Bericht zum Vortrag
„Beschreibung und Lokalisierung von Diensten in drahtlosen Netzen“

Von: Leanza Antonino
Betreuer: Frank Siegemund
Professor: Friedemann Mattern

Zusammenfassung

Es wird von mobilen Geräten mehr und mehr erwartet, die Fähigkeit zu haben, sich in vorhandene Umgebungen dynamisch integrieren zu können. Das soll möglichst ohne manuelle Konfiguration und auch später möglichst ohne manuellen Wartungsaufwand geschehen. Das System sollte „selbstkonfigurierend“ sein.

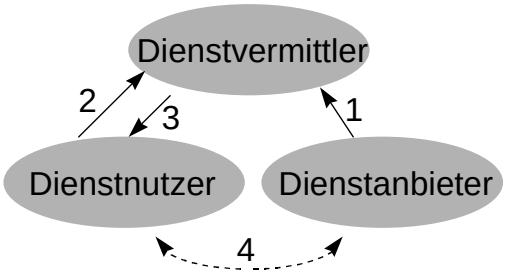
In solchen Szenarien *spontaner Vernetzung* ist es wichtig, ein Mechanismus zu haben, welcher es erlaubt, nach vorhandenen Diensten nachzufragen oder die eigenen anzubieten. Um dies zu ermöglichen, muss man sich auf eine Form der Dienstbeschreibung einigen und Suchmechanismen anbieten.

Es wird zunächst auf einige generelle Konzepte der Dienstbeschreibung eingegangen. Dann werden einige konkrete Architekturen aus Industrie und Hochschulen vorgestellt und auf deren besondere Eigenschaften hingewiesen.

1. Genereller Ablauf der Dienstvermittlung

Obwohl sich die heute verfügbaren Architekturen zur Dienstbeschreibung und Dienst-Vermittlung z. T. beträchtlich unterscheiden, weisen sie dennoch gewisse Ähnlichkeiten auf.

1.1 Dienstvermittlung



1: Registrierung
2: Dienstanfrage
3: Antwort
4: Dienstnutzung

Abbildung 1: Die nötigen Interaktionen gemäss [1], die vor der eigentlichen Dienstnutzung stattfinden

In einem System in welchem Dienstvermittlung angeboten wird, unterscheidet man i. A. die drei in Abbildung 1 dargestellten Komponenten. Bei denjenigen Architekturen, die einen Dienstvermittler zur Verfügung stellen, können die Dienstanbieter zunächst einmal ihre Dienste *registrieren*. Später kann ein anderes Gerät, der Dienstnutzer, welcher nach einem bestimmten Dienst sucht, den Dienstvermittler befragen. Der Dienstvermittler kann, falls ein geeigneter Dienst existiert, diesen ermitteln und das dem suchenden Dienstnutzer mitteilen. Dieser kann nun direkt den Dienstanbieter kontaktieren.

Um eine Kenntnis der lokalen Gegebenheiten zu erhalten, müssen die Dienstnutzer resp. Anbieter zuerst eine Instanz eines Dienstvermittlers ausfindig machen. Diesen Prozess bezeichnet man als *discovery*.

Die Mächtigkeit der Dienstvermittlung der verschiedenen Architekturen ist sehr unterschiedlich: Einige Systeme liefern als Resultat einer Dienstsuche lediglich die Information zurück, ob ein bestimmter Dienst vorhanden ist oder nicht, andere Systeme liefern zusätzlich noch genügend Information, um den gewünschten Dienst zu benutzen.

2. Möglichkeiten der Diensteschreibung

Prinzipiell kann man bei der Beschreibung eines Dienstes zwei Möglichkeiten unterscheiden: Entweder sind die Dienste im Voraus bekannt, oder nicht. Im ersten Fall spricht man von der sog. *Closed-World Assumption*, im letzten von der *Open-World Assumption.*

2.1 Open-World Assumption

Im Falle der *Open-World Assumption* ist kein standardisiertes, gemeinsames Wissen auf Dienstebene bekannt. Neue, modifizierbare Dienste können jederzeit erstellt werden. Beim Definieren der Dienstschnittstelle oder Dienst-Struktur haben die Teilnehmer eine ganz bestimmte Semantik im Sinn (*explizite Semantik)*. Es ist dazu nötig, eine semantische Beschreibung eines Dienstes zu definieren, damit im Voraus unbekannte Kombinationen von Diensten zugelassen werden können.

