
- Kan uttrykke alle begreper i CORBA objekt modell
- OMG/IDL er
  - ikke avhengig av bestemt programmeringsspråk
  - syntaktisk orientert mot C++
  - ikke beregningsmessig fullstendig
- Forskjellige bindinger til programmeringsspråk tilgjengelig

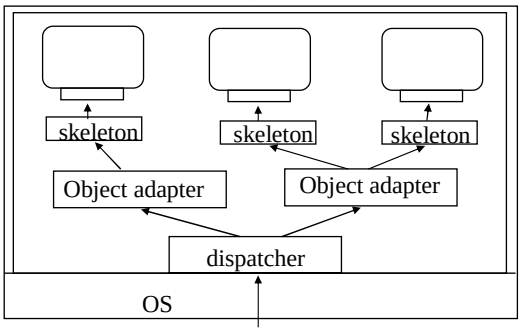
CORBA objektmodell og IDL



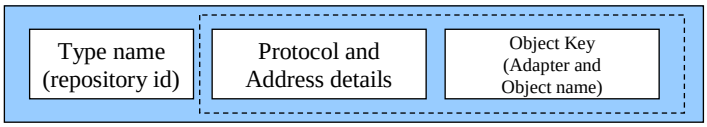
CORBA arkitektur



CORBAs ROR format (IOR) reflekterer organiseringen av objekt-tjenere

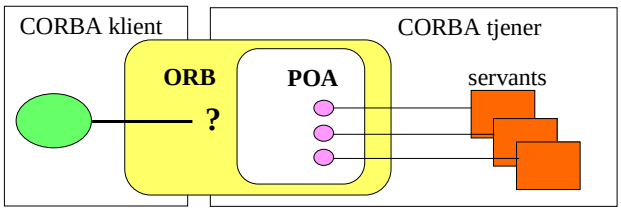


CORBA Interoperable Object Reference (IOR)



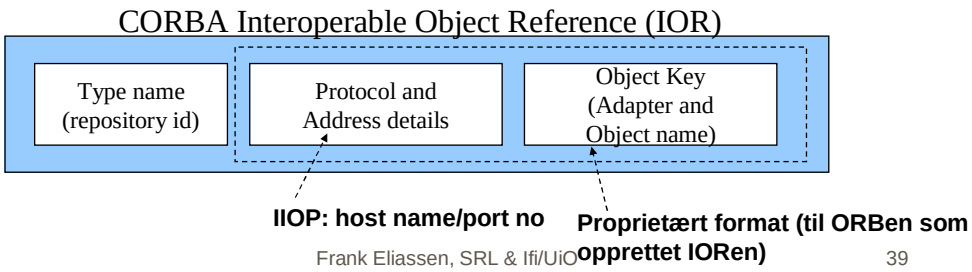
CORBA
Portable object adapter (POA)

- Muliggjør portabilitet av objektimplementasjoner mellom ulike ORBer
- Understøtter lettvekts transiente objekter og persistente objektidentifikatorer (f.eks. For objekter lagret i databaser)
- Tillater tjenere å implementere mange objekter
- Understøtter transparent objekt-aktivisering
- Utvidbar mekanisme for aktiviseringspolicies
- Flere POAer i en tjener



CORBA RMI Lokalisering

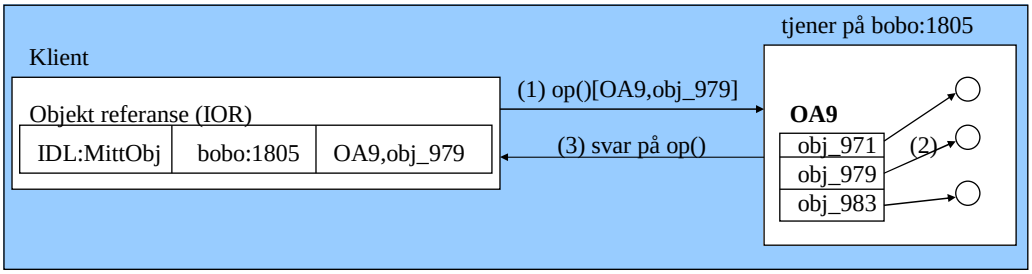
- Lokalisering i RMI tilsvarer å avbilde objektreferansen (ROR) til "servant"
  - servant: implementasjonen av et (eller flere) CORBA objekt
- ROR I CORBA kalles: Interoperable Object Reference (IOR)
- Lokaliseringssprosessen:
  - er basert på informasjon kodet i objekt-referansen



