[KiSchSt02] *Holger Kirchner, Wolfgang Schönfeld, Ralf Steinmetz;* **Verfahren zur Diensterkennung und Dienstinformationsbereitstellung im Vergleich;** ITG Fachtagung "Neue Kommunikationsanwendungen in modernen Netzen", Duisburg, März 2002, S. 19-27.

Verfahren zur Diensterkennung und Dienstinformationsbereitstellung im Vergleich

Holger Kirchner[1], Prof. Dr. Wolfgang Schönfeld[1] und Prof. Dr. Ralf Steinmetz[2],
[1]Fraunhofer Institut für Integrierte Publikations- und Informationssysteme (IPSI), Darmstadt, Deutschland
[2]Technische Universität Darmstadt, Darmstadt, Deutschland

Kurzfassung

Protokolle wie Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), CORBA Trader Service (TS), Salutation, Service Location Protocol (SLP), Jini, Universal Plug and Play (UPnP), Secure Service Discovery Service (SSDS) und Dynamic Host Configuration Protocol (DHCP) sind Verfahren zur Diensterkennung und Dienstinformationsbereitstellung. Sie bieten Benutzern die Möglichkeit, Information über Dienste zu erfragen und zu nutzen. Dieser Beitrag zeigt die wesentlichen Unterschiede der o. g. Verfahren auf und vergleicht sie ausführlich.

1 Einführung

Eine Vielzahl von Diensten, wie beispielsweise Faxen, Drucken, Scannen, oder das Versenden von E-mails, wird in Rechnernetzen zur Verfügung gestellt. Um diese Dienste nutzen zu können, müssen die Benutzer die Möglichkeit haben, Informationen über sie in Erfahrung zu bringen. Dabei steht ihnen jedoch kein „generelles Informationssystem" zur Verfügung, das ihnen erlaubt, für unterschiedliche Betriebssysteme und Anwendungen Dienstinformationen abzufragen. Mobile Endgeräte, die in unterschiedlichen Netzen agieren, unterstreichen noch den Bedarf an einem Mechanismus, der dieses Kommunikationsproblem löst und es dem Benutzer ermöglicht, im eigenen oder in einem fremden Netz Dienste gezielt zu finden und zu nutzen.

Von den Grundbegriffen ausgehend wird in Abschnitt 3 die Dienstinformationsvermittlung erläutert. Im darauffolgenden Abschnitt 4 werden einzelne Verfahren anhand ausgewählter Eigenschaften untersucht und in Tabelle 1 gegenüber gestellt. Im letzten Abschnitt 5 wird eine Zusammenfassung und ein Ausblick gegeben.

2 Einheitliche Begriffe

Da die Terminologie im Bereich der Diensterkennung und Dienstinformationsbereitstellung noch nicht standardisiert ist, definieren bzw. verwenden die bisherigen Untersuchungen die relevanten Begriffe unterschiedlich. McGrath [1] versteht unter einem Dienst: „a service includes any sort of network service that might be available". Andere definieren einen Dienst anhand seiner Eigenschaften, z. B. [2]: "A service is typically implemented by a set of distributed hardware and software components which cooperate to provide the required service to users", und [3]: "Services are deployed in various forms and with different levels of complexities." Es zeigt sich, daß der Begriff Dienst in unterschiedlichen Komplexitäten und für unterschiedliche Ausprägungen definiert werden kann. Jedoch wird er oftmals nicht ausreichend eingegrenzt. Die für die Diensterkennung und Dienstinformationsbereitstellung relevanten Begriffe werden in den nachfolgenden Abschnitten folgendermaßen verwendet:

- Ein ***Dienst***, der über das Netzwerk zugänglich ist, erbringt eine Dienstleistung. Wird z. B. ein Druckservice oder ein Faxservice (oft handelt es sich dabei um elektrische Geräte, die über einen Netzzugang verfügen) in Anspruch genommen, ist die geographische Lage (der Ort, an dem das Gerät steht) von entscheidender Bedeutung. Neben solchen „realen" Diensten gibt es auch virtuelle Dienste. Ihre Dienstleistung wird häufig von verschiedenen Servern angeboten. Dabei ist unter einem Server ein Programm zu verstehen, das auf einem Rechner gestartet wird und den Dienst über eine Rechneradresse und einen Port anbietet. So kann z. B. der Mailservice über einen Mailserver mit einem verbindungsorientierten (Kommunikations-) Dienst der Vermittlungsschicht (TCP) und Port 25 erreicht werden.
- Unter ***Dienstinformation*** ist die Beschreibung eines Dienstes zu verstehen. Dienstinformationen werden von einem Dienst beim Dienstinformationsvermittler registriert. Die Informa-

tionen über einen Dienst werden in einer Datenhaltung (Repository) gehalten.

- ξ Ein ***Client*** stellt den Dienstnutzer dar, der Dienstinformationen für die Benutzung des Dienstes verwendet.
- ξ Ein ***Server*** bietet Dienste an (Dienstanbieter).
- ξ Ein ***Dienstinformationsvermittler*** (service information trader) übernimmt die Datenhaltung sämtlicher Dienstinformationen. Als unabhängige Instanz vermittelt er die Dienstinformation zwischen Server und Clienten.

Zur Einschränkung des Such-Resultats besteht für den Clienten die Möglichkeit, sog. *Filter* einzusetzen. Um den Netzverkehr zu reduzieren und eine Skalierung zu ermöglichen, wird der Dienstinformationsvermittler in einem Netzwerk mehrfach in der gleichen Konfiguration eingesetzt.

3 Allgemeine Architektur

Das nachfolgende Schaubild beschreibt die grundsätzliche Architektur der Dienstinformationsvermittlung:

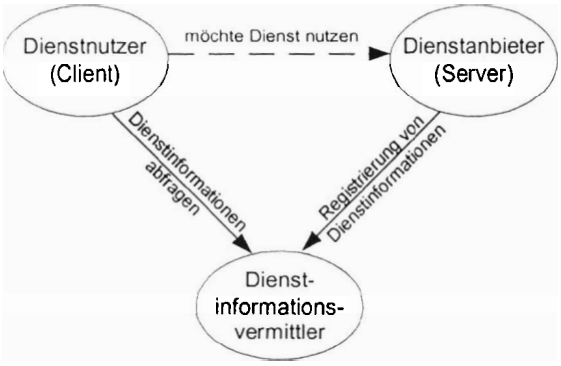

Bild 1 Grundmodell Dienstinformationsvermittlung

Die wichtigsten Funktionen im Grundmodell der Dienstinformationsvermittlung sind:

- ξ Auffinden der Dienstinformationsvermittlungsinstanz
- ξ Registrierung sowie Deregistrierung von Diensten
- ξ Unterstützung des Benutzers beim Browsen nach Dienstinformationen
- ξ Einsatz von Filtern in Suchanfragen, um das Abfrageergebnis einzuschränken

Nach [1] werden Protokolle zur Dienstinformationsvermittlung in ***Lookup*** und ***Service Discovery*** eingeteilt. Bei Lookup benötigt der Suchende mindestens eine Instanz (einen Verzeichnisagenten oder einen anderen Agenten), die die Anfrage beantworten kann. Lookup-Mechanismen werden statisch konfiguriert. Im Gegensatz dazu werden bei Service Discovery die Prozesse zur Dienstinformationsvermittlung spontan gestartet. Diese können mehrere Instanzen haben (nicht nur den Verzeichnisagenten), um in einer spontanen (adhoc) Vernetzung Dienstinformationen zu vermitteln.

Weiterhin unterscheidet man ***lokalen*** (in LANs) und ***globalen*** (in WANs) Verfahren. Bei globalen Netzen muß eine hierarchische Struktur aufgebaut werden, um die Skalierung zu gewährleisten: Mehrere Dienstinformationsvermittler verteilen dann die Dienstinformationen nach dem Prinzip „Teile und Herrsche" [1].

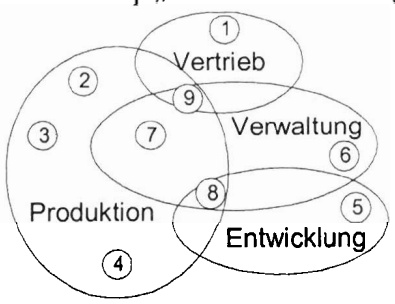

Bild 2 Verteilung von Dienstinformationen

In Bild 2 wird anhand einer einfachen organisatorischen Struktur wie „Vertrieb", „Produktion", „Entwicklung" und „Verwaltung" die logische Plazierung von Dienstinformationsvermittlern (1 bis 9) gezeigt. Dabei organisieren sie die Dienstinformationen nach Sichtbarkeitsbereichen (der Dienstinformationsvermittler 8 ist z. B. für die Bereiche „Entwicklung", „Verwaltung" und „Produktion" sichtbar). Die Informationen werden dadurch nach organisatorischen Gesichtspunkten unter den dafür zuständigen Dienstinformationsvermittlern verteilt. Die Vermittlungsinstanzen können mehrfach besetzt sein (Redundanz), um eine ***Robustheit*** des Systems zu erreichen und zu garantieren (in „Produktion": 2, 3 und 4).

Durch ***zentrale*** Datenhaltung werden Inkonsistenzen in Dienstinformationen vermieden. Jedoch wird der Netzverkehr konzentriert, und es kann zu Störfällen oder Fehlern des Systems kommen. Zusätzlich verursacht die zentrale Datenhaltung eine hohe Last für das Verzeichnis [4].

In einer ***verteilten*** Architektur verwenden mehrere Instanzen ganz individuell Push- oder Pull-Mechanismen oder eine Kombination aus beiden für die Verteilung und die Bekanntmachung der Dienstinformationen. Wird ein Push-Verfahren verwendet, so macht jeder Dienst seine Dienstinformation selbst bekannt. Bei einem Pull-Verfahren fordert jeder, der einen Dienst benötigt, selbst Dienstinformationen an.

Dienstinformationen werden in unterschiedlichen ***Formaten*** beschrieben. Im allgemeinen unterscheiden

sie sich dadurch, daß dass entweder Objekte oder Schnittstellen, die verschiedene Dienste beschreiben, oder einfache Zeichenketten (Uniform Resource Locator - URL [5, 6]) verwendet werden. Dieser Unterschied hat Auswirkungen auf das ***Laufzeitverhalten*** oder die Reaktionszeit bei Abfragen. Weiterhin sind Dienstinformationen nur für eine vordefinierte Zeit gültig (Time to Live - TTL). Das bedeutet, daß beim Registrieren von Diensten eine ***Lebensdauer*** *(*Lease*)* vereinbart wird, die nach einer vorbestimmten Zeit verlängert werden muß. Der Vorteil eines solchen Konzeptes besteht darin, daß der ***Administrationsaufwand*** [7] reduziert wird. Unter Umständen ist es sogar möglich, daß kein administrativer Aufwand (***zero configuration***) entsteht und die Systeme selbst in der Lage sind, sich zu konfigurieren (***self-configuration***).

Dienstinformationen werden in Systemen unterschiedlich beschrieben: entweder ***fest*** oder ***flexibel***. Letztere sind vor allem im mobilen Umfeld wichtig [8]. Oftmals wird eine einfache Grammatik dafür verwendet. Desweiteren wird die Flexibilität dadurch verbessert, da zur Laufzeit Dienste bzw. die Dienstvermittlungsinstanzen spontan gefunden werden können. Dies geschieht mit Multicast. Der Umfang von Anfrageergebnissen kann durch verschiedene ***Filterverfahren*** reduziert werden. So kann nicht nur gezielt nach dem gewünschten Diensttyp, sondern auch nach Attributen oder Tags gefragt werden. Andererseits kann der Client aus der erhaltenen Resultatliste die für ihn wichtigen Resultate selektieren.

Die Verfahren zur Dienstfindung und Dienstinformationsbereitstellung unterscheiden sich auch in der unterschiedlichen Unterstützung von ***Sicherheitsaspekten***. Viele von ihnen haben unzureichende Möglichkeiten, Dienstinformationen zu schützen. Daher sind sie in Bezug auf Sicherheit auf andere Konzepte angewiesen. In [9] werden Sicherheitsaspekte analysiert.

4 Verfahren zur Diensterkennung und Dienstinformationsbereitstellung

Nachfolgend werden die verschiedenen bekannten Verfahren zur Diensterkennung und Dienstinformationsbereitstellung erläutert. Die zuvor erwähnten Merkmale werden dabei berücksichtigt.

4.1 Domain Name System (DNS)

DNS ist ein standardisiertes Protokoll [10, 11], das von der Internet Engineering Task Force (IETF) entwickelt wurde. Es ist ein weltweites Namenskonzept, das durch sog. Resolver (DNS-Server) Rechnernamen in IP-Adressen auflöst [12]. DNS ist als ein hierarchisches System von Servern aufgebaut und skaliert zu einem globalen System. Es verwendet eine statische Datenbank bzw. Dateien für die Speicherung von Dienstinformationen. Jedoch garantiert DNS nicht die Verfügbarkeit der Dienstinformationen. Diensttypen werden durch Resource Records (z. B. MX, SOA, PTR) [13] beschrieben und in Namensservern gehalten. Anhand eines SRV-Records wird z.B. ein *telnet* Dienst folgendermaßen beschrieben:

```
Dienst      SRV Pri. Gew. Port Server.Domäne
telnet.tcp SRV 0    1    23   tapsi.gmd.de
```

Für die Suche nach Dienstinformationen ist eine begrenzte Semantik vorgegeben. Die Attribute von Diensttypen können im DNS nicht erweitert werden. Daher eignet es sich weniger für die spontane Erkennung von Diensten [14].

Bei DNS handelt es sich um einen vertrauenswürdigen Dienst, dessen Sicherheit durch die Beschränkung des Zugriffs auf privilegierte Benutzer gewährleistet wird. Nur privilegierte Administratoren können Informationen verändern. Um Datensätze der Namensserver-Datenbank satzweise zu ändern oder einzufügen, wurde das Dynamische DNS [15] eingeführt. Bislang mußte immer die komplette Datenbank nach einer Änderung neu geladen werden, damit die neuen Datensätze wirksam wurden. Mit dem dynamischen Update-Mechanismus ist es möglich, einen einzelnen Eintrag anzulegen bzw. einen existierenden Datensatz zu überschreiben. So vereinfacht Dynamisches DNS den Administrationsaufwand.

4.2 Lightweight Directory Access Protocol (LDAP)

LDAP ist ebenfalls ein IETF Standard [16], der eine skalierbare Hierarchie von Namensräumen ermöglicht. In ihm werden Dienste bekannt gemacht oder von Clienten lokalisiert. Im Unterschied zu DNS ist es erweiterbar und unterstützt viele unterschiedliche Typen von Dienstinformationen. LDAP ist hierarchisch organisiert und auch als globaler Verzeichnisdienst (Globus Metacomputing Directory Service [17]) einsetzbar, wobei Abfragen auf bestimmte Teile der Hierarchie begrenzt werden können. Anfragen über große Domänen bleiben jedoch ineffizient.

Die Abfragesprache von LDAP ähnelt denen von

anderen Discovery Protokollen [18], wie z. B. SLP. Trotz seines Namens ist LDAP im Vergleich zu vielen anderen Verfahren wie SLP ein sehr „schweres" Protokoll: Die Dienstinformationen sind im allgemeinen sehr umfangreich und für mobile Endgeräte deshalb weniger geeignet. Selbst die Komplexität des Protokolls macht es mobilen Endgeräten nicht einfach, Dienstinformationen in einer großen Anzahl genügend schnell abzufragen [1]. LDAP selbst ist nicht als Protokoll für spontane Erkennung von Diensten definiert: Es macht keine Dienstinformationen über Multicast bekannt und überprüft nicht die Existenz von registrierten Diensten. Weiterhin benötigt LDAP andere Netzwerkdienste, um Sicherheit garantieren zu können [9].

