Analyse von Service Discovery Protokollen für P2P-Anwendungen

Studienarbeit am Institut für Betriebs- und Dialogsysteme
Prof. Dr. Frank Bellosa
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Nicole Nöldner

Betreuer:
Prof. Dr.-Ing. Frank Bellosa
Dr.rer.nat. Thomas Fuhrmann
Dipl.-Ing. Björn Saballus

Tag der Abgabe: 27. Juni 2007

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 27. Juni 2007

Inhaltsverzeichnis

1. Einleitung

Peer-to-Peer gilt als eines der meistdiskutierten Themen aus dem Bereich der Informationstechnologie und erfreut sich stark wachsender Beliebtheit. Ein Grund dafür mag u.a. der rasante Aufstieg und Fall des ersten großen bekannt gewordenen File-Sharing-Programmes Napster sein, in dem Millionen von Benutzern untereinander Daten, wie z.B. Musik und Filme, austauschen und der darauffolgende Boom des Filesharing. Dies führte auch dazu, dass Peer-to-Peer-Systeme vor allem als Tauschbörsen für Files bekannt sind.
Obwohl ursprünglich dafür entworfen, sind Peer-to-Peer-Systeme viel mehr als File-Sharing-Programme a la Napster, Gnutella, Kazaa und andere. Zu den Anwendungsbereichen gehören neben den File-Sharing-Systemen noch das Instant Messaging (IP-Telefonie oder IP-Videokonferenzen), Grid bzw. Distributed Computing (verteiltes Rechnen) und die Collaboration bzw. P2P Groupware (Zusammenarbeiten, Kommunizieren von Teilnehmern). Oftmals werden auch noch die Web Services zum Bereich des P2P dazugezählt.

Da in den letzten Jahren Peer-to-Peer-Systeme in immer mehr Bereichen verwendet werden, werden diese immer komplexer. Durch die steigende Anzahl an Teilnehmern werden auch die Aufgaben, die ein solches Netz zu erfüllen hat, immer umfangreicher und die Skalierbarkeit der Netze wird gefordert. Durch eine steigende Anzahl der mobilen Teilnehmer, muss auch die dadurch entstandene Dynamik im Netz beachtet werden, was zu einem weiteren Problem führt, das gelöst werden muss. Es muss verstärkt darauf geachtet werden, dass das Netz trotz dieser neu auftretenden Eigenschaften noch voll funktionstüchtig ist.
Eines dieser Probleme stellt das Suchen und Finden von Diensten unterschiedlichster Art in einem Peer-to-Peer-System, das Service Discovery, dar. Dieses Thema der Service Discovery Protokolle soll in dieser Arbeit behandelt werden.

1.1 Aufgabenstellung

In dieser Studienarbeit soll das Problem von Verzeichnisdiensten (engl. Look-up and Directory Services) diskutiert werden, und zwar mit speziellem Blick auf die dezentralisierten, d.h. vollständig verteilten Dienste, im Bereich der Ambient Technology. Ziel eines solchen Systems ist die effiziente Vermittlung zwischen Prozessen,

die einen bestimmten Dienst anbieten (zum Beispiel Zugang zu speziellen Geräten, Speicher, Rechenleistung) auf der einen Seite, und „Verbraucher“-Prozessen auf der anderen Seite, die diese Dienste benutzen möchten. Ziel der Studienarbeit ist es, die Anforderungen eines solchen Systems zu beschreiben und wichtige Aspekte schon existierender Lösungen zu skizzieren.

1.2 Gliederung der Arbeit

Die Studienarbeit ist in 5 Kapitel untergliedert.

Kapitel 2 führt in die Grundbegriffe dieser Arbeit ein. Nach der Beschreibung der Hauptmerkmale eines Peer-to-Peer-Systems, wird der Bereich der Ambient Technology, in dessen Bereich die Peer-to-Peer-Systeme im Zusammenhang mit dieser Arbeit eingesetzt werden sollen, erläutert. Anschließend wird in den Bereich des Service Discovery eingeführt, das für diese Arbeit das zentrale Thema bildet.

Kapitel 3 widmet sich der genaueren Betrachtung des Service Discovery. Nachdem in Abschnitt 3.1 einige Suchstrategien mit ihren Vor- und Nachteilen näher beschrieben werden, beschäftigt sich Abschnitt 3.2 mit mit dem Thema Protokolle mit speziellem Hinblick auf das Service Discovery.

Anschließend werden vielversprechende Service Discovery Protokolle in Kapitel 4 untersucht. Aufgrund der großen Anzahl kann nur auf die wichtigsten eingegangen werden. In Abschnitt 4.2 werden diese näher betrachtet, da hierbei vermutet wird, dass sie in Peer-to-Peer-Systemen im Ambient Technology Bereich einsetzbar sind.

Mit der Zusammenfassung und einem Ausblick in Kapitel 5 wird diese Arbeit abgeschlossen.

2. Grundlagen

In diesem Kapitel werden die Grundlagen der Schwerpunkte dieser Arbeit: Peer-to-Peer-Systeme, Ambient Technology und Service Discovery in Peer-to-Peer-Systemen behandelt. Hauptaugenmerk soll dabei auf Eigenschaften, Funktionsweise und Anwendungsgebiete gelegt werden.

Im ersten Abschnitt 2.1 werden die Grundzüge der Peer-to-Peer-Systeme aufgezeigt. Da Peer-to-Peer-Systeme bereits in einer Reihe anderer Arbeiten ausführlich erklärt wurden (siehe z.B. [p2p04]), werden nach einer kurzen Einführung und Definition noch die verschiedenen Ausprägungsformen aufgezeigt.

Als nächstes gibt Abschnitt 2.2 eine kurze Einführung in den Bereich des Ambient Technology. Nach einer genaueren Untersuchung des Begriffes werden die verschiedenen Eigenschaften und Einsatzmöglichkeiten aufgezeigt. Ein Szenario erklärt diesen Bereich anschaulich.

Abschnitt 2.3 schließlich führt in das Thema des Service Discovery ein. Nach einer allgemeinen Beschreibung des Begriffes „Dienst“ wird der Bereich des Service Discovery mit Schwerpunkt Peer-to-Peer-Systeme vorgestellt.

2.1 Peer-to-Peer-Systeme

Obwohl Peer-to-Peer-Systeme erst seit etwa Mitte des Jahres 2000 bekannt sind, kann man sie doch zu den ältesten Architekturen im Bereich der Informationstechnologie zählen. Im Grunde genommen ist auch das Telefonnetz und das Internet ein Peer-to-Peer-System: Ein Netz, in dem Ressourcen, die zu der Zeit noch sehr teuer und kostbar waren, auf mehrere Rechner verteilt wurden und so besser von allen Teilnehmern genutzt werden konnten. [Ora01] schreibt, dass Peer-to-Peer das Internet auf seinen eigentlichen Grundgedanken zurückbringt: Ein selbstorganisiertes Netz, in dem alle Teilnehmer gleichberechtigt sind, und jeder dieser Teilnehmer sowohl Dienste zur Verfügung stellen als auch andere Dienste nutzen kann.
Der Begriff Peer-to-Peer (engl. peer „Gleichgestellter“, „Ebenbürtiger“) macht dies auch deutlich.

Definfition eines Peer-to-Peer-Systems nach [Zit05]:
Selbstorganisierendes System gleichberechtigter, autonomer Systeme ohne Nutzung zentraler Dienste auf der Basis eines unzuverlässigen Netzwerks.

Ein Peer-to-Peer-Netz kann nach dieser Definition folglich als ein verteiltes System aufgefasst werden.

Nachfolgend werden die Peer-to-Peer-Systeme anhand des **Hierarchiegrades** ihrer Architektur charakterisiert. Der Vollständigkeit halber werden dabei alle möglichen Netzwerktopologien für Peer-to-Peer-Systeme aufgeführt.

2.1.1 Client-Server-System

In diesem System stellt ein zentraler Server die Ressourcen zur Verfügung und die Clients, welche diese Ressourcen verwenden wollen, müssen auf diesen Server zugreifen. Zwischen den Clients findet keine direkte Kommunikation statt, der Server steuert jegliche Interaktion. Beispiele hierfür sind u.a. das World Wide Web (WWW) und das Domain Name System (DNS).
Es handelt sich hierbei also nicht um ein Peer-to-Peer-System nach der Definition.

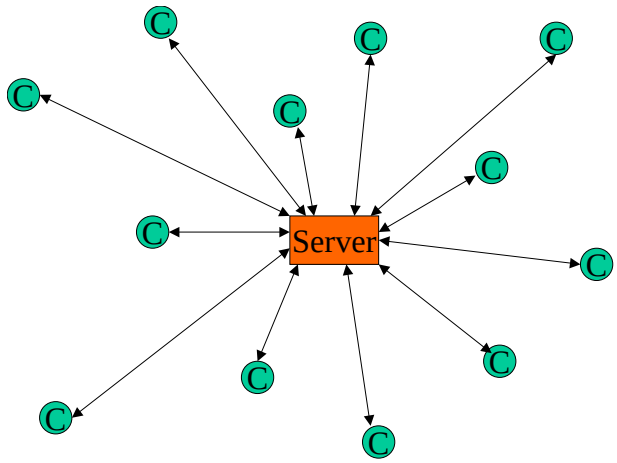

Abbildung 2.1: Client-Server-Prinzip.

2.1.2 Hybrides Peer-to-Peer-System

Dieses System unterscheidet sich zu dem herkömmlichen Client-Server-Ansatz bereits entscheidend in der Art, wo die Ressourcen lokalisiert sind. Die Ressourcen sind auf verschiedene Clients verteilt und dort gespeichert. Es existiert immer noch ein zentraler Server, welcher aber lediglich der Koordination dient. Dieser zentrale „Server“ wird in einigen Quellen auch als Super-Peer bezeichnet. Super-Peers verfügen meist über eine größere Bandbreite als die anderen Peers und haben daher weitergehende und tieferreichendere Aufgaben. Die Clients nutzen die verteilten Ressourcen gemeinsam und können auch direkt miteinander kommunizieren. Beispiele hierfür sind FastTrack und ICQ.

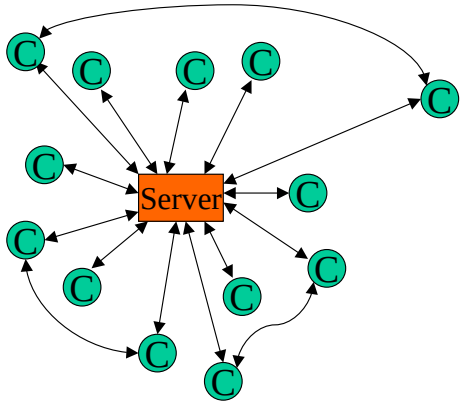

Abbildung 2.2: Hybrides Peer-to-Peer-System.

2.1.3 Reines Peer-to-Peer-System

Ein reines Peer-to-Peer-System schließlich ist ein völlig verteiltes Netzwerk, das auch die Definition erfüllt. Hierbei werden die Ressourcen vollständig dezentral organisiert und genutzt. Es existiert folglich auch kein zentraler Server. Innerhalb eines solchen Systems verwendet jeder Peer die Ressourcen, die von den anderen Peers zur Verfügung gestellt werden.
Beispiele hierfür sind Gnutella und Freenet. Ein „reines“ Peer-to-Peer-System hat dabei folgende Eigenschaften: Alle Peers sind autonom und kommunizieren direkt miteinander, um die geteilten Ressourcen zu nutzen. D.h. sie verwalten sich selbstständig und müssen nicht von einer zentralen Instanz gesteuert werden. Eine weitere Eigenschaft ist, dass alle Peers gleichberechtigt sind. Ein Peer agiert hier als sogenannter Servent, welcher sowohl Client- als auch Server-Dienste anbietet bzw. diese in Anspruch nimmt. Dies steht im Gegensatz zu dem ursprünglichen Client-/Server-Ansatz des WWW.

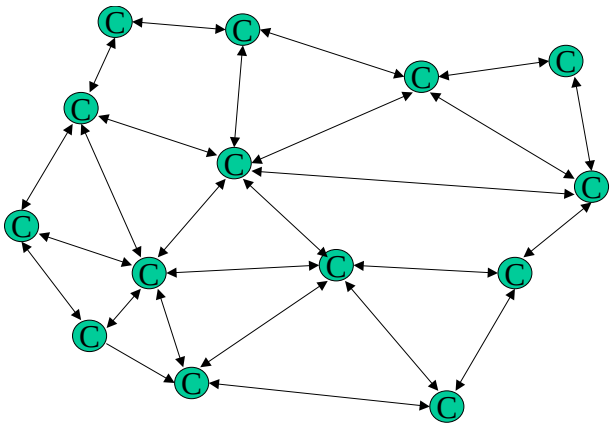

Abbildung 2.3: Reines Peer-to-Peer-System.

Dieses „reine“ Peer-to-Peer-System lässt sich weiter charakterisieren nach der **Strukturiertheit** des Systems: Es existieren sowohl als unstrukturierte als auch strukturierte Systeme.

Unstrukturierte Peer-to-Peer-Systeme

In einem unstrukturierten System sind die Peers zufällig angeordnet. Prinzipiell kann jeder Peer jeden beliebigen anderen Peer, zu dem er eine direkte Verbindung hat, als

seinen Nachbarn wählen. Dem jeweiligen Peer sind dabei nur benachbarte Peers innerhalb seiner Kommunikationsreichweite bekannt. Außerdem hängt die Anzahl der bekannten benachbarten Peers auch noch von der Leistungsfähigkeit (z.B.Speicher) des jeweiligen Peer ab.

Nachrichten können in einem solchen System mittels Fluten übermittelt werden, was einen enormen Overhead bei einer großen Anzahl an Knoten darstellen kann. Ein solches System ist auch schlecht skalierbar. Eine andere Möglichkeit der Suchanfrage stellt der Random Walk dar. Es kann allerdings beiden Möglichkeiten nicht garantiert werden, dass eine Suchanfrage ihr Ziel erreicht. Näheres zu diesen Möglichkeiten wird in Kapitel 3.1 näher erläutert.

Ein Vorteil dieser Struktur ist ihre Robustheit. Denn bei einem gezielten Angriff bzw. Ausfall eines bestimmten Peers kann sich das System schnell und selbstständig neu organisieren.