CORBA RMI lokalisering over IIOP

- Transient IOR
  - Virker bare så lenge den tilsvarende tjener-prosessen er tilgjengelig
  - Etter at tjener-prosessen er terminert, vil IOREn aldri virke mer
  - Tjenerprosessens lokasjon er kodet i objektets IOR
- Persistent IOR
  - fortsetter å virke (betegner samme CORBA objekt) selv om tjener-prosessen terminerer og starter på nytt.
  - En *activator* (implementation repository) kan automatisk starte en tjener-prosess når en klient benytter en persistent objekt referanse og terminere den igjen etter en viss ledig tid
  - Lokasjonen til *activator* er kodet i objektets IOR. Tjener-prosessens faktiske lokasjon må oppklares via activator.
  - En persistent POA (lager persistente IOR) med registreres hos activator.

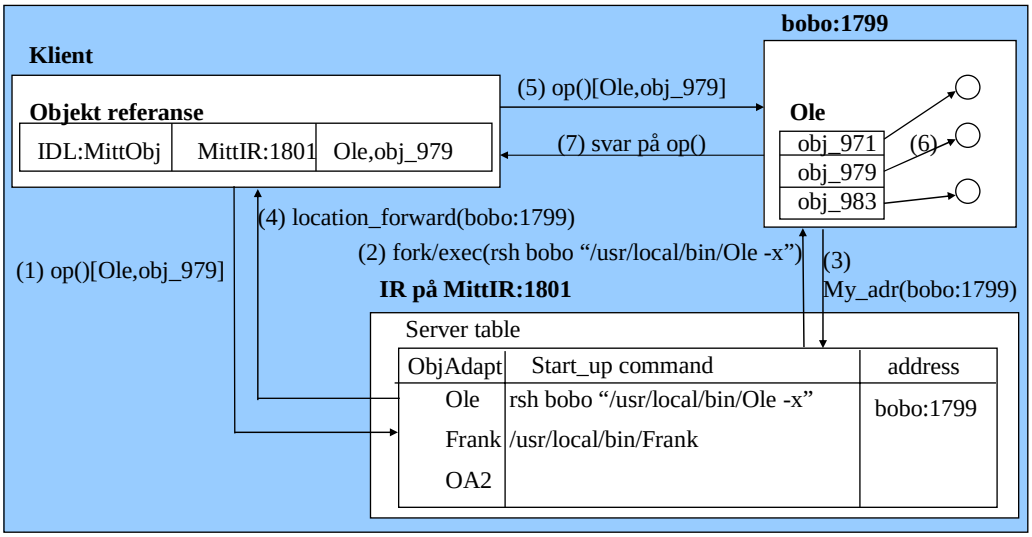
Lokalisering av transient IOR over IIOP



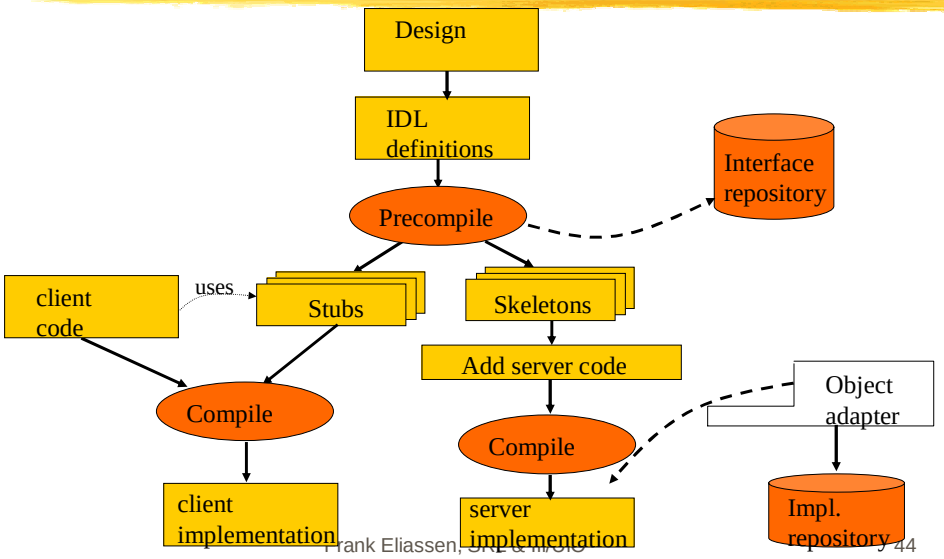
Lokalisering av persistent IOR over IIOP

- Benytter Implementation Repository (IR) som activator.
- IR:
  - handterer prosess/tråd-oppretting og -terminering, m.v.
  - er ikke portabel (spesifikk for en ORB implementasjon)
  - ikke standardisert
    - spesialsydd for spesifikke omgivelser
    - ikke mulig å skrive spesifikasjoner som dekker alle omgivelser
    - kommunikasjon mellom en ORB og dens IR er ikke synlig
  - Objekt migrering, skalering, ytelse, og feil toleranse er avhengig av IR
