Jini

Andreas Zeidler
az@ubicomp.de

Java Intelligent Network Infrastructure

Jini Is Not Initials

Jini

- Framework von APIs zu nützlichen Funktionen / Services
- Hilfsdienste (discovery, lookup,...)
- Standardprotokolle und Konventionen

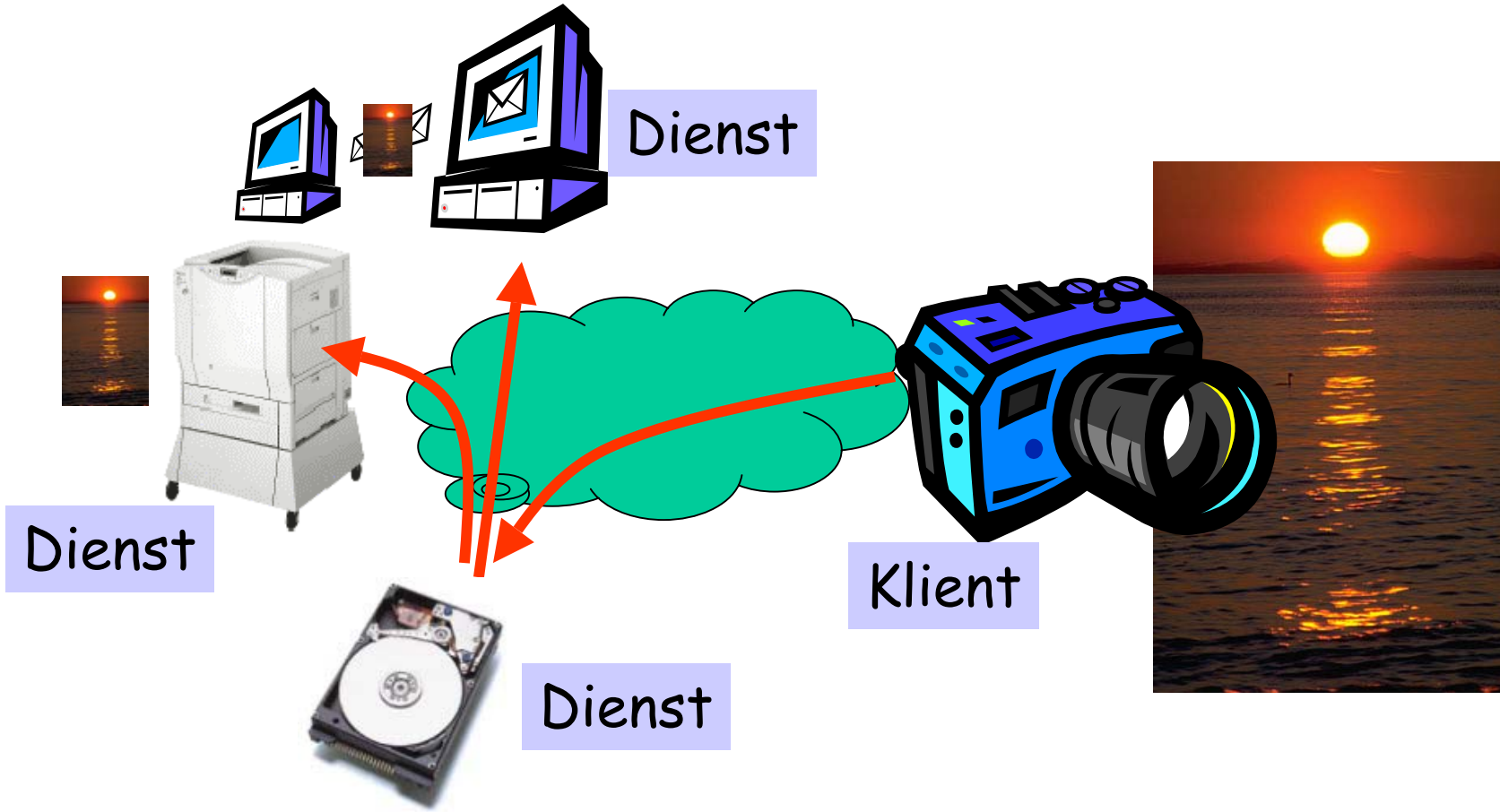
- Infrastruktur („Middleware“) für dynamische, kooperative, spontan vernetzte Systeme
  - Zur Programmierung / Realisierung verteilter Anwendungen

- Dienste, Geräte,... finden sich „automatisch“ („plug and play“)
- Neu hinzukommende Komponenten bzw. Interaktionsbeziehung
- Mobilität

Jini

- Zusammenführen ("Föderation") von Benutzern und benötigten Ressourcen (Dienste)
- Netzzentriert:
  - Netz als einfach zu verwendendes Werkzeug
  - Netz ist immer da, Dienste und Klienten nicht -> Dynamik
  - Einfache Administration
    - spontane Vernetzung
    - Zero Administration
    - Flexibles Hinzufügen und Löschen von bel. Diensten
  - Einfaches Auffinden von benötigten Ressourcen
    - Hardware-Dienste (Drucker, Platten, ...)
    - Software-Dienste (Auskunftsdienste, Email-Dienst, ...)

Föderation



Jini

- Auf Java aufbauend und in Java implementiert
  - Typsichere (objektorientierte) Kommunikationsstruktur
  - Nutzt RMI (Remote Method Invocation)
  - „Überall" JVM / Bytecode vorausgesetzt
  - Code shipping, mobiler Code (RMI)
- Erweitert Java für verteilte Systeme
  - Netzweiter Zugriff auf verschiedene JVMs
  - Eine Applikation, verschiedene Rechner
  - Ein Service, viele Klienten/Applikationen -> "Re-usability"

Jini Domäne

Büro

Jini

Haushalt/Privc

Mobil

Vom Supercompu

...bis zum Toaster

Herausforderungen

- Heterogenität
  - "Wie redet der Toaster mit dem Supercomputer?"
  - Hardware, Software, Ressourcen, Protokolle

  -> Einheitliche Sprache? ("Java vs. XML/SOAP")
- Mobilität
  - Trend zu mehr Mobilität
  - Handys, PDAs, Notebooks, Bluetooth, UMTS, GPRS, ...

  -> Wie kann man Mobilität unterstützen/möglich machen?
- User-zentriert
  - Einfache und intuitive Verwendung (spontan...)

  -> DHCP? Nameserver? Kennt das meine Oma?

  -> Was heisst Mailserver auf russisch?

Sichtweisen

- Systemsicht: Infrastruktur für verteilte Systeme
- Entwicklersicht: Unterstützung zur einfachen Entwicklung verteilter Applikationen (Netzprogrammierung!)
- Anwendersicht: Unterstützung zur einfachen Verwendung von verteilten Systemen und Applikationen

Erweiterung von Java

	Infrastruktur	Programmiermodell	Dienste
Java	- JVM - RMI - ...	- Swing - Beans - ...	- Transaction Se - Enterprise Java - ...
+ Jini	- Discovery/Join -Lookup -Lookup-Service	- Transaktionen - Verteilte Ereignisse - Leasing	- JavaSpaces -Email-Dienste -Transaktionsdie -Benutzerdienste -...

Infrastruktur

- Minimaler Kern von Jini
  - Discovery&Join Protokolle
    - Suchen&Finden einer Registrierungsinstanz ohne Kenntnis des N
    - Dienste machen sich publik, werden Teil der Föderation
    - Beschreiben sich selbst und ihre Fähigkeiten mit Hilfe von Java-Objekten
  - Lookup-Service
    - Zentraler Dienst in Jini
    - Läuft autonom irgendwo im lokalen Netz
    - "Yellow pages" für Clients (Lookup)
    - Kann ohne Wissen über lokales Netz "gefunden" werden
    - Verwaltet Dienst An- und Abmeldungen
    - Ähnlich X.500/LDAP
      - Java-Typen, statt "Textbeschreibungen"

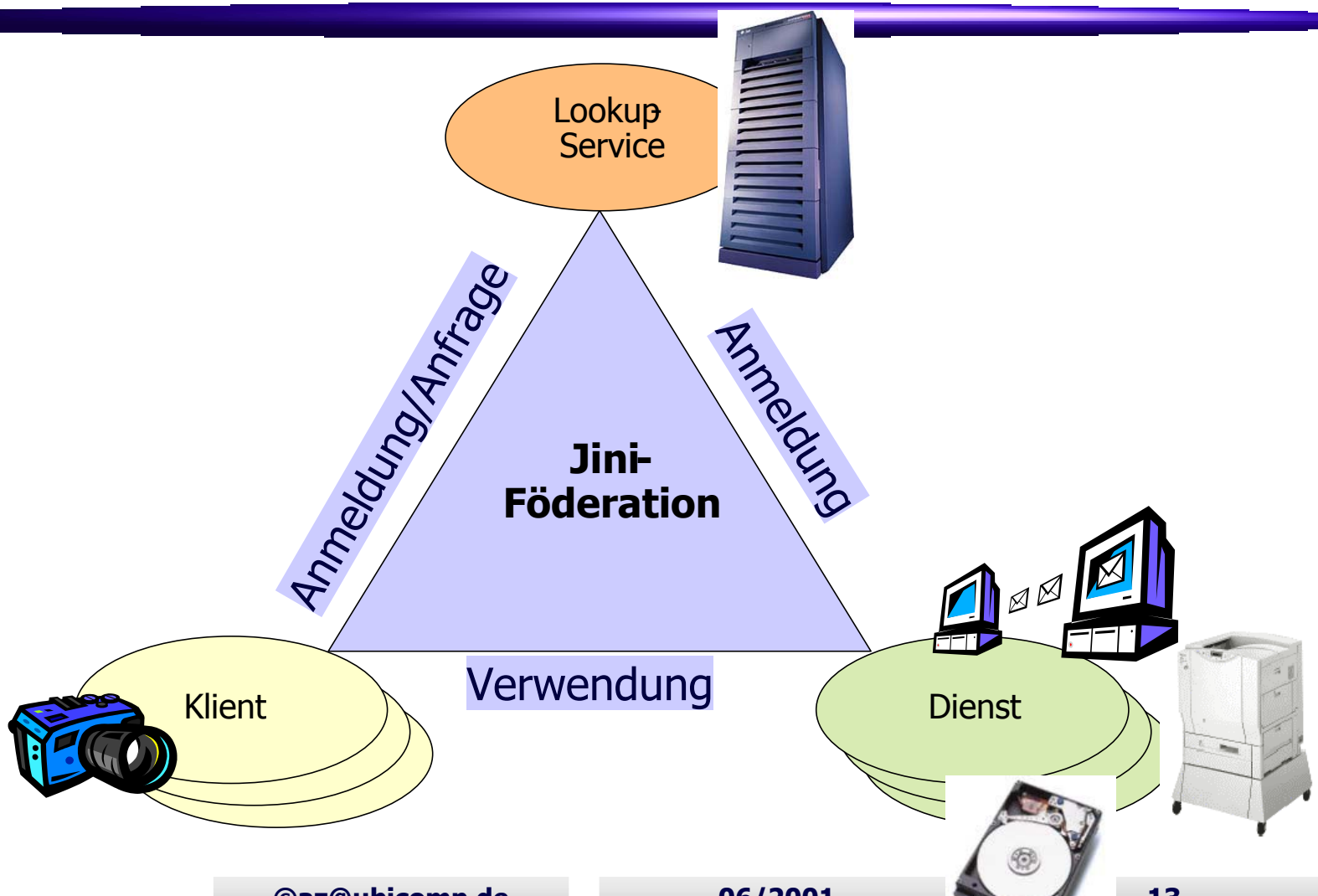
Programmiermodell/APIs

- Design für verteilte Applikationen
  - Leases
    - Token für bestimmte Dienstleistung -> Ressourcen-Verwaltung
    - Muss periodisch erneuert werden -> Zeitmodell in Jini
    - Time-Out: Lease nicht verlängert
    - Heart-Beat: Lease wird verlängert -> "Alive"
    - Beispiel: Eintrag in LUS -> Aktuelle Sicht auf Gesamtsystem
  - Distributed Events
    - Asynchrone Benachrichtigungen wenn Änderungen eintreten ("pu
    - Publish/Subscribe: Registrierung für bestimmte Ereignisse und Benachrichtigung wenn Ereignis eintritt
    - Beispiel: "Neuer Dienst ist angemeldet", "Dienst reagiert nicht"
  - Verteilte Transaktionen: Transaktionen über JVM-Grenzen hinweg