Beispiele für unstrukturierte Peer-to-Peer-Systeme sind Gnutella und Freenet, welche in [KA02] näher erläutert werden.

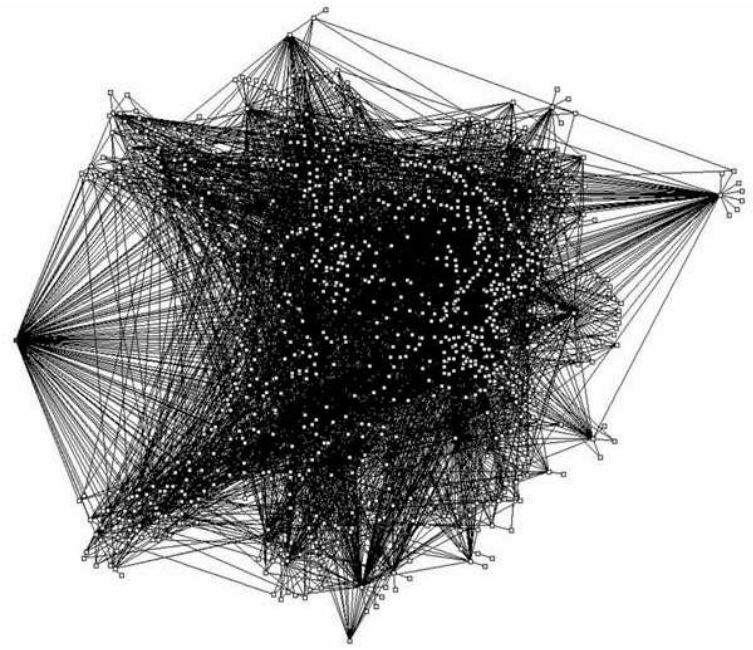

Abbildung 2.4: Topologie eines Gnutella-Netzwerkes [JAB01].

Strukturierte Peer-to-Peer-Systeme

In einem strukturierten System werden die Knoten nach einer bestimmten Struktur angeordnet. Im Gegensatz zu dem unstrukurierten System werden den Peers hier feste Nachbarn und Verbindungen zugwiesen. Dies macht eine zielgerichtete Suche nach bestimmten Ressourcen möglich. Die Anfragen können also direkt an den entsprechenden Peer weitergeleitet werden, ohne erst womöglich willkürlich das ganz System durchlaufen zu müssen. Dadurch wird die Suche effizienter, schneller und es kommt auch nicht zum Verlust von Suchanfragen. Somit kann garantiert werden, dass angeforderte Ressourcen, wenn sie im System existieren, auch gefunden werden.

Ein Beispiel für strukturierte Peer-to-Peer-Systeme sind verteilte Hashtabellen, wie sie z.B. Chord [IS01] nutzt.

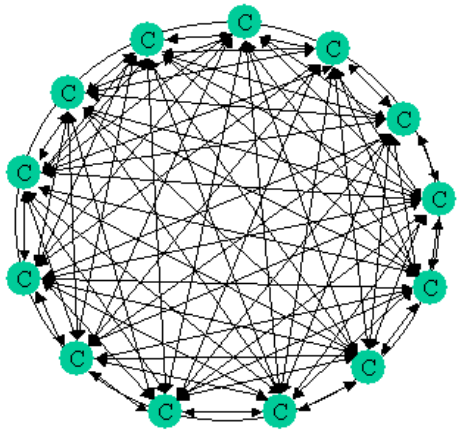

Abbildung 2.5: Beispielabbildung für eine verteilte Hashtabelle: Chord.

Da der Schwerpunkt dieser Arbeit auf den „reinen“ Peer-to-Peer-Systemen liegt, wird nachfolgend der Begriff des Peer-to-Peer-Systems synomym zu den „reinen“ Peer-to-Peer-Systemen verwendet.

Vor- und Nachteile

Das Fehlen von zentralen Instanzen in Peer-to-Peer-Systemen hat einen enormen Vorteil: Das Netz ist ausfallsicherer und vor gezielten Angriffen geschützt. Ein weiterer Vorteil durch das Fehlen des zentralen Servers sind geringere Kosten, da keine zusätzliche Hardware zur Realisierung des Servers (Konfiguration und Administration) benötigt wird. Außerdem wird die Einrichtung eines solchen Systems einfacher, da auf fundiertere Fachkenntnisse eines Server-Administrator verzichtet werden kann. Nicht zu vernachlässigen den bereits zuvor erwähnten Schutz gegenüber gezielten Angriffen.

Trotz dieser Vorteile und Vereinfachungen, die ein solches System auf den ersten Blick mit sich bringt, existieren auch nicht zu vernachlässigende Nachteile. Durch die Verteilung der Aufgaben auf die einzelnen Peers, werden diese auch mehr belastet. Dies kann besonders für leistungsschwache Peers problematisch werden. Neben ihren eigenen Aufgaben müssen sie zusätzlich die Aufgaben zur Aufrechterhaltung des Netzes übernehmen. Außerdem ist es schwieriger solche Systeme gegen äußere Angriffe jeglicher Art abzusichern. Denn anstatt lediglich einer zentralen Instanz muss nun der Zugriff auf die einzelnen verteilten Ressourcen überwacht und geregelt werden. Eine weitere Schwierigkeit ergibt sich bei Softwareupdates oder Backups durch die auf mehreren Geräten verteilten Ressourcen.

Auf die Suche in den einzelnen Ausprägungen von Peer-to-Peer-Systemen wird in Kapitel 3.1 näher eingegangen.

2.2 Peer-to-Peer-Systeme im Bereich des Ambient Technology

„Ambient technology or ambient intelligence is defined as an electronic environment sensitive to its inhabitants and responsive to the user's needs without the use of strange or complicated commands.“ [1]

In dem Paper [LL05] wird ein Szenario vorgestellt, dass nachfolgend als Einstiegsbeispiel in den Bereich des Ambient Technology genannt werden soll.

> Man stelle sich eine Person mit Namen Anita vor, welche in einem Supermarkt einkaufen gehen wird. Anita betritt also den Supermarkt und nimmt sich einen Einkaufswagen. Am Einkaufswagen ist eine Halterung für PDAs und Mobiltelefone angebracht. Nachdem sie ihren PDA in die Halterung eingelegt hat, erscheint eine Meldung auf dessen Bildschirm. Sie wird aufgefordert sich zu identifizieren. Dafür ist am Einkaufswagen ein biometrischer Fingerscanner angebracht. Nach erfolgreicher Identifkation kann Anita nun mit dem Einkauf beginnen. Ihr Einkaufszettel erscheint auf dem Bildschirm. Da Anitas Zuhause vernetzt ist, wird der Einkaufszettel automatisch erstellt. Dafür sind verschiedene Geräte, wie z.B. Kühlschrank, Mülleimer und Schrankmonitor mit Sensoren ausgestattet und kommunizieren unsichtbar mit ihrem PDA.
> Das Supermarkt-System informiert Anita über besondere Angebote und Aktionstage.
> Der PDA registriert die Position von Anita im Supermarkt und kann Anita so die benötigten Gegenstände aus dem aktuellen Gang anzeigen. Außerdem registriert er jeden Gegenstand mit Name und Preis, der in den Wagen gelegt wird. Genauso werden Informationen wieder gelöscht, wenn sie aus dem Wagen entfernt werden.
> Wenn Anita ihren Einkauf beendet hat, wird auf ihre Anforderung automatisch die Rechnung erstellt. Die Bezahlung erfolgt durch Eingabe des biometrischen Fingerabdruckes. Eine Möglichkeit wäre dabei die automatische Abbuchung des Betrages von ihrem Konto. Der Vorteil dabei ist, dass sie sich nicht erst noch an eine Kasse anstellen muss.

Wie aus diesem Beispiel ersichtlich wird, bedeutet „Ambient Technology“ Alltagsgegenstände mit Rechenleistung und Kommunikation auszustatten, so dass sie gemeinsam und ggf. selbständig auf Benutzeranforderungen reagieren können. Dabei werden Prozessoren, Speicherbausteine und Sensoren in viele dieser Alltagsgegenstände eingebaut. Dies wird erst durch die geringe Größe, den geringen Energiebedarf und den mittlerweile (fast) vernachlässigbaren Preis möglich. Somit können eine Vielzahl auch kleinster Geräte miteinander verknüpft werden und so sogenannte „intelligente“ (smarte) Umgebungen geschaffen werden. Smarte Umgebungen sind folglich Umgebungen, in denen Alltagsgegenstände mit Informationstechnologie, wie z.B. Sensoren ausgestattet sind, um zusätzliche Funktionen, wie z.B. Datenaustausch möglich zu machen.

[1] Bruno, Lee (2002, July 24) A Window into the future, Red Herring

Das System muss sich seiner Umgebung und den darin vorkommenden Geräten „bewusst“ sein. Dazu gehört auch die Fähigkeit verschiedene Objekte und/oder Personen innerhalb der Umgebung zu lokalisieren und identifizieren. Die Intelligenz des Systems liegt darin neue Daten zu bearbeiten und Verbindungen zu den existierenden Informationen zu ziehen. Schließlich muss das System auch eine natürliche und für den Benutzer einfach zu bedienende Schnittstelle besitzen, so dass grundlegende Anforderungen übermittelt und interpretiert werden können und diese ohne unnötigen und unnatürlichen Aufwand von seitens des Benutzers integriert werden können. Eine natürliche Benutzerschnittstelle bedeutet auch den Gebrauch von Sprach- und Gestenerkennung. Die verschiedenen Komponenten sollten dabei mit dem Hintergrund der Umgebung so gut wie möglich verschmelzen.

Damit diese Komponenten miteinander kommunizieren können, müssen robuste Adhoc-Netzwerke gebildet werden, was erst durch fortschrittliche Netzwerktechnologien möglich wird.
Solche Adhoc-Netzwerke haben folgende zwei Haupteigenschaften:

- mobile Knoten (Peers)
- keine feste Infrastruktur

Im Gegensatz zu herkömmlichen Client-Server-Arbeitsumgebungen, die mit großen Betriebs- und Middlewaresystemen ausgestattet sind, gibt es im Bereich der Ambient Technology nur sehr eingeschränkte Ausführungsmöglichkeiten für Software. Dies resultiert u.a. dadurch, dass die meisten der Endgeräte-Technologien, die in eine Umgebung integriert sind, klein sind und so über nur geringe Ressourcen verfügen. Solche Geräte besitzen eine geringe Speicherkapazität, eine niedrige Datenübertragungsrate, eine begrenzte Stromversorgung.
Desweiteren darf in solchen Systemen der Aspekt der Mobilität nicht vernachlässigt werden. Dadurch dass die Peers autonom sind und eventuell auch mobil (d.h. an keinen Ort gebunden), können sie sich überall bewegen. Verlassen sie dabei die Reichweite des Netzwerks, bricht die Verbindung ab. Es kann auch vorkommen, dass ein Peer über längere Zeit ausgeschaltet ist (z.B. Stromsparmodus) oder plötzlich ausfällt.
Jedes dieser Möglichkeiten bedeutet, dass ein Peer nicht konstant über die gleiche Netzwerkadresse erreichbar ist. Somit müssen diese Systeme unabhängig vom herkömmlichen DNS-System (Domain Name System) in der Lage sein, ein Netzwerk aufzubauen.

Technologisch gesehen umfasst der Bereich des Ambient Technology die Sensorik, eingebettete Systeme, sowie deren Optimierung hinsichtlich des Energieverbrauchs, multi-modale I/O Systeme und adaptive Softwaresysteme.

Vorhergehend beschriebene Umgebungen der Ambient Technology besitzen nach [Loe03] folgende 4 Eigenschaften:

Allgegenwart: Allgegenwart bedeutet in diesem Zusammenhang, dass wir von einer Vielzahl an eingebetteten Systemen umgeben sind. Diese Systeme werden dadurch unsichtbar und verschwinden in den Hintergrund unserer Wahrnehmung. Solche Systeme können somit in beliebige Gegenstände integriert werden und praktisch überall vorhanden sein.

Bewusstsein: Bewusstsein bezieht sich hier auf die Fähigkeit des Systems, Objekte und Personen, die sich in der näheren Umgebung aufhalten, zu lokalisieren.

Intelligenz: Mit Intelligenz ist hier gemeint, dass solche „bewussten“ Umgebungen dazu fähig sind, den Kontext zu erkennen. Das bedeutet, sie können sich an die in der Umgebung befindlichen Personen anpassen, deren Verhalten erkennen und somit auch davon lernen.

Natürliches Verhalten: Natürliches Verhalten umfasst hier fortschrittliche Fähigkeiten, wie Sprach-, Gesten- und Objekterkennung, welche eine natürliche Kommunikation mit der „intelligenten“ Umgebung ermöglichen. Damit verbunden ist auch eine natürliche Benutzerschnittstelle.

An eine Ambient Technology Umgebung werden teilweise hohe Anforderung bezüglich

- Sicherheit
- Reaktionszeit
- Zuverlässigkeit

gestellt.

Ziel des Ambient Technology soll dabei sein, Sensoren und Computerprozessoren (vorallem mittels Funktechnologie) so zu vernetzen, dass das Leben der Menschen, die in dieser Umgebung leben, vereinfacht wird. Anwendungsmöglichkeiten sind u.a. smarte Räume, Katastrophenschutz, Gesundheitsüberwachung. Eine weitere Anwendungsmöglichkeit stellt dabei das „intelligente Haus“ [WK04] dar. Hier wurden sämtliche Gegenstände – Waschmaschine, Kaffeemaschine, Spülmaschine, etc. – mit eingebetteten Systemen ausgestattet und Kameras und Sensoren wie Druck-, Licht-, Bewegungs- und Hitzesensoren in die Umgebung integriert. Diese können dann mittels mobiler Geräte (z.B. PDA) gesteuert werden.
Bereits jetzt besitzt ein durchschnittlicher Haushalt um die 50 Prozessoren [RH04], die für uns unsichtbar in Geräte integriert sind. Diese Prozessoren sollen miteinander kommunizieren können, um uns noch effektiver unterstützen zu können.

Abbildung 2.6: Beispielabbildung für eine Verkehrsanwendung [fra06].

