Fachseminar SS 2000
Ubiquitous Computing

Spontane Vernetzung
Dienstbeschreibung und service discovery

Roland Kurmann

7. Juni 2000

Zusammenfassung

In dieser Arbeit wird auf die spontane Vernetzung, also die selbstkonfigurierende, sofortige Vernetzung und Nutzung von Geräten eingegangen. In einer ubiquitären Welt werden Dienste von Geräten nachgefragt und angeboten. Damit eine Nachfrage möglich wird, müssen Dienste beschrieben werden können. Es werden kurz die Grundlagen besprochen, danach werden verschiedene konkrete Lösungsansätze, wie z.B. Jini, exemplarisch vorgestellt.

Inhaltsverzeichnis

1 Einleitung

In einer ubiquitären Welt treten viele Geräte zur Kooperation miteinander in Beziehung. Es ist daher notwendig, dass sich die Geräte sehr einfach vernetzen können. Die Vision ist die spontane Vernetzung, also die selbstkonfigurierende, sofortige Vernetzung und Nutzung der Geräte.

Diese Geräte stellen dann einen Dienst, wie das Drucken einer Datei im Netz zur Verfügung. Um miteinander kooperieren zu können, müssen die Geräte in der Lage sein ihre Dienste anzupreisen. Umgekehrt muss es möglich sein, die für sich nützlichen Dienste zu finden.

Um die Vision der Ad-hoc Vernetzung zu verdeutlichen, kann man folgendes Szenario betrachten: In einem Seminar muss ein Vortrag vorgetragen werden. Der Vortragende hat seine Folien auf seinem Notebook vorbereitet und gespeichert. Im Seminarraum steht ein Videobeamer zur Verfügung. Der Vortragende packt nun sein Notebook aus, stellt es auf den Tisch und teilt dem Computer mit, dass der Vortrag gehalten werden soll. Der Computer meldet sich nun bei der Netz-Infrastruktur des Raumes mittels drahtloser Funktechnologie an und sucht den Videobeamer. Die Infrastruktur liefert die Adresse des Beamers, umgehend stimmen sich die Geräte ab und konfigurieren sich. Es werden die Auflösung der Projektion und weitere Parameter eingestellt. Der Vortrag kann nun ohne weitere Konfigurationsprobleme beginnen.

2 Spontane Vernetzung

Es stellt sich die Frage wie die spontane Vernetzung erreicht werden kann. Die spontane Vernetzung betrifft verschiedene Ebenen der Kommunikationshierarchie. Es lassen sich drei Ebenen ausmachen, die zur spontanen Vernetzung beitragen:

Netzwerk-Ebene: Diese Ebene ist für die Kommunikation zwischen den verschiedenen Diensten zuständig. Die Ad-hoc Vernetzung erfordert nun die automatische Zuweisung der Kommunikationsparameter. Bei TCP/IP-Netzen ist dies beispielsweise eine IP-Adresse, die in diesem Fall per DHCP [14] zugewiesen wird.

Infrastruktur-Ebene: Auf dieser Ebene erfolgt der Austausch und die Vermittlung von Informationen über Dienstanbieter und -nehmer. Die Beschreibung eines Dienstes, die Anfragemöglichkeiten und die Protokolle zur Anmeldung und Nachfrage sind die zentralen Punkte dieser Ebene.

Dienst-Ebene: Nach der Vermittlung der Dienste wollen diese genutzt werden. Die Nutzung kann unabhängig von der Infrastruktur-Ebene sein, kann aber von der Infrastruktur-Ebene beeinflusst werden.

Diese Arbeit befasst sich im Weiteren mit der Infrastruktur-Ebene.

In Abbildung 1 ist der generelle Ablauf einer Dienstvermittlung dargestellt. Zuerst erfolgt eine Registrierung des angebotenen Dienstes bei einem Dienstvermittler. Später kann ein anderes Gerät für einen bestimmten Dienst eine Anfrage stellen. Der Dienstvermittler ermittelt einen geeigneten Dienst und teilt das Resultat dem Gerät mit. Dieses kann nun den Dienstanbieter kontaktieren.

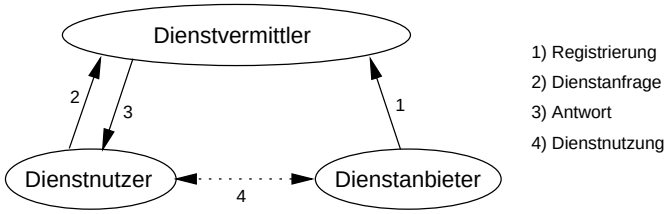

Abbildung 1: Genereller Ablauf einer Dienstvermittlung.

