Studienarbeit

Identifikationsauflösung mittels eines Peer-to-Peer Netzwerks

von Michael Leuchtner

Betreuer:
Prof. Dr. rer.nat. W. Juling
Dipl.-Inform. Christian Decker

Institut für Telematik, Telecooperation Office (TecO)
Universität Karlsruhe

August 2003

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Identifikationssysteme finden sich überall in unserem täglichen Leben in unterschiedlichsten Formen: Strichcodes auf Lebensmitteln, Bank- und Scheckkarten, Türöffner um nur einige Beispiele zu nennen. Wurden sie jedoch lange tatsächlich nur zur Identifikation von Personen und Gegenständen benutzt, können sie heute wesentlich umfangreicher eingesetzt werden. Aufgrund der Allgegenwärtigkeit und Mobilität von verschiedenen Kommunikationsmitteln und Datennetzen - z.B. Internet oder GSM - bieten sich dem Nutzer fast überall eine Flut von Informationen. Oft ist es jedoch schwierig diese sinnvoll zu koordinieren und gezielt die richtige Information aus den Datennetzen relativ schnell und ohne Aufwand in einen Kontext mit der realen Welt zu bringen. An dieser Stelle kommen Identifikationstechniken, wie z.B. Radio Frequency Identification (RFID)[8], zum Einsatz. Sie dienen als Brücke zwischen der virtuellen und der realen Welt, oder anders ausgedrückt, ein realer Gegenstand wird durch sein Identifikatinsmechanismus einem virtuellen Dienst, z.B. einer Beschreibung, zugeordnet und umgekehrt. Dadurch bieten sich natürlich eine Reihe von Anwendungsmöglichkeiten z.B. in den Bereichen Supply Chain Management oder Kundenbetreuung und -service.
Diese allgemeine Beschreibung soll folgendes kurzes Beispielszenario veranschaulichen:
Typisch wäre beispielsweise ein RFID-Tag auf Waren im Supermarkt. Ein, oder bei Bedarf auch mehrere, RFID-Tags auf einem Produkt könnte verschiedenste Dienste durch vorbeiführen an einem RFID-Leser anfordern. Dies könnte eine Produktbeschreibung sein, die auf einem Display angezeigt wird oder ein 'Warenkorb-Dienst' der einen Preis liefert, um den Gesamtpreis des Einkaufswagens festzustellen. Möglich ist aber auch ein Diebstahl-Service, der beim Verlassen des Kaufhauses durch ein Lesegerät an der Tür Alarm schlägt. Die Abblidung 1.1 skizziert nochmal ein solches Supermarkt-Szenario.

An dieser Stelle entstehen eine Reihe von zu lösenden Aufgaben: diese Zuordnung von virtuellen Diensten zu realen Gegenständen sollte eindeutig, zuverlässig und korrekt, aber auch schnell, einfach und intuitiv, z.B. ohne bewußtes Zutun des Benutzers ablaufen.
Diese Studienarbeit beschreibt ein System, das die Assoziation zwischen Identifkationsmechanismen - wie z. B. RFID-Tags - und virtuellen Informationen mit Hilfe eines Peer-to-Peer