Services

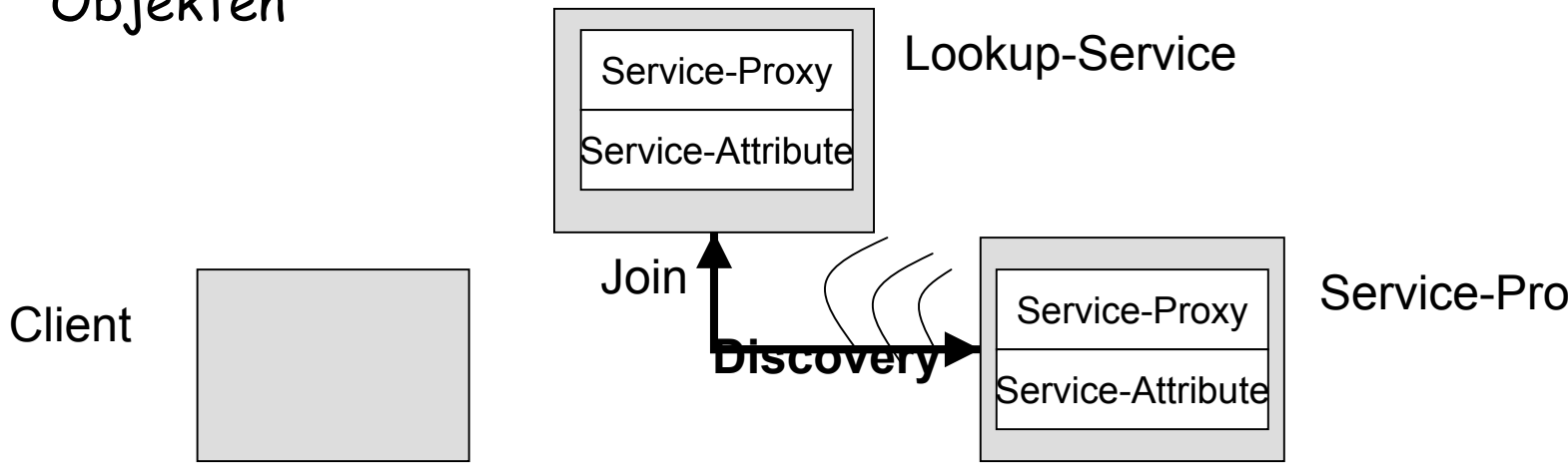
- Verwenden Infrastruktur und Programmiermodell
- Sind Objekte im Sinne von Java
- Funktion/Semantik wird definiert durch das Interface, das der Service implementiert
  - Z.B. Printer-Service implementiert `printer`
  - Äquivalent zu "normalem" Java
- Services können andere Services verwenden
- Verwenden i.d.R. mobilen Code (Service-Proxies), d.h. Teile des Dienstes werden beim Client ausgeführt
  - Beispiel: Benutzer-Interaktion, Filterung

Das Jini-Dreieck



Discovery&Join

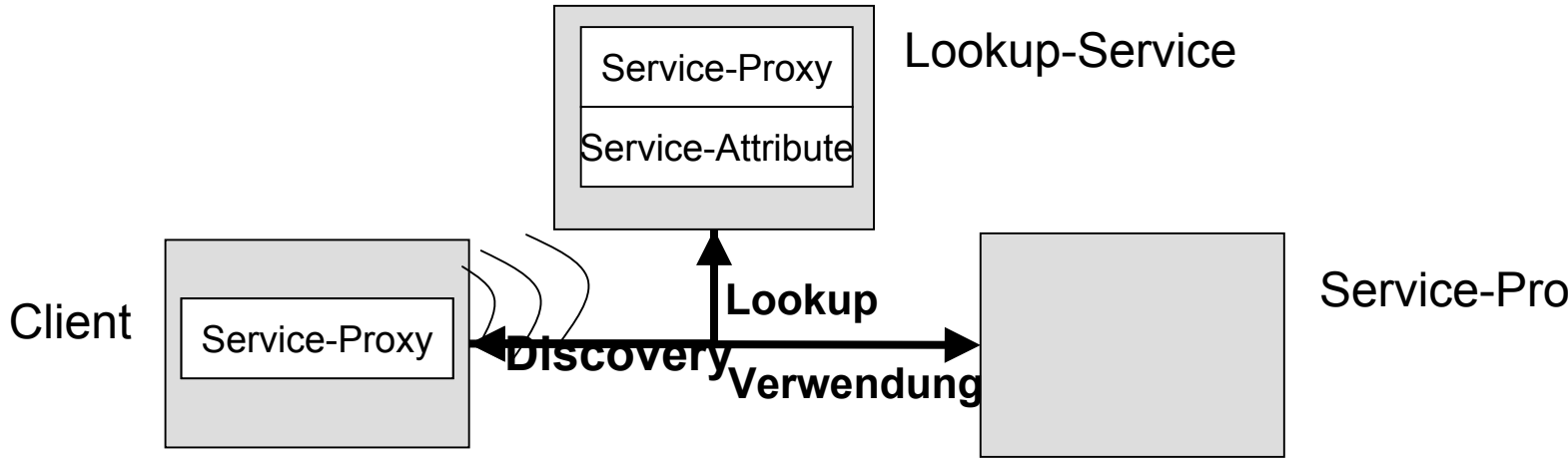
- Discovery:
  - Dienst versucht sich neu zu registrieren
  - Multicast-Protokoll (keine festen Adressen nötig!)
  - Dient zum Auffinden eines/mehrerer LUS
- Join:
  - Dienste registrieren sich beim LUS mit: einem Service-Proxy-Objekt und Menge von Attribut-Objekten



Lookup

- Klient sucht Lookup-Service (wie bei Discovery)
- Beschreibt den gesuchten Dienst
  - Möglichst "genau"

  -> Art von Schablone (engl.: Template)
- Bekommt eine Kopie der Dienst-Registrierung als Antwort
- Kann Dienst sofort verwenden (mobiler Code)



Jini in Forschung&Lehre

- Einsatz von Jini in Forschung&Lehre
  - "Verteilte Systeme" Programmierpraktikum
  - Einige Prototypen in der Forschung
  - Studien- & Diplomarbeiten
- "Industrie-Kurs"
  - Verschiedenster Wissenstand
  - Verschiedenste Programmiererfahrungen
  - 2-tägiger "Crash-Kurs"
    - 1 Tag Theorie
    - 1 Tag Programmierpraxis

Erfahrungswerte...

- Jini ist...
  - ...sehr einfach zu erlernen
    - Konzepte sind in wenigen Stunden erlernbar
    - Kern-APIs sind relativ klein und überschaubar
  - ...am Anfang schwer zu verwenden
    - Verteilte Systeme -> Netzprogrammierung
    - Code Mobilität (von RMI geerbt) ist sehr schwer zu begreifen ("Was wandert wann, von wo nach wo und warum?")
    - Classloader & Codebase-Property
    - Mangelhafte verteilte Debug-Möglichkeiten
  - ...schwer in Betrieb zu nehmen
    - Dauert immer eine Weile, bis die Pfade, etc. stimmen -> Frustrationspotenzial
    - "Jini-out-of-the-box"

Pro...

- Einfach zu lernen
- Einfach zu programmieren
- Gute Konzepte
  - Spontane Vernetzung
  - Leases, Events, Transaktionen, APIs
  - Robuste, fehlertolerante Programme
  - "Selbstheilende" Netze
- Einfache und elegante Programme

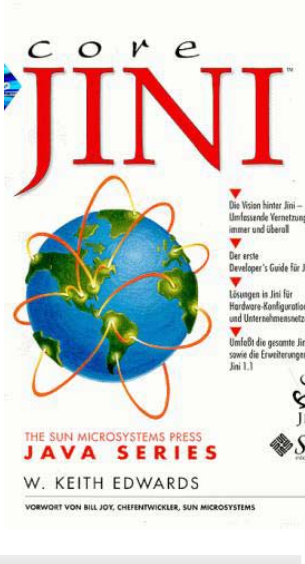
-> Sehr gut für verteilte Applikationen

...und Kontra

- Java2 Standard Edition auf jedem Gerät nötig
- Ressourcen-Bedarf
  - "Kleinstes" JRE2SE auf Compaq iPAQ (Beta)
    - 12MB für Klassen
    - Min. 32MB RAM für Laufzeitumgebung
    - Min. 190MHz. Prozessor
- Mangelhafte Sicherheitskonzepte
  - Momentan keine "eingebaute" Sicherheit
  - "Nur" normale Java2 Security
  - Nicht immer hinreichend für kritische Anwendungen
- Interfaces für "Alles"
  - Festplatten, Drucker, Autos, Kühlschränke, Mobiltelefone,...

Informationen zu Jini

- Internet
  - Jini Homepage bei Sun
    `http://www.sun.com/jini/`
  - Jini Community Homepage
    `http://www.jini.org/`
  - Jini Mailingliste
    `JINI-USERS@JAVA.SUN.COM`
    `http://www.jini.org/openforum.html`
- "Bestes" Buch ("Jini-Bibel"):
  "Core Jini", W. K. Edwards,
  Prentice Hall, 2nd Ed.,
  Juli 2000, ca. 75€



Das war's...

Andreas Zeidler
Datenbanken & Verteilte Systeme
Fachbereich Informatik
TU Darmstadt
mailto:az@ubicomp.de
mailto:az@informatik.tu-darmstadt.de