Bei der Beschreibung eines Dienstes kann man grundsätzlich zwei Möglichkeiten unterscheiden: Die Dienste sind im Voraus (zur Entwicklungszeit des Dienstnutzers) bekannt (*Closed World Assumption*) oder nicht (*Open World Assumption*). Im Falle der *Open World Assumption* muss eine semantische Beschreibung eines Dienstes gewählt werden, damit im Voraus unbekannte Kombinationen ermöglicht werden können. Das Vermitteln mit Hilfe von semantischen Informationen ist sehr schwierig und wird in der künstlichen Intelligenz behandelt. Das AI-Trader Projekt [12] von der Universität Frankfurt beschäftigt sich mit dieser Problematik. Im Falle der *Closed World Assumption* reicht es, wenn die Dienste syntaktisch mittels spezifizierter Schnittstellen beschrieben werden . Die Auswahl eines passenden Dienstes ist in diesem Fall einfacher, es muss eine passende Schnittstelle unter der Menge der vorhandenen Schnittstellen ausgewählt werden.

3 Überblick über Produkte und Projekte

In den folgenden Kapiteln werden verschiedene aktuelle Infrastrukturtechnologien vorgestellt. Sie gehen alle von einer *Closed World Assumption* aus. Die Technologien werden bezüglich ihrer Dienstbeschreibung, Dienstanfrage und dem *Service Lookup* analysiert. Der Beitrag zu Jini ist ausführlicher, da diese Technologie am weitesten fortgeschritten ist und die anderen Technologien sich meistens im Vergleich zu Jini präsentieren.

3.1 Jini (Java Intelligent Network Infrastructure)

Jini [1, 2, 3, 4] wurde von Sun Microsystems zur dynamischen Dienstvermittlung in lokalen Umgebungen, wie zum Beispiel das Netzwerk eines Unternehmens konzipiert.

Jini baut auf der Java-Technologie auf und ist vollständig in der Programmiersprache Java realisiert. Es ist deshalb erforderlich, dass alle Geräte die Jini einsetzen wollen eine *Java Virtual Machine* besitzen.

Jini ist auf der Infrastruktur-Ebene angesiedelt und erwartet eine funktionierende Netzwerk-Ebene, zum Beispiel einen initialisierten IP-Protokollstack. Jini kümmert sich also nicht selbst um die Netzwerk-Ebene.

Die Architektur von Jini ist dienstorientiert. Ein Gerät, wie zum Beispiel ein Drucker, bietet einen *Dienst* im Netz an. Ein solcher Dienst wird durch ein *Proxy* repräsentiert. Ein Proxy ist ein Stück *mobiler Code* und ist der lokale Stellvertreter des Dienstes, der die Kommunikation mit dem eigentlichen Dienst übernimmt. Daneben gibt es Klienten, die Dienste nachfragen. Weiter gibt es eine zentrale Instanz, den

Beispiel 1 Dienstbeschreibung und Dienstanfrage mit Jini.

Dienstschnittstelle

```
interface PrinterInterface implements Serializable {
    void print();
}
```

Dienstimplementation

```
public class PrinterImpl implements PrinterInterface {
    void print() { ... }
}
```

Attribute

```
class Resolution extends AbstractEntry {
    int dpi;
    Resolution(int dpi) {this.dpi = dpi;}
}
class Info extends AbstractEntry {
    String vendor;
    int year;
    Info(String vendor, int year) { this.vendor = vendor; this.year = year; }
}
```

Dienstbeschreibung und Registrierung

PrinterInterface *printerProxy* = new PrinterImpl();
Entry[] attr = {new Resolution(600), new Info(“HP”, 1996)};
JoinManager(printerProxy, attr, ...)
Der JoinManager erzeugt im LUS ein ServiceItem-Objekt, welches die eigentliche Beschreibung des Dienstes darstellt:
⇒new ServiceItem(id, printerProxy, attr)

Anfrage

Class[] classes = {PrinterInterface.class};
Entry[] attr = {new Resolution(600), new Info(“HP”, *null*)};
ServiceTemplate anfrage = new ServiceTemplate(null, classes, attr);

Antwort

PrinterInterface *printerProxy* = (PrinterInterface) LUS.lookup(anfrage);

Nutzung

printerProxy.print();

Lookup-Service (LUS), der für die Registrierung und Vermittlung von Diensten zuständig ist. Das Lookup-Protokoll funktioniert wie im allgemeinen Schema in Abbildung 1 angegeben mit der Ergänzung der Proxies. Zur Registrierung eines Dienstes wird ein Proxy zum LUS geschickt. Das Resultat einer Anfrage ist wiederum ein Proxy. Der Klient steuert nun den Dienst via dem Proxy, der die Befehle weiterleitet.

