

# G JINI

---

## G.1

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 1 Überblick

---



- Dienstplattform für Java
  - ◆ von Sun Microsystems
  
  - ◆ Programmiermodell für Dienste
  - ◆ Infrastruktur
  - ◆ Unterstützungsdienste
  
- Hintergrund
  - ◆ Nomadic computing
    - mobile Geräte – dynamische Dienstnutzung
  - ◆ einfacher Zugang zu Geräten
    - Gerät bietet Dienste an
  - ◆ allgemeine Dienstnutzung

## G.2

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

# 1.1 Programmiermodell

---



- JINI Dienst
  - ◆ konzeptioneller Dienst
    - keine vorgeschriebene Zuordnung zu bestimmter Implementierung
    - Implementierung in Hardware
      - z.B. Lichtschalter, Heizungsregler
    - Implementierung in Software
      - z.B. „Hallo“-Sager, Druckseitenaufbereiter
    - Kombination
      - z.B. Drucker
  - ◆ Implementierung des Dienstes muss nicht vollständig in Java erfolgen
  - ◆ Dienstveröffentlichung (Registrierung)
    - muss in Java erfolgen
    - Dienst erhält weltweit eindeutige Service-ID
      - sollte für die selben Dienste wiederverwendet werden

**G.3**

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

# 1.1 Programmiermodell (2)

---



- JINI-Client
  - ◆ Nutzer eines Dienstes
    - Implementierung in Java
  - ◆ dynamischer Dienstzugang
    - Dienstzugang (Proxy) in Java
    - Proxy muss ein dienstspezifisches Java-Interface implementieren
- JINI-Lookup-Service
  - ◆ ähnlich Namensdienst
  - ◆ registriert JINI-Dienste
  - ◆ ermittelt registrierte JINI-Dienste für Clients

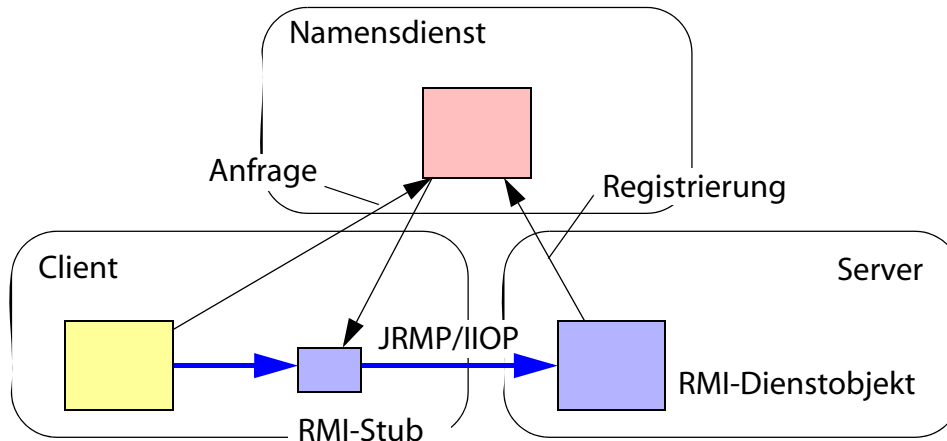
**G.4**

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2 Client-Server-Interaktion



- Klassische Interaktion (z.B. bei Java-RMI)
  - ◆ Dienstimplementierung als RMI-Objekt
  - ◆ Ermitteln einer Dienstreferenz über Namensdienst
  - ◆ Aufruf des Dienstes über RMI



G.5

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2 Client-Server-Interaktion (2)



- JINI-Interaktionsmodell
  - ◆ klassische Interaktion
    - Einsatz von RMI
  - ◆ proxybasierte Interaktion („Smart-Proxy“-Ansatz)
    - Teil der Dienstimplementierung vor Ort beim Client
      - automatische Installation des Proxys
      - Proxy ist ein Java-Objekt
    - Einsatz andere Middleware-Systeme
      - z.B. CORBA, JMS, EJB, Web-Services
    - teillokale Dienstimplementierung
      - Caching
      - Vorverarbeitung von Anfragen
    - lokale Dienstimplementierung
      - keine zentrale Implementierung

