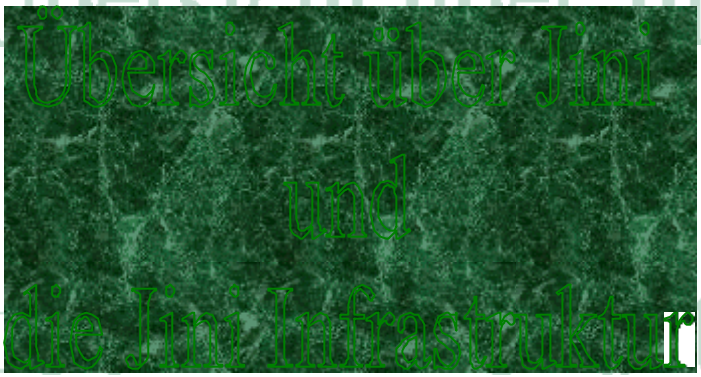

Seminararbeit
von
Wolfgang Klebsattel

Inhaltsverzeichnis

Einleitung

Das Internet als Grosses Ganzes ist an Inhomogenität kaum zu übertreffen. Home-PCs, Server, Router, Switche und sonstige Komponenten der unterschiedlichsten Hersteller werden vereint. Zudem laufen auf den Komponenten unterschiedliche Betriebssysteme und Anwendungen. Trotz dieser Tatsache wird versucht dem Chaos Herr zu werden und zuverlässige, sichere Dienste zu schaffen, um Qualitiy of Service zu gewährleisten.
Die Entwicklung und Lauffähigkeit des Internet dient als Vorbild für viele Visionäre. Weg vom Internet geht es heute zu Endgeräten, wie Handys, PDA, Organizern etc. Auch Haushaltsgeräte spielen eine Rolle.
Die Vision könnte so aussehen, dass der PC über Handy seinen Benutzer auf einen wichtigen Termin hinweist. Der Kühlschrank stellt fest, dass die Milch leer ist und setzt dieses Produkt auf eine Einkaufsliste, die prompt via Email oder SMS übermittelt wird.
Die Kommunikation dieser Menge an Geräten ist eine Herausforderung, der man sich bei SUN gestellt hat. Die Geräte benötigen nicht nur eine eigene Intelligenz, sondern müssen, wie auch das Internet, zuverlässig ihre Dienste erfüllen.
Wegen einer großen Benutzerfreundlichkeit sollen sich Geräte automatisch in ein Netzwerk integrieren. Dabei sollen sie sich automatisch im Netzwerk anmelden, so dass der Benutzer seine Zeit nicht mit der Installation aufwendiger Software verbringt. Ähnlich, wie bei USB, soll sich ein Gerät authentifizieren. Die erforderliche Einstellung für die Kommunikation läuft in einem automatisierten Prozess transparent für den Benutzer ab.
Einen entscheidenden Teil dieser Anforderungen versucht SUN mit Jini (Java Intelligent Network Infrastructure) zu erfüllen. Mit großem PR-Aufwand wurde Anfang 1999 Jini ins Leben gerufen. Auf der Seite "www.jini.org" kann man die Entwicklung von Jini und der Jini-Community verfolgen.
Durch Jini haben Entwickler die Möglichkeit Gerätetreiber zu abstrahieren und in Netzwerken mit Jini-Komponenten (Jini-Netzwerken) zu integrieren. Daraus folgt, dass nicht nur Softwaredetails der Applikationen, die mit Jini zur Verfügung gestellt werden, sondern auch Hardware-Details für den Benutzer transparent sind. Da Jini sowohl Software, als auch Hardware berücksichtigt, wird ein Standard geschaffen.
Neben dem Anmeldeprozess eines Gerätes im Netzwerk als Service nimmt Jini auch Protokollaufgaben war, um den Austausch von Informationen zwischen den Geräten einheitlich zu gestalten.
Grundsätzlich können vier Ziele von Jini zusammengetragen werden.

- Plug-and-Play von Dienstleistungen innerhalb eines Jini-Netzwerks
- Verbergen der Lokalität von Diensten gegenüber dem Benutzer
- Schnelle Verfügbarkeit von Diensten
- Einfachheit

Jini ist in Java realisiert und nutzt RMI (Remote Methode Invocation).
Im weiteren Verlauf wird noch auf Java-Spaces eingegangen, womit wir zwei Kernbausteine von Jini genannt hätten.

Die gesamte Java-API von SUN kann genutzt werden. Des weiteren lässt sich Jini auch mittels Java-Beans realisieren.
Hier zeigt sich wieder eine Stärke von Java. Es bietet für eine große Menge von Problemen Lösungsmöglichkeiten.

Grundlegende Architektur

Allen Teilnehmern in einem Netzwerk stehen bestimmte Funktionen zur Verfügung. Diese Funktionen werden von einem Service-Anbieter bereit gestellt.
Der Service-Anbieter (Dienstleister) ist beispielsweise ein Drucker, welcher eine Einkaufsliste ausdrucken soll. Der Client ist der PC, welcher vom Kühlschrank die zu beschaffenden Produkte erhalten hat. Der PC als Servicenehmer sendet die Produktdaten an den Service-Anbieter, den Drucker. Auch Kühlschrank und PC stehen in einem Serviceanbieter- und Servicenehmer-Verhältnis. Der Look-up-Service sorgt für eine Verbindung der beiden Teilnehmer. Über den Look-up-Service lassen sich alle Service-Anbieter über ein Protokoll anmelden und eintragen. Auf das Protokoll mit dem Namen "Discovery-and-Join" wird später detaillierter eingegangen.
Beim Eintrag eines Anbieters wird ein Proxy-Objekt erzeugt, welches ein Interface implementiert (z.B. das einer print-Methode).
Ein Servicenehmer nimmt mit einem Look-up-Service Kontakt auf und stellt eine Anfrage, ob eine von ihm gewünschte Dienstleistung und die damit verbundene Implementierung zur Verfügung steht. D.h. der Dienstnehmer stellt das Interface und der Look-up-Service bzw. das erzeugte Proxy-Objekt hat die Aufgabe dieses Interface zu implementieren.
Die Kontaktaufnahme geschieht über die Funktion "Discovery". Auch darauf wird später genauer eingegangen.
Der LUS sucht nach der Anfrage des Clienten in seinen Einträgen nach einem Server (Dienstanbieter) und dem damit verbundenen Proxy-Objekt, welches die geforderte Interface-Implementierung anbietet.
Ist dieser gefunden, kann der Client einen Proxy über den LUS herunterladen, welchen er für die Ausführung der entfernten Methoden benötigt wird.
Der Proxy wird, verglichen mit einem Treiber, für den Zeitraum, in dem der Dienst ausgeführt wird, auf dem Clienten installiert.
Ist die Verbindung hergestellt bzw. ist ein Dienst vermittelt worden, wird der LUS nicht mehr benötigt. Die weitere Kommunikation geschieht über eine Verbindung, welche über das Proxy-Objekt aufgebaut wird.
Anbieter und Nehmer wickeln den weiteren Datenaustausch über ein eigenes Kommunikationsprotokoll ab. Dadurch ist eine Unabhängigkeit vom LUS und eine Synchronisation zwischen den Teilnehmern gewährleistet.
Durch die Vermittlerrolle des LUS bleibt die interne Struktur eines Netzwerks und anderer Teilnehmer für Client und Server transparent. Der Client bekommt lediglich die Informationen, die er für eine Kontaktaufnahme

zum Dienstanbieter benötigt. Ein Interface dient als Kommunikationsschnittstelle. Durch diese Schnittstelle braucht weder der Anbieter, noch der Dienstnutzer von der Existenz des anderen zu wissen. Vom Service-Anbieter wird erwartet, das er das erforderliche Interface implementiert und eine Klasse als Proxy zum Download bereitstellt. Auf Client-Seite wird die Kommunikation mit dem angeforderten Dienst vom Proxy abgewickelt. Das Kommunikationsprotokoll bleibt für den Dienstnehmer transparent.

Programmierer sind in der Entwicklung von Diensten deswegen frei, da die Komponenten im Großen und Ganzen propretär bleiben. Damit die Clienten unter Jini das Interface und dessen Implementierung als Proxy-Objekt verstehen, muss dieses in Java geschrieben sein.
Zu einem Proxy-Objekt können noch weitere Objekte nachgeladen werden. Daraus ergibt sich, dass das Proxy-Objekt in seiner Größe nicht beschränkt ist. Für den Dienst heißt dies, dass er komplett auf die Seite des Dienstnehmers verlagert werden kann.
Das Netzwerk wird anschließend nicht mehr so stark und unnötig belastet, wie es bei Softwarediensten der Fall wäre.

Die Jini Protokollschichten

Jini, mit seinen Klasen und Interfaces, ist komplett in Java zwei geschrieben und erweitert so die Standardklassenbibliotheken von Java. Jini-Programme werden genau, wie andere Java-Programme von der Virtuellen-Maschine ausgeführt. Daher sind sie ebenfalls plattformunabhängig. Die Objektorientierung verleiht Jini einen großen Vorteil. Damit können Jini-Services in Objekten gekapselt werden. Auf diese Weise können die Objekte, ohne etwas über ihre innere Struktur zu verraten, von außen angesprochen werden.
Jini setzt auf dem gleichen Protokoll auf, wie es von RMI (Remote Methode Invocation) genutzt wird. Mit diesem Protokoll ist es möglich Methoden von Objekte aufzurufen, die auf einer anderen virtuellen Maschine erzeugt wurden. Dadurch wird die Kommunikation zwischen LUS, Service und Client im Netzwerk realisiert. Mit RMI werden zudem die Proxy-Klassen des Dienstanbieters auf den Client geladen.
Mit Subsets, die RMI bietet, ist die Anbindung an andere Standards, wie CORBA, möglich. Damit kann Jini auf andere Protokolle für entfernte Methodenaufrufe aufsetzen.
Auf Systemebene sorgen Protokolle, die unterhalb von Java und RMI angesiedelt sind, für den Datenaustausch. TCP/IP und UDP bilden hier den gängigen Standard für den Datenaustausch über Netzwerk und Internet.
Die erwähnten Protokolle haben lediglich die Aufgabe alle Funktionen für eine Java-Anbindung zu Verfügung zu stellen. Daraus folgt, dass Jini von Netzwerkprotokollen unabhängig ist, da diese transparent bleiben und nur

das System bzw. die virtuelle Maschine über die Implementierung bescheid wissen muss. Jini setzt auf Protokollen auf, die in unterschiedliche Abstraktionsstufen aufgeteilt sind.

- Netzwerkprotokoll:
  Dies ist systemunabhängig und für den Datenaustausch zwischen Dienstanbieter und Dienstnehmer verantwortlich.
- Java Virtuelle Maschine:
  Diese setzt auf dem Netzwerkprotokoll auf und führt den Programmcode unabhängig vom übrigen System aus.
- Java RMI:
  RMI ermöglicht die Kommunikation zwischen zwei unterschiedlichen VMs in einem Netzwerk
- Jini-Lookup-Service:
  Durch diesen wird RMI abstrahiert. Dadurch werden alle Teilnehmer in einem Netzwerk öffentlich bekannt und verfügbar gemacht .
- Service-Protokoll:
  Jeder Netzteilnehmer verfügt über sein eigenes Service-Protokoll. Über dieses kann er mit anderen Teilnehmern im Netz kommunizieren. Dazu muss das Service-Protokoll über die Proxy-Klasse an den Client eines Services übergeben werden. Über ein öffentliches Interface ist das Protokoll abstrahiert, so dass es für Objekte transparent ist und nicht bekannt sein braucht.
  Die Struktur der genannten Protokollschichten zeigt, dass sich Teilnehmer in einem Netzwerk durch die starke Abstraktion über eine gemeinsame und unabhängige Kommunikationsebene austauschen. Diese Tatsache soll auch weiterhin mit der Jini-Technologie verbreitet und weiterentwickelt werden.

Discovery and Join

Jini arbeitet neben den externen Komponenten und Protokollen auch intern mit Protokollen. Diese sind Bestandteil des folgenden Abschnitts. Die zur Jini-Architektur gehörenden Konzepte Discovery and Join bilden den Anfang der Betrachtung der internen Struktur. Geht es um die Anmeldung eines Dienste bei einem LUS, ist das Hauptaugenmerk auf die erwähnten Konzepte Discovery and Join zu richten. Die derzeitige Jini-Version 1.0 setzt bzgl. Netzwerkprotokollen auf IP auf.
Ein neues Gerät bzw. Dienst wird in ein Netzwerk eingefügt. Die Registrierung übernimmt das Gerät ohne äußeres Eingreifen des Benutzers. Das Gerät und seine Dienste werden automatisch allen bereits vorhandenen Teilnehmern im Netzwerk zur Verfügung gestellt.
Eine Voraussetzung dafür ist die Anmeldung an einem oder mehreren Look-up-Services. Dieser Prozess gliedert sich in mehrere Arbeitsschritte:

- "Discovery". Dabei wird nach den Teilnehmern im Netzwerk gesucht. Die Discovery-Entity, wie das gerät genannt wird, welches neu ins Netzwerk integriert wurde, sendet Anfragen durch das Netz und lokalisiert auf diese Weise existierende Look-up-Services.
  Das Aussenden der Anfragen kann über drei verschiedene Methoden erfolgen:
  - Multicast Request Protokoll
    UDP-Datagramme mit maximal 512 Byte werden versendet. Beim Multicast Request Protokoll werden eine bestimmte Gruppe von LUS angesprochen. Man spricht in diesem Zusammenhang auch von der Multicast-Gruppe. Der involvierte Service braucht nichts über die dabei verwendete Netzadresse zu wissen.
  - Unicast Request Protokoll
    Es können nicht nur Gruppen angesprochen werden. Das Unicast Request Protokoll ermöglicht, dass über den Look-up-Service mittels TCP eine bestimmte IP-Adresse angesprochen wird.
  - Multicast Announcement Protokoll
    Die besten Rechner müssen gelegentlich neu gestartet werden. Auch kommt es in den besten Familien schon mal zu einem Absturz des Systems. In diesem Fall würden Dienstanbieter, um auf dem laufenden zu bleiben, ständig Anfragen (Discovery) über das Netz senden. Da dies zu überflüssigem Verkehr in einem Netzwerk führt liegt nah. Um gerade dies zu vermeiden hat das Multicast Announcement Protokoll die Aufgabe in einem Notfall, wie Neustart oder Absturz des Servers, einem oder mehrere LUS die Möglichkeit zu bieten, Dienstnehmer (Clients) im Netz über ihre Erreichbarkeit zu informieren.
    Auf diese Weise ist auch bei Notfällen für einen reibungslosen Ablauf gesorgt, so dass der Benutzer nicht eingreifen braucht.

- Joining:
  Nachdem die einzelnen Anfragemöglichkeiten gezeigt wurden zurück zum nächsten Schritt. Bei erfolgreicher Suche wird die Anmeldung, das "Joining", vorgenommen.
  Ein Objekt wird in Form einer Proxy-Klasse bei den gefundenen LUS gespeichert. Ein Service benötigt für eine eindeutige Identifikation folgende Informationen:
  - Service-ID
  - Menge von Attributen
  - Menge von Gruppen bei welchen sich der Service hat registrieren lassen
  - Menge an dedizierten Look-up-Services

Mit dem Joining ist der Anmeldeprozess einer Komponente abgeschlossen und der Betrieb in einem Netzwerk kann aufgenommen werden.

Quellenverzeichnis

Boger M.: Java in verteilten Systemen, Heidelberg, 1999
Oaks S.; Wrong H.: Jini in a Nutshell, 2001
Bader. H; Huber W.: Jini. Die intelligente Netzwerkarchitektur in Theorie und Praxis, München, 1999

http://www.htw.uni-sb.de/fb/gis/people/wpauly/vortraege_internet/ws99_00/JINI/vortrag.html