Vorteile in der Verwendung von Ambient Technology bestehen vor allem darin, dass damit unser Leben erleichtert werden kann. Bei Bedarf kann die „intelligente“ Umgebung den Menschen unterstützen. Dies ist vor allem für behinderte Menschen, wie z.B. Blinde, sehr hilfreich. Zudem sind keine besonderen Kenntnisse über diese Technologien oder Kenntnisse über eine bestimmte Bedienungsweise notwendig, da sie unsichtbar in unsere Umgebung einfügen und sich automatisch an unser Verhalten anpassen kann.

Allerdings hat diese allgegenwärtige Existenz der Technologie auch seine Nachteile. Durch die „intelligenten“ Umgebungen wird der Mensch förmlich zum gläsernen Menschen und kann prinzipiell ständig kontrolliert werden. Solche Informationen können auch missbraucht werden. Dagegen gilt es sich zu schützen.

Um nun solche robusten ad-hoc Netzwerke zu bilden und die Kommunikation zwischen den einzelnen Geräten zu organisieren, bieten sich Peer-to-Peer-Systeme an. Denn gerade die „reinen“ Peer-to-Peer-Systeme haben einige Gemeinsamkeiten mit den Umgebungen des Ambient Technology. So besitzen beide Systeme z.B. autonome mobile Geräte, die unvorhergesehen nicht mehr erreichbar sein können. Man geht folglich von einer hohen Dynamik aus.

Abbildung 2.7: Beispieldarstellung einer Ambient Technology Umgebung [vdM].

Speziell der Bereich des Service Discovery spielt im Ambient Technology, genauso wie in den Peer-to-Peer-Systemen selbst, eine überaus wichtige Rolle in der Realisierung dieser „intelligenten“, den Menschen unterstützenden Umgebung. Im nächsten Kapitel wird daher der Bereich des Service Discovery näher erläutert.

2.3 Service Discovery in Peer-to-Peer-Systemen

Ein Service wird auch als Dienst oder Dienstleistung bezeichnet. Im Bereich der Telekommunikation wird ein Dienst von einem oder mehreren Servern erbracht und von einem Client genutzt. Es gibt dabei verschiedene Arten von Diensten, z.B. Rechenleistung, Druck-Service, Download von Dateien oder auch Umgebungsinformationen wie Verkehrsinformationen. Im Bereich des Ambient Technology kann ein Dienst z.B. das automatische Lichtein- und ausschalten beim Betreten oder Verlassen eines Raumes sein.

[Zit05] beschreibt das Service Discovery grob als ein „effizientes Auffinden von und effiziente Kommunikation mit mobilen Dienstgebern“.

Service Discovery bedeutet im Allgemeinen, dass ein Teilnehmer (Peer) in einem Netz einen gewünschten Dienst mit Hilfe eines bestimmten Protokolls finden kann. Sofern er denn in dem Netz vorhanden ist. Zu Service Discovery kann dabei auch das Service Advertisement gehören. Dabei können sich die Dienste auch selbst bei einem Netz anmelden und registrieren.

Für den Teilnehmer ist es dabei wichtig, einen gewünschten Dienst möglichst einfach und schnell, d.h. ohne sein System neu konfigurieren zu müssen, nutzen zu können. Insbesondere mit der weit verbreiteten Entwicklung mobiler Geräte und der selbst verwalteten Adhoc-Netzwerke wird das dynamische und effektive Entdecken und Verwenden von Diensten in einem Netzwerk sowie die automatische Systemkonfiguration eine immer wichtigere Rolle spielen.

Wie schon erwähnt, stellt das Suchen und Finden von Diensten, das Service Discovery, in großen Peer-to-Peer-Systemen mit steigender Teilnehmerzahl ein Problem dar. Besonders wenn es sich um völlig verteilte Netzwerke handelt, wie es im Bereich der Ambient Technology der Fall sein kann.

Die Suche in kleinen Systemen gestaltet sich im Vergleich zu großen Systemen mit einigen hundert oder tausend Peers dagegen relativ einfach. Denn in kleinen Systemen, die nur einige wenige Peers besitzen, muss nicht auf die manchmal nur spärlich vorhandenen Ressourcen wie Bandbreite oder Prozessorleistung geachtet werden. Außerdem müssen eine viel geringere Anzahl an Peers nach einem gewünschten Dienst durchsucht werden. Das System kann folglich trotzdem funktionsfähig bleiben.
Die Skalierbarkeit, d.h. bis zu welcher Peer-Anzahl ein System noch funktionsfähig ist, hängt dabei vom verwendeten Protokoll und der verwendeten Technik (wie Netzwerk-Struktur, Übertragungstechnik) ab. Damit größe Systeme mit einigen hundert oder tausend Peers funktionsfähig bleiben, muss auf eine besonders effiziente Nutzung der Ressourcen Wert gelegt werden. Daher sollte es auch Ziel einer effizienten Suchstrategie sein, den gesuchten Dienst, die ein beliebiger Peer im System besitzen kann, über möglichst wenig Zwischenstationen und in minimaler Zeit zu lokalisieren. Desweiteren sollte im Fall der Nichtexistenz dieses gesuchten Dienstes der suchende Peer in ebenso minimaler Zeit darüber in Kenntnis gesetzt werden. Die Beachtung der Kriterien Entdeckungszeit und Antwortzeit bestimmen ganz stark die Funktionstüchtigkeit und Effizienz Service Discovery in großen Systemen.

Grundlegender Ablauf einer Suchanfrage:

Ein Peer bietet seinen Dienst, mittels einiger Attribute, im System an. Das kann entweder an einer zentralen Stelle, an einer Anzahl besonders ausgezeichneter Peers oder allgemein jedem Peer bekannt gemacht werden. Die Attribute stellen dabei Informationen über den Dienst bzw. über den anbietenden Peer dar. Solche Informationen können z.B. Diensttyp, Fähigkeiten oder auch Zugriffsinformationen (Accessrights) sein.
Ein Client wiederrum stellt eine Dienstanfrage unter Angabe des Diensttyps an eine zentrale Stelle, an jene ausgezeichneten Peers oder an die Allgemeinheit. Er spezifiziert die Attribute dabei in der Weise, wie er es benötigt. Sollten mehrere Instanzen des gewünschten Dienstes zur Verfügung stehen, wählt der Client selbstständig eine aus.

Entsprechend der Einteilung von Peer-to-Peer-Systemen nach dem Hierarchiegrad bzw. der Strukturiertheit ihrer Architektur, kann auch die Dienstesuche in verschiedene Suchstrategien unterteilt werden.

Für ein **Client-Server-System** wird die Suchanfrage an den

- **zentralen Server**

gestellt.

Für **unstrukturierte Systeme** kann man vor allem zwischen

- **Fluten** (Breitensuche) und
- **Routing Indizes**

unterscheiden.

Für **strukturierte Systeme** wird die Suche mittels

- **verteilter Hashtabellen**

realisiert.

Im nachfolgenden Kapitel werden diese Suchstrategien näher erläutert und untersucht.

3. Analyse

Nachdem die Grundlagen des Service Discovery im Umfeld von Peer-to-Peer-Systemen in Kapitel 2 behandelt wurden, wenden wir uns nun dem eigentlichen Haupthema dieser Arbeit, der Analyse des Service Discovery und einiger dafür zuständiger Protokolle zu.

Beginnend im Abschnitt 3.1 mit der Erklärung des Ablaufs von Suchanfragen nach den einzelnen Suchstrategien, wird darauf aufbauend in Abschnitt 3.2 das Thema Protokolle der Dienstesuche eingeführt.

3.1 Ablauf des Service Discovery nach den verschiedenen Suchstrategien

Es gibt vier grundlegende Mechanismen des Service Discovery in einem Peer-to-Peer-System:

- Suchanfragen an **zentralen Server**
- Suchanfragen durch **Fluten**, also ohne genaues Ziel
- Suchanfragen durch **gezieltes Routing** und
- Suchanfragen in **verteilten Hashtabellen**

Die Suche mittels verteilter Hashtabellen gehört eigentlich in den Bereich des gezielten Routings. Sie werden hier aber aufgrund ihres Bekanntheitsgrades für die Verwendung bei strukturierten Netzen extra aufgelistet. Daneben gibt es noch einige weitere Algorithmen und Strategien, die hier aber aufgrund des Umfanges nicht näher erwähnt werden sollen. Die dargestellten Suchstrategien stellen die bekanntesten und am häufigsten verwendeten dar.

Zur Erklärung der Abläufe nehmen wir ein Peer-to-Peer-System mit einer unbestimmten Anzahl an Peers. Da es hierbei um den prinzipiellen Ablauf geht, wird von der Anzahl der Peers und der darunterliegenden Netzwerktechnologie abstrahiert.

Desweiteren definieren wir einen beliebigen Peer, der um eine Aufgabe zu erfüllen, einen dazu entsprechenden Dienst sucht. Dafür muss er eine Suchanfrage generieren und an das Netzwerk, in dem er sich momentan befindet, senden.
Nachfolgend werden nun die verschiedenen Möglichkeiten wie eine solche Suchanfrage in einem Peer-to-Peer-System ablaufen kann, erklärt.

3.1.1 Zentralisierter Ansatz

Der zentralisierte Ansatz basiert auf dem Client/Server-Modell bzw. dem hybriden Peer-to-Peer-Ansatz (Kapitel 2.1). Hierbei gibt es ein oder auch mehrere Peers, die als Server fungieren und für das Service Discovery verantwortlich sind. Um Suchanfragen und die Anmeldungen von Diensten sowie die Administration des Systems übernehmen zu können, müssen sie ausreichend Speicher, Rechenleistung und Bandbreite besitzen. Außerdem müssen sie stets erreichbar sein.
Die Clients dagegen können kleinere Geräte mit geringerer Leistung sein. Sie müssen lediglich ihre Dienste zur Verfügung stellen können.
Eine Darstellung der Struktur wurde bereits in Abbildung 2.1 bzw. 2.2 gegeben.

Möchte ein Client nun seine Dienste bekanntgeben, schickt er ein Service Advertisement mit einer Beschreibung seines Dienstes an einen Server. Das Service Advertisement enthält neben Informationen über den Dienst wie z.B. Typ, Name, Metainformationen etc. auch Informationen darüber, wo sich der Dienst befindet und wie man darauf zugreifen kann. Der Server ist an einer systemweit eindeutigen und bekannten Adresse erreichbar. Er speichert dieses Service Advertisement in seinem Index, einem lokalen Speicher. Benötigt nun ein Client einen bestimmten Dienst, dann stellt er eine Suchanfrage an den Server. Der Server wiederum schaut in seinem Index nach, ob im System ein solcher Dienst vorhanden ist, und schickt dem Client eine Antwort. Diese Antwort enthält alle notwendigen Informationen, um auf den Dienst zugreifen zu können. Anschließend kann der suchende Client eine direkte Verbindung mit dem Client, der den Dienst zur Verfügung stellt, aufbauen.

Zur Verdeutlichung ist in Abbildung 3.1 die Struktur einer Anfragebearbeitung dargestellt.

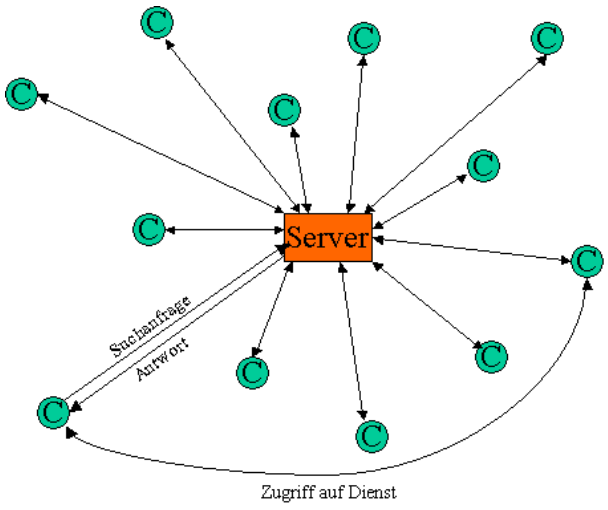

Abbildung 3.1: Anfragebearbeitung im zentralisierten Ansatz

Meldet sich ein Client vom System ab, dann benachrichtigt er den Server darüber, dass sein Dienst nicht mehr zur Verfügung steht. Um einen Ausfall eines Clients festzustellen, besteht die Möglichkeit, dass der Server regelmäßige Ping-Machrichten an die bei ihm angemeldeten Clients verschickt. Bekommt er keine Antwort von einem Client, dann steht dieser Client nicht mehr zu Verfügung. Sämtliche Service Advertisements von diesem Client werden aus seinem Index des Servers gelöscht.

Die Vorteile dieses Ansatzes liegen vor allem darin, dass sich die Suche und die Anmeldung von Diensten relativ einfach gestaltet. Auch die Administration dieses Systems, das der Server übernimmt, gestaltet sich nicht allzu kompliziert. Je mehr Clients sich in einem solchen System befinden, desto mehr Anfragen werden auch an den Server gestellt. Ein solches System besitzt eine hohe Performance, solange der oder die Server nicht ausgelastet sind.

Ein großer Nachteil dieses Ansatzes besteht darin, dass der Server ein „Single Point of Failure“ gegenüber Angriffen darstellt. Eine mögliche Lösung dafür wäre, wenn man mehrere Server an unterschiedlichen Stellen des Systems zur Verfügung stellt. Steht nur ein Server zur Verfügung und fällt der Server aus, bricht das ganz Netz zusammen. Desweiteren ist dieser Ansatz nicht für dynamische Umgebungen wie den Adhoc-Netzen oder des Ambient Technology geeignet. Denn in solchen Umgebungen existieren hauptsächlich kleine, leistungsschwache Geräte, die als Server nicht fungieren könnten. Außerdem zeichnen sich dynamische Umgebungen durch einen hohen Grad an Mobilität aus. Bewegt sich ein Client aus der Reichweite des Servers, kann er folglich auch nicht mehr auf andere Dienste zugreifen.

Als Beispiele für diesen Ansatz können Jini [Wal99] und INS (Intentional Naming System) [WAW99] genannt werden.

3.1.2 Zielloses Weiterleiten von Suchanfragen – Broadcaststrategien

Diese Verfahren sind zwar verhältnismäßig einfach zu implementieren, meistens jedoch auch recht ineffizient. Suchanfragen erzeugen hier enorme Mengen überflüssigen Netzverkehrs, was auch den Grund darstellt, warum Netzwerke mit solchen Strategien ab einer bestimmten Größe nicht mehr skalieren.

3.1.2.1 Fluten

Dieser Suchstrategie liegt prinzipiell eine relativ einfacher Ablauf zugrunde. Ein Peer sendet eine Suchanfrage an alle seine Nachbarn. Die Suchanfrage enthält hierbei neben Spezifikationen zu dem gewünschten Dienst, einen TTL(Time-To-Live)-Zähler und eine GUID (globally unique identifier, global eindeutiger Identifikator). Die GUID dient dazu eine Suchanfrage eindeutig zu identifizieren. Der TTL-Zähler beschreibt die Lebenszeit einer Suchanfrage in Hops. Ein Hop kann dabei als Übergang zu einem Peer angesehen werden. Die Suchanfrage wird nun von Peer zu Peer an alle Nachbarn eines Peers weitergeleitet.

Jeder Peer verarbeitet die Suchanfrage folgendermaßen:

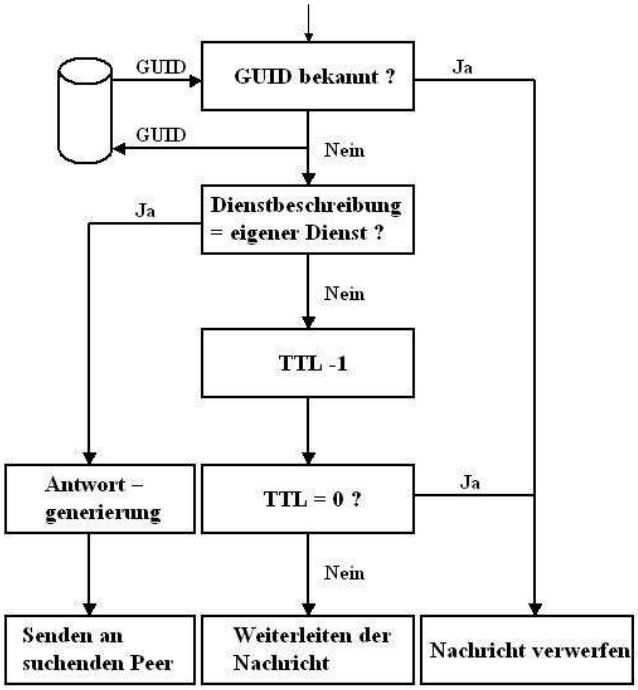

Abbildung 3.2: Ablauf der Suchanfragebearbeitung am Beispiel Fluten.

Die Anfrage wird solange weitergeleitet bis entweder der TTL-Zähler = 0 ist oder ein Treffer erzielt wird. Bei einem Treffer kann die Antwort-Nachricht mit Hilfe der in den Listen der Peers gespeicherten GUIDs und IP-Adressen direkt zum suchenden Peer zurückgeleitet werden. Damit kann der suchende Peer mit dem Dienstanbieter-Peer eine direkte Verbindung aufbauen und den Dienst nutzen.

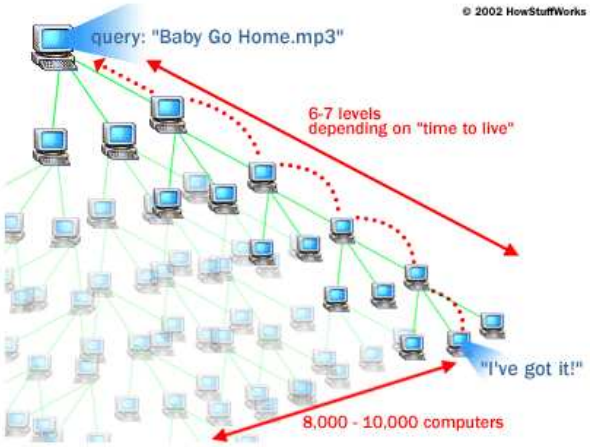

Abbildung 3.3: Weiterleitung einer Anfrage in einem System am Beispiel von Gnutella [wwwc].

Bei dieser Suchstrategie ergeben sich zwei Hauptprobleme:

1. Welcher Wert wird für den TTL-Zähler gewählt? Ist er zu niedrig, kann ein gesuchter Dienst eventuell nicht gefunden werden. Ist er zu hoch, könnte das System unnötig belastet werden, indem eine Anfrage erfolglos immer weiter dupliziert und weitergeleitet wird. Die Wahl eines geeigneten TTL-Zählers spielt also eine große Rolle.

2. Im Laufe der Zeit, wird die Zahl der Duplikate der Suchanfrage im System immer größer. Dadurch erhalten die Peers immer mehr Nachrichten, die verarbeitet werden müssen. Damit steigt sowohl die Auslastung der einzelnen Peers als auch die allgemeine Netzauslastung.

Es scheint somit verständlich, dass je größer die Systeme werden, diese Strategie immer schlechter skalieren. Somit kann es sehr lange dauern, bis eine Suchanfrage beantwortet wird bzw. schlimmstenfalls das ganze System wegen Überlastung ausfällt. Als weiterer Nachteil dieses Verfahrens kann die schlechte Performanz durch die hohe Netzwerkauslastung genannt werden.

Vorteile sind u.a. dass diese Suchstrategie sehr einfach zu realisieren ist und dass für die Peers keine großen Verwaltungsaufgaben außer das Speichern der GUIDs zu erledigen sind.

Ein Beispiel für ein Protokoll, das diese Suchstrategie verwendet, ist Gnutella [wwwc].

Expanding-Ring

Um einen der Hauptprobleme des Flutens, die Wahl eines geeigneten TTL-Zählers, zu lösen oder zumindest abzuschwächen, wurde der Expanding Ring Algorithmus [QL02] entwickelt. Dabei bleibt der grundlegende Ablauf des Flutens erhalten. Nur die Wahl eines TTL-Zählers ändert sich geringfügig. Beim Expanding Ring Algorithmus wählt ein Peer für seine Suchanfrage zunächst einen niedrigen Wert des TTL-Zählers. Wird der gewünschte Dienst mit diesem TTL-Zähler nicht gefunden, wird der TTL-Zähler erhöht und die Suchanfrage erneut gesendet.

Iterative Deepening

Das Iterative Deepening (u.a. [BY02], [Sos04]) ist eine Weiterentwicklung des Expanding Ring Algorithmus. Hier wird bereits vor Beginn der Suche eine Policy definiert, die im ganzen System einheitlich ist. Diese Policy besteht aus einer festen Anzahl vordefinierter Suchtiefen $\{ a_1, a_2, ..., a_n \}$. a_n stellt dabei die maximale Suchtiefe dar.

Wenn ein Peer nun eine Suchanfrage erstellt, wählt er als TTL-Zähler den kleinsten Wert der Policy a_1, und schickt die Suchanfrage an alle seine Nachbarn. Die Suchanfrage wird nun wie beim Fluten von den Peers verarbeitet (Abbildung 3.2). Im Unterschied zum Expanding Ring Algorithmus wird die Suchanfrage hier aber nicht verworfen, wenn der TTL-Zähler abgelaufen ist, sie also die Suchtiefe a_1 erreicht hat. In diesem Fall wird die Suchanfrage von dem Peer bei Suchtiefe a_1 für die Wartezeit W zwischengespeichert.

Der suchende Peer erhält in der Zwischenzeit Antworten von den einzelnen Peers, die die Suchanfrage bereits verarbeitet haben. Hat er einen passenden Dienst darunter gefunden, ist die Suche damit beendet. Er sendet keine neue Suchanfrage mehr. Die Peers, die die Suchanfrage zwischengespeichert haben, verwerfen diese nach der Wartezeit W.

Hat der suchende Peer allerdings nach Ablauf der Wartezeit W noch keinen für ihn passenden Dinest gefunden, generiert er eine neue Nachricht. Anstelle nun aber eine komplett neue Suchanfrage wie beim Expanding Ring Algorithmus zu generieren, erstellt er eine sogenannte *Resend Message*. Als TTL-Zähler für die Resend Message wählt der Peer den aktuellen Policy-Wert a_1, schickt aber den nächsten Policy-Wert, in diesem Fall a_2, mit. Zur Erkennung der Resend Message existiert eine systemweiter eindeutiger Identifier. Außerdem enthält die Resend Message die GUID der originalen Suchanfrage. Der Vorteil einer solcher Resend Message ist, dass diese Nachricht nicht erneut von allen Peers bis zur aktuellen Suchtiefe nochmals verarbeitet werden muss. Die Resend Message wird wieder an alle Nachbarn weitergeleitet, aber eben ohne Verarbeitung. Dies belastet die Peers nicht so als wenn sie die Resend Message auch noch verarbeiten müssten.

Hat diese Resend Message nun die Suchtiefe a_1 erreicht, aktivieren die Peers, welche die vorhergehende Suchanfrage zwischengespeichert haben, diese wieder und leiten sie, mit einem aktualisierten TTL-Zähler von $a_2 - a_1$ an alle ihre Nachbarn weiter. Welche Suchanfrage wieder aktiviert werden muss, erkennen die Peers an der mitgelieferten GUID der Resend Message. Die Resend Message wird ohne weitere Verarbeitung verworfen.

Dieser Prozess wird solange wiederholt, bis entweder der suchende Peer einen passenden Dienst gefunden hat, oder der maximale Policy-Wert a_n erreicht ist.

Der Vorteil dieses Verfahrens liegt darin, dass bei einer erfolgreichen Suche die Netz- und Peerauslastung wesentlich geringer ausfällt. Einen Nachteil ergibt sich bei erfolgloser Suche. Dann ist die Zeitdauer, die für eine komplette Suche aufgewendet werden muss, höher als bei dem ursprünglichen Verfahren. Dies liegt daran, dass erst W gewartet werden muss, bis ein neuer Suchdurchlauf mit größerer Suchtiefe gestartet werden kann.

Beide vorgestellten Weiterentwicklungen des Flutens, Expanding Ring und Iterative Deepening, versuchen eines der Probleme, Wahl eines geeigneten TTL-Zählers, zu lösen, indem sie schrittweise diesen Zähler erhöhen. Allerdings werden auch hier keine der Eigenschaften eines skalierbaren Suchalgorithmus erfüllt, wie z.B. nur die relevanten Peers so schnell wie möglich und mit möglichst wenig Aufwand zu besuchen. Das zweite Problem, welches beim Fluten entsteht sind Duplikate. Diese bestehen auch bei den Erweiterungen, so dass die Suchanfragen immer noch unnötig dupliziert werden. Und schließlich muss jeder Peer innerhalb des TTL-Zählers die Suchanfrage bearbeiten. Auch wenn das Iterative Deepening diesem Problem entgegen wirkt, wird auch hier das Netz unnötig belastet. Die Suche terminiert immer noch statisch nach einer gewissen Anzahl Hops – nach dem Ablauf des TTL-Zählers.

3.1.2.2 Random Walk

Während bei den vorangegangenen Broadcaststrategien das Prinizip der Breitensuche angewendet wurde, liegt dem Random Walk ([QL02], [CG06]) die Tiefensuche

zugrunde. Dabei wird eine Suchanfrage nicht an alle benachbarten Peers versendet, sondern nur an einen zufällig ausgewählten benachbarten Peer.

Um zu vermeiden, dass dieser Peer dieselbe Suchanfrage nochmals erhält, wird in der Suchanfrage der Pfad, den sie bis dahin genommen hat, verzeichnet. Dadurch können auch mögliche Schleifendurchläufe verhindert werden.

Desweiteren speichern die Peers Informationen über die Dienste, welche ihre Nachbarn ersten und zweiten Grades zur Verfügung stellen. Erhält ein Peer nun so eine Suchanfrage, schaut er zuerst lokal bei sich, ob er oder einer seiner nächsten Nachbarn den gesuchten Dienst besitzt. Erst wenn er nicht lokal fündig geworden ist, leitet er die Suchanfrage an einen wieder zufällig ausgewählten Nachbarn weiter. Dieses Weiterleiten wird solange wiederholt, bis der gesuchte Dienst gefunden wurde oder der TTL-Zähler der Suchanfrage abgelaufen ist.

Zur Verhinderung, dass eine Suchanfrage endlos im Netz herumgeschickt wird, gibt es neben dem TTL-Zähler einer Suchanfrage noch eine Variante, die *Checking* genannt wird. Diese beinhaltet, dass die Suchanfrage regelmäßig überprüft, ob die Suchanfrage schon erfolgreich war. Dazu sendet sie nach jedem Hop, bevor die Nachricht an den nächsten Hop weitergeleitet wird, eine Checking Message an den Ursprungspeer. Darin fragt sie an, ob dieser Peer den gesuchten Dienst schon gefunden hat. Ist dies nicht der Fall, wird die Suchanfrage weitergeleitet. Zur Absicherung gibt es hier zusätzlich noch einen TTL-Zähler mit einem hohen Wert für den Fall, dass die Suche überhaupt nicht erfolgreich sein sollte.

Eine Weiterentwicklung des Random Walk wird als k-Random Walk bezeichnet und bedeutet, dass mehr als nur eine Suchanfrage weitergeleitet werden – nämlich k Stück. Sie dient dazu die Verzögerung, die beim herkömmlichen Random Walk durch die Versendung an nur einen Peer auftritt, zu verringern. Man muss dabei aber auch bedenken, dass dadurch die Netzauslastung wieder ansteigt. Bei schlechter Wahl von k, kann dies die gleiche Auslastung bedeuten wie beim herkömmlichen Fluten-Algorithmus.

In [QL02] wurden zu dieser Variante Analysen durchgeführt. Dabei konnte die Annahme gemacht werden, dass k Suchanfragen nach T Hops etwa die gleiche Anzahl an Peers erreicht haben, wie einen Suchanfrage nach $k * T$ Hops.

3.1.2.3 Local Indices

B.Yang und H. Garcia-Molina präsentieren in [BY02] einen Suchalgorithmus, welcher dem Random Walk sehr ähnlich ist. Der Hauptunterschied besteht darin, dass ein Peer eine Suchanfrage an alle seine Nachbarn weiterleitet, also Broadcast. Außerdem verwaltet ein Peer einen lokalen Index über alle zur Verfügung gestellten Dienste der Peers in einem Umkreis von r Hops. Dabei wird r als Indexradius bezeichnet und von einer systemweiten Policy festgelegt. Diese Policy enthält außer dem Radius r noch die Information $P = \{a, b\}$, die die Peers angibt, die die Suchanfrage bearbeiten.

Ein praktisches Beispiel wäre $P = \{2, 6\}$ und $r = 2$. D.h. der Peer hat einen lokalen Index seiner benachbarten Peers bis zweiten Grades. Desweiteren leiten Peer 1 und 3 bis 5 die Suchanfragen nur weiter, während Peer 2 und 6 die Suchanfragen auch bearbeiten werden. Dabei ist Peer b (hier also Peer 6) der letzte Peer dieser Suchanfrage und folglich wird sie nach seiner Verarbeitung auch verworfen.

Bei der Bearbeitung durchsucht ein Peer dabei seinen lokalen Speicher, ob er oder einer der benachbarten Peers den gesuchten Dienst zur Verfügung stellt. Bei einem positiven Ergebnis wird dies dem suchenden Peer mitgeteilt.

Auch hier muss beim Random Walk, für die Verwaltung der Indizes zusätzlicher Aufwand betrieben werden. Tritt ein Peer einem Netzwerk bei, muss eine sogenannte „Join-Message“ mit einem TTL von r versendet werden. Somit erhalten alle Peers, die sich im Umkreis von r Hops befinden die aktuellen Informationen über diesen Peer. Jeder Peer, der eine solche Join-Message erhält, sendet seinerseits eine Join-Message mit seinen Diensten direkt an den neu beigetretenen Peer.

Verlässt ein Peer das Netzwerk geregelt, d.h. mit Abmeldung, dann kann er dies mit einer Leave-Message mit TTL $= r$ tun. Somit können die Peers, die ihn in ihren lokalen Indizes haben, löschen. Bei einem ungeregelten Verlassen des Netzwerks, d.h. der Peer fällt plötzlich aus, werden die bei den Peers gespeicherten Indizes nach einer Wartezeit automatisch gelöscht.

Als letzte Möglichkeit, die lokalen Indizes zu aktualisieren, kann ein Peer, der seine Dienste erweitert bzw. aktualisiert hat, eine Update-Message wieder mit TTL $= r$ versenden. Somit werden seine nächsten Nachbarn über neue Dienste von ihm informiert.

Ein Vorteil dieses Algorithums ist, dass die allgemeine Netzauslastung dadurch reduziert wird, dass nicht jeder Peer die Suchanfragen bearbeiten muss. Somit können leistungsschwache Peers absichtlich übergangen werden. Ein Nachteil dieses Algorithmus ist, dass durch die lokalen Indizes mehr Speicher aufgewendet werden muss und dies für kleinere Geräte zum Problem werden könnte. Und durch die Join-/Leave- und Update-Messages zur Aktualisierung der lokalen Indizes entsteht für die Peers auch ein erhöhter Aufwand.

3.1.3 Gezieltes Weiterleiten von Suchanfragen

Neben Broadcaststrategien, gibt es noch den Bereich des gezielten Weiterleitens einer Suchanfrage. In diesen Bereich gehören u.a. Verfahren wie das Directed Breadth-First Search (DBFS) oder auch die Routing Indizes.
Der hohe Datenverkehr, der bei den vorangegangenen Suchalgorithmen aufkommt, soll bei diesen Verfahren reduziert werden. Um dies zu erreichen wird die Suchanfrage nicht an alle benachbarten Peers weitergeleitet, sondern nur an eine bestimmte Teilmenge (Multicast). Um hier eine Auswahl treffen zu können, werden z.B. beim DBFS Kriterien festgelegt, nach denen die Anzahl und die Peers selber bestimmt werden. Beispiele für solche Kriterien könnten eine kurze Warteschlange oder die Anzahl der Nachrichten, die von einem benachbarten Peer durchschnittlich verschickt werden. Eine hohe Anzahl könnte für eine gute Netzanbindung und Leistungsfähigkeit des Peers stehen. Jedes Verfahren verwendet dabei andere Kriterien, um die Peers zu bestimmen, an die die Suchanfgrage weitergeleitet wird.

Nachfolgend werden einige Verfahren aus dem Bereich des gezielten Weiterleitens von Suchanfragen kurz vorgestellt. Da diese Suchalgorithmen eher auf statische als dynamische Verbindungen, wie es bei Adhoc-Netzen im Allgemeinen der Fall ist, zurückgreifen, werden sie für den hier relevanten Arbeitsbereich der Adhoc-Netze als eher ungeeignet angesehen.

Beim DBFS z.B. ist es eher unwahrscheinlich, dass die einzelnen Peers solange mit einem Netz verbunden bleiben, dass dadurch solche Kriterien, die auch auf der Historie der Kommunikation innerhalb dieses Netzes beruht, aufgestellt werden können. Ebenso scheint es eher weniger vorteilhaft zu sein, wenn Kleinstgeräte bzw. -komponenten, die z.B. in eine Kaffeemaschine eingebaut sind, Routingtabellen, wie sie bei den Routing Indizes verwendet werden, verwalten müssen.

3.1.3.1 Directed BFS

Dieses Verfahren [YGM02] soll die Antwortzeiten auf Suchanfragem minimieren. Dazu wird eine Suchanfrage nur n eine Teilmenge von benachbarten Peers gesendet. Die Auswahl, welche Peers zu dieser Teilmenge gehören, wird mit Hilfe von Statistiken von jedem Peer selbst getroffen. Diese Statistiken können z.B. aus folgenden Kriterien bestehen:

- Menge an gesendeten Suchergebnissen für vorherige Suchanfragen.
- Menge der Nachrichten, die von einem Peer durchschnittlich versendet werden. Eine hohe Zahl könnte für eine gute Netzanbindung und Leistungsfähigkeit des Peers stehen.
- Die Größe der Warteschlange eines Peers. Ist die Warteschlange lang, könnte der Peer z.B. ausgelastet sein und bräuchte folglich mehr Zeit, um die Suchanfragen zu bearbeiten.

Um solche Statistiken erstellen zu können, muss jeder Peer die relevanten Daten über seine benachbarten Peers sammeln.

Ein Vorteil dieses Verfahrens ist eine geringere Netzbelastung.
Nachteile sind u.a. dass seltene Dienste eventuell nur sehr schwer gefunden werden, wenn überhaupt. Zudem gestaltet sich das Durchsuchen des kompletten Netzes als sehr schwierig.

3.1.3.2 Routing Indizes

Im Unterschied zu den lokalen Indizes und dem Random Walk, wo die Peers Informationen über die zur Verfügung gestellten Dienste ihrer nächsten Nachbarn lokal speichern, werden bei diesem Mechanismus [AC02] nur Verweise auf die Peers gesichert, bei denen die Suche am geeignetesten weitergeführt werden kann. In der Routingtabelle steht demzufolge, wieviele Dienste über diesen Peer prinzipiell erreicht werden können. D.h. wenn ein Peer eine Suchanfrage erhält, prüft er seine lokale Routingtabelle, ob er diesen Dienst selbst besitzt oder an welchen erfolgversprechensten Peer die Nachricht als nächstes geschickt werden kann. Somit werden die Nachrichten nur an einen nächsten Nachbarn versendet. Dies wird solange weitergeführt, bis der gewünschte Dienst lokalisiert wurde.

Um die lokalen Routingtabellen aufzubauen, bedarf es der gegenseitigen Kommunikation mit den benachbarten Peers. Wird erstmalig eine Verbindung zwischen zwei Peers aufgebaut, tauschen die Peers untereinander Informationen über die von den jeweiligen Peer erreichbaren Dienste aus. Die neuen Informationen werden dann an die anderen bereits bekannten benachbarten Peers der beiden weitergeleitet. Dies

wird aber nur gleich anschließend getan, falls es sich um umfangreiche bzw. größere Änderungen in den lokalen Routingtabellen handelt. So kann unnötiger Nachrichtenaustausch und somit unnötige Netzbelastung vermieden werden.

A. Crespo und H. Garcia-Molina haben in [AC02] Untersuchungen zu diesem Thema durchgeführt.

Der Vorteil dieses Verfahrens liegt darin, dass für diese Routingtabellen weniger Speicherplatz aufgewendet werden muss, als bei den lokalen Indizes. Außerdem ist die Netzwerkbelastung geringer, das die Nachrichten nicht an alle Nachbarn versendet werden.

Zur Bestimmung der Peers, an die die Suchanfrage weitergeleitet wird, existieren unterschiedliche Ansätze. Zwei von diesen Ansätzen sollen nachfolgend kurz beschrieben werden.

Compound Routing Indices (CRI)

Bei diesem Verfahren [AC02] werden die Dienste, die in einem Netzwerk prinzipiell zur Verfügung gestellt werden können, in verschiedene Kategorien eingeteilt. Beispiele für Kategorien könnten Drucker, Laserdrucker und Farbdrucker sein. Ein Peer speichert nun in seiner lokalen Routingtabelle die Anzahl der Dienste, die er pro Kategorie selbst zur Verfügung stellt als auch die, die über einen bestimmten benachbarten Peer erreicht werden können.

Die Berechnung, wie geeignet ein benachbarter Peer für die Weiterleitung der Suchanfrage an ihn ist, wird mit Hilfe einer Abschätzfunktion durchgeführt.

Dieses Verfahren bringt noch den weiteren Vorteil, dass Suchanfragen erst gar nicht in Bereiche des Netzes weitergeleitet werden, in denen Dienste einer bestimmten Kategorie nur sehr wenig oder gar nicht vorhanden sind.

Hop-Count Routing Indizes (HCRI)

Hier [AC02] wird, zusätzlich zu der Anzahl der über einen bestimmten Peer erreichbaren Dienste, die Anzahl der Hops angegeben, die für das Erreichen der Dienste benötigt werden. Dabei wird in der Routingtabelle zusätzlich nach Anzahl der Hops sortiert. Beispiel: Nach 1 Hop über Peer X können 2 Druckerdienste erreicht werden, während nach 2 Hops weitere 3 Druckerdienste zur Verfügung stehen würden.

Der Vorteil, der sich durch diese Methode ergibt ist, dass nun für die Auswahl eines geeigneten Peer, an den die Suchanfrage weitergeleitet wird, auch die Anzahl der weiteren Schritte, die benötigt werden, um einen bestimmten Dienst zu erreichen, mit einbezogen werden. Dies geht in die Berechnung des besten Nachbarn als zusätzliches Qualitätskriterium ein. Der offentsichtliche Nachteil dieser Methode ist allerdings, dass nun zusätzlicher Speicherbedarf und Aufwand für das Anlegen und Aktualisieren dieser Routingtabellen aufgewendet werden muss.

3.1.3.3 Verteilte Hashtabelle (VHT)

Eine weitere Möglichkeit des gezielten Weiterleitens einer Nachricht stellt die verteilte Hashtabelle [IS03] dar. Eine verteilte Hashtabelle (VHT) (engl. Distributed Hashtable (DHT)) wird zu den strukturierten Peer-to-Peer-Systemen gezählt. Sie

kann als System zur Speicherung von verschiedenen Diensten unter Identifikationsschlüsseln angesehen werden. D.h. die Beschreibungen der einzelnen, von einem Peer zur Verfügung gestellten Dienste, werden mit Hilfe einer Hashfunktion in numerische Zahlen umgewandelt.
Das Ziel dabei ist, mit möglichst geringem Aufwand Dienste in einem System zu finden und dabei die Peers möglichst gleichmäßig zu beanspruchen. Außerdem soll ein stabiles und konsistentes System zur Verfügung gestellt werden.
Eine DHT ermöglicht durch Zuordnung von Hashwert zu Peer eine effiziente, zielgerichtete Suche. Die Dienste sind dabei möglichst gleichmäßig auf die Peers des Systems verteilt. Jedem Peer ist weiterhin eine eindeutige ID zu Identifikation zugeordnet.

Durch die Zuordnung der Hashwerte zu den einzelnen Diensten lässt sich jeder Dienst eindeutig indentifizieren. Der Bereich der Hashwerte wird auch als Schlüsselraum bezeichnet.
Dieser Schlüsselraum wird gleichmäßig auf die vorhandenen Peers verteilt. Da meistens mehr Schlüssel, also Dienste in einem System exisitieren als Peers, erhält jeder Peer mehrere dieser Schlüssel. Welcher Peer dabei welche Schlüssel erhält, regelt das jeweilige Protokoll. Prinizipiell ist es jedoch sehr hilfreich, demjenigen Peer solche Schlüsselwerte zuzuordnen, dessen ID in dem Wertebereich liegt.
Die Aufgabe eines Peers besteht nun darin, einen bestimmten Bereich dieses Schlüsselraums zu verwalten. Dazu speichern die Peers diejenigen Dienste, deren Schlüssel in ihren zuständigen Bereich passen. Eine Alternative ist die Speicherung eines Verweises auf die Adresse des Peers, der die Dienste zur Verfügung stellt.

Für die Suche verwendet jedes DHT-Protokoll seinen eigenen Algorithmus. Allgemein betrachtet haben aber alle Protokolle eine Grundstrategie, die nachfolgend kurz erläutert wird.

Für die Dienstesuche wird eine Funktion *lookup(key)* verwendet. Dabei soll unter Angabe des Schlüssels eines Dienstes der zugehörige Knoten, der diesen Dienst zur Verfügung stellt, gefunden werden. Für die effiziente Weiterleitung der Suchanfragen verwaltet jeder Peer eine Routingtabelle, in der die IDs der benachbarten Peers aufgelistet sind. Wer dabei die nächsten Nachbarn sind, wird hier nicht, wie in einem unstrukturierten System dem Zufall überlassen, sondern von der Struktur des verwendeten Protokolls festgelegt. Chord [IS01] z.B. verwendet eine Ringstruktur, wie in Abbildung 3.4 zu sehen ist.
Mit Hilfe dieser Routingtabellen können die Peers nun Suchanfragen zu denjenigen Peers weiterleiten, deren ID näher an dem Schlüsselwert liegt. Dies wird solange wiederholt, bis der gesuchte Dienst gefunden wurde.

Abbildung 3.4 macht das Prinzip der Dienstesuche in DHTs am Beispiel von Chord deutlich.

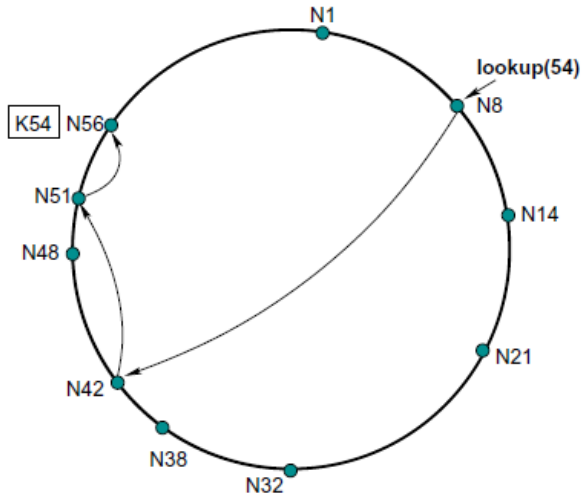

Abbildung 3.4: Lookup bei Chord [IS01].

Verteilte Hashtabellen besitzen nach [wwwe] folgende Eigenschaften:

Selbstorganisation: Durch möglichen Ausfall, Beitritt und Austritt von Peers wird eine ständige Umorganisation des System notwendig. Dies erledigen die Peers automatisch. Somit wird eine manuelle Konfiguration vermieden.

Skalierbarkeit: Das System sollte in der Lage sein, auch mit einer großen Anzahl von Knoten funktionsfähig zu bleiben. Dies ist mit diesem Ansatz im Gegensatz zu den zentralisierten Ansätzen möglich.

Lastenverteilung: Durch die Aufteilung der Schlüssel auf alle Peers werden die Peers bei einer Suche gleichmäßig belastet. Es gibt keinen Peer, welcher zentral den kompletten Verwaltungsaufwand übernimmt.

Fehlertoleranz: Das System sollte zuverlässig funktionieren, auch wenn Knoten ausfallen oder das System verlassen. Dies ist durch die Selbstorganisation gewährleistet.

Sicherheit/Robustheit: Das System sollte „korrekt“ funktionieren können, auch wenn ein Teil der Peers oder Angreifer von „aussen“ versuchen, das System zu stören. Dies wird maßgeblich durch die Verteilung der Ressourcen gewährleistet.

Systeme, die auf dem Ansatz der verteilten Hashtabellen aufbauen, sind u.a.:

- CAN (Content Addressable Network) [SR01]
- Chord [IS01]
- Tapestry [IS01]
- Pastry [AR01]

Ein Vorteil bei der Verwendung einer DHT liegt darin, dass eine einfache Suche praktiziert werden kann. Dies ist darin begründet, dass jeder Schlüsselwert und somit jeder Dienst einem Peer zugewiesen werden kann. Dadurch kann eine zielgerichtete Suche erreicht werden.

Ein Nachteil ist die Komplexität der Hashfunktionen.

3.2 Protokolle für das Service Discovery

Nachfolgend soll auf das Thema Protokolle für das Service Discovery näher eingegangen werden. Die einzelnen Protokolle werden in Kapitel 4.2 genauer beschrieben.

Die Hauptaufgabe, die Service Discovery Protokolle bei ihrem Einsatz erfüllen sollen, ist es Nutzern, Geräten und auch Anwendungen in einem Netzwerk das Finden und Auswählen von geeigneten Diensten in diesem Netzwerk zu erleichtern. Das Ganze soll dabei möglichst ohne bzw. mit minimalem Vorwissen möglich sein. Der Idealfall wäre, dies ohne vorhergehende Konfiguration zu ermöglichen. Um dies realisieren zu können, bedarf es geeigneter Protokolle.

Service Discovery Protokolle stellen folglich einen Mechanismus für das automatische Finden von Diensten in einem System zur Verfügung. Sie sollten also Hilfestellungen für :

- Suche nach einem Dienst
- Auswahl eines geeigneten Dienstes
- Benutzung des Dienstes und auch
- Anbieten des Dienstes

geben. Desweiteren sollten sie eine geeignete Beschreibungsmöglichkeit für die Dienste und die Schnittstellen für den Zugriff auf diese Dienste zur Verfügung stellen. Diese Beschreibung sollte systemweit einheitlich und eindeutig sein, damit die Dienste bei einer Suche identifiziert und zugeordnet werden können.

Anforderungen an Service Discovery Protokolle

Friday at al. haben in [AF04] einige grundlegende Anforderungen an Service Discovery Protokolle vorgestellt.

Interoperabilität

Es gibt viele verschiedene Geräte, die für Peer-to-Peer-Systeme verwendet werden können. Für diese Geräte gibt es auch die unterschiedlichste Systemsoftware. Somit sollte ein Protokoll unabhängig von darunterliegender Hardware und Systemsoftware sein, um auf den verschiedenen Geräten eingesetzt werden zu können.
Ebenso werden von System zu System die Repräsentation von Geräten, Diensten und Gerätestati unterschiedlich gehandhabt. Es ist daher auch wichtig möglichst einheitliche Mechanismen zur Dienstelokalisierung, -beschreibung, -verwendung und -bekanntmachung zu realisieren, welche unabhängig von verwendeter Hard- und Software sind.

Skalierbarkeit

Die Skalierbarkeit stellt eines der wichtigsten Kriterien im Service Discovery dar. In kleinen Netzen stellen die Anzahl der veschickten Nachrichten oft keine große Rolle, da genügend Kapazitäten vorhanden sind, um sie zu verarbeiten. Werden die Systeme allerdings größer, d.h. nimmt die Anzahl der Peers bis zu einigen tausend und mehr zu, skalieren viele der exisiterenden Protokolle nicht mehr. Das System wird von Nachrichten geflutet und die einzelnen Peers können diese Anfragen kaum noch verarbeiten. Schlimmstenfalls fallen einige Peers oder sogar das ganze System zusammen. Folglich ist es wichtig, Protokolle zu entwickeln, die auch in großen Systemen noch funktionsfähig bleiben. Dies könnte z.B. durch eine geeignete Suchstrategie realisiert werden.

Lokalitäts-basierte Services

Es gibt Dienste, die sich auf einen bestimmten Ort beziehen, und solche, die ortsunabhängig sind. Wenn man z.B. einen Drucker in einem bestimmten Gebäudeabschnitt sucht, dann ist dieser Drucker ortsabhängig. Wenn man jedoch Informationen über etwas bestimmtes benötigt, im Sinne einer Wissendatenbank z.B., dann handelt es sich eher um einen ortsunabhängigen Dienst. Denn hier spielt der Standort keine Rolle.
Folglich sollten Protokolle sicherstellen, dass Dienste sowohl einen physischen als auch einen logischen Ort besitzen können.

Historie

Dieser Aspekt hat sowohl mit dem wiederholten Benutzen bzw. mit der Anzahl der Anfragen nach bestimmten Diensten, als auch mit der Anzahl der verschickten Anfragen und Antworten von Peers zu tun. Solche Informationen werden z.B. bei Suchstrategien wie den Routing Indizes verwendet. Für administrierte Systeme können diese Informationen Hinweise liefern, wo und wann in einem System z.B. Engpässe oder Ausfälle auftreten können.

Status

Ebenfalls nicht zu vernachlässigen ist der Status von Geräten, also Peers, und zur Verfügung gestellte Dienste im System. So spielt dieser Apekt z.B. bei Protokollen, welche die Suchstrategien Random Walk oder die lokalen Indizes verwenden, eine große Rolle. Hier sollten die lokalen Indizes immer möglichst aktuell sein. Aber auch bei den Routing Indizes ist dieser Aspekt wichtig. Denn wenn bestimmte Dienste oder Peers, die diese Dienste zur Verfügung stellen, in einem Bereich des Systems nicht mehr verfügbar sind, kann dadurch Zeit gespart werden. Denn dann werden in diesen Bereich erst gar keine Suchanfragen der nicht mehr verfügbaren Dienste geschickt. So kann auch eine unnötige Netzbelastung in diesem Bereich vermieden werden.

Sicherheit, Authentifikation und Zugangkontrolle

Da in dieser Arbeit dieser Aspekt nicht näher betrachtet wurde, wird dieser Abschnitt übergangen.

Meta-Daten

Im Allgemeinen obliegt die Beschreibung der einzelnen Dienste den jeweiligen Protokollen, sofern dies zu ihrem Aufgabengebiet gehört. Diese Beschreibungen sind jedoch nicht immer für einen Nutzer verständlich. Sie verwenden gerne Metainformationen, also Umschreibungen für die Bezeichnung von Dingen. Daher wird in [AF04] vorgeschlagen, einen zusätzliches Verzeichnis bei jedem Peer zur Verfügung zu stellen, um die wenig sagenden Dienstbeschreibungen der Systeme mit Metainformationen dieser zu verknüpfen.

Komplexe Service-Anfragen

Neben einer Abfrage nach einem Dienst mit einer bestimmten Funktionalität, also einem bestimmten Typ, kann es auch solche Anfragen geben, die nicht nur einen Diensttyp, sondern mehrere ansprechen. Einen wichtige Aufgabe eines Protkolls wäre es nun, solche Anfragen, die über mehrere Diensttypen gehen, korrekt aufzulösen. Das ganze soll dabei möglichst automatisch und ohne Eingreifen der Peers möglich sein. Damit soll erreicht werden, dass die Komplexität von den Peers ferngehalten wird.

Drahtloser Zugang

Schließlich muss ein Protokoll auch im drahtlosen Bereich funktionsfähig sein. Dies liegt u.a. darin begründet, dass immer mehr Geräte mobil sind. Dieser Trend wird sich auch noch weiter fortsetzen. Ein Protokoll muss folglich in dynamischen Systemen und mit von Zeit zu Zeit nicht erreichbaren Peers umgehen können.

4. Evaluation

In diesem Kapitel wendet sich die Arbeit nun wieder dem Service Discovery des Ambient Technology zu.

In Abschnitt 4.1 wird geklärt, warum einige Protokolle für den Adhoc-Bereich nicht geeignet sind. In Abschnitt 4.2 wird aus der Fülle an Service Discovery Protokollen eine kleine Anzahl herausgesucht, die sich theoretisch gesehen am ehesten für den Einsatz in diesem Bereich eignen. Diese werden dann genauer betrachtet, wobei der Fokus auf die Suche gelegt wird. Für nähere Informationen sei auf weiterführende Literatur verwiesen.

4.1 Einsatzbarkeit verschiedener Service Discovery Protokolle

Wie bereits in Kapitel 3.1 genauer erklärt wurde, gibt es im Service Discovery unterschiedliche Suchstrategien. Bei der näheren Untersuchung der gefundenen Service Discovery Protokolle wurde festgestellt, dass dementsprechend auch nicht alle Protokolle auf dem „reinen“ Peer-to-Peer-Ansatz basieren. Vielmehr können sie in die unterschiedlichen Ansätze der Suchstrategien eingeordnet werden. Dabei kann man auch erkennen, dass die Entwicklung der Service Discovery Protokolle sehr stark mit der Entwicklung der Netzwerktechnologien zusammenhängt. Da in letzter Zeit der Fokus immer mehr auf die völlig verteilten Netze, die in MANETs und auch im Ambient Technology Bereich zum Einsatz kommen, gelegt wird, wird immer mehr Wert auf Protokolle gelegt, die diese Anforderungen erfüllen können. Deswegen sind die Anstrengungen in der Entwicklung neuer geeigneter Protokolle gestiegen.

Als ungeeignet gelten solche Protokolle, die auf einem Infrastrukturnetz als darunterliegende Netzwerktechnologie aufbauen. Bei der Untersuchung wurde weiter deutlich, dass sich die meisten der bereits existierenden Service Discovery Protokolle nicht für völlig verteilte Netze eigenen. Viele der bekanntesten Protokolle wie das Service Location Protokoll (SLP) [wwwd] und Jini [Wal99], können für diesen Bereich genannt werden. Diese Protokolle erlauben keine große Dynamik der Netze. Ihre Peers übernehmen oft komplexere Aufgaben, was in Adhoc-Netzen nur bedingt

einsatzfähig ist. Sie verwenden als Suchstrategie meistens den zentralisierten Ansatz. Dies ist für große Netze und die Skalierbarkeit vielleicht von Vorteil, jedoch in mobilen Umgebungen wie den Adhoc-Netzen, die für das Ambient Technology benötigt werden, nicht realisierbar.

In Abbildung 4.1 sind Folgen der Verwendung des zentralen Ansatzes in Adhoc-Netzen dargestellt.

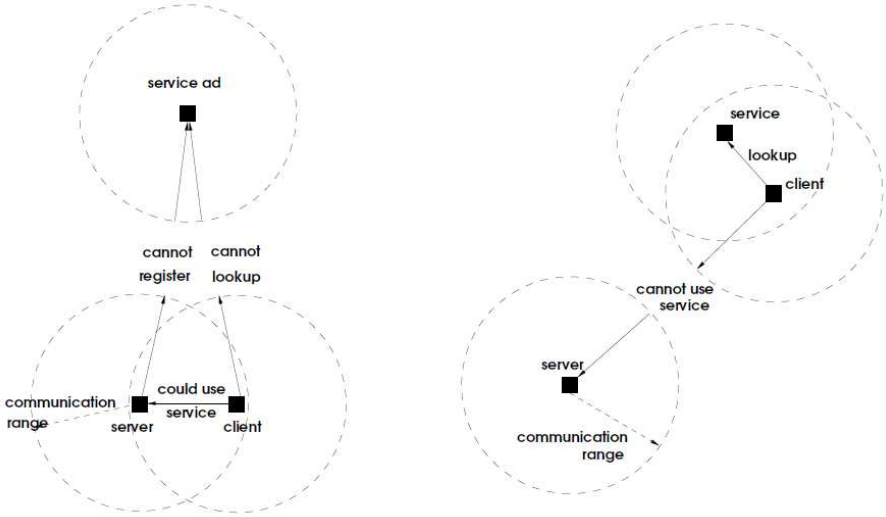

Abbildung 4.1: Lookup-Probleme in einem Adhoc-Netz am Beispiel von Jini [RH03].

Um diese Probleme zu lösen, wurden in den letzten Jahren verstärkt neue, für den Adhoc-Bereich geeignetere Protokolle entwickelt. Diese zeichnen sich vor allem dadurch aus, das sie unabhängig von der darunterliegenden Netzwerktechnologie sind und auf eine zentrale Instanz verzichten. Diese werden nachfolgend aufgezeigt und näher erläutert. Solche Protokolle könnten also prinizipiell auch im drahtlosen Bereich eingesetzt werden. Dabei hängt es folglich nur noch davon ab, wie das Service Discovery realisiert ist. Nachfolgend werden einige Protokolle vorgestellt und näher erläutert.
Als Ausnahme wird Bluetooth SDP [Gry] gezählt, da es bereits auf einer drahtlosen Technologie basiert und somit prinzipiell im Ambient Technology Bereich angewendet werden kann.

4.2 Mögliche Service Discovery Protokolle für Ambient Technology Anwendungen

Nachfolgend werden nun einige der gefundenen Service Discovery Protokolle vorgestellt. Es wurden eine große Anzahl an Protokollen identifiziert, die sich um das Thema Service Discovery kümmern. Die nachfolgend kurz beschriebenen Protokolle stellen nur einen Auszug aus den mittlerweile zahlreich vorhandenen Ansätzen dar. Da für diese Arbeit hauptsächlich Protokolle für vollständig verteilte Dienste im Bereich der Ambient Technology von Interesse sind, wird auch nur für diesen Bereich in Frage kommende Protokolle näher eingegangen.

Bluetooth SDP

Bluetooth ist eine Adhoc-Netzwerk Technologie für Peers, die sich in Kommunikationsreichweite voneinander befinden. Die Kommunikation bei Bluetooth erfolgt direkt zwischen zwei benachbarten Peers. Normalerweise werden sogenannte Piconetze, aus zwei bis max. sieben Peers gebildet. Werden größere Netzwerke gebraucht, können sogenannte Scatternetze, die aus mehreren Piconetzen bestehen, aufgebaut werden.

Ein wichtiger Bestandteil der Bluetooth-Spezifikation ist das sogenannte Service Discovery Protocol (SDP) [Gry]. Dieses Protokoll ermöglicht das Erkennen und Abfragen von Informationen über die von anderen Bluetooth-Geräten angebotenen Dienste. Während diese Dienste in traditionellen, statischen Netzwerken vom Administrator konfiguriert werden müssen, besteht bei den Bluetooth-Geräten der Bedarf, die angebotenen Dienste dynamisch zu erkennen. Das Bluetooth SDP verwendet dabei so genannte Universal Unique Identifiers (UUID), die einen Dienst eindeutig charakterisieren. Sie enthalten sowohl Informationen über die Klasse des Dienstes (z.B. Drucken, E-Mail-Server, LAN) als auch spezifische Attribute. Durch das Bereitstellen der Dienst-Informationen werden zwei grundsätzliche Formen des Service Discovery ermöglicht. Zum einen kann ein Client durch ein „Service Browsing“ eine Anfrage nach vorhandenen Diensten einer gewissen Klasse aussenden und darauf unter den erhaltenen Angeboten einen Dienst auswählen. Andererseits kann ein Client, falls dieser bereits die UUID des gewünschten Services kennt, das Vorhandensein des Angebots überprüfen (Searching).

Die Informationen über einen bestimmten Service werden in einem Service Record zur Verfügung gestellt. Ein solcher Service Record enthält dabei ein oder mehrere Service Attribute, die den Dienst näher spezifizieren. Es gibt einige vordefinierte Attribute, wie z.B. ServiceID, ServiceInfoTimeToLive. Darüber hinaus hat jeder Dienstanbieter (SDP Server) die Möglichkeit eigene Attribute zu definieren. Ein Service Attribut wiederum besteht aus einer ID und einem Wert. Die ID dient zur eindeutigen Identifikation eines Attributes innerhalb eines Service Records. Der Wert ist ein Datenelement und wird von der Attribut ID und der Serviceklasse des Service Records spezifiziert.

Jeder Service gehört einer bestimmten Serviceklasse, die die Service Attribute definiert, an. Jede dieser Serviceklassen besitzt einen eindeutigen Identifikator, den UUID (universal unique identifier). Von ihm wird garantiert, dass jeder UUID stets sowohl räumlich wie auch zeitlich eindeutig ist.

Um einen Dienst zu finden, sendet ein Peer ein ServiceSearchRequest an jeden anderen Peer des Netzes. Er fragt dabei jeden Peer einzeln, also via Unicast, an. Diese Peers wiederum antworten alle mit einer ServiceSearchResponse, die entweder positiv oder negativ ausfallen kann.

Allia (Alliance Based Service Discovery)

Allia [OR02] ist ein agenten-basiertes Service Discovery Protokoll für Peer-to-Peer Systeme in Adhoc-Umgebungen. Es verzichtet dabei vollständig auf zentrale Instanzen. Stattdessen macht es sich das Cachen von Service Advertisements in einem Peer zu nutzen. Das Hauptziel dieses Protokolls soll dabei sein, es Peers zu ermöglichen, auf Dienste in ihrer näheren Umgebung besser zuzgreifen.

Das Protokoll soll sowohl gerätespezifische Eigenschaften, wie z.B. Speicherkapazität oder Batterieladezustand, als auch Benutzerpräferenzen mit einbeziehen können. Diese Eigenschaften werden mit Hilfe von sogenannten Policies widergespiegelt. Eine Policy legt dabei lokal Regeln für jeden Peer fest. So bestimmt eine Policy z.B., wie ein Peer sich gegenüber anderen in seiner Nähe befindlichen Peers präsentiert. Desweiteren legt sie die Häufigkeit fest in der Service Advertisements versendet werden, welche Advertisements von anderen Peers gecached und welche verworfen werden. Man kann also sagen, dass eine Policy sämtliche Aktionen eines Peers festlegt. Dabei besitzt jeder Peer seine eigene Policy.

Jeder Peer, der im Netz Dienste zur Verfügung stellen möchte, schickt Service Advertisements mittels Fluten (Broadcast) [Kapitel 3.1.2] an alle Peers in seiner näheren Umgebung. Jeder Peer, der ein solches Service Advertisement erhält, entscheidet gemäß seiner Policy, ob das Advertisement in seinen Cache mitaufgenommen oder verworfen wird. Durch dieses Cachen bilden sich sogenannte Allianzen. Da jeder Peer unterschiedliche Service Advertisements *cached*, erstellt jeder Peer seine eigene Allianz. Dadurch dass die Allianzen im System nicht bekannt gemacht werden, kennt jeder Peer nur die Mitglieder seiner Allianz und weiß also nicht, in welchen Allianzen er überall Mitglied ist. Eine Allianz besteht folglich aus den lokal gespeicherten Service Advertisements der anderen Peers und werden auch durch die Policy bestimmt.

Fragt ein Peer einen bestimmten Dienst an, durchsucht er zuerst seine Allianz im lokalen Cache. Ist dort ein solcher Dienst nicht vorhanden, hat er die Möglichkeit durch Broadcast oder Multicast (Nachricht geht nur an eine Untermenge der benachbarten Peers) einen Service Request an die Peers in seiner Umgebung und somit an weitere Allianzen zu senden. Dieser Service Request kann – je nach Policy – mit einem *MaxHop*-Feld ausgestattet sein, welches die Aufgabe eines TTL-Zählers hat.

Eine Beschreibungsmöglichkeit für Dienste wurde in Allia nicht realisiert. Es ist somit unabhängig von jeglichen Beschreibungsmechanismen und -protokollen. Folglich wurde auch kein Mechanismus für das Prüfen von Übereinstimmung der Dienstbeschreibungen während des Service Discovery Prozesses realisiert. Es wird folglich den Nutzern überlassen, welche Dienstbeschreibung verwendet werden soll. Daraus ergibt sich die Möglichkeit evtl. auch mit anderen Protokollen gleichzeitg zu arbeiten und deren Dienstbeschreibung zu verwenden. Denkbar wäre z.B. auch eine Dienstbeschreibung in XML, das bei einigen Protokolle bereits zum Einsatz kommt.

Konark

Konark (z.B. [SH03], [Bie06]) ist ein Service Discovery und Delivery (Übergabe) Protokoll, das speziell für geräteunabhängige Dienste in Adhoc-Netzen entwickelt wurde. Um Konark verwenden zu können, muss jeder Peer allerdings dazu in der Lage sein, einen „micro-HTTP-Server“ zu verwenden. Dies ist ein HTTP-Server mit eingeschränkter Funktionalität, speziell für eingebettete Systeme. Bei Konark wird dieser Server für die Behandlung von Dienstanfragen der Peers verwendet und soll die Peers beim Versenden ihrer Nachrichten unterstützen. D.h. der micro-HTTP-Server nimmt die mit Hilfe von HTTP verschickten Anfragen entgegen und gibt sie an die darüberliegende Anwendungsschicht (in diesem Fall Konark) weiter.

Konark kümmert sich nicht um die Konfiguration des Netzes und ist somit unabhängig von der darunteliegenden Netzwerktechnologie. Denoch gehen die Entwickler von

Konark in [SH03] davon aus, dass die verwendete Technologie IP-basiert ist. Dies begründen sie damit, dass viele der heutigen Betriebssysteme automatische Konfigurationsmechanismen für Netzwerke auf Basis von IP, auch im Adhoc-Bereich, besitzen. Folglich wurde für Konark eine Multicast-Gruppe im Bereich von 239.255.0.0/16 definiert.

Im Gegensatz zu Allia ist bei Konark die Dienstbeschreibung ein zentrales Thema. Hierfür wird eine XML-basierte Dienstbeschreibungssprache verwendet. Desweiteren wird als Dienstregister eine Baumstruktur, welche in Dienstklassen unterteilt ist, verwendet. In diesem Diensteregister können die Peers lokal ihre Dienste ablegen. Damit werden auch die Dienste, die ein Peer bei einem Service Discovery entdeckt oder durch ein Service Advertisement erhalten haben, verwaltet. Im oberen Bereich dieser Baumstruktur sind die Klassen generisch aufgebaut, werden aber immer genauer, je tiefer man sich im Baum zu den Blättern hin bewegt. Die Blätter stellen dabei die spezifischen Dienste dar. Desweiteren werden die Dienste nochmal zusätzlich mittels der Kennzeichnung „all“ (Wurzel des Baumes), „generic“ und „specific“ (Blätter des Baumes) näher spezifiziert und klassifiziert.

Beim Service Discovery erstellt der Peer, der einen bestimmten Dienst sucht, eine Suchanfrage, sofern er den Dienst nicht in seinem eigenen Diensteregister-Baum finden konnte. Diese Dienstanfrage enthält hier entweder einen bestimmten Pfad aus dem Dienstergister-Baum oder einen Schlüsselwort, das den Dienst spezifizieren soll. Desweiteren wird mit einer Suchanfrage eine Portnummer mitgesendet, damit die Empfänger-Peers wissen, auf welchen Port der suchende Peer eine Antwort erwartet. Anschließend wird die Anfrage an eine der spezifizierten Multicastgruppen geschickt. Jeder Peer, der diese erhält, überprüft, ob er den Pfad oder das Schlüsselwort in seinem Diensteregister-Baum findet. Kann er einen solchen anbieten, generiert er ein Service Advertisement, das an den suchenden Peer an der genannten Portnummer geschickt wird. Ein solches Service Advertisement enthält dabei den genauen Servicenamen, den Pfad aus dem Diensteregister, einen Typ, die URL an der eine Dienstebeschreibung für diesen Dienst erhältlich ist und einen TTL (Time-To-Live)-Wert.

Damit der Ursprungs-Peer diesen Dienst nutzen kann, muss er mittels der mitgelieferten URL erst auf eine genaue Dienstebeschreibung zugreifen. Dies wird in Konark Service Delivery genannt und mittels SOAP und dem mirco-HTTP-Server ausgeführt.

Schließlich gibt es auch bei Konark die Möglichkeit, dass Peers ihre Dienste im Netz anbieten (Service Advertisement). Hierbei kann ein Peer, gemäß der Diensteklassifizierung, entweder „all“ (alle) seine Dienste, einen „generic“ (generischen) Dienst, repräsentiert durch einen Pfad des Diensteregister-Baumes oder einen „specific“ (spezifischen) Dienst angeben. Über die Häufigkeit dieser Advertisements wurde keine Aussage getroffen. Man könnte jedoch annehmen, dass nach Ablauf der TTL-Werte der Dienste, diese automatisch von ihren Diensterbringern neu bekannt gemacht werden.

DEAPSpace

DEAPSpace [RH00] ist ein System, das für das Service Discovery in drahtlosen Netzwerken entwickelt wurde. Ziel dabei ist, Dienste zwischen Peers auszutauschen, die sich in Kommunikationsreichweite voneinander befinden.

Nachrichten werden in DEAPSpace mittels Broadcast verschickt. Im Unterschied zu der Suchstrategie, werden die Nachrichten aber nicht weitergeleitet, sondern die Suche endet bei dem Peer. Dies bedeutet, dass die Suche nur für Netzwerke von einer Reichweite von einem Hop gedacht ist.

In dem Protokoll existieren in erster Linie Service Advertisements, sogenannte *SAMs* (Service Advertisement Messages). Diese SAMs bestehen aus einer Liste der eigenen und aller verfügbaren Dienste aus der nächsten Umgebung eines Peers. Damit wird folglich die Weltsicht („world-view“) eines Peers verbreitet. Ein Dienst wird dabei als Service Element (SE) bezeichnet. Diese SAMs werden zu bestimmten Zeitpunkten innerhalb eines Fensters (Window) verschickt. Dabei wird pro Zeitfenster nur eine Broadcastnachricht im System versendet. Somit kann wenigstens einer temporären Überlastung des Systems vorgebeugt werden. Bevor ein Peer erneut eine SAM verschickt, wartet er eine bestimmte Zeit.
Jeder Peer, der eine SAM erhält, vergleicht diese mit seiner eigenen „world-view“. Bei keinem Unterschied und angemessenen TTL-Zeiten der Services in der „world-view“, verarbeitet der Peer die SAM nicht weiter. Er setzt seine Wartezeit neu, da aktuelle Informationen erst von einem anderen Peer verbreitet wurden.
Existieren allerdings größere Unterschiede zwischen „world-view“ und SAM, wie z.B. neue Dienste sind in der SAM verzeichnet, setzt er seine eigene SAM früher an. Dazu verkürzt er seine Wartezeit, um möglichst schnell die neuen Informationen an die anderen Peers propagieren zu können.

Benötigt ein Peer nun einen bestimmten Service, generiert er keine Suchanfrage, wie bei den meisten anderen Protokollen. Stattdessen schaut er in seiner lokalen „world-view“, ob er einen solchen Dienst verzeichnet hat. Wird er fündig, so stellt er mit Hilfe der in der Dienstebeschreibung vorhandenen Adresse des Peers eine direkte Verbindung zu diesem her. Da bei diesem Verfahren keine Suchanfragen generiert werden, wird die Netzwerklast und die Belastung der einzelnen Peers möglichst gering gehalten.

5. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, den Bereich der völlig verteilten Dienste, wie es in einer Ambient Technology Umgebung der Fall ist, zu beschreiben und wichtige Aspekte bereits existierender Service Discovery Protokolle zu skizzieren. Schließlich sollte eine Auswahl an Protokollen für den Ambient Technology Bereich näher analysiert werden.

Im Zuge dessen wurden zuerst Peer-to-Peer-Systeme untersucht. Diese sind vor allem durch Filesharing-Programme bekannt geworden, bieten jedoch für viele weitere Anwendungsmöglichkeiten eine Lösung. Es wurden drei verschiedene Strukturen identifiziert: zentrale, hybride und reine Peer-to-Peer-Systeme.

Bei der Betrachtung der Eigenschaften und Anwendungsmöglichkeiten der Ambient Technology wurde festgestellt, dass teilweise hohe Anforderungen an darin eingesetzte Peer-to-Peer-Systeme bezüglich Ressourcenauslastung und Dynamik der Geräte gestellt werden. Es wurde weiter bemerkt, dass nicht alle Peer-to-Peer-Systeme für den Einsatz in diesem Bereich geeignet sind.
Im Allgemeinen kann man sagen, dass Ambient Technology eine große Spanne an Anwendungsmöglichkeiten besitzt und dem Menschen einen nicht zu vernachlässigenden Mehrwert im alltäglichen Leben sowie eine Erleichterung bei der Erledigung verschiedenster Aufgaben bringen kann.

Aufbauend auf die Peer-to-Peer-Strukturen wurden mehrere unterschiedliche Suchstrategien für das Service Discovery gefunden. Diese Ansätze können unterteilt werden in zentraler Server, Broadcast und gezieltes Routing. Für das Service Discovery wurden eine große Anzahl an Protokollen gefunden und es wurde erläutert, warum sich nicht alle für völlig veteilte Netze eignen.

Schließlich wurden vier Protokolle – Bluetooth SDP, Allia, Konark und DEAPSpace – als geeignete Lösungen für völlig verteilte Netze ausgewählt und näher analysiert. Dabei wurde festgestellt, dass Allia einen guten Ansatz (Policies) zur Behandlung dynamischer Umgebungen, auch mit ressourcenschwachen Geräten, bietet.
Bluetooth ist eine bereits bekannte und mittlerweile weitverbreitete Technologie, die in adhoc-Bereichen (Druckerverbindungen, Kopplung von Mobiltelefonen) eingesetzt wird. Bluetooth SDP ist folglich ein durchaus geeignetes Service Discovery Protokoll, allerdings mit dem Nachteil, dass für das Netzwerk nur Bluetooth-Technologie

eingesetzt werden kann.
DEAPSpace erscheint durch die Kommunikationsreichweite von nur einem Hop etwas eingeschränkt bzgl. der Bildung von größeren Netzen. Jedoch wäre durch eine Bildung von z.B. Scatternetzen, wie es bei Bluetooth der Fall ist, eine Vergrößerung der Reichweite denkbar.

Als weiterführende Aufgabe können die Protokolle nun auf ihre Eignung für bestimmte Ambient Technology-Szenarien überprüft werden. Dafür sollte zunächst ein Protokoll ausgewählt und implementiert werden. Anschließend kann dies dann an einem Simulator getestet werden. Dazu kann ein Simulator wie z.B. Jist/Swans zum Einsatz kommen.

Desweiteren können weiteregehende Untersuchungen von Service Discovery Protokollen für den Ambient Technology Bereich weitere geeignete Ansätze hervorbringen. Durch die Weiterentwicklungen in diesem Bereich, k

Seit einigen Jahren beschäftigen sich Computer-Wissenschaftler mit dem Service Discovery und deren Protokollen. Dabei wird verstärkt auf den Adhoc-Bereich und seiner Komplexität Wert gelegt. Mittlerweile exisitieren bereits einige geeignete Ansätze.

Grundsätzlich haben alle aufgeführten Protokolle eine Gemeinsamkeit: Sie haben wenig Schwierigkeiten im Umgang mit einer geringen Anzahl an Services und Nutzern. Die eigentlichen Probleme entstehen bei der Vergrößerung des Systems (Skalierbarkeit). Ebenso kommen viele Protokolle mit der Dynamik in Adhoc-Netzen nicht zurecht. Diese gilt es näher zu erforschen.

Zudem ist es denkbar, dass sich mehrere Technologien durchsetzen werden, was die Notwendigkeit der Zusammenarbeit unterschiedlicher Protokolle erforderlich macht.

Dabei gilt es speziell im Bereich des Service Discovery auf Fragen der Diensteverteilung, -anfrage und -beschreibung näher einzugehen, um Suchen möglichst effizient zu gestalten.

Literaturverzeichnis

[AC02] A. Crespo, H. Garcia-Molina: *Routing indices for peer-to-peer systems.* In: *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, Seite 23. IEEE Computer Society, 2002.

[Aca05] Acar, J.: *Supporting Service Discovery and Interaction.* Seminararbeit in „Services Computing and Service-Oriented Architectures", Westfälische Wilhelms-Universität Münster, 2005.

[AF04] A. Friday, N. Davies, N. Wallbank und E. Catterall und S. Pink: *Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments.* Wireless Networks, Seiten 631–641, 2004.

[Ang03] Angstmann, K.: *Entwicklung eines energieeffizienten Verfahrens zur dynamischen Erkennung entfernter Dienste in ubiquitären Informationssystemen.* Diplomarbeit, Universität Stuttgart, 2003.

[AR01] A. Rowstron, P. Druschel: *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.* In: *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Seiten 329–350. Springer-Verlag, 2001.

[Bie06] Bier, M.: *Multicast Service Discovery in Mobile Ad Hoc Networks.* Seminararbeit, Institut für Informatik, FU-Berlin, Juni 2006.

[BY02] B. Yang, H. Garcia-Molina: *Efficient Search in Peer-to-Peer Networks.* In: *Proceedings of ICDCS*, Seiten 5–14, 2002.

[CG06] C. Gkantsidis, M. Mihail und A. Saberi: *Random walks in peer-to-peer networks: algorithms and evaluation.* Perform. Eval., 63(3):241–263, 2006.

[DC] D. Chakraborty, H. Chen: *Service Discovery in the Future for Mobile Commerce.* `http://www.cs.umbc.edu/~dchakr1/papers/mcommerce.html` (Zuletzt besucht: 06. Juni 2007).

[DC01] D. Chakraborty, F. Perich, S. Avancha und A. Joshi: *DReggie: A smart service discovery technique for e-commerce applications.* 20th Symposium on Reliable Distributed Systems, Oktober 2001.

[DC02] D. Chakraborty, A. Joshi et al.: *GSD: A Novel Group-based Service Discovery Protocol for MANETS*. In: *Proceedings of International Workshop on Mobile and Wireless Communications Networks*, Seiten 140–144, 2002.

[fra06] *Gefahrenmeldungen verbreiten sich wie ein Lauffeuer*. Innovisions, Zukunftsmagazin der Frauenhofer IuK-Gruppe, Seiten 20, 21, 2006.

[Ger04] Gerdes, E.: *Agentenbasierte Suche in Peer-to-Peer Netzen*. Studienjahresarbeit, Universität Illmenau, 2004.

[Gry] Gryazin, E.: *Service Discovery in Bluetooth*. `http://citeseer.ist.psu.edu/392311.html`.

[Gut99] Guttmann, et al: *Service Location Protocol*. RFC 2608, Juni 1999.

[HC00] H. Chen, A. Joshi, T. Finin: *Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether*. Technischer Bericht, Department of Computing Science and Electrical Engineering, UMBC, Februar 2000.

[IS01] I. Stoica, R. Morris, D. Karger und F. Kaashoek und H. Balakrishnan: *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications*. In: *Proceedings of the 2001 ACM SIGCOMM Conference*, Seiten 149–160, 2001.

[IS03] I. Stoica, R. Morris, D. Karger und F. Kaashoek und H. Balakrishnan: *Looking up data in P2P systems*. Commun. ACM, 46(2):43–48, 2003.

[JAB01] Jovanovic, M., F. Annexstein und K. Berman: *Scalability issues in large peer-to-peer networks - a case study of gnutella*. Technischer Bericht, University of Cincinnati, Januar 2001.

[KA02] K. Aberer, M. Hauswirth: *An overview of peer-to-peer information systems*. In: *WDAS*, Seiten 171–188, 2002.

[KHZ02] K. Hildrum, J. Kubiatowicz, S. Rao und B. Zhao: *Distributed object location in a dynamic network*. In: *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, Seiten 41–52. ACM Press, 2002.

[LK03] L. Küderli, M. Mayer: *Peer-to-Peer und andere Ad-hoc-Dienste*. Hauptseminararbeit in „Aktuelle Trends in Kommunikationsnetzen“ WS 02/03, TU München, 2003.

[LL05] L. Little, P. Briggs: *Designing Ambient Intelligent Scenarios to Promote Discussion of Human Values*. Presented at Ambient Intelligence workshop, Interact, Rome, September 2005, 2005.

[Loe03] Loenen, E. J.: *On the role of Graspable Objects in the Ambient Intelligence Paradigm*. Presented at the Smart Objects Conf., 2003.

[Mur03] Murdock, S.: *Ambient Intelligence and the home of the future: Are we ready for it?* Technischer Bericht, INSS690, Juli 2003.

[OR02] O. Ratsimor, D. Chakraborty, A. Joshi und T. Finin: *Allia: Alliance-based Service Discovery for Ad-Hoc Environments*. In: *ACM Mobile Commerce Workshop*, 2002.

[Ora01] Oram, A.: *Peer-to-Peer: Harnessing the Benefits of a Desruptive Technology*. O'Reilly, 2001.

[p2p04] *P2P-Technologien*. Hauptseminar P2P-Technologien, Universität Ulm, 2004.

[QL02] Q. Lv, P. Cao, E. Cohen und K. Li und S. Shenker: *Search and replication in unstructured peer-to-peer networks*. In: *Proceedings of the 16th international conference on Supercomputing*, Seiten 84–95, 2002.

[RH00] R. Hermann, D. Husemann, M. Moser und M. Nidd und C. Rohner und A. Schade: *DEAPspace: transient ad-hoc networking of pervasive devices*. In: *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, Seiten 133–134. IEEE Press, 2000.

[RH03] R. Handorean, Gruia-Catalin Roman: *Secure Service Provision in Ad Hoc Networks*. In: *Service-Oriented Computing - ICSOC 2003*, Seiten 367–383. Springer Berlin/Heidelberg, Volume 2910/2003 Auflage, 2003.

[RH04] R. Harbird, S. Hailes und C. Mascolo: *Adaptive resource discovery for ubiquitous computing*. In: *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, Seiten 155–160. ACM Press, 2004.

[Sch05] Schwarz, G.: *Unstrukturierte Peer-to-Peer-Netze*. Seminararbeit in „P2P Networking", BTU Cottbus, Dezember 2005.

[SH03] S. Helal, N. Desai, V. Verma und C. Lee: *Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks*. In: *Third IEEE Conference on Wireless Communication Networks (WCNC)*, Seiten 2107–2113, 2003.

[Sos04] Sosna, D.: *Vorlesung P2P*. Skriptum zur Vorlesung. Uni Leipzig, 2004.

[SR01] S. Ratnasamy, P. Francis, M. Handley und R. Karp und S. Schenker: *A scalable content-addressable network*. In: *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, Seiten 161–172. ACM Press, 2001.

[uCR00] C. Renner, C. Bettstetter und: *A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol*, 2000. `http://citeseer.ist.psu.edu/article/bettstetter00comparison.html`.

[vdM] Meulen, B. van der: *Designing ambient intelligence: from use to practice; from users to networks.* Design and Consumption: Ideas at the Interface Workshop, Durham University, UK, 12-13 January, 2005. http://www.dur.ac.uk/designing.consuming/wkshpsjan06/papers/12-13th/van%20der%20meulen.pdf.

[Wal99] Waldo, J.: *The Jini architecture for network-centric computing.* Communications of the ACM, 42(7):76–82, 1999.

[WAW99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan und J. Lilley: *The design and implementation of an intentional naming system.* In: *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, Seiten 186–201. ACM Press, 1999.

[WK04] W. Klenner, M. Jenisch, S. Rupp: *The Intelligent Inhouse Ambient.* FITCE Congress 2004, Ghent, 2004.

[wwwa] *Ambient Intelligence @ INI-GraphicsNet.* http://www.ambientintelligence.de.

[wwwb] *The Chord/DHash Project.* http://pdos.csail.mit.edu/chord/ (Zuletzt besucht: 18. Juni 2007).

[wwwc] *How Gnutella Works.* http://computer.howstuffworks.com/file-sharing3.htm (Zuletzt besucht: 18. Juni 2007).

[wwwd] *OpenSLP.* http://www.openslp.org (Zuletzt besucht: 4. Juni 2007).

[wwwe] *Verteilte Hashtabelle.* http://de.wikipedia.org/wiki/Verteilte_Hashtabelle (Zuletzt besucht: 16. Juni 2007).

[YGM02] Yang, Beverly und Hector Garcia-Molina: *Improving Search in Peer-to-Peer Networks.* In: *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, Seite 5. IEEE Computer Society, 2002.

[Zit05] Zitterbart, Prof. Dr. Martina: *Next Generation Internet.* Skriptum zur Vorlesung. Institut für Telematik, Fakultät für Informatik, Universität Karlsruhe (TH), 2005.