  - Implementerers vanligvis som en prosess på en fast adresse
    - vertsmaskiner som er konfigurert under samme IR utgjør et “location domain”

Lokalisering av persistent IOR over IIOP



Utvikling av CORBA applikasjon Generering av stubs/skeletons



Plan

- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

Java Remote Method Invocation (RMI)

Java RMI

- Remote Method Invocation (RMI) understøtter kommunikasjon mellom forskjellige Java Virtual Machines (VM), og eventuelt over et nettverk
- Gir tett integrasjon med Java
- Minimaliserer endringer i Java language/VM
- Virker i homogene omgivelser (Java)
- RMI-IIOP muliggjør kommunikasjon med CORBA objekter
  - Automatisk oversetting Java "Remote Interface" til CORBA IDL
- Klient kan implementeres som *applet* eller *application*

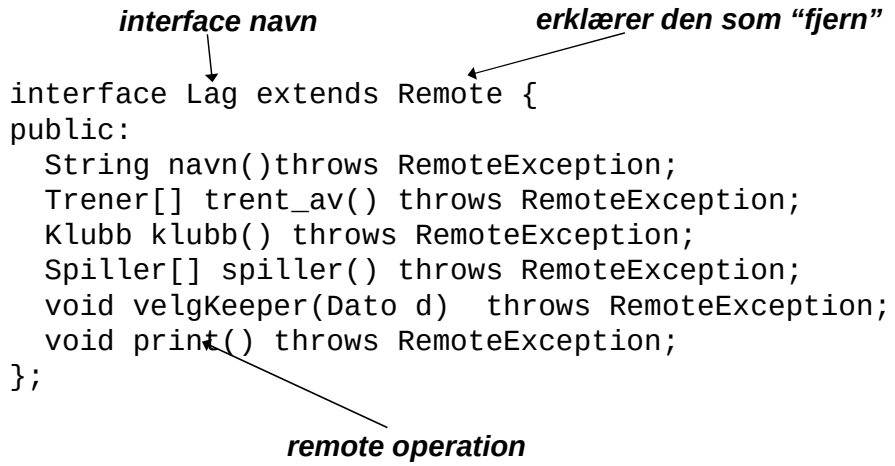
Java Objekt Modell

- Grensesnitt (interfaces) og fjerne objekt (Remote Objects)
- Klasser
- Attributter
- Operasjoner/metoder
- Exceptions
- Arv

Java grensesnitt og fjerne objekt

- Basert på Javas vanlige grensesnitt-begrep
- RMI har ikke separat språk (IDL) for å definere grensesnitt
- Predefinert grensesnitt `Remote`
- All RMI kommunikasjon er basert på grensesnitt som utvider (extends) java.rmi.Remote
- Fjerne klasser ("remote classes") implementerer remote grensesnittet
- Fjerne objekter ("remote objects") er instanser av fjerne klasser

Java fjerngrensesnitt: Eksempel



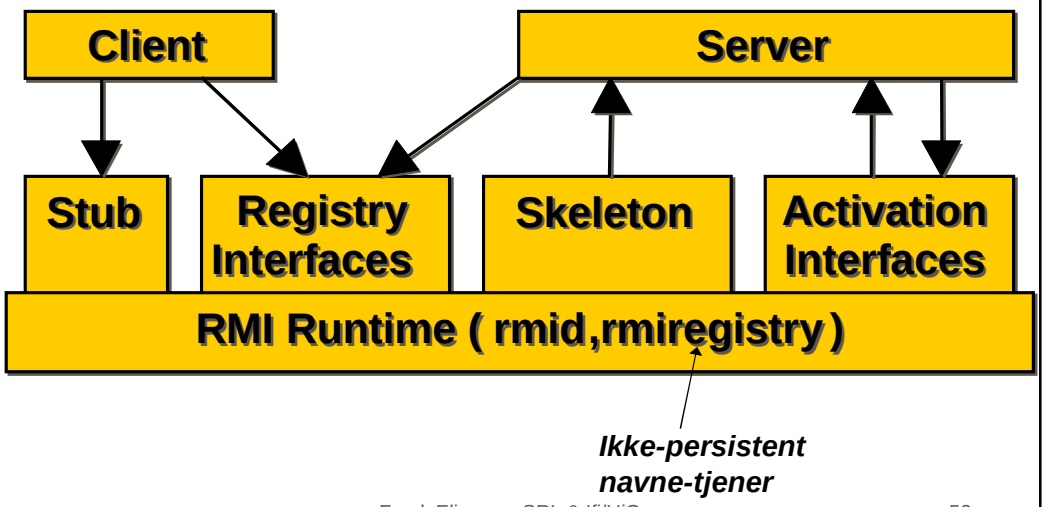
RMI parameteroverføring

- Atomiske typer overføres *by value*
- Fjerne objekt overføres *by reference*
- Ikke-fjerne objekt overføres *by value*

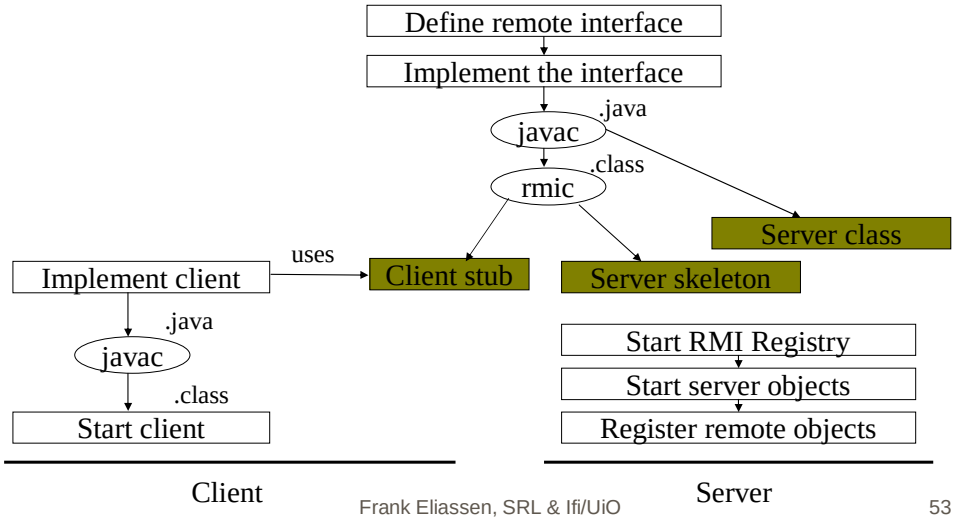
```
class Adresse {
  public:
  String gate;
  String Postnr;
  String Poststed;
};
interface Klubb extends Organisasjon, Remote {
  public:
  Adresse adr() throws RemoteException;
  ...
};
```

Returnerer en kopi av Adresse-objektet

Arkitektur til Java RMI



Java RMI utviklingsprosess



Plan

- Prinsipper for realisering av fjernmetodekall (RMI)
- Objekt-tjenere
- CORBA RMI
- Java RMI
- Fler-trådede objekt-tjenere

Objekt-tjenere må tilordne prosesserings-ressurser til objekter når de aktiviseres

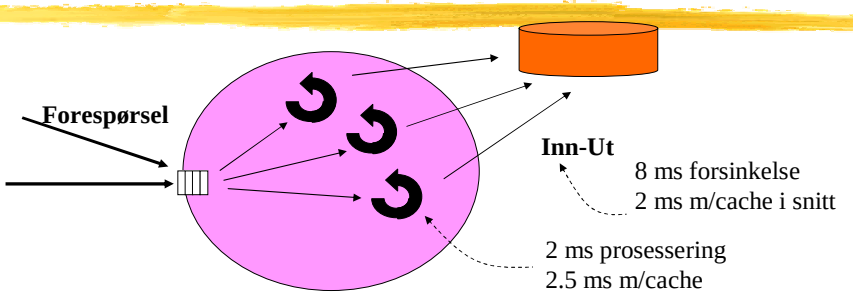
- Når et objekt aktiviseres, hvilke prosesseringsressurser skal implementasjonen av det tilordnes?
  - Skal en ny prosess eller tråd opprettes?
  - Finnes mange måter dette kan gjøres på?
  - Finnes det én beste måte?
- Hvordan bruke prosesser og tråder i distribuerte system?
  - Bedre ytelse
  - Bedre utnyttelse av ressurser

Tråder er billigere å forvalte enn prosesser

- kjøretidsomgivelser (adresserom) er dyre å opprette
- tråder er mye billigere å opprette og terminere
- tråder kan dele kjøretidsomgivelser
  (f.eks. globale variable, åpne filer, timere, signal, ...)

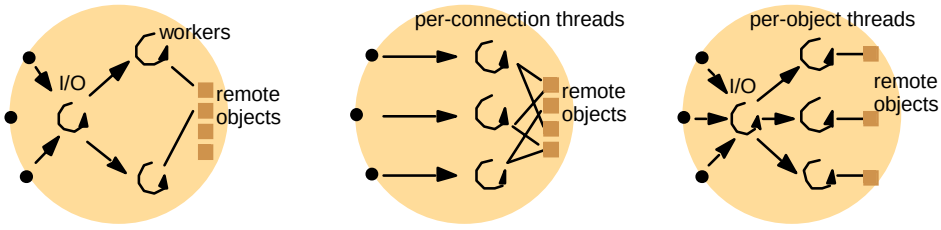
- prosesser definerer *beskyttelsesdomener*
  - en tråd i en prosess kan (normalt) ikke aksessere data og ressurser i en annen prosess

Flertrådet tjener



	# prosessorer	# tråder	Maks. anrop/sekund
ingen disk caching	1	1	100
ingen disk caching	1	2	125
disk caching	1	2	400
disk caching	2	2	444
disk caching	2	3	500

Alternative tråd-policies for objekt-aktivisering



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

Oppsummering

- Implementasjon av RMI
  - proxies, skeletons, dispatcher
  - grensesnittprosessering, binding, lokalisering, aktivisering
- Kallsemantikk
  - Kanskje, minst-en-gang, høyst-en-gang
  - Pålitelighet til RMI er i beste fall “høyst-en-gang”
- Prinsipper for CORBA
  - Klienter kaller metoder til fjerne objekter uten bekymring for objekt-lokasjon, programmeringsspråk, operativsystem-plattform, kommunikasjonsprotokoller og maskinvare.
- Prinsipper for Java RMI
  - Tilsvarende som CORBA men avgrenset til Java-miljø
- Flere-trådede tjenere
  - kan i noen tilfeller benyttes til å øke gjennomstrømningen (metodekall/tidsenhet) dersom f.eks. I/O er flaskehalsen