4.3 Corba Trader Service

Der CORBA Trader Service (TS) ist ein Standard CORBA Service [19] der Object Management Group (OMG). Er ermöglicht über eine Schnittstelle die Bekanntmachung von Diensten und Serviceanfragen mit Attributen. Jeder Dienst, der eine CORBA Implementierung z. B. über einen Proxy oder einen Wrapper verwendet, kann den Trader Service verwenden. Hierbei müssen sie sich der CORBA-Objekte bedienen, um Objekte zu lokalisieren, die in Beziehung zu anderen Objekten stehen. Bei allen Bekanntmachungen, Anfragen und Antworten handelt es sich um CORBA-Objekte. Anfragen können durch Beziehungsausdrücke (constraints) eingeschränkt werden. Sie können durch boolesche oder arithmetische Ausdrücke mit Werten von CORBA-Variablen erstellt werden. Der TS garantiert nicht, daß registrierte Objekte auch verfügbar sind und liefert keine Mitteilung bei Statusänderungen. Dagegen unterstützt er den Zusammenschluß von mehreren Trader-Diensten zu einer einfachen logischen Struktur, deren Zusammensetzung und Verteilung beliebig ist.

4.4 Salutation

Das Salutation-Protokoll [20] ist eine offene Spezifikation des Salutation Consortiums, das spontane Konfiguration von Netzwerkgeräten und Diensten unterstützt. Salutation wird zur Zeit von vielen Unternehmen entwickelt, die Drucker oder ähnliche Geräte herstellen (HP, IBM, Xerox und andere) [21]. Die Salutation-Architektur definiert ein abstraktes Modell mit drei Komponenten: Client, Server und Salutation-manager (SLM). Der SLM organisiert sämtliche Verbindungen und Übergänge (bridges) zwischen den verschiedenen notwendigen Kommunikationsmedien. Salutation basiert auf Remote Procedure Calls von Sun (SunRPC). Das Modell kann mit oder ohne SLM implementiert werden. Die einzelnen Verzeichnisse können hierarchisch oder willkürlich vernetzt werden. Wenn Salutation ohne den SLM implementiert wird, können sich Clients und Services direkt mit lokalen Broadcastanfragen bekanntmachen. Mit dieser Konfiguration kann Salutation ohne Administrationsaufwand arbeiten. Weiterhin definiert es ein spezielles (erweiterbares) Format für die Beschreibung von Dienstinformationen. Dieses Format enthält Diensttypen. Dienste machen sich durch eine Registrierung bei einem oder mehreren SLMs bekannt. Clienten können sie lokalisieren, indem sie die Standardfunktionalitäten einsetzen (siehe Abschnitt 3). Zusätzlich können für die Abfrage von Dienstinformationen Funktionen beim SLM hinterlegt werden. Die Verwendung von SunRPC gibt die Grenzen von Salutation vor [1]. Die einzige multicastähnliche Funktionalität ist Broadcast-RPC. Sie wird verwendet, um lokale Kopien des Salutationmanagers zu entdecken.

4.5 Service Location Protocol (SLP)

SLP ist ein IETF Standard [22, 23, 24], der für die spontane Erkennung von Diensten und die Bereitstellung von Dienstinformationen gedacht ist. SLP definiert eine abstraktes Modell mit User Agents (UA), Service Agents (SA) und Directory Agents (DA). Der SA liefert die Dienstinformationen, und der DA sammelt sie. Mehrere DAs können für die Verteilung von Dienstinformationen verwendet werden oder eine Hierarchie bilden. SLP skaliert in lokalen Netzen durch die Verwendung von mehreren DAs. Es kann in verschiedenen Konfigurationen eingesetzt werden. In der ***passiven*** Konfiguration sendet der DA Bekanntmachungen (Advertisements) periodisch über Multicast. In der ***aktiven*** Konfiguration können UA oder SA ebenfalls DAs entdecken, indem sie DHCP verwenden [25], statische Konfiguration einsetzen (Informationen aus einer Konfigurationsdatei) oder über Multicast eine Anfrage stellen. SLP kann auch ohne DA implementiert werden. In diesem Fall entsteht kein administrativer Aufwand. Ohne DA arbeitet SLP beim Austausch von Dienstinformationen mit Multicast. Ist jedoch ein DA oder sind sogar mehrere DAs im Netz verfügbar, arbeitet das Protokoll effizienter [26], da UA und SA gezielt Nachrichten über Unicast an den DA schicken. Das Format für Dienstinformationen heißt in SLP *Service URL-Schema* und beinhaltet Adresse, Typ und Attribute eines Dienstes. Nachfolgend ein Beispiel für

die Beschreibung eines Druckers, der DIN A4 farbig druckt:

```
URL:service:lpr://PAPER_FORMAT=A4,
                  PAPER_COLOR=COLOR
```

Registrierungen von Diensten werden mit einer Lebenszeit (TTL) versehen, die die Aktualität sicherstellen soll [26].
UAs senden Serviceanfragen, die Suchfilter enthalten können, an den DA. Diese gleichen syntaktisch LDAPv3 [16] und filtern häufig nach Diensttypen und Attributen. Attribute werden durch Templates [27] nach einem LDAPv3 Prädikat beschrieben.
SLP erkennt nicht die Größe der Netze, in denen es sich befindet. Es kann daher in größeren Netzen keine Multicastanfragen unterbinden und die TTL angemessen anpassen.
SLP ist ein String-basiertes Protokoll, das binäre Nachrichten-Header verwendet. Für den Nachrichtenaustausch wird meist UDP verwendet; TCP wird zum Versenden großer Nachrichten eingesetzt. Dabei verwendet SLP Multicast, um Dienstinformationen an sämtliche DAs zu verteilen und um DAs bzw. SAs ausfindig zu machen. Weiterhin ermöglicht SLP durch standardisierte Funktionen eine Browserunterstützung, indem bei einem DA zuerst die Diensttypen abgefragt werden, danach mit Hilfe der Diensttypen die Dienste und zuletzt die Dienstinformationen der Dienste selbst.

4.6 Jini

Suns Jini basiert auf der Java-Umgebung (SUN, Javasoft) und unterstützt die spontane Erkennung von Diensten [8, 19, 28]. In vielerlei Hinsicht ähnelt Jini SLP und wurde auch davon beeinflußt [1]. Das Protokoll verwendet serialisierte Javaobjekte, um Dienstinformationen auszutauschen (mit Java Remote Method Invocation (RMI) [29]). Jini erfordert eine Instanz des Lookup-Services. Wie SLP macht der Lookup Server (LS) über Multicast seine Verfügbarkeit bekannt. Dies kann auf unterschiedliche Weise geschehen: Entweder startet die Instanz erfolgreich einen oder mehrere LS, und die Clients und Server erlangen dadurch mit einem RMI Stub Zugriff zu dem Jini Lookup Service, oder die Server machen sich bekannt, indem sie den Java RMI Stub beim Jini Lookup Server registrieren. Clients lokalisieren Dienste durch Anfrage nach einem besonderen Diensttyp.
Die Anfrage hat grundsätzlich ein einfaches Muster von String-Attributen. Jedoch muß sie als Java-Objekt implementiert werden, das durch ein Jini Interface spezifiziert ist. Wenn ein Service lokalisiert wird, liefert Jini ein Java RMI Stub, um darauf zugreifen zu können. Dies ermöglicht Clienten, Code während der Laufzeit zu laden, und Servern, Schnittstellen zur Verfügung zu stellen, die die Clients vorher nicht kannten. Das Jini Protokoll verwendet TTLs für Bekanntmachungen und Registrierungen. Client und Server müssen ständig ihre TTL verlängern. Entitäten, die abgestürzt sind, werden automatisch aus dem Lookup Service entfernt, wenn die TTL abgelaufen ist.
Jini bietet zusätzlich einen Mitteilungsdienst für Ereignisse (Notification), wenn Entitäten hinzugefügt oder entfernt werden. Jini kann nicht mit anderen Protokollen als RMI oder in anderen Sprachumgebungen arbeiten. Das Protokoll ist abhängig von dem Einsatz von Java Stubs, d. h. ein Gerät muß entweder eine Java Virtual Machine (JVM) implementiert haben oder einen Proxy verwenden. Theoretisch kann Jini für große Systeme verwendet werden [1]. Es basiert auf Java und garantiert dieselbe Sicherheit wie Java. Der Jini Lookup Server bietet zwar Sicherheitsmechanismen, ist aber dennoch vom Sandkastenmodell abhängig (das in Java zur Verfügung gestellt wird). Im Vergleich zu SLP ist Jini kein genereller Mechanismus für heterogene Systeme, da es nur auf Java-Plattformen funktioniert. Für mobile Endgeräte scheint der Einsatz von Jini unattraktiv zu sein, da die zur Verfügung stehenden Ressourcen sehr begrenzt sind. Außerdem benötigt eine JVM Speicher und Prozessressourcen, die bekanntlich sehr kostbar sein können [26].

4.7 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) ist ein Microsoft Standard für spontane Konfiguration [21]. UPnP behandelt Netzwerk-Adreßauflösung und ist mit dem Simple Service Discovery Protocol (SSDP) der IETF verknüpft [30]. Es hat eine ähnliche Architektur wie SLP. UPnP verwendet XML für die Beschreibung von Diensten und Abfragen. Es bietet Diensterkennung, bei der Netzwerkadressen entdeckt werden. Bekanntmachungen werden durch lokale Broadcasts angekündigt. Bei erfolgreicher Erkennung wird eine Adresse oder eine URL und der Diensttyp übergeben.
Dienste werden durch erweiterte URLs beschrieben, ähnlich wie bei SLP. Die URL weist auf eine XML-Datei mit einer ausführlichen Beschreibung eines Dienstes. Ein UPnP Gerät gibt einen oder mehrere Dienste bekannt. Die XML Beschreibung kann von Programmen oder Browsern verwendet werden. Mit ihr können bestimmte Dienstinformationen durch Verwendung von Filtern oder XML-Tags oder einer

Kombination aus beiden gefunden werden. XML ist flexibel, da es jede Art von Information liefern kann. Diese Information ist ausführlicher als die, die SLP, LDAP, Salutation oder Jini bietet. UPnP erfordert jedoch HTTP und XML, die einen Webserver voraussetzen.

4.8 Secure Service Discovery Service (SSDS)

SSDS ist eine Entwicklung der University of California Berkeley (Ninja Projekt) [31]. Das Protokoll ähnelt anderen Discovery Protokollen mit einigen speziellen Verbesserungen in der Stabilität, Skalierung und Sicherheit. Die Secure Discovery Service (SDS) Server sind als Hierarchie organisiert. Obwohl SSDS in Java implementiert ist, nutzt es XML anstelle von Java-Objekten für die Dienstbeschreibung. Das SSDS Modell umfaßt Clients, Dienste und SDS Server. Die Information über die Verfügbarkeit der Dienste wird dadurch erreicht, daß die SDS Server periodisch Multicast-Nachrichten senden. Die Nachrichten enthalten URLs über verfügbare SDS Server. Die SDS Server (und Clients) können Dienstinformationen zwischenspeichern. Dabei kann die Verteilung der Dienstinformationen überall durch Multicast erreicht werden. Dieses Protokoll bietet Skalierung, Fehlererkennung und Selbstkonfiguration, die das System sehr robust machen. SSDS verwendet Java RMI Remote-Methoden. Das Protokoll ist für den Austausch von XML-Dokumenten konzipiert.Nach [31] können alle Formate für Dienstinformationen, einschließlich Jini-Objekte, nach XML übersetzt werden.
Ein wichtiges Unterscheidungsmerkmal von SSDS gegenüber anderen Verfahren ist die Sicherheit. Alle Instanzen sind authentifiziert, und sämtliche Nachrichten werden verschlüsselt. Eine Neuimplementierung des RMI Protokolls stellt Mechanismen zur Authentifizierung und verschlüsselten Kommunikation zur Verfügung [31]. Weitere Server können hinzugefügt werden, z. B. um eine Hierarchie aufzubauen. Werden fehlerhafte Server erkannt, so wird versucht, den Fehler durch Neustart zu beheben.

4.9 Dynamic Host Configuration Protocol (DHCP)

DHCP bietet eine weitere Möglichkeit, Dienstinformationen zu vermitteln. Über optionale Parameter bei der Vergabe einer IP-Lease werden diese Informationen, neben einer temporären IP-Adresse, als Konfigurationsparameter übertragen. Die Informationen sind standardisiert und können beliebig als Konfigurationsparameter eingesetzt werden [32, 33]. Über die angegebenen Parameter haben Rechner somit die Möglichkeit, Konfigurationsinformationen zu verwenden und Informationen dynamisch zu erhalten, z. B. Informationen über Druckserver, die in einem Netzwerk zur Verfügung stehen. Druckserver (LPR) können somit z. B. über ID 9 bekanntgemacht werden.

5 Zusammenfassung und Ausblick

Die verschiedenen untersuchten Dienstzugangs- und Diensterkennungsverfahren machen auf unterschiedliche Art und Weise Dienstinformationen zugänglich. Dabei sind die Unterschiede in Bezug auf den Einsatzbereich, die Dienstinformationsvermittler, die notwendige Mindestanzahl von Servern, den Administrationsaufwand, die Organisation der Server, die Darstellung und das Format der Dienstinformation, die Aktualität der Dienstinformation, ihre Granularität und die Sicherheit deutlich geworden. Möglichkeiten bzgl. Leistungsfähigkeit und Dynamik der Dienstbereitstellungsverfahren sind in Tabelle 1 dargestellt.

	DNS	LDAP	TS	Salutation	SLP	Jini	UPnP	SSDS	DHCP
Quellen	RFC 1035, 1035, 1101, 1183, 2136	RFC 2251	www.omg.org	www.salutation.org	www.srvloc.org	www.sun.com/jini	www.upnp.org	iceberg.cs.berkeley.edu/	RFC 2131, 2132
Lookup / discovery	lookup	discovery	lookup	discovery	discovery	lookup	discovery	lookup	lookup
Scope (Bereich)	global	global	lokal	lokal	lokal	lokal	lokal	global	lokal
Dienstvermittler (service broker)	Namensserver	LDAP Server	Lookup Server	Salutationmanager	Directory Agent	Lookup Server	Simple Service Discovery Server	Secure Service Discovery Server	DHCP Server
Mindestanzahl an Servern (service broker)	1	1	1	0	0	1	1	1	1
Administrations-aufwand	hoch	hoch	hängt vom Szenario ab	niedrig	niedrig	hängt vom Szenario ab	mittel	hängt vom Szenario ab	hoch
zero configuration	nein	nein	hängt von der Implementierung ab	ja	ja	ja	ja	ja	nein
selbstorganisierend	nein	nein	ja	ja	ja	ja	ja	nein	nein
Darstellung / Format der Dienstinformation	resource record	String	Objekt	Objekt	String (URL)	Objekt	XML	XML	String
Aktualität der Dienstinformation	start of authority record	nein	hängt von der Implementierung ab	ja	lease	lease	lease	lease	lease
Dienstinformation erweiterbar	nein	ja	ja	ja	ja	ja	ja	ja	nein
Granularität der Dienstinformation	grob	fein	hängt von der Implementierung ab	mittel	mittel	hängt von der Implementierung ab	fein	fein	grob
Sicherheit	privilegierte Administratoren	auf andere Systeme angewiesen	auf andere Systeme angewiesen	auf andere Systeme angewiesen	auf andere Systeme angewiesen	auf andere Systeme angewiesen	auf andere Systeme angewiesen	ausgefeiltes Sicherheitskonzept	auf andere Systeme angewiesen
Sonstiges	mehrere Namensserver bilden eine Hierarchie	mehrere LDAP Server bilden eine Hierarchie	Ereignisse werden bei Statusänderungen erzeugt	basiert auf SunRPC; verwendet Broadcast zur Bekanntmachung von Dienst-informationen	verwendet Unicast und Multicast zum Auffinden der Instanzen und zur Verteilung von Dienstinformationen	tauscht serialisierte Javaobjekte aus; Events werden bei Statusänderungen erzeugt	basiert auf Simple Service Discovery Protocol (SSDP); benötigt HTTP	verwendet eigene RMI Implementierung	

Tabelle 1: Vergleich

Die aufgeführten verschiedenen Verfahren haben unterschiedliche Eigenschaften und sind somit für bestimmte Anwendungen geeignet. Bestimmte Protokolle setzen einige der fehlenden Funktionalitäten anderer Systeme um und ermöglichen z. B. die spontane Vernetzung mit geringem Aufwand an Netzwerkkonfiguration – Aspekte, die v. a. beim Einsatz in einer mobilen Umgebung wichtig sind. Die wichtigsten Funktionalitäten bzw. Eigenschaften der Verfahren sind:

- ξ Spontane Erkennung und Konfiguration von Netzwerkinstanzen und -diensten
- ξ Auswahlmöglichkeit und Selektion von Diensttypen und Diensten
- ξ Geringe (oder keine) manuelle Administration
- ξ Automatische Anpassung zur mobilen und sporadischen Verfügbarkeit

  Interoperabilität für Hardwarehersteller und Plattformen

Auch Sicherheit, wie z. B. Anonymität, ist in unterschiedlichen Netzen (Heimat- und Fremdnetzen) wichtig. Sie sollte aber individuell einsetzbar sein.

Zur Zeit sind zu viele Protokolle im Einsatz für die Verwaltung und Organisation von Dienstinformationen. Viele der Protokolle sind logisch kompatibel und können teilweise verknüpft werden.

Diensterkennung sollte universal sein, d. h. neue Attribute sollten immer hinzugefügt werden können, ohne daß das Basisprotokoll verändert werden muß.

Bei der Unterstützung der Mobilität müssen in diesem Zusammenhang zwei Sachverhalte berücksichtigt werden: Einerseits der wechselnde Ort des mobilen Rechners und andererseits die durch die ständige Positionsänderung ungültig werdenden Dienstinformationen. Der Client, der die Dienstinformation abfragen möchte, muß in der Lage sein, diese Änderungen zu erkennen und sich den neuen Gegebenheiten anzupassen. U. U. muß er zusätzlich mit mehreren DAs kommunizieren können. Da der Mobilitätsaspekt immer mehr in den Vordergrund rückt, gilt es, die Bereitstellung von Informationen an die Situation, in der sich der mobile Nutzer befindet, und an seinen Bedarf anzupassen. Man kann daher vom Wunsch nach *situationsgesteuerter und bedarfsgerechter Bereitstellung von Informationen* sprechen. Ein solcher Mechanismus muß in der Lage sein, für den Benutzer Dienste in unterschiedlichen Umgebungen zu finden und deren Konfiguration dynamisch anzupassen. Abhängig von der Rolle des Nutzers (z. B. Administrator, Mitarbeiter oder Gast) muß er dann den geeigneten Dienst erkennen und zur Verfügung stellen. Zusätzlich soll er sowohl bei mobilen als auch bei stationären Rechnern eingesetzt werden. Dann kann er ebenfalls Administratoren bei ihren Verwaltungsaufgaben unterstützen. Seine Aufgabe besteht darin, neue Dienste bekanntzumachen bzw. Konfigurationsänderungen von Diensteanbietern durchzuführen, ohne eine Mitteilung an die betreffenden Benutzer senden zu müssen.

6 Literatur

[1] Robert McGrath: Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing, Center for Excellence in Space and Information Science, NASA Goddard Space Flight Center, April 2000.

[2] Morris Sloman: Management Issues for Distributed Services, IEEE Second International Workshop on Services in Distributed and Networked Environments (SDNE 95), 52-59, IEEE Computer Society Press, June 1995.

[3] Harry Chen, Dipanjan Chakraborty, Liang Xu, Anupam Joshi, and Tim Finin: Service Discovery in the Future Electronic Market, In Proceedings Workshop in Knwoledge, AAAI, 2000.

[4] Jonathan Rosenberg: Internet Telephony Gateway Discovery, Proceedings of IEEE Bell Laboratories and Columbia University, December 1997.

[5] Tim Berners-Lee, Larry Masinter, and Mark McCahill: Uniform Resource Locators (URL), Request for Comments (Proposed Standard) 1738, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc1738.txt>, December 1994.

[6] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter: Uniform Resource Identifiers (URI): Generic Syntax, Request for Comments (Proposed Standard) 2396, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2396.txt>, August 1998.

[7] Christian Bettstetter: Toward Internet-based Car Communications: On Some System Architecture and Protocol Aspects. In Proceedings EUNICE 2000, Sixth EUNICE Open European Summer School, Twente, Netherlands, September 2000.

[8] Javier Govea, and Michel Barbeau: Comparison of Bandwidth Usage: Service Location Protocol and Jini, School of Computer Science, Carleton University, September 2000.

[9] Bradley J. Rhodes, Nelson Minar, and Josh Weaver: Wearable Computing Meets Ubiquitous Computing: Reaping the best of both worlds, The Proceedings of The Third International Symposium on Wearable Computers (ISWC '99), San Francisco, CA, October 18-19 1999.

[10] Paul Mockapetris: Domain Names - Concepts and Facilities, Request for Comments (Proposed Standard) 1034, Internet Engineering Task Force,<URL:http://www.ietf.org/rfc/rfc1034.txt, November 1987.

[11] Paul Mockapetris: Domain Names - Implementation and Specification, Request for Comments (Proposed Standard) 1035, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc1035.txt>, November 1987.

[12] Paul Mockapetris: DNS Encoding of Network Names and other Types, Request for Comments (Proposed Standard) 1101, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc1101.txt>, April 1989.

[13] Craig F. Everhart, Louis A. Mamakos, Robert Ullmann, and Paul Mockapetris: New DNS RR Definitions, Request for Comments (Proposed Standard) 1183, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc1183. txt>, October 1990.

[14] Gerd Aschemann, Svetlana Domnitcheva, Peer Hasselmeyer, Roger Kehr, and Andreas Zeidler: A Framework for the Integration of Legacy Devices into a Jini Management Federation. In Proceedings of the Distributed Systems: Operations and Management (DSOM99), pp.257-268. Springer Verlag Berlin Heidelberg, 1999.

[15] Paul Vixie, Susan Thomson, Yakov Rekhter, and Jim Bound: Dynamic Updates in the Domain Name System (DNS UPDATE), Request for Comments (Proposed Standard) 2136, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2136.txt>, April 1997.

[16] Mark Wahl, Tim Howes, and Steve Kille: Lightweight Directory Access Protocol (v3), Request for Comments (Proposed Standard) 2251, Internet Engineering Task Force, <URL: http://www.ietf.org/rfc/rfc2251.txt>, December 1997.

[17] Steven Fitzgerald, Ian Foster, Carl Kesselman, Gregor vol Laszewski, Warren Smith, and Steven Tuecke: A Directory Service for Configuring High-Performance Distributed Computations. In Sixth IEEE International Symposium on High Performance Distributed Computing, 1997.

[18] Tim Howes: The String Representation of LDAP Search Filters, Request for Comments (Proposed Standard) 2254, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2254.txt>, December 1997.

[19] Object Management Group: CORBAservices: Common Object Services Specification, 1995.

[20] Salutation Consortium, <URL:http://www.salutation.org>.

[21] Bob Pascoe: Salutation Architectures and the newly defined service discovery protocols form Microsoft and Sun, Salutation Consortium, White Paper <URL:http://www.salutation.org/whitepaper/Jini-UPnP>, June 6, 1999.

[22] John Veizades, Erik Guttman, Charles E. Perkins, and Scott Kaplan: Service Location Protocol, Request for Comments (Proposed Standard)

2165, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2165.txt>, June 1997.

[23] James Kempf, and Pete St. Pierre: Service Location Protocol for Enterprise Networks, Internet Engineering Task Force, Wiley, ISBN 0-471-31587-7, 1999.

[24] Erik Guttman, Charles E. Perkins, John Veizades, and Michael Day: Service Location Protocol Version 2, Request for Comments (Proposed Standard) 2608, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2608.txt>, June 1999.

[25] Charles E. Perkins, and Erik Guttman: DHCP Options for Service Location Protocol, Request for Comments (Proposed Standard) 2610, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2610.txt>, June 1999.

[26] Erik Guttman: Service Location Protocol: Automatic Discovery of IP Network Services, pages 71-80, IEEE Internet Computing, Vol. 3, No. 4, July/August 1999.

[27] Erik Guttman, Charles E. Perkins, and James Kempf: Service Templates and Service: Schemes, Request for Comments (Proposed Standard) 2609, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2609.txt>, June 1999.

[28] W. Keith Edwards: Core Jini. Upper Saddle River, MJ: ISBN 0-13-014469-X , Prentice Hall, 1999.

[29] Sun Microsystems: Java Remote Method Invocation (RMI), <URL:http://http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html>, 2001.

[30] Yaron Y. Goland, Ting Cai, Paul Leach, and Ye Gu: Simple Service Discovery Protocol/1.0 – Operating without an Arbiter, Internet Draft, Internet Engineering Task Force, <URL:http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt>, April 2000.

[31] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz: An architecture for a secure service discovery service. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom '99), pages 24-35, Seattle, Washington, August 1999.

[32] Ralph Droms: Dynamic Host Configuration Protocol, Request for Comments (Proposed Standard) 2131, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2131.txt>, March 1997.

[33] Steve Alexander, and Ralph Droms: DHCP Options and BOOTP Vendor Extensions, Request for Comments (Proposed Standard) 2132, Internet Engineering Task Force, <URL:http://www.ietf.org/rfc/rfc2132.txt>, March 1997.