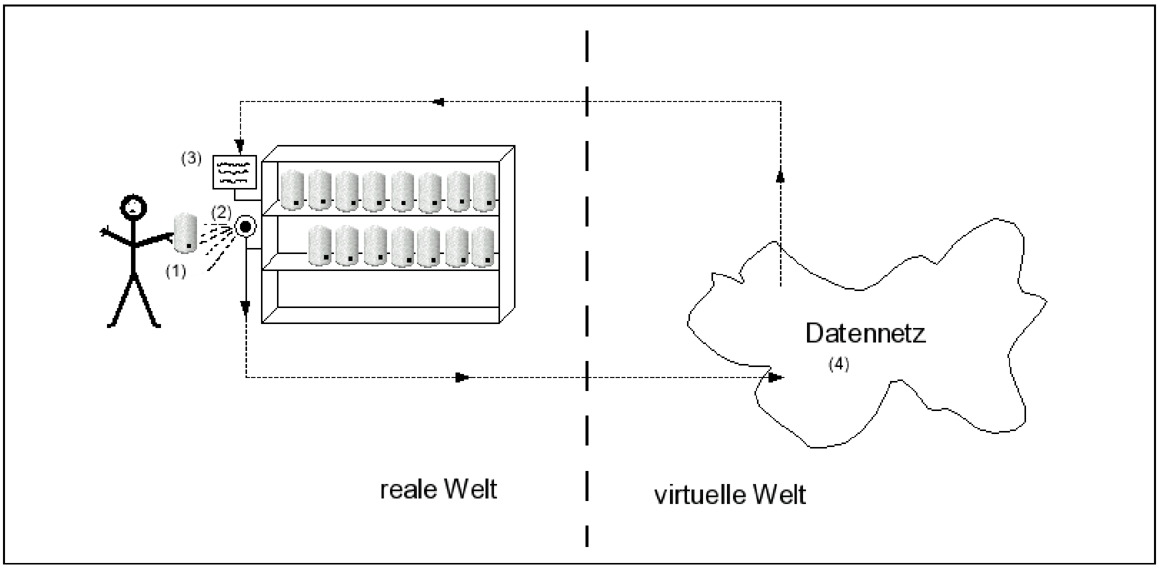

Abbildung 1.1: Ein Produkt(1) mit einem RFID-Tag wird aus dem Regal geholt und an einem RFID-Leser(2) vorbeibewegt. Ein Display(3) zeigt eine Beschreibung - erhalten aus einem Datennetz(4) - der Ware an.

Netzwerks auflöst: Beliebige Objekte in der realen Welt, z.B. Lebensmittel, elektrische Geräte etc., ausgestattet mit einer eindeutigen Kennung, sollen mit Hilfe dieser Kennung über ein Peer-to-Peer Netz (P2P-Netz) identifizert werden und anhand dieser entsprechende Informationen - z.B. Produktinformation, Dokumentationen, Softwareupdates etc. - zurückliefern.
Die derzeit größten Informationsnetze, wie Telefon oder Internet, basieren auf hierarchischen Strukturen. Der Einsatz von Peer-to-Peer Netze bringt aber eine Reihe von Vorteilen mit sich: Der wesentliche Punkt ist die Dezentralität. Eine Befürchtung, die oft im Zusammenhang mit Identifikationstechniken und deren Auflösung geäußert wird, ist die Überwachung, das Problem des gläsernen Menschen. Da ein P2P-Netz grundsätzlich dezentral organisiert ist, ist jeder Benutzer im Netz gleichberechtigt. Es gibt keine zentralen Stellen die Dienste anbieten oder die nötig sind um das Netz zu betreiben und aufrecht zu erhalten. Dieser Umstand erschwert somit z. B. das eventuelle protokollieren von Anfragen. Weiterhin ist es in einem P2P-Netz relativ einfach anonym zu bleiben, da sich kein Teilnehmer im Netz z. B. an einer speziellen Stelle registrieren muß, sondern das Netz beliebig betreten und verlassen kann. Ein weiter Vorteil ist die Erweiterbarkeit: Da es in einem P2P-Netz keine Instanzen mit hervorgehobener Stellung gibt, kann das Netz relativ leicht durch Hinzunahme von zusätzlichen Peers vergrößert werden.

1.2 Ziel und Umfang der Arbeit

Zentrales Thema dieser Arbeit ist dabei die Integration des P2P-Netzes. Zielsetzung ist es, ein dezentrales, anonymes und vor allem sicheres Netz mit Hilfe bereits existierender P2P- und Sicherheitstechniken aufzubauen, über welches verschiedenste Dienste z.B. per RFID-Tag ange-

fragt werden können. Der P2P-Ansatz bietet vor allem die folgenden Vorteile: Um das Netz zu betreiben, ist keine zentrale Instanz nötig, welche z. B. ausgenutzt werden könnte, um Anfragen zu protokollieren. Des weiteren ist ein P2P-Netz relativ einfach erweiterbar.
Der erste Teil der Arbeit betrachtet zunächst die Auswahl und Verteilung des Identifiationssystems. Der Einfacheit halber bezieht sich die Studienarbeit o.B.d.A. hauptsächlich auf RFID-Tags, das Netz kann aber auch - je nach Bedarf - mit anderen ID-Systemen als Eingabe, z.B. Barcode, betrieben werden. Das folgende Kapitel stellt kurz die wichtigsten Systeme vor und beschreibt die wesentlichsten Unterschiede.
Anschließend standen dann zwei wesentliche Aufgaben im Mittelpunkt: Zunächst mußte eine geeignete P2P-Umgebung gefunden werden. Es war hierbei nicht nur die Leistungsfähigkeit der einzelnen Netze zu vergleichen, sondern zu prüfen, ob eine benutzbare Implementierung hinsichtlich Stabilität, Lizenzen, Plