



Aktuelle SW teknologier for distribuerede systemer

Dansk Selskab for Datateknik

Kommunikationskonference - 29. maj 2001

Finn Overgaard Hansen

Ingeniørhøjskolen i Århus - www.ih.dk


Elektro- og IKT-afdelingen

foh@e.ih.dk



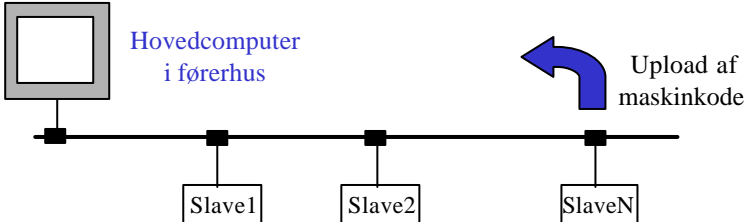
Agenda

- Introduktion til distribueret softwareteknologi
- Middleware teknologier:
 - CORBA
 - Java RMI
 - DCOM
- Auto-discovery teknologier
 - Jini, UPnP, Salutation, HAVI
- Opsummering




En case på et distribueret system

- Dronningborg mejetærsker (ca. 1986/87)
 - Der kunne tilsluttes nye ”devices” til en fælles bus - uden at hovedcomputeren skulle omprogrammes



- Løsning:
 - Hver device uploadede den nødvendige maskinkode - til hovedcomputerens RAM lager med display rutiner mm.
 - Samme maskinkode i hovedcomputer som i satellit microprocesserne
 - Definerer af faste memorygrænser for upload af nye programmer
 - En simpel low cost løsning


3



Hvorfor distribuering ?

- Distribuering i problemområdet
 - Vejrprognosesystem: Central server med decentrale målestationer (klienter) rundt omkring i landet
 - Pervasive computing: mobile trådløse enheder
 - Bærbare computere, mobiltelefoner, PDA'er
- Distribuering valgt som systemarkitektur
 - For at opnå fleksibilitet og større genbrug
 - For at spare ”wiring” og opnå en bedre pålidelighed
 - For at opnå nødvendig ”performance”
 - For at kunne integrere forskellige platforme og programmeringssprog


4



Hvad vi gerne vil opnå

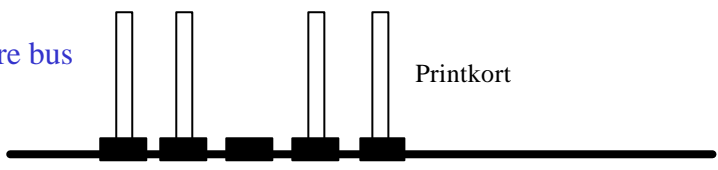
- Software udviklere vil gerne:
 - Kunne programmere distribuerede systemer uden at skulle programmere hele kommunikationslaget
 - Kunne kalde lokale og remote objekter på samme måde
 - Kunne genbruge eksisterende kode skrevet i forskellige sprog
- Brugere og installationsfolk vil gerne:
 - kunne tilslutte nye enheder uden en omfattende konfigurering
 - nye typer af mobile "devices" forudsætter "plug and play" for at være anvendelige i praksis

5



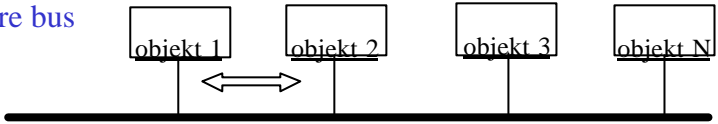
Løsning: Software bussen

Hardware bus



Veldefineret grænseflade: elektrisk og mekanisk


Software bus



Programmøren ønsker at kunne skrive følgende ved kodning af objekt 1:

`pObjekt2->print(file)`


6



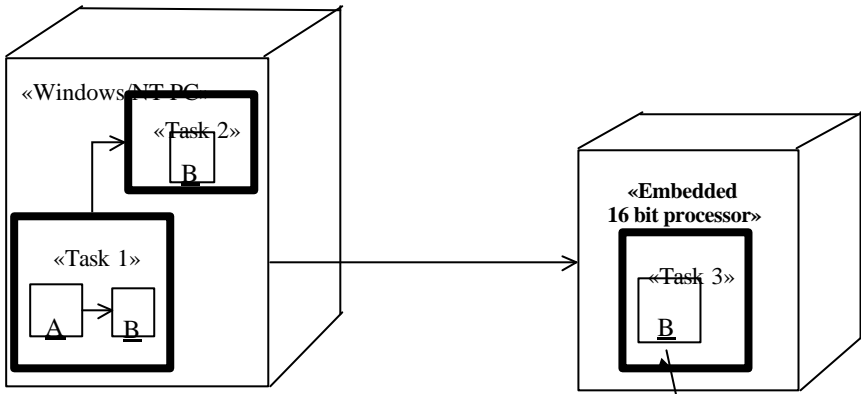
To former for kommunikation

- **Synkron kommunikation**
 - **RPC - Remote Procedure Call**
 - En klient kan kalde procedurer i server programmer, der kører i andre processer eller på andre computere
 - **Remote Method Invocation (RMI)**
 - En objektorienteret udgave af **RPC**
 - Objekter i forskellige processer kan kommunikere ved at kalde hinandens operationer (metoder) (**RMI anvendes her som generel term**)
- **Asynkron kommunikation**
 - **Event - notification**
 - Et objekt kan abonnere på at få en notification fra et andet objekt når en bestemt hændelse (event) indtræffer
 - Kaldes også for publish/subscribe

7



Objektorienteret kommunikation




Objekt A ønsker at kalde operationer i objekt B på samme måde uanset om objekt B er i:

- samme task (Task1),
- et andet Task 2
- eller befinder sig på en helt anden processor

Kan indkapsle ikke OO kode som f.eks. C som et objekt


8



Lokations transparens

- Ved RMI:
 - Et klientobjekt der kalder et serverobjekt behøver ikke at vide om det er et lokalt objekt eller et objekt på en anden maskine
- Ved event baseret kommunikation:
 - objekter der generer hændelser og de objekter der modtager *notifications* herpå behøver ikke at kende hinandens lokation
- I praksis:
 - Man tilstræber at kaldesyntaksen er den samme
 - Klienter bør dog vide om det er et lokalt objekt eller et remote objekt f.eks. er fejlhåndteringen forskellig

9



Remote interfaces

Anvendelse af interfaces er en fundamental teknik:

- kontrakter mellem klient og server

Local client objekt

→

Remote interface 1

 op1(parameters)
 op2(): return value
 op3(parameters): return value

←

Implementerer Remote interface 1

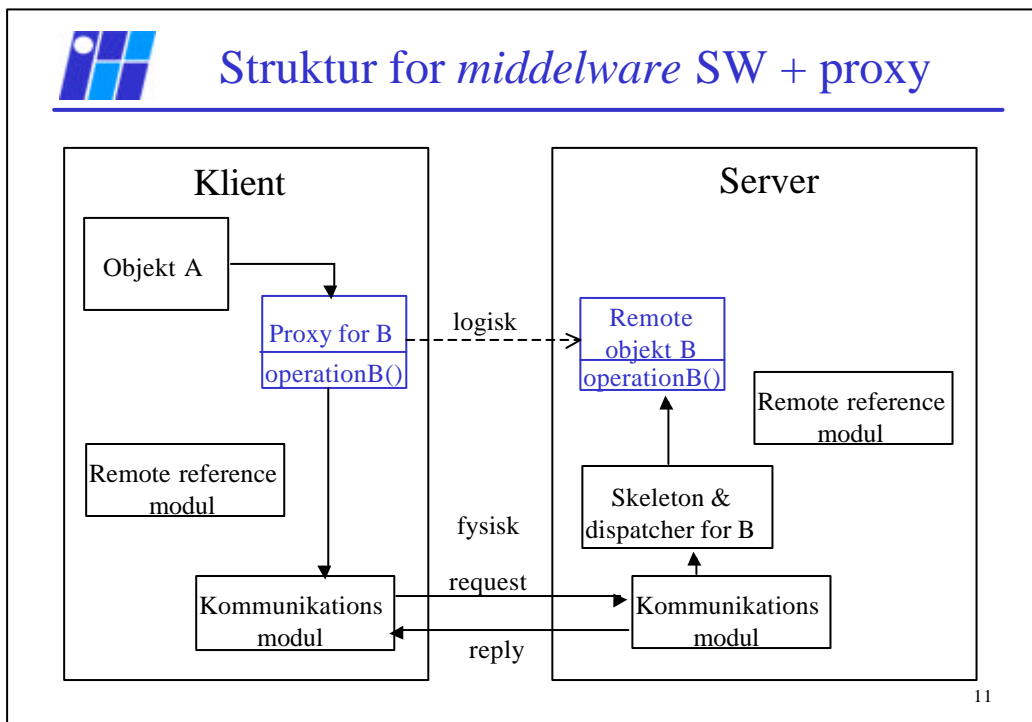
Remote class

 private data

 op1(parameters)
 op2(): return value
 op3(parameters): return value
 op4()
 op5(): return value
 op6(paramter)

kan f.eks. kaldes lokalt

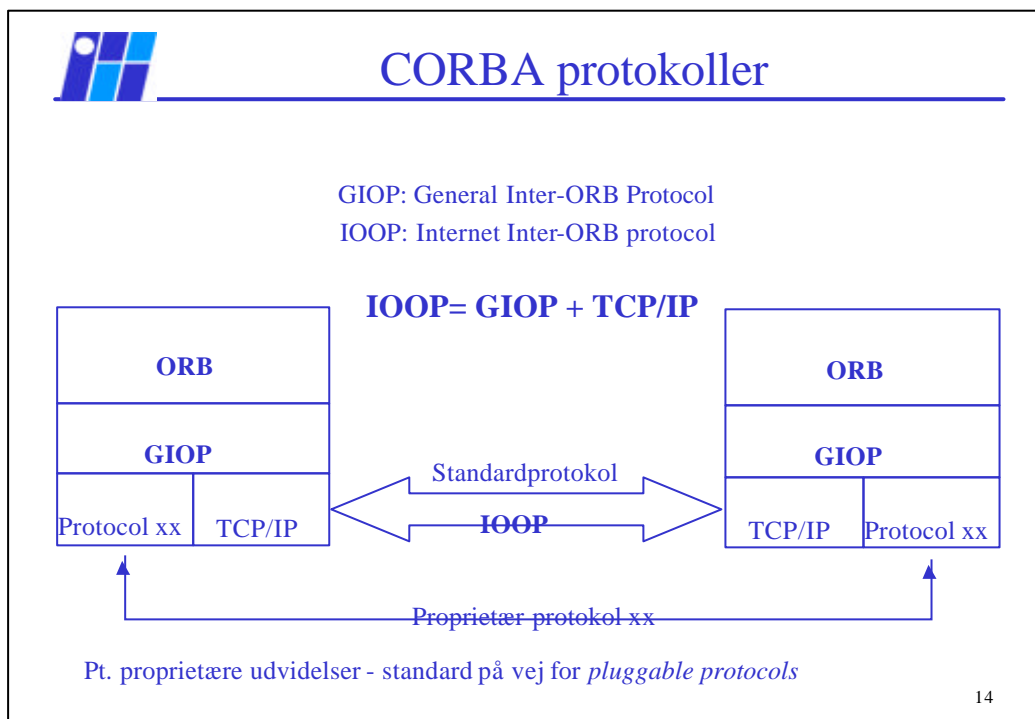
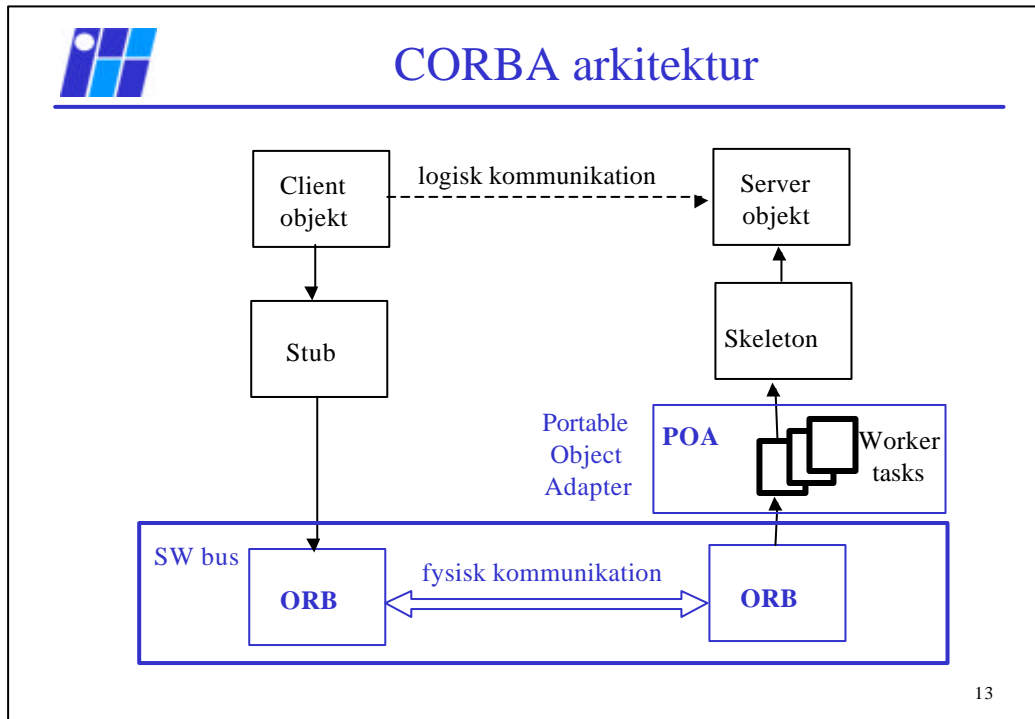
10



CORBA

- CORBA (**C**ommon **O**bject **R**equest **B**roker **A**rchitecture)
 - En **OMG standard** (Object Management Group) introduceret i 1991 - CORBA 2.0 i 1996 - CORBA 3.0 er på vej.
 - OMG er en sammenslutning af ca. 800 firmaer
 - CORBA er tilgængeligt på over **50 operativsystemer** (f.eks. Windows9x/NT/2000, UNIX, Linux og embedded RTOS)
 - Supporterer **syv programmeringssprog** (C++, Java, C, Ada, COBOL, Smalltalk og Lisp)
 - **Central mekanisme: ORB - Object Request Broker**
 - Plus et stort antal CORBA services
 - f.eks. Naming, Event & Notification, Concurrency, Persistency
- Reference: www.omg.org

12



IDL: Interface Definition Language

- Alle interfaces til objekter defineres i IDL
 - IDL ligner klassedefinitioner i C++ og Java
 - IDL filen oversættes til det aktuelle sprog vha. IDL kompilere

```

    graph LR
      IDL[IDL definitions file] --> idl2cpp((idl2cpp compiler))
      IDL --> idl2java((idl2java compiler))
      idl2cpp --> C++_stub[C++ stub kode]
      idl2cpp --> C++_skeleton[C++ skeleton kode]
      idl2java --> java_stub[java stub kode]
      idl2java --> java_skeleton[java skeleton kode]
    
```

15

Maskinovervågnings eksempel

Logisk model

```

            classDiagram
                class MachineMonitoring {
                }
                class AlarmManager {
                    +receiveAlarm(Alarm)
                    +cancelAlarm()
                }
                class AlarmReceiver {
                }
                MachineMonitoring "*" -- "1" AlarmManager
                MachineMonitoring ..|.. AlarmReceiver
            
```

Fysisk model

```

            graph LR
                subgraph EmbeddedProcessor [«Embedded processor»]
                    MM[":Machine Monitoring"]
                    ORB1[ORB]
                end
                subgraph WindowsNTPC [«Windows/NT PC»]
                    AM[":AlarmManager"]
                    ORB2[ORB]
                end
                EmbeddedProcessor -- «TCP/IP» --- WindowsNTPC
            
```

16

Eksempel: AlarmReceiver IDL file

```

classDiagram
    class AlarmManager {
        +receiveAlarm(Alarm)
        +cancelAlarm()
        +xx()
    }
    class AlarmReceiver
    AlarmManager <|-- AlarmReceiver
            
```

```

// IDL file example

enum AlarmType {Hardware, Memory, Network};
enum Priority {High, Medium, Low};

struct Alarm
{
    Priority priority;
    AlarmType alarmType;
    string message;
};

interface AlarmReceiver
{
    void receiveAlarm(in Alarm a);
};
            
```

17

Eksempel: kald af Java objekt fra C++

```

graph TD
    IDL[AlarmReceiver interface IDL file]
    subgraph Cplusplus [C++ klient objekt]
        MM[":Machine Monitoring"]
        Stub["C++ AlarmReceiver stub"]
        ORB_C[ORB]
        MM --> Stub
        Stub --> ORB_C
    end
    subgraph Java [Java server objekt]
        AM[":AlarmManager"]
        Skeleton["Java AlarmReceiver skeleton"]
        ORB_J[ORB]
        Skeleton --> AM
        ORB_J --> Skeleton
    end
    IDL -- "auto genereres af IDL compilere" --> Stub
    IDL -- "auto genereres af IDL compilere" --> Skeleton
    ORB_C <--> ORB_J
            
```

18



CORBA styrker/svagheder

- **Styrker:**
 - En åben standard, der er meget anvendt - og videreudvikles
 - Supporterer mange sprog pt. syv - let at tilføje flere
 - Uafhængig af operativsystem
 - Er venlig overfor Java (Java 2 er CORBA enabled)
 - Har et stort antal nyttige services
- **Svagheder:**
 - Kommercielle CORBA implementeringer er ekstremt dyre
 - Standard CORBA er ekstremt pladskrævende og komplekst
 - Virker ikke godt gennem firewalls
 - Dårlig opførelse i "disconnected" environments

Kilde: Mike Preradovic, Intrinsic Software, ESC2001

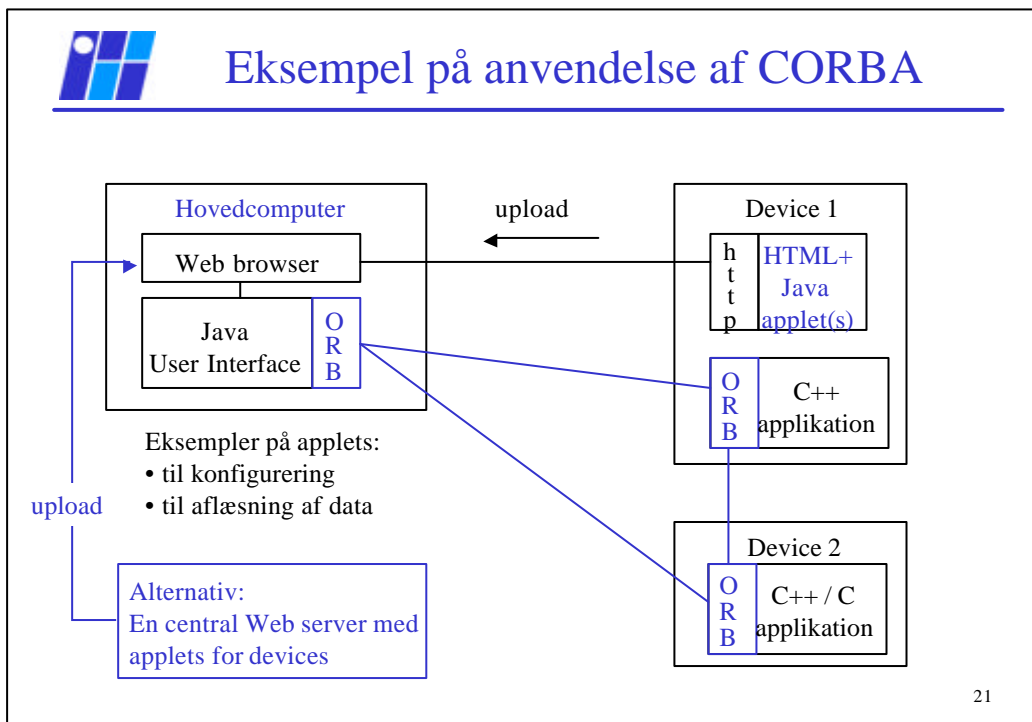
19



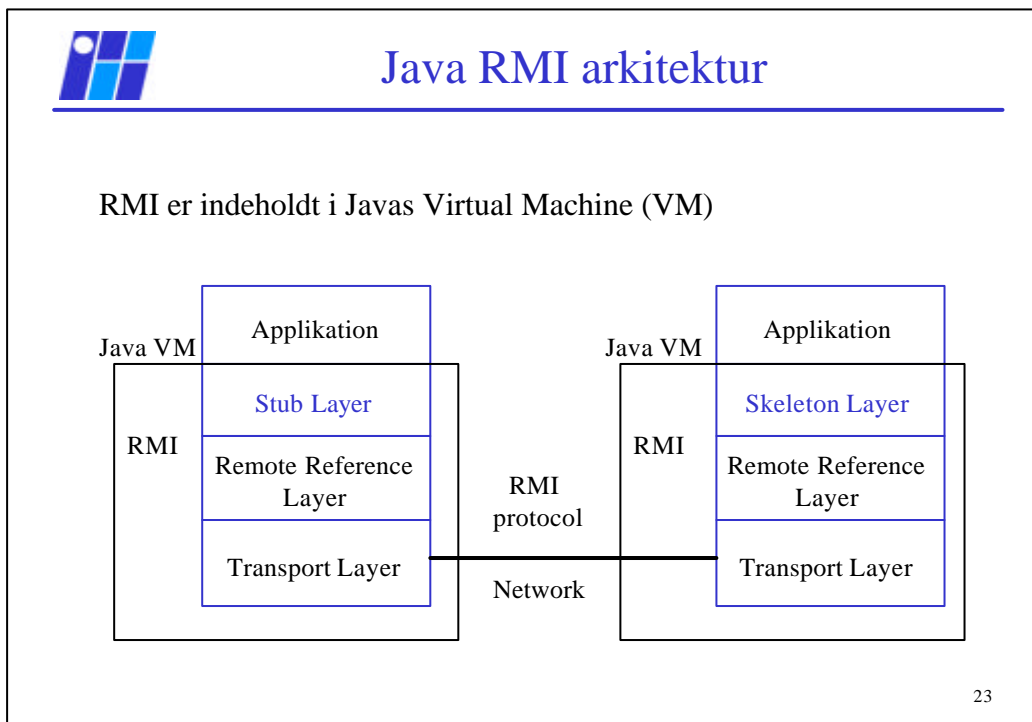
CORBA til apparater

- **Minimum CORBA for embedded system 1998**
 - en nedroslet version f.eks. uden *Dynamic Invocation if.*
- **Real-Time CORBA standard 1999**
 - udvidelser: prioritetsbegreb, threadpool, konfigurerings og valg mellem forskellige protokoller
- **ORB til embedded systemer**
 - Et eksempel: OrBacus/E på Windows 2000/PC (www.iona.com)
 - C++ "Hello world" applikation: client 98 Kb, Server 164 Kb
 - Java "Hello world" applikation; client 63 Kb, Server 92 Kb
 - Embedded Event services: 270 Kb, Naming Services: 260 Kb
- **Anvend et bibliotek *IIOP engine***
 - tillader at man anvender IIOP protokollen (fås ned til 15 Kb)


20



- ## Java RMI
- Javas RMI - Remote Method Invocation
 - Er Javas ORB (Object Request Broker)
 - Er en ren Sun standard
 - Er et middleware layer indbygget i Java Virtual Machines
 - Protokol: Java Remote Method Protocol (JRMP)
 - Java 2 standard and enterprise editions har support for både RMI og CORBA
 - Muliggør at objekter inklusiv kode flytter rundt på et netværk
 - Reference: www.sun.com
- 22



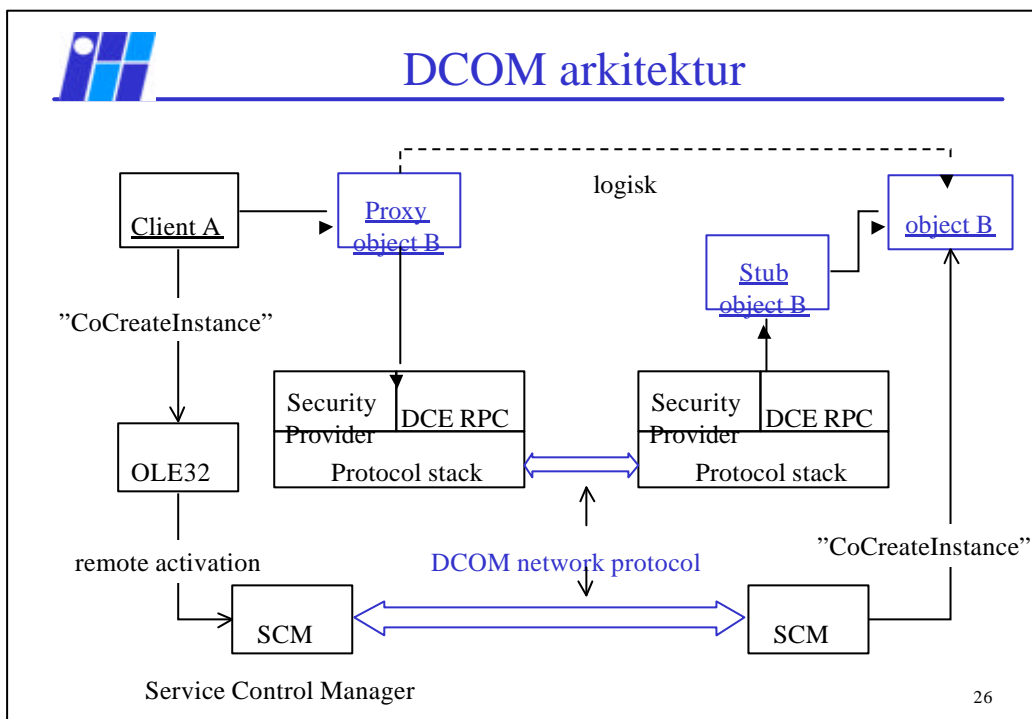
-
- ## Java RMI styrker/svagheder
- **Styrker:**
 - Er ren Java - en styrke i et Java miljø
 - Kan samarbejde med CORBA via RMI-IIOP
 - Følger med som en integreret del af Java
 - Kan overføre objekter inklusiv koden for operationerne
 - Håndterer *remote garbage collection*
 - **Svagheder:**
 - Er ren Java teknologi, men kan dog samarbejde med CORBA
 - Virker ikke godt gennem firewalls
 - Kræver nogen erfaring at anvende (dog lettere end CORBA)
- Kilde: Mike Preradovic, Intrinsic Software, ESC2001
- 24




DCOM

- Microsoft standarderne COM og DCOM:
 - COM - Component Object Model
 - DCOM - Distributed COM
 - released i 1997, Microsofts version af CORBA
 - DCOM er integreret i alle Window platforme
 - incl. Windows CE 3.x og supporteres på andre platforme f.eks. Unix, Linux
 - Protocol: Object Remote Procedure call (ORPC)
 - Er en binær standard
 - Interfaces specificeres med et memory layout svarende til C++'s vtable (virtual method table)
 - Kan anvendes fra f.eks. C++, Delphi, Visual Basic og Java
- Reference: www.microsoft.com/com/dcom.htm

25



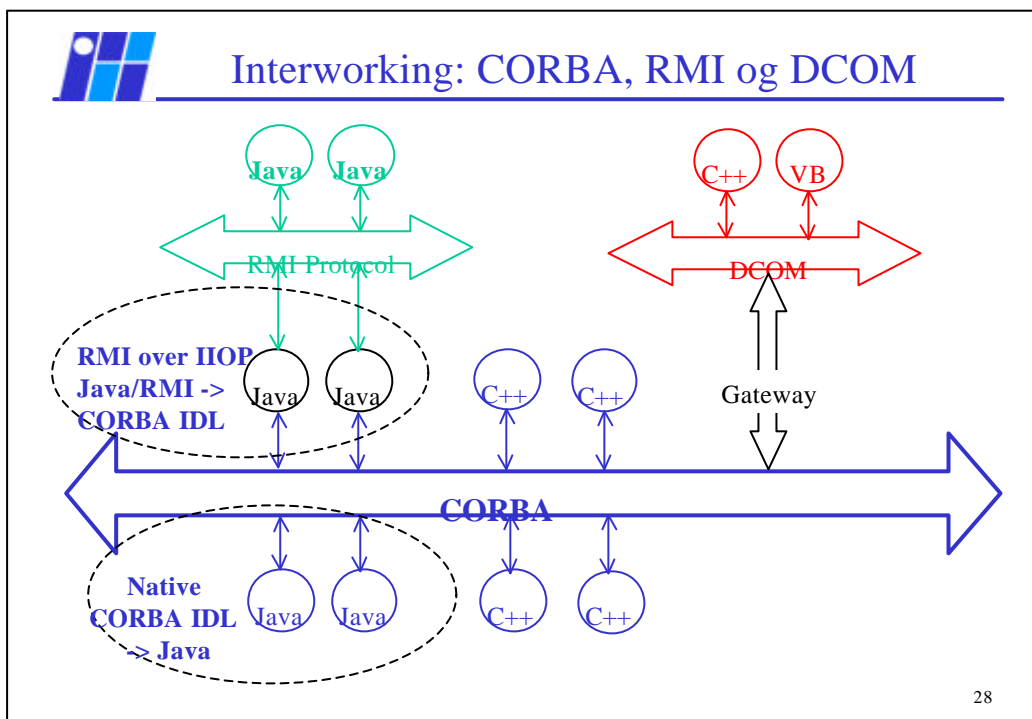


DCOM styrker/svagheder

- **Styrker:**
 - Er på alle Microsoft platforme og er anvendt i industrien i flere år
 - Er sproguafhængig
 - Mange Microsoft applikationer er DCOM enabled
 - DCOM er gratis
- **Svagheder:**
 - Er en Microsoft standard
 - Er ikke Java venlig
 - Er meget kompleks at programmere i C++
 - Har ingen reelle services som f.eks. CORBA
 - Egner sig ikke til indlejrede decives pga. størrelsen
 - Virker ikke godt gennem firewalls
 - Dårlig opførsel i "disconnected" environments

Kilde: Mike Preradovic, Intrinsic Software, ESC2001

27





Auto-discovery teknologier

- Enheder der skal kobles på et netværk bør fremover overveje at have funktionalitet, der muliggør *auto discovery* og *auto configuration*
- Der er pt. flere konkurrerende *defacto* standarder:
 - UPnP (Universal Plug and Play)
 - Jini
 - Salutation
 - HAVI
- Disse er pt. ikke særligt udbredte

29



UPnP

- **UPnP: Universal Plug and Play** - oprindeligt en Microsoft standard
 - Version 1.0 af arkitekturen er defineret
 - Support for UPnP i Windows Me - ikke pt. i Window CE
 - **UPnP forum**: konsortium med 200 leverandører
- Defineret som standardudvidelser til TCP/IP, DHCP og andre Internet standarder
- *Discovery* via en decentral *broadcast* process
- Data udveksling foregår via **XML**
- Kontrol via **SOAP**
- Reference: www.upnp.org

30



XML remote call/object modelling

- XML (eXtensible Markup Language)
 - Er en World Wide Web Consortium (W3C) standard
 - Anvendes til at repræsentere komplekse data i et læsbart tekstformat
- Reference: www.w3.org/XML/
- Pt. er der 3 XML baserede distribuerede skemainitiativer:
 - SOAP (Simple Object Access Protocol) - Microsoft
 - XML/RPC - en åben standard for modellering af RPC kald som XML
 - Et OMG initiativ for anvendelse af XML sammen med CORBA

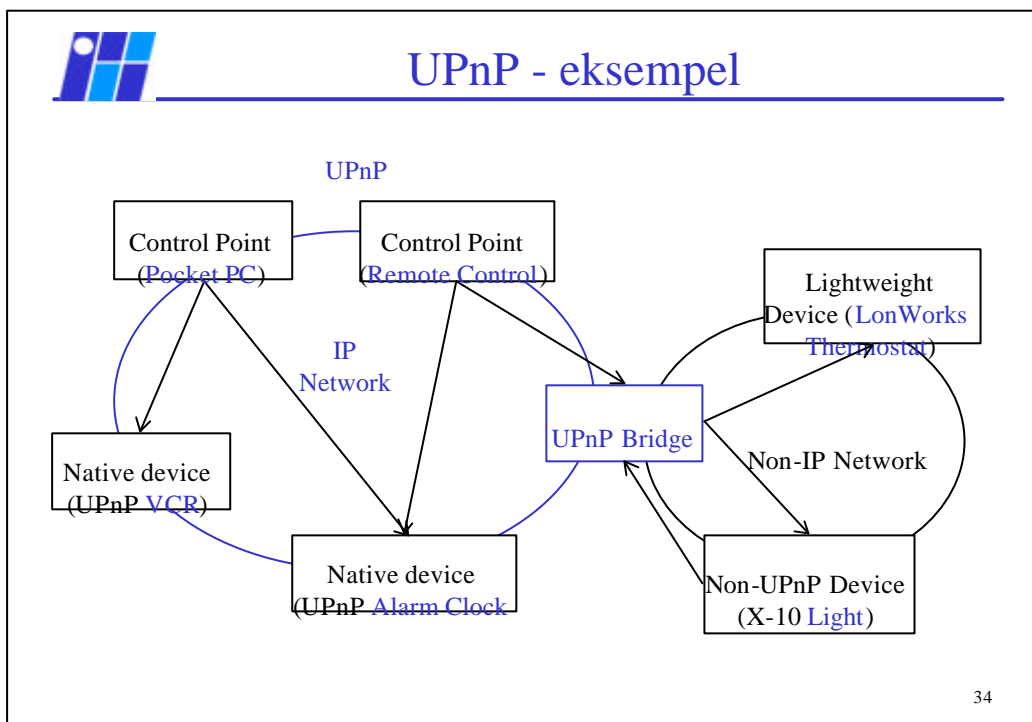
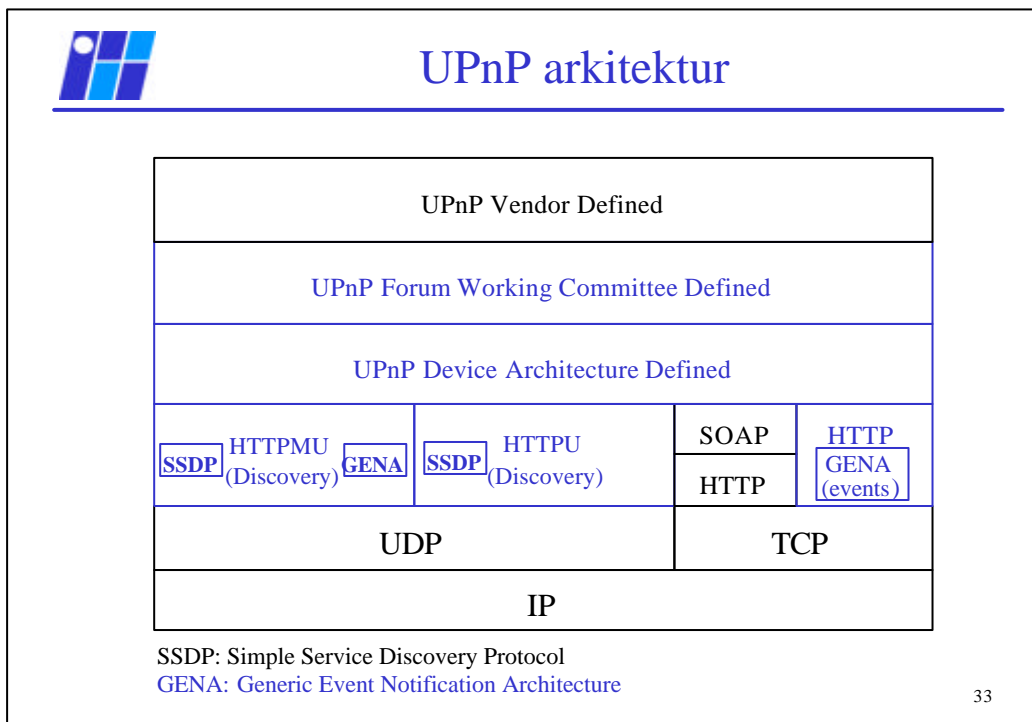
31

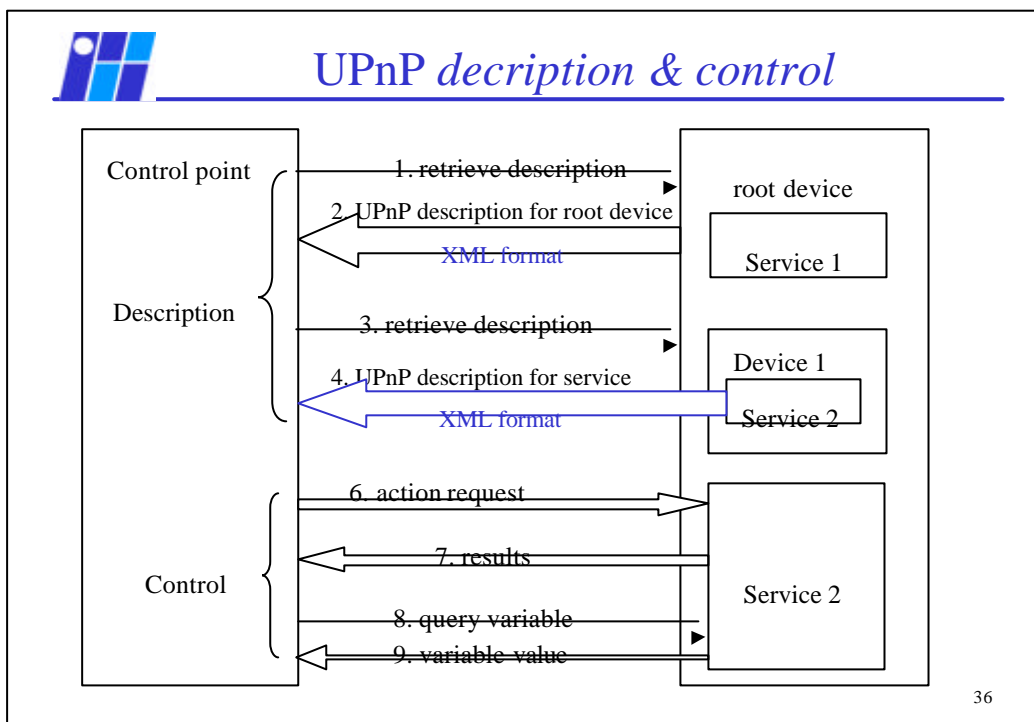
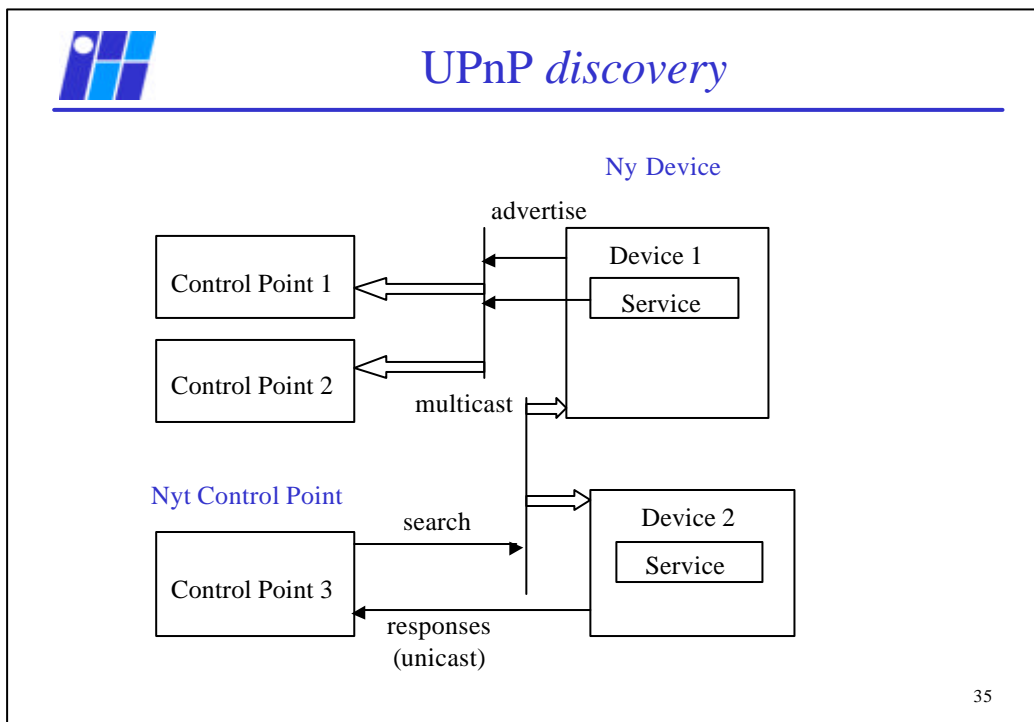


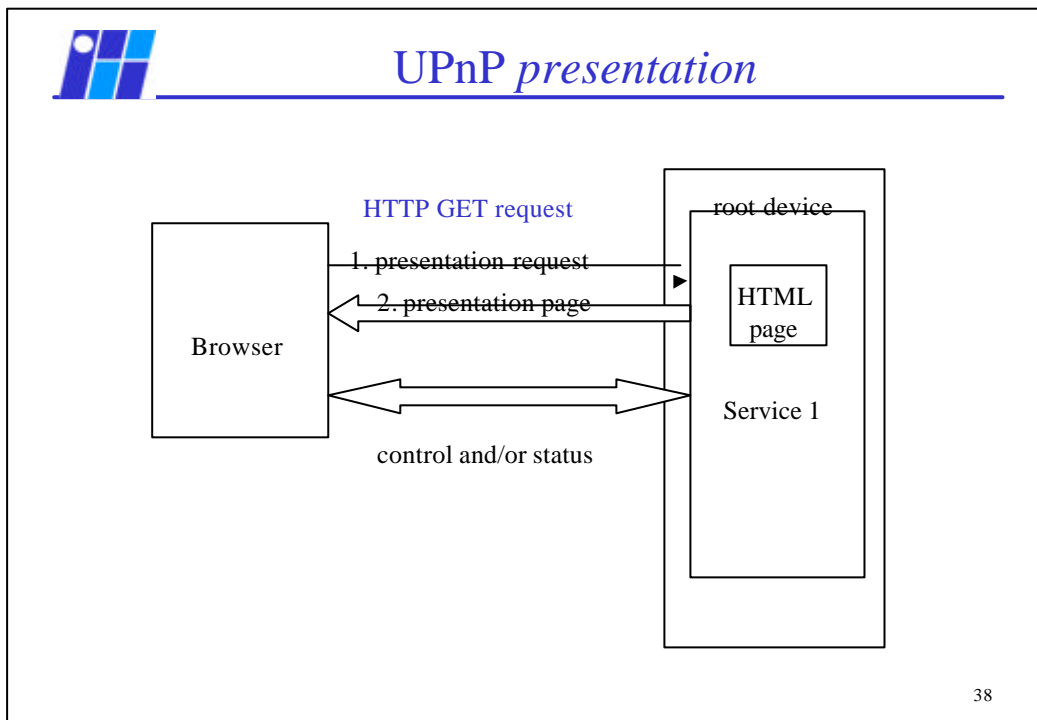
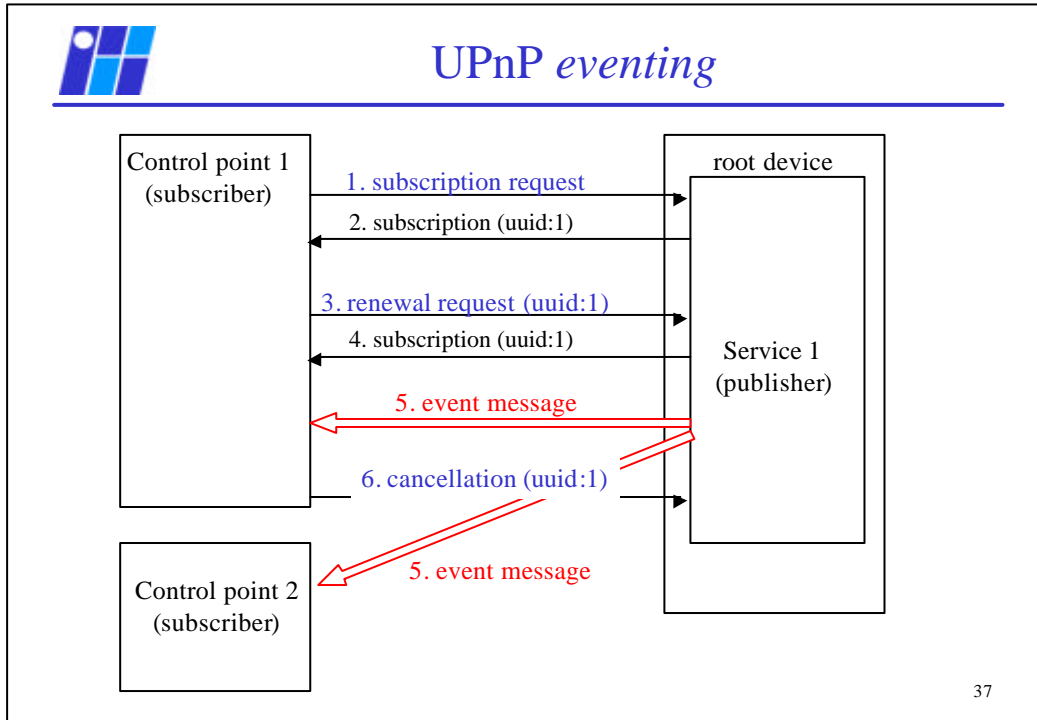
SOAP


- SOAP: (Simple Object Access Protocol)
 - Definerer anvendelsen af XML og HTTP til at eksekvere remote procedure call (RPC)
 - En standard for RPC over internettet
 - Kan arbejde effektivt med proxier og firewalls
 - Kan anvende Secure Socket Layer (SSL) for sikkerhed
 - Hvert UPnP control request samt retursvar er en SOAP message
- SOAP og XML kan anvendes uafhængig af UPnP

32





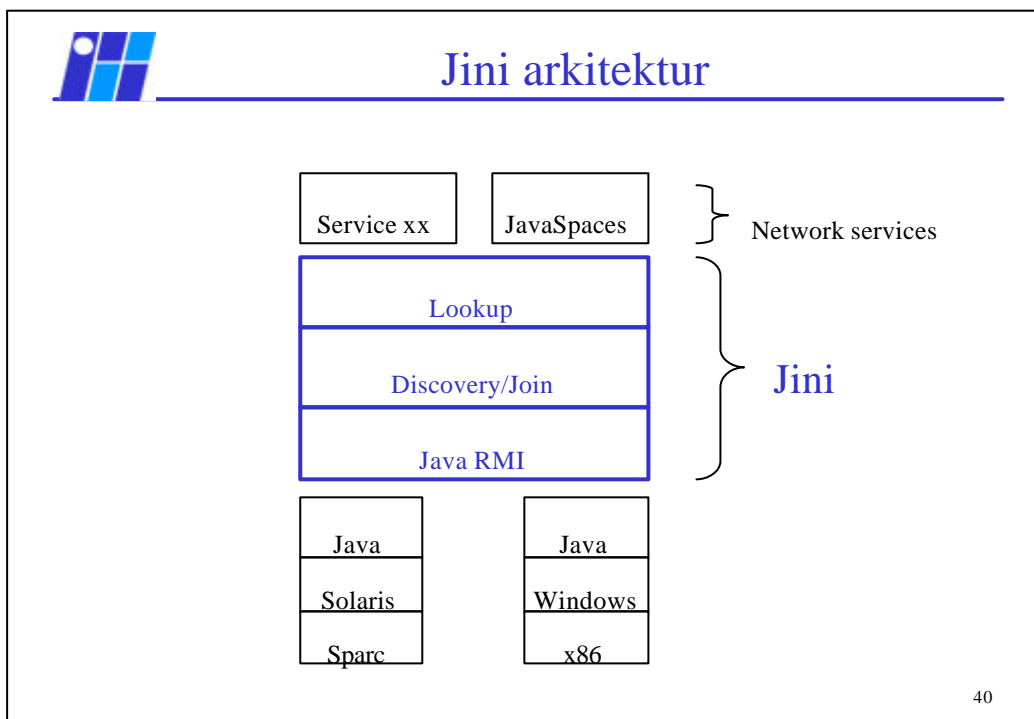


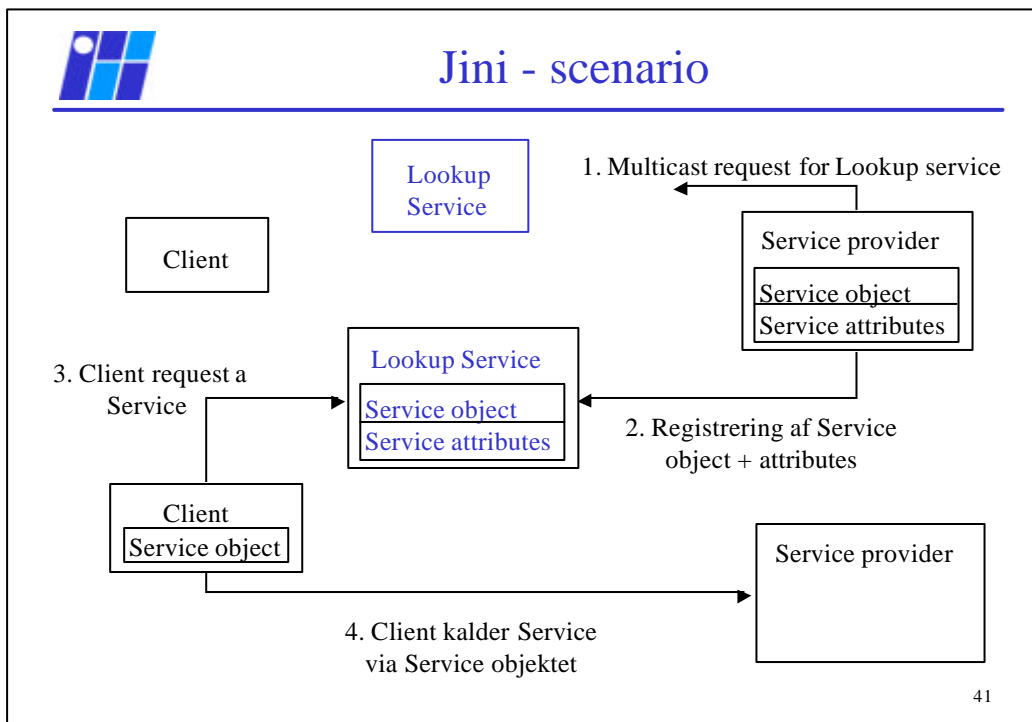


Jini

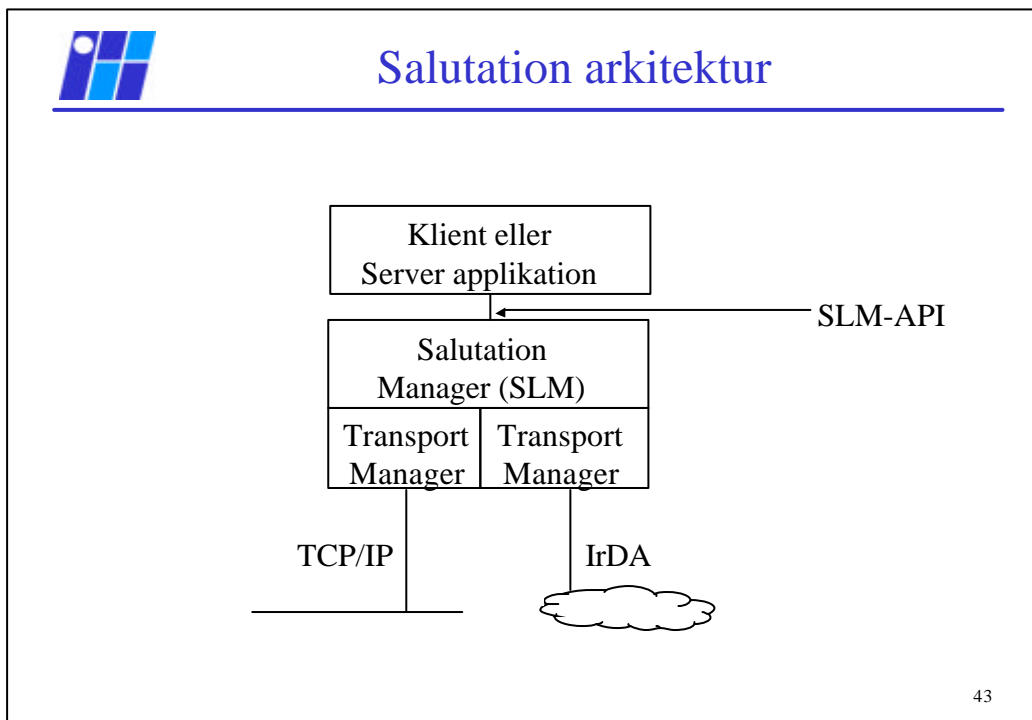
- Jini er en **SUN standard**
- Jini er en ren Java teknologi, der kræver:
 - Java 2 (JDK 1.2) eller højere
 - Java's Remote Method Invocation (RMI)
- Jini kræver at netværket og enhederne supporterer IP multICASTING
- Jini supporterer **distribuerede events med notifications**
- En "device" kan f.eks registrere en brugerdialog til styring af "devicen" som en service
- Reference: www.jini.org

39





- ### Salutation
- Salutation konsortium - **et åbent standard initiativ**
 - Salutation arkitekturen er uafhængig af:
 - operativsystem og kommunikations protokol
 - Muliggør implementering i meget små resurse-begrænsede enheder
 - Kræver at transport protokollen understøtter flere pålidelige bidirektionelle kommunikationskanaler
 - Reference: www.salutation.org
- 42

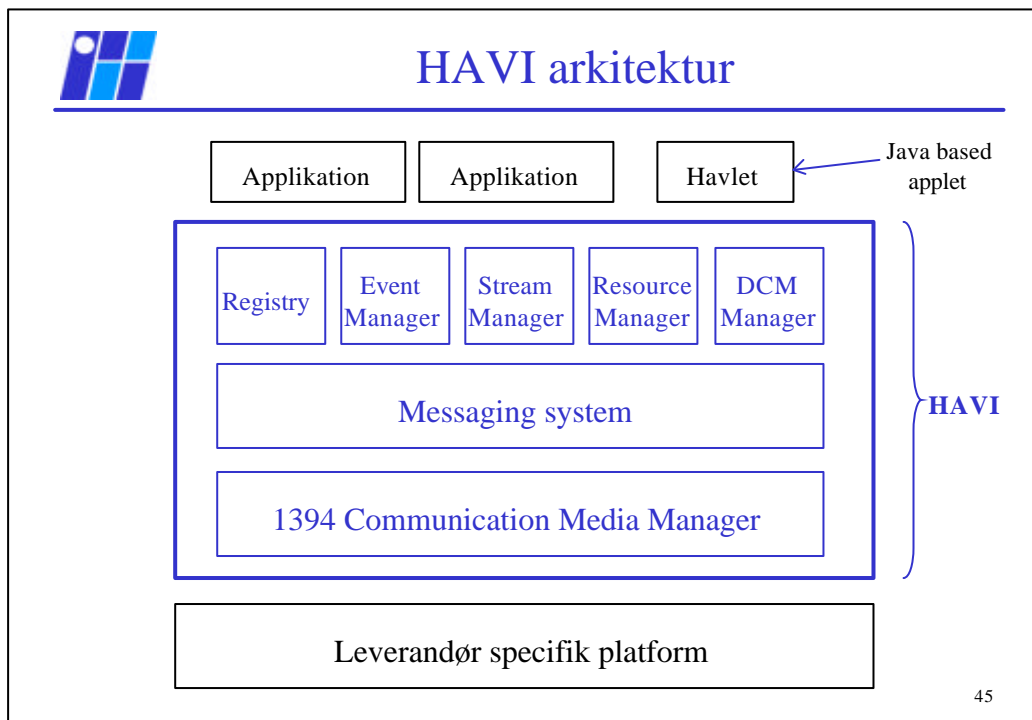


43

HAVI

- HAVI - Hardware Audio/Video Interoperability
 - Version 1.0 af specifikationen er released
 - Defineret af Grundig, Hitachi, Matsushita, Philips, Sharp, Sony, Thomson og Toshiba
- Defineret ovenpå standarden IEEE-1394 (Firewire)
- Anvender 1394's *device & service discovery* mekanisme
- Definerer et statisk kontrol interface for enheder
- Der eksisterer *bridges* til Jini
- Bridges til UPnP er under overvejelse
- Reference: www.havi.org

44



-
- The slide, titled "Opsummering: UPnP, Jini, Salutation, HAVI", contains a bulleted list of characteristics for these technologies. The number "46" is located in the bottom right corner of the slide's frame.
- Jini centraliseret struktur (er en Sun/java løsning)
 - Er afhængig af en central server (lookup service)
 - Jini har som den eneste run-time licens !
 - Sun forsøger pt. at gøre Jini mere letvægts
 - UPnP decentraliseret struktur (er en Microsoft løsning)
 - Jini, UPnP og Salutation er transport uafhængige
 - Jini og UPnP kræver begge IP multicasting
 - ikke understøttet pt. i mange embedded operativsystemer
 - Salutation kan implementeres på små devices
 - HAVI er forholdsvis lukket - hvor Jini, UPnP og Salutation er mere åbne for tilpasninger



Opsummering

- Der er en rivende udvikling inden for distribueret kommunikation og middleware
- Der er gode, brugbare men desværre konkurrerende standarder
- Standarderne er pt. "store" i forhold til apparatsystemer men på vej i mindre og mere RT orienterede udgaver
- Automatisk tilslutning og konfiguration bliver et must for nogle typer af apparater
 - Der er her tre generelle bud (Jini, UPnP, Salutation) samt HAVI
- XML forventes at blive anvendt til udveksling af kontrol- og statusinformation til apparater

47



Referencer

- CORBA: www.omg.org
- RMI: www.sun.com
- DCOM: www.microsoft.com/com/dcom.htm
- Jini: www.jini.org
- UPnP: www.upnp.org
- Salutation: www.salutation.org
- HAVI: www.havi.org
- XML: www.w3.org/XML/

48