Es ist sehr schwierig, nur mit Hilfe semantischer Information die Dienste zu vermitteln. Man versucht, der komplexen Natur dieser Problematik mit Hilfe von *künstlicher Intelligenz* zu begegnen.

2.2 Closed-World Assumption

Einfacher ist es im Falle der *Closed-World Assumption*: Die Struktur der Dienste wird syntaktisch, im Vornherein festgelegt und ist allen Teilnehmern bekannt. Die Suche nach einem Dienst ist in einem solchen Fall sehr einfach: man muss lediglich nach einer 'Schnittstelle' oder einer Kombination von 'Schnittstellen' suchen, die den Bedürfnissen des Klienten genügen.

Alle Architekturen, die im Folgenden beschrieben werden, gehen von einer *Closed-World Assumption* aus.

3. Übersicht der Dienstbeschreibungs- und Dienstvermittlungs-Technologien

Spontane Vernetzung beinhaltet Kommunikation auf verschiedenen Ebenen der Kommunikationshierarchie nach [2] :

Abbildung 2: In diesem Bericht werden nicht die Netzwerk- und proprietären Geräteprotokolle behandelt. Es geht vielmehr um die sich 'dazwischen' befindliche Dienstvermittlungsschicht.

3.1 Bluetooth

Bluetooth ist ein Standard für Mobilfunk-Kommunikation für kurze Distanzen [3]. ***SDP** (Service Discovery Protocol)* ist ein Teil der Bluetooth-Spezifikation ([3] Teil E).

Dienstbeschreibung: Die Dienste werden in Bluetooth als *Service-Klassen* beschrieben, welche die Attribute vorgeben und durch eine *UUID* (Universally Unique ID, 128 bit) eindeutig repräsentiert sind. Diese Klassen implizieren die 'Natur' der Applikations-Entität. Sie definieren die Semantik der Datenströme und der Attribute. Die Attribute sind in Form von Schlüssel-Wert-Paaren angegeben. Ein Bluetooth-Gerät innerhalb des *Piconet*s, welche Dienste erbringen möchte, muss einen lokalen *SDP-Server* implementieren um dort seine Dienste zu registrieren. Einen zentralen Dienstvermittler gibt es also nicht.

Dienstlokalisierung: Die Anfrage, die an einen *SDP-Server* gesendet wird, besteht aus einer Liste der UUID's von gewünschten Service-Klassen und Attributen. Andere Typen wie Zahlen oder Zeichen können bei der Suche nicht verwendet werden. Es wird nur die Information zurückgeliefert, ob ein *SDP-Server* eine Obermenge der gesuchten UUID's beinhaltet oder nicht. Der Nachteil, dass nicht nach anderen Attributen ausser UUID-Typen und deren Werten gesucht werden kann, wird in der Praxis durch einfache *Navigationsmöglichkeiten* ein wenig kompensiert.

3.2 IrDA (Infrared Data Association)

IrDA [4] ist gleichzeitig der Name einer Vereinigung und die Bezeichnung eines von ihr spezifizierten Standards für Infrarot-Kommunikation. Wie im Falle von *Bluetooth*, ist auch *IrDA* als „intelligenter Kabelersatz" zu betrachten, welcher heute sehr verbreitet und günstig ist.

IAS (*Information Access Service*) [5] ist ein Teil des *IrDA*-Protokolles, welches, unter Anderem, ein Informationsmodell spezifiziert und die Operationen beschreibt die es von einem Geräten erlaubt, die Dienstklassen und Attributwerte eines anderen Gerätes herauszufinden. Diese Operationen sind als Teil von *IAS* festgehalten und dieses Protokoll heisst ***IAP*** (*Information Access Protocol*).

Dienstbeschreibung: Im Informationsmodell von *IrDA* ist eine Hierarchie von Dienst-Objekt-Klassen definiert. Diese Objekt-Klassen enthalten dienstspezifische Attribute in Form von Schlüssel-Wert-Paaren. Die Objekt-Klassen und deren Attribute werden durch Strings identifiziert.

Dienstlokalisierung: Nachdem sich zwei Geräte gegenseitig erfolgreich identifiziert haben, können sie mittels *IAP* herausfinden, welche Klassen das entfernte Gerät implemen-tiert und deren Attributwerte lesen.

Ein wichtiges Attribut innerhalb der Dienst-Klassen ist der ***LSAP*** (*Link Service Access Point*), welches als 'Selektor' betrachtet werden kann. Da ein *IrDA*-Gerät mehrere Klassen implementieren kann, kann man über diesen 'Selektor' die entsprechenden Dienste direkt aufrufen.

3.3 SLP (Service Location Protocol)

SLP [6] ist ein Vorschlag einer Arbeitsgruppe der *Internet Engineering Task Force* (*IETF*), um Service-Discovery in IP-Netzen durchzuführen. *SLP* definiert sowohl die Protokolle zwischen den einzelnen Komponenten als auch die einen Service definierenden Informationen. *SLP* kann wahlweise mit oder ohne 'zentrale Komponente', *Directory-Agent* (DA) genannt, konfiguriert werden. DA's offerieren ein 'zentrales Repository' für Serviceinformationen, die den Klienten bereitgestellt werden. Für den Betrieb in grossen administrativen Einheiten ist aus Gründen der Skalierbarkeit angedacht, neben 'Klienten' (*User-Agents*) und 'Server' (*Service-Agents*) auch DA's zu verwenden [7], [2].

Dienstbeschreibung: Ein Service ist definiert durch einen Typnamen in Form eines Strings, einer URL, über welche der Service lokalisierbar ist, und einer Menge von Schlüssel-Wert-Paaren, die Attribute des Service beschreiben.

```
Lifetime: 10800
URL: service:lpr://my.com:151
Attributes:
    (PAPER COLOR = WHITE),
    (LANGUAGE = POSTSCRIPT),
    (LOCATION = 12th FLOOR)
```

Abbildung 3: Beispiel einer Dienstbeschreibung bei Registrierung.

Dienstlokalisierung: Eine Service-Suche besteht im Wesentlichen aus einem Prädikat, das den Typ des gewünschten Dienstes und eine Menge von Randbedingungen, welche die Attribute des Dienstes erfüllen müssen, beschreibt.

```
lpr//(&
    (LANGUAGE == POSTSCRIPT)
    (LOCATION == 12th FLOOR)))
```

Abbildung 4: Dienst-Anfrage

Falls ein DA oder ein Service-Agent einen entsprechenden Dienst kennt, wird dieser dem User-Agenten in Form einer URL mitgeteilt:

```
service:lpr://my.com:151
```

Abbildung 5: Antwort

3.4 Jini (Java Intelligent Network Infrastructure)

Jini [2] ist eine von *Sun Microsystems* entwickelte Infrastruktur für die Dienstvermittlung in lokalen Netzen. *Jini* ist vollständig in *Java* implementiert und definiert ein komplettes API, welches es Dienstanbietern erlaubt, ihren Dienst in Form eines *Java-Proxy-Objektes* zu publizieren. Klienten versuchen, angebotene Dienste zu lokalisieren und über das Proxy-Objekt ein Nutzungsverhältnis mit den Anbietern einzugehen. *Jini* basiert auf *Java-RMI* und *Mobile Code* (Proxy-Objekte werden übermittelt). Alle Geräte, die Jini einsetzen wollen, brauchen darum zwingend eine *Java Virtual Machine. Jini* ist zudem noch Programmiersprachenabhängig (*Java*).

Es wird, nebst Klient und Dienst, auch ein sog. *Lookup-Service* (LUS) definiert. Dieser muss im System vorhanden sein und funktioniert als zentrale Registrierungs- und Vermittlungsinstanz.

Dienstbeschreibung: Im Sinne der *Closed-World Assumption* werden die Dienste in Jini über ihre Java-Interfaces definiert. Nebst dieser Dienst-Typ-Information, erlaubt es Jini, bei der Registrierung noch weitere Objekte beim entsprechenden LUS-Eintrag abzulegen: Ein Proxy-Objekt, um diesen Dienst entfernt anzusprechen und eine Menge von weiteren 'Attribut-Objekten', den sog. *Entries*. Diese erlauben es, erweiterte Suchmöglichkeiten anzuwenden. Man kann z.B. ein String-Objekt als *Entry* gebrauchen und dann nach Diensten suchen, die nebst einem bestimmten Dienst-Typen (Interface) auch einen bestimmten String als Beschreibung beinhalten. All diese Objekte werden vom Server in ein *Service-Item-Objekt* eingepackt und als Registrierungsanfrage zum LUS gesendet. Diese Dienstregistrierung (*Join*) ist zeitlich begrenzt. Ein Server muss sich vor Ablauf des *Lease*, (ein 'Zeitintervall', welches bei erfolgreicher Registrierung vom LUS zurückgeliefert wird) beim LUS wieder zurückmelden, andernfalls läuft der 'Vertrag' ab, und der LUS geht davon aus, dass der Dienst nicht mehr verfügbar ist und die Ressourcen davon können wieder freigestellt werden.

Dienstlokalisierung: Um nach einem Dienst zu suchen (*Lookup*), muss ein Klient zunächst einmal ein *Service-Template-Objekt* ausfüllen. Darin können eine Interface-ID, ein Interface und/oder eine Menge von *Entries* angegeben werden. *null*-Referenzen besitzen dabei *Wildcard*-Semantik. Der LUS liefert eine Menge von Service-Item-Objekten, die der Anfrage genügen. Der Klient wird ein Service-Item-Objekt auswählen und den darin enthaltenen Dienst-Proxy aktivieren. Falls weitere Klassen benötigt werden, werden diese dynamisch nachgeladen.

Erwähnenswert dabei ist, dass das Protokoll zwischen Dienst-Proxy und Dienst komplett der Implementierung des Proxies überlassen ist.

3.5 UPnP (Universal Plug and Play)

UPnP ist eine von *Hewlett-Packard* und *Microsoft* definierte Architektur nebst Protokollen, mit der eine Vermittlung zwischen Klienten und Dienstanbietern für kleine Geräte in einem IP-Netzwerk geleistet werden soll. Zielrichtung ist vor allem der Bereich von vernetzbaren Geräten im Umfeld eines 'home/office network'. Die Dienstvermittlung ist dabei mit oder ohne zentrale Komponente (*Proxy*) möglich. UPnP be-

findet sich, im Vergleich zu den anderen Architekturen, noch in einem sehr frühen Stadium, da die Spezifikationen teilweise noch recht jung sind und sich stellenweise noch ändern werden.

Dienstbeschreibung: Der Service-Typ besteht in UPnP einfach aus einem Bezeichner (String) für den Dienst.

Dienstlokalisierung: Server, die den vom Klienten gewünschten Dienst anbieten, senden eine entsprechende Antwort in Form eines UDP-Paketes an den Klienten zurück. Es gibt keinerlei Mechanismen die Suche durch bestimmten Eigenschaften (Attribute) eines Dienstes einzuschränken. Das UDP-Paket beinhaltet jedoch eine *Description-URL* die auf eine Beschreibungsseite des Dienstes verweist. Man kann dadurch eine XML-basierte Beschreibung des Dienstes beantragen, die einer bestimmten DTD konform ist.

3.6 Salutation

Salutation [8], [9] ist eine Netzwerkunabhängige Architektur zur Dienstvermittlung vom *Salutation Consortium* (Industrie und Hochschulen). Jede Netzwerk-Entität hat ihren eigenen Dienstvermittler, den *Salutation Manager*, wo das Gerät seine Dienste lokal registrieren kann (vgl. *Bluetooth, IrDA*). Die Netzwerkunabhängigkeit wird durch den *Transport Manager* erreicht, welcher in jeder Netzwerk-Entität vorhanden ist und die höheren Protokolle von Netzerkspezifischen Parameter und Protokollen 'abschirmt'. Diverse Produkte mit dieser Technologie existieren bereits.

Dienstbeschreibung: In jedem Gerät, welches Dienste anbietet, (*Functional Unit*) befindet sich eine Liste von *Functional Unit Records*, welche im Wesentlichen, nebst einer Service-Typ-Bezeichnung, eine Menge von Attribut-Tripeln beinhaltet. Ein solcher Tripel bestet aus: einer ID, welche den Typ des Attributes festlegt; einer ID, welche Vergleichsfunktionen für diesen Attribut festlegt; und schliesslich den Wert selbst.

Dienstlokalisierung: Die Suche nach gewissen Service-Typen wird dem lokalen *Salutation Manager* übergeben. Falls die vom Klienten verlangten Service-Typen lokal nicht vorhanden sind, können *Salutation Manager* über RPC kooperieren um den benötigten Dienst ausfindig zu machen.

4. Vergleich

Es sollen nun an dieser Stelle die wichtigsten Unterschiede tabellenartig dargestellt werden:

Kriterien	Bluetooth	IrDA	SLP	Jini	UPnP	Salutation
Service-Registrierung	Nur lokal	Nur lokal	Authent. Multicast	Multicast m. TCP-Callback/RMI	Multicast/ Unicast	Nur lokal
Service-Typ	UUID	String	String	Java-Interface	String	String
Service-Beschreibung	Attribute/ Werte	Attribute/ Werte	Attribute/ Werte	Serialisierte Java-Objekte	String u. XML-Instanz	Attribute/ Werte
Klienten-Anfrage	UUID	String	Boolsche Ausdrücke	Java-Objekt-Template	String	String
Dezentraler Betrieb	Ausschliesslich	Ausschliesslich	Möglich	(nein)	Möglich	Ausschliesslich
Service Discovery	ja	ja	ja	ja	ja	ja
Service Announcement	nein	nein	ja	ja	ja	ja
Interoperability	ja	ja	ja	ja	nein	ja
Security	ja	ja	ja	ja	nein	nein

Tabelle 1: Die wichtigsten Eigenschaften der betrachteten Architekturen.

Wie in **Tabelle 1** ersichtlich ist, bestehen die wichtigsten Unterschiede in der Dienstbeschreibung selbst: Es werden ganz unterschiedliche Konzepte eingesetzt. Von Objekten, Strings, Schlüssel-Wert-Paaren bis hin zu XML-Dokumenten. Einige Systeme arbeiten mit einer zentralen Dienstinformationsverwaltung, andere mit einer dezentralen. Einige bieten diese zentrale Komponente optional an, andere brauchen sie zwingend. In Bezug auf Sicherheits-Aspekte gibt es Systeme, die etwas anbieten, andere beinhalten praktisch keinerlei Sicherheits-Features. Auch in Bezug auf die Anwendungsgebiete gibt es grosse Überlappungen, und es gibt keinen klaren Sieger der sich für alle Aufgaben besser eignet. Man vergleiche z.B. Infrarot- vs. Funk-Technologie. Was ist nun bezüglich Sicherheit oder Komfort 'besser'? Eine prägnante Antwort könnte lauten: „Es kommt im Wesentlichen auf Anwendungs-Szenarien an".

Man merkt sofort, dass es schwierig ist abzuschätzen, welche der hier diskutierten Systeme in Zukunft eine verstärkte Rolle spielen werden. Es wird im Moment ziemlich viel Aufwand in die *Interoperabilität* zwischen den Systemen investiert: Man hat zum Beispiel eine *Salutation-Bluetooth-Bridge* entwickelt, um es *Salutation* zu erlauben, auf einige Vorteile von Bluetooth zuzugreifen. Oder es existiert eine *Jini-SLP-Bridge,* um Geräte, die nicht über eine *Java Virtual Machine* verfügen, 'mächtiger' zu machen, usw. Ob sich diese Strategie in Zukunft bewähren wird, bleibe dahingestellt.

Sicher ist im Moment nur, dass von vielen Seiten solche Systeme, die hier betrachtet wurden, zur Vernetzung von Geräten und Diensten gewünscht werden.

Referenzen

[1] S. Domnitcheva, K. Römer, M. Rohs, *Service Discovery and Description*, Folien, *Doktorandenseminar Ubiquitäre Information*, ETH Zürich, 2001.

[2] R. Kehr, *Spontane Vernetzung, Infrastrukturkonzepte für die Post-PC-Ära*, *Informatik Spektrum*, Juni 2000

[3] *Bluetooth Consortium: Specification of the Bluetooth System Version 1.0 A-Core*, http://www.bluetooth.com, Juni 2001.

[4] Williams Stuart, *IrDA-Past, Present and Future*, *IEEE Personal Communications*, http://www.irda.org, Juni 2000.

[5] A. Seaborne, S. Williams, F. Novak, *Infrared Data Association Link Management Protocol*, January 1996.

[6] E. Guttmann, *Service Location Protocol: Automatic Discovery of IP Network Services*, *IEEE Internet Computing*, July-August 1999.

[7] Golden G. Richard III, *Service Advertisement and Discovery: Enabling Universal Device Cooperation*, *IEEE Internet Computing*, University of New Orleans, September-October 2000.

[8] *The Salutation Consortium, Salutation Architecture Specification (Part-1) Version 2.0c*, June 1999, http://www.salutation.org.

[9] D. Chakraborty, H. Chen, *Service Discovery in the Future for Mobile Commerce*, January 2001, http://www.acm.org/crossroads/xrds7-2/service.html.