In Jini wird ein Dienst durch Java-Interfaces und Attribute beschrieben. So gibt es zum Beispiel ein Java-Interface *Printer* für Drucker und das Proxy-Objekt muss dieses Interface implementieren. Durch Vererbung der Java-Interfaces können Dienste hierarchisch organisiert werden, zum Beispiel ein Farbdrucker implementiert das Interface *ColorPrinter*, welches von *Printer* erbt. Die Proxies können durch hinzufügen von Attributen weiter spezifiziert werden. Attribute sind vom Interface *Entry* abgeleitete Objekte und spezifizieren Dinge wie die Auflösung eines Druckers. Siehe Beispiel 1 zur Illustration. Eine Anfrage an den LUS enthält die Interfaces, die vorhanden sein müssen, wie auch Attribute die erfüllt sein müssen. Bei einer Anfrage sucht der LUS ein Proxy-Objekt, das die Interface-Anforderungen erfüllt, wobei die Vererbungshierarchie beachtet wird. Die Attribute werden mit einer exakten *Match-Semantik* geprüft. Attribute können durch ein *Wildcard* (*null* Referenz) offen gelassen werden. Dem Anfrager wird als Resultat das ermittelte Proxy-Objekt zugesandt.

Die Proxies haben weitere Eigenschaften, die von Nutzen sein können. Durch die lokale Stellvertretung kann die Kommunikation mit dem eigentlichen Dienst über die Proxies abgewickelt werden, was eine Kapselung proprietärer Protokolle ermöglicht. Durch die Verwendung von Proxies (*mobiler Code*) ist es möglich ein *Graphical User Interface* (GUI) dem Klienten mitzuschicken. Damit kann die Anfrage von der Maschine-Maschine-Ebene auf die Mensch-Maschine-Ebene verlagert werden und mit Hilfe des Menschen “bisher unbekannte” Dienste (im Sinne der *Open World Assumption*) zu nutzen.

3.2 SLP (Service Location Protocol)

SLP [1, 5, 6] ist zur Dienstvermittlung in einer lokalen Umgebung spezifiziert worden. Wie Jini erwartet SLP eine funktionsfähige Netzwerk-Ebene. Die Architektur ist ähnlich zu Jini, es gibt Dienste, Klienten und eine Vermittlungsinstanz.

Das Forschungsprojekt RDP [7] hat zum SLP Standard beigetragen.

Ein Dienst wird durch einen Servicetyp spezifiziert, welcher in der URL zum Dienst als Protokolltyp angegeben wird. Die weitere Dienstbeschreibung in SLP erfolgt textuell mit Attributen in der Form von Name-Wert-Paaren, wie zum Beispiel SIZE = A4.

Eine Anfrage richtet sich nach einem bestimmten Servicetyp. Die Anfrage kann eine Boole’sche Verknüpfung von Attributen enthalten, die erfüllt sein müssen. Als Resultat wird die URL des Dienstes zurückgeliefert. Siehe dazu Beispiel 2.

3.3 S SDS (Secure Service Discovery Service)

S SDS [8] ist ein Forschungsprojekt mit dem Ziel der Entwicklung einer Infrastruktur für die spontane Vernetzung für grössere Umgebungen wie das Internet.

Ein Dienst wird in S SDS mit Hilfe von XML [13] beschrieben. Dies erlaubt im Vergleich zu SLP die hierarchische Schachtelung von Attributen. In XML ist es zu-

Beispiel 2 Dienstbeschreibung und Dienstanfrage mit SLP.

Beschreibung

Lifetime:	10800
URL:	service:lpr://vpp.ethz.ch:515/queue
Attributes:	(SCOPE=STUDENTS), (PAPER COLOR=WHITE), (PAPER SIZE=A4), (UNRESTRICTED_ACCESS), (LANGUAGE=POSTSCRIPT), (LOCATION=4th FLOOR)

Anfrage

lpr//(&	(LANGUAGE==POSTSCRIPT), (UNRESTRICTED_ACCESS), (LOCATION==4th FLOOR))/

Antwort

service:lpr://vpp.ethz.ch:515/queue

dem möglich die Struktur der XML-Datei mit einer *Document Type Definition* (DTD) vorzuschreiben.

Die Anfrage wird in Form einer *XML Template Datei* formuliert. Die *Service Discovery* wird bei diesem Forschungsprojekt mit dem Programm *XSet* realisiert, welches eine XML *Search Engine* ist. Siehe dazu Beispiel 3.

3.4 Bluetooth/SDP (Service Discovery Protocol)

Das Protokoll SDP [1, 9] ist ein Teil des Standards Bluetooth, welcher eine Möglichkeit zur drahtlosen Vernetzung in einem Raum spezifiziert. Mit SDP möchte man den Anwendern der Bluetooth-Technologie eine einfache Möglichkeit zum Auffinden von Diensten bieten.

Die Dienste werden in Serviceklassen eingeteilt, die durch *Universally Unique Id's* (UUID) bezeichnet werden. Eine Serviceklasse kann mit einem Java-Interface verglichen werden, da die Serviceklasse alle zu beschreibenden Attribute vorschreibt und Serviceklassen vererbt werden können. Die Attribute sind wie bei SLP in Form von Name-Wert-Paaren anzugeben. Bei den Attributen wird eine *Wildcard-Semantik* verwendet.

Die Anfrage geschieht mit einem Suchmuster, welches die gewünschten Serviceklassen und Attribute angibt. Ein Dienst wird als Resultat zurückgeliefert, wenn das Suchmuster eine Teilmenge der Attribute aller nachgefragten Serviceklassen ist.

SDP ermöglicht zusätzlich das *Browsen* durch die angebotenen Dienste.

3.5 HAVi (Home Audio Video Interoperability)

HAVi [10] ist eine Infrastruktur zur digitalen Vernetzung von Audio/Video-Heimgeräten. Es wird Firewire (IEEE 1394, i-Link) als Netzwerktechnolgie verwendet. Es sind

Beispiel 3 Dienstbeschreibung und Dienstanfrage in S SDS mit XML.

Beschreibung

```
<?xml version="1.0"?>
<!DOCTYPE printcap SYSTEM "http://vpp.ethz.ch/printer.dtd">
<printcap>
    <name>CLC1000</name>
    <location>rzspez</location>
    <postscript>yes</postscript>
    <color>yes</color>
    <duplex>yes</duplex>
    <rmiaddr>http://rzspez.ethz.ch/CLC1000</rmiaddr>
</printcap>
```

Anfrage

```
<?xml version="1.0"?>
<printcap>
    <color>yes</color>
    <postscript>yes</postscript>
</printcap>
```

vier Gerätekategorien definiert, wobei die Umfassendste eine *Java Virtual Machine* vorschreibt.

Die Dienstbeschreibung erfolgt mit Name-Wert-Paaren. Die Dienstanfrage findet mit einem Boole'schen Ausdruck statt.

Es wird an einer Kooperation mit Jini gearbeitet.

3.6 UPnP (Universal Plug and Play)

UPnP [1, 11] wird zur Vernetzung von Home/Office-Geräten konzipiert und ist momentan noch im Entwicklungsstadium.

Ein Dienst wird in UPnP mit einem Servicetyp beschrieben. Es gibt keine Attribute. Ein Dienst kann durch seinen Servicetyp nachgefragt werden. Als Resultat bekommt man die Menge der in Frage kommenden Dienste.

In UPnP ist es möglich, zu einem Dienst eine XML-Beschreibung auf einem *Description Server* zu hinterlegen. Dies erlaubt dem Klienten selbst den optimalen Dienst aus der Resultatmenge der zurückgelieferten Dienste wählen, nämlich durch das Heranziehen der XML-Beschreibungen.

3.7 Vergleich

Die oben vorgestellten Technologien werden in der nächsten Tabelle verglichen. Es werden die Flexibilität der Beschreibungs- und Anfragemöglichkeiten verglichen, wie oben nicht betrachtete Aspekte der verschiedenen Infrastrukturtechnologien. Es wird angegeben, ob der Betrieb mit einer zentralen Instanz, mit mehreren dezentralen Instanzen oder beiden Varianten möglich ist. Weiter wird untersucht, ob es möglich ist, Gruppen von Diensten, zum Zwecke der Administration oder der Skalierbarkeit zu

bilden. Es wird angegeben, ob Sicherheitsüberlegungen bei der Entwicklung miteinbezogen worden sind.