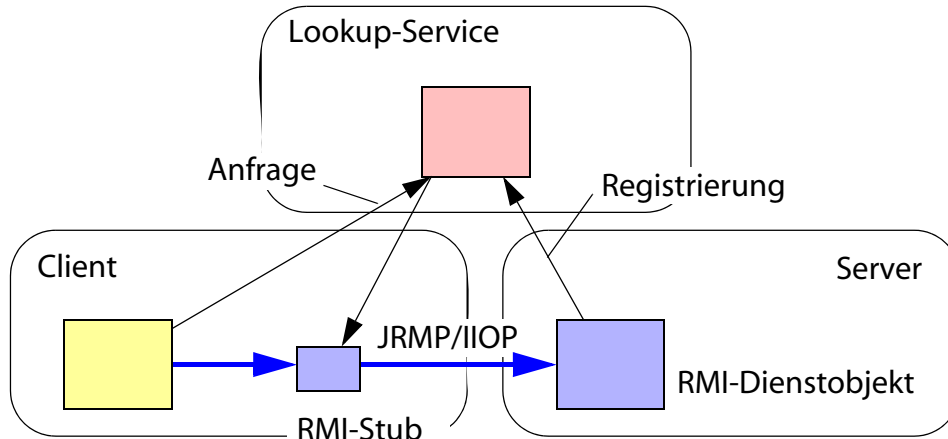
G.6

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2.1 RMI-basierte Interaktion



- Dienstimplementierung als RMI-Objekt
  - ◆ Ermitteln eines Dienstproxys (RMI-Stub) über Lookup-Service
  - ◆ Aufruf des Dienstes über RMI



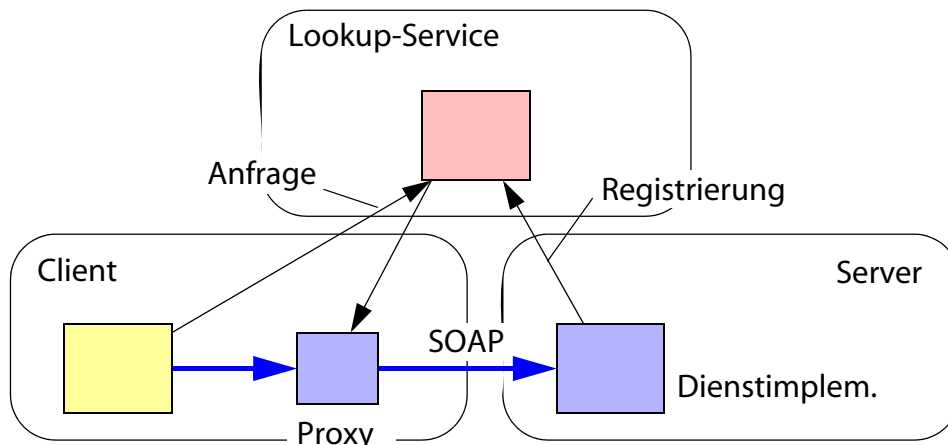
G.7

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2.2 Proxybasierte Interaktion



- Dienstimplementierung über Netz erreichbar
  - ◆ Ermitteln eines Dienstproxys über Lookup-Service
  - ◆ Aufruf des Dienstes über Methoden des Proxys
    - beliebige Kommunikation (z.B. SOAP)
    - teillokale Dienstimplementierung



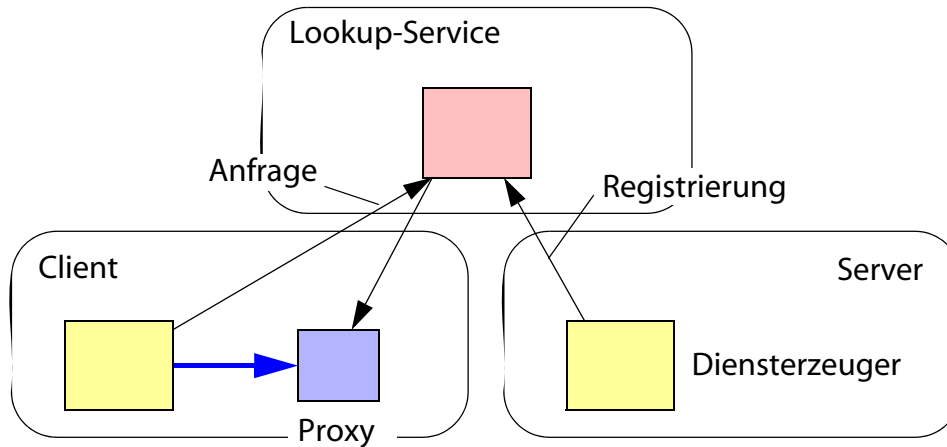
G.8

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2.3 Proxylokale Dienstimplementierung



- Proxy implementiert den Dienst vollständig
  - ◆ Registrierung des Dienstes notwendig
  - ◆ Ermitteln eines Dienstproxys über Lookup-Service
  - ◆ Aufruf des Dienstes über Methoden des Proxys



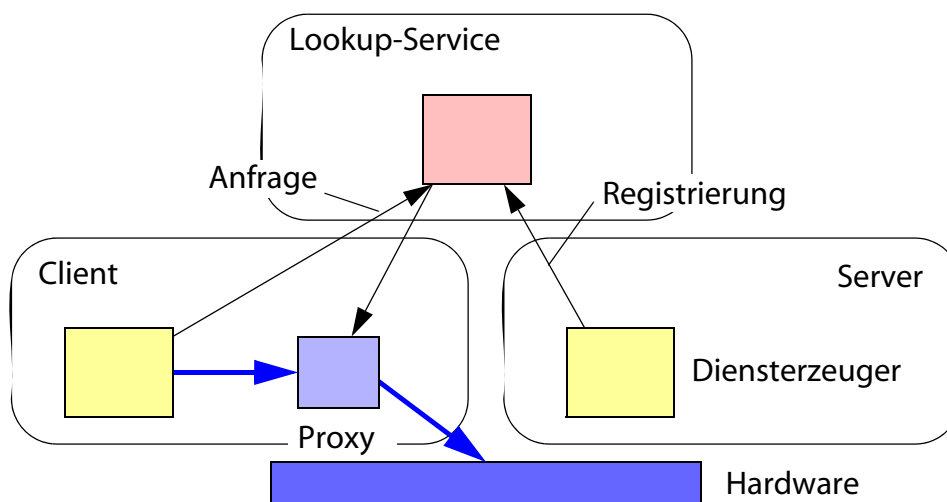
G.9

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 2.4 Dienstimplementierung in Hardware



- Proxy implementiert Zugang zur Hardware
  - ◆ Registrierung des Dienstes notwendig
  - ◆ Ermitteln eines Dienstproxys über Lookup-Service
  - ◆ Aufruf des Dienstes über Methoden des Proxys



G.10

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3 Lookup-Service

---



- Zentrale Instanz bei JINI
  - ◆ implementiert als JINI-Dienst
    - Dienst-Proxy erforderlich
      - Henne-Ei-Problem (Wo kann man den bekommen?)
    - Lösung: Discovery-Protokolle

### 3.1 Discovery

---

- Zwei Varianten
  - ◆ Unicast-Discovery
    - Rechner mit Lookup-Service bekannt
  - ◆ Multicast-Discovery
    - unbekannte Umgebung

G.11

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

### 3.1 Discovery (2)

---



- Discovery-Anfragenachricht über Multicast (IP: 224.0.1.85:4160)
  - ◆ Protokollversion
  - ◆ Portnummer des Client
    - Client hält TCP-Port offen für Antwort
  - ◆ Liste von bekannten Lookup-Services
    - Service-IDs
  - ◆ Liste von JINI-Gruppennamen
    - öffentliche Gruppe: leerer String
    - beliebiger Gruppenname
      - z.B. „AvID“
  - ◆ Nachricht bekommt TTL-Angabe (Time-To-Live)
    - lokales Netz, Organisationsnetz, regionales Netz, gesamtes Internet
    - korrektes Routing an den Netzgrenzen vorausgesetzt

G.12

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AvID-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3.1 Discovery (3)

---



- Anfrage über TCP/IP (Unicast)
  - ◆ Protokollversion
  
- Generische Antwort über TCP/IP
  - ◆ Service-Proxy
    - doppelt serialisiertes Java-Objekt
      - serialisiertes Proxy-Objekt wird in ein `MarshaledObject` eingebettet
      - `MarshaledObject` wird serialisiert
    - Vorteil: eigentliches Objekt braucht nur bei Bedarf deserialisiert werden
      - Klasse muss nicht geladen werden
      - Klasse von `MarshaledObject` ist bekannt
  - ◆ Liste der unterstützten Gruppennamen

G.13

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3.1 Discovery (4)

---



- Multicast-Announcement-Protocol (IP: 224.0.1.84:4160)
  - ◆ Lookup-Service verbreitet periodisch seine Adresse
    - etwa alle zwei Minuten
  - ◆ Nachrichteninhalt:
    - Protokollversion
    - eigener Hostname plus Portnummer
    - eigene Service-ID
    - Liste der unterstützten Gruppen

G.14

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3.1 Discovery (5)

---



### ■ Unterstützung

- ◆ Klasse `net.jini.discovery.LookupDiscovery`
  - Erzeugung einer Instanz
    - Liste der interessierten Gruppennamen als Konstruktorparameter
  - sucht automatisch nach Lookup-Services
  - Registrierung einer `DiscoveryListener`-Instanz
    - wird benachrichtigt über neu entdeckte und verschwundene Lookup-Services
    - erhält Referenz auf Proxy
- ◆ Klasse `net.jini.discovery.DiscoveryManager`
  - komfortablere Buchhaltungsfunktionen über entdeckte Lookup-Services
- ◆ Lookup-Discovery-Service
  - JINI-Dienst sammelt Informationen über Lookup-Services und benachrichtigt interessierte Clients

G.15

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3.2 Registrierung

---



### ■ Registrierung eines JINI-Dienstes (*Join*)

- ◆ Übergabe eines `ServiceItem`-Objekts an Lookup-Service
  - Service-ID (oder `null`, falls noch nicht bekannt)
  - Proxy-Instanz
  - Liste von Attributen
    - z.B. `PrinterType=Colour`
    - Realisierung der Attribute über speziell serialisierte Attributobjekte (Klasse `Entry`)
- Unterstützung
  - ◆ Klasse `net.jini.lookup.JoinManager`
  - ◆ sorgt automatisch für Registrierung an allen gefundenen Lookup-Services
    - insbesondere bei neuen Lookup-Services

G.16

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 3.3 Dienstsuche

---



- Anfrage an einen Lookup-Service
  - ◆ Übergabe eines `ServiceTemplate`-Objekts
    - Service-ID oder `null`
    - Liste von Interface-Typen oder `null`
    - Liste von Attributmustern oder `null`
  - ◆ Rückgabe eines oder mehrere Dienst-Proxies zu gefundenen Diensten
    - typisch: Cast auf Dienst-Interface und Nutzung des Dienstes
- Unterstützung
  - ◆ Klasse `net.jini.lookup.ServiceDiscoveryManager`
  - ◆ erzeugt Cache von bekannten und interessanten Diensten

G.17

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases

---



- Motivation
  - ◆ Fehlererkennung
    - ausgefallene Server und Clients
  - ◆ Selbstheilung
    - Reaktion aus früher Fehlererkennung
      - z.B. nach anderen Lookup-Services suchen
      - z.B. anderen Drucker suchen
  - ◆ Begrenzung der Dienstlebenszeit in langlaufenden Systemen
    - z.B. zur Aktualisierung auf neue Version
  - ◆ Garbage-Collection
    - Löschen oder Deaktivierung eines Dienstes, falls nicht genutzt

G.18

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases (2)

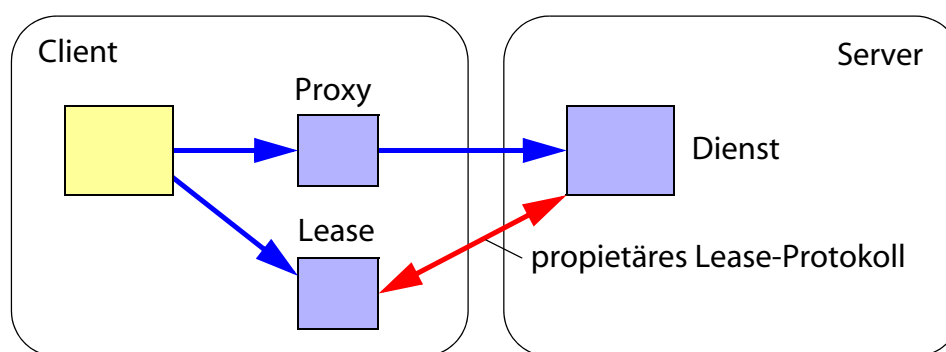
- Betriebsmittel werden mit Leases versehen
  - ◆ Leases haben begrenzte Gültigkeit
  - ◆ nach Ablauf wird Betriebsmittel unter Umständen entzogen oder ungültig
  - ◆ **hier:** Lease für Proxies und Dienstregistrierungen
    - nach Ablauf wird Proxy ungültig oder Dienst deregistriert
  - ◆ initiale Lease-Zeit wird verhandelt
    - Vorschlag durch Betriebsmittelnutzer
      - z.B. beim Registrieren an einem Lookup-Service
  
- Lease
  - ◆ Java-Interface für beliebige Lease-Implementierungen
  - ◆ Methode zur Erneuerung der Lease (Vorschlag durch Nutzer)
    - tatsächliche Verlängerungszeit abhängig vom Anbieter
  - ◆ Methode für unmittelbaren Ablauf der Lease

G.19

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases (3)

- Lease-Architektur
  - ◆ Beispiel: Lease für einen Dienst-Proxy



- ◆ verschiedene Implementierungsmöglichkeiten
  - Lease-Implementierung auf Seite des Client/Proxy
  - Lease-Implementierung auf Seite des Dienstes

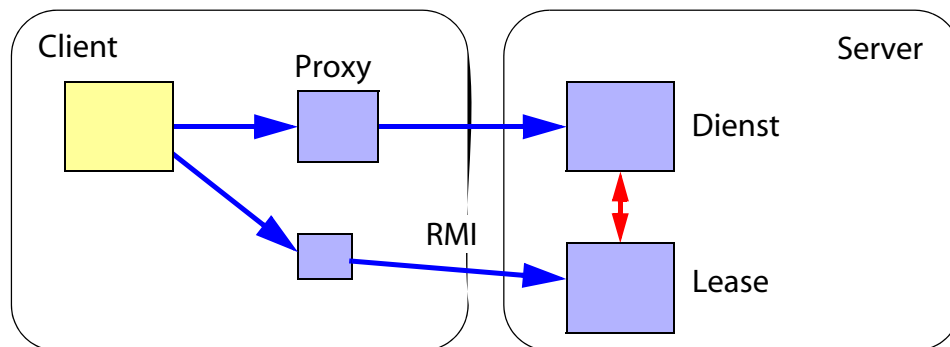
G.20

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases (4)

### ■ Lease-Implementierung mit RMI

#### ◆ Lease-Implementierung auf der Seite des Dienstes



#### ◆ z.B. Methode `getLease()` am Dienst-Interface ermittelt Lease

G.21

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases (5)

### ■ Unterstützung auf Client-Seite

#### ◆ Klasse `net.jini.lease.LeaseManager`

#### ◆ Registrierung von Leases zur Verwaltung

- zur automatischen Verlängerung
- zur automatischen Verlängerung bis zu gewissem Zeitpunkt

#### ◆ Leasing-Service

- JINI-Service zur Verlängerung von Leases
- sinnvoll für langdauernd inaktive Clients zum „Outsourcing“ der Lease-Verwaltung

G.22

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 4 Leases (6)

---



- Unterstützung auf Server-Seite
  - ◆ Sun's Landlord Implementierung bestehend aus Interfaces für
    - Landlord: Verwaltet die Ablaufzeiten aller Leases
    - LandlordLeaseFactory: erzeugt neues Lease-Objekt für bestimmten Landlord
    - LandlordLease: Lease, die mit Landlord zusammen arbeitet
  - ◆ Identifikation der Betriebsmittel in Lease und Landlord über Cookie-Objekte

**G.23**

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 5 Randbereiche

---



- Dynamisch zu ladender Code
  - ◆ Proxy-Code muss dynamisch geladen werden
    - Standard bei RMI-Stubs
    - für alle JINI-Proxies notwendig
  - ◆ Web-Server erforderlich
    - stellt Stub- und Proxy-Klassen bereit
  - ◆ Security-Manager erforderlich
    - muss erzeugt werden
    - benötigt entsprechende Berechtigungen
      - muss in Policy-Datei festgelegt werden

**G.24**

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 5 Randbereiche (2)

---



### ■ RMI-Activation

- ◆ RMI erlaubt aktivierbare Objekte
  - Objektreferenz vorhanden
  - Objekt wird bei Bedarf (bei Zugriff) instanziiert (aktiviert)
- ◆ Lookup-Service oder Jini-Service könnte sich deaktivieren
- ◆ Dämon-Prozess `rmiid` notwendig für Aktivierung

G.25

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 6 JavaSpaces

---



### ■ Implementierung eines Tupel-Space in Java bzw. JINI

- ◆ „Dateisystem“ für JINI-Services

### ■ Programmiermodell

- ◆ JavaSpace enthält Java-Objekte
- ◆ Einfügen eines Objekts in den Space (als Kopie) (`write`)
- ◆ Lesen eines Objekts vom Space (als Kopie) (`read`)
- ◆ Herauslösen eines Objekts aus dem Space (löschendes Lesen) (`take`)
- ◆ Benachrichtigungsfunktion beim Einfügen von Objekten mit bestimmten Eigenschaften (`notify`)
  
- ◆ speicherbare Objekte erben von `Entry`
  - vgl. Attribute bei Dienstregistrierung und -suche
  - Suche und Suchmuster sind äquivalent

G.26

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 6 JavaSpaces (2)

---



- Implementierung als JINI-Service
  - ◆ muss installiert, registriert, gesucht und gefunden werden
  - ◆ Entry-Objekte bekommen eine Lease
    - Client muss Lease-Management übernehmen
    - Entries mit abgelaufener Lease werden gelöscht
  
- Suns JavaSpaces Implementierung Outrigger
  - ◆ transienter oder persistenter Objektspeicher
  - ◆ Zusammenarbeit mit Transaktionen
    - weiterer Jini-Service: Transaktionsmanager
      - Suns Implementierung: Mahalo
    - Erzeugung eines Transaktionskontexts am Transaktionsmanager
      - Aufruf von `abort()` macht alle Änderungen im Space rückgängig
      - Aufruf von `commit()` macht alle Änderungen sichtbar

G.27

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>

## 7 Literatur

---



- JINI
  - ◆ Sun Microsystems: Jini Spezifikation- und Download-Seite  
<<http://java.sun.com/products/jini/>>
  - ◆ W. K. Edwards: *Core Jini*. Prentice Hall, 2001.
  - ◆ R. Flenner: *JINI and JavaSpaces Application Development*. Sams, 2001.
  - ◆ Jan Newmarch's Guide to Jini Technologies  
<<http://jan.netcomp.monash.edu.au/java/jini/tutorial/Jini.xml>>

G.28

© 2002-2006, Franz J. Hauck, Verteilte Systeme, Univ. Ulm, [2006s-AviD-G-Jini.fm, 2006-07-18 14.44] <http://www-vs.informatik.uni-ulm.de/teach/ss06/avid/>