	Jini	SLP	S SDS	SLD	HAVi	UPnP
Flexibilität der Dienstbeschreibung und der Dienstanfrage	++	++	+++	+	++	-
zentral / dezentral	√ / -	√ / √	√ / -	- / √	- / √	√ / √
Gruppenkonzept	√	√	√	-	-	?
Skalierbarkeit	+	+	++	-	-	-
Sicherheitsüberlegungen	-[1]	+	++	?	+	?
jetzt einsetzbar	√	√	-	√	?	-

[1] Sun arbeitet an einer sicheren Variante von RMI, welche bei Jini eingesetzt werden soll.

4 Ausblick

Das Gebiet der spontanen Vernetzung erfordert weitere Forschungsanstrengungen.

Die Sicherheit muss in verschiedenen Bereichen verbessert werden, zum Beispiel der Schutz des Klienten vor dem Missbrauch von Proxies für andere Aufgaben, das Verstecken von geheimen Diensten und das Sicherstellen der Authentizität von Diensten. Dabei muss man bedenken, dass zusätzliche Sicherheit die Spontanität einschränken kann.

Die Dienstbeschreibung und Dienstvermittlung findet jetzt vor allem auf syntaktischer Ebene statt und ist eingeschränkt. Wünschenswert wäre eine “intelligentere” Dienstvermittlung, die aber trotzdem einfach und effizient ist. XML bietet für die Dienstbeschreibung weitere Möglichkeiten, die genutzt werden können.

Die Skalierbarkeit der Technologien muss sichergestellt werden, um es vielen ubiquitären Geräten zu ermöglichen sich spontan zu vernetzen.

Bei den oben vorgestellten Technologien, die auf der *Closed World Assumption* beruhen, ist die Standardisierung der Schnittstellen weiter voranzutreiben, um die Verwendbarkeit im grösseren Rahmen sicherzustellen.

Zudem ist es denkbar, dass sich mehrere Technologien durchsetzen werden, was die Notwendigkeit der Zusammenarbeit erfordert. Beim Übergang zur spontanen Vernetzung muss überlegt werden, wie mit bestehenden Geräten (*Legacy Devices*) umgegangen wird.

Literatur

[1] R. Kehr, *Infrastruktur-Konzepte für die spontane Vernetzung von Diensten und Geräten*, Internes Dokument TU Darmstadt, Version 3, 2000.

[2] Sun Microsystems, *Jini*, `http://www.sun.com/jini`, 1998.

[3] P. Hasselmeyer, F. Mattern und A. Zeidler, *Jini*, Folien, `http://www.inf.ethz.ch/department/IS/vs/publ/`, 2000.

[4] A. Zeidler und M. Gruteser, *Praktische Einführung in Jini*, `http://www.inf.ethz.ch/department/IS/vs/publ/`, 1999.

[5] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan, *Service Location Protocol (SLP)*, Internet RFC 2165, 1997.

[6] *Service Location Protocol Homepage*, `http://srvloc.org`, 1998.

[7] C. Perkins and H. Harjono, *Resource discovery protocol for mobile computing*, Mobile Networks and Applications, 1, 447-455, 1996.

[8] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz, *An Architecture for a Secure Service Discovery Service*, Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99), Seattle, 1999.

[9] Bluetooth Consortium, *Specification of the Bluetooth System Version 1.0 A - Core*, `http://www.bluetooth.com`, 1999.

[10] HAVi Consortium, *The HAVi Specification: Specification of the Home Audio/Video Interoperability Architecture Version 1.0*, `http://www.havi.org`, 2000.

[11] Microsoft, *Universal Plug and Play*, `http://www.upnp.org`, 1999.

[12] S. Markwitz, F. Gudermann, C. Kaiser, H. Röhrig, K. Römer, R. Farsi und A. Puder, *AI-Trader*, `http://www.vsb.cs.uni-frankfurt.de/projects/aitrader/`, 1996.

[13] T. Bray, J. Paoli, and C. Sperberg-McQueen, *Extensible Markup Language (XML) 1.0*, `http://www.w3.org/TR/1998/REC-xml-19980210.html`, 1998.

[14] R. Droms, *Dynamic Host Configuration Protocol (DHCP)*, Internet RFC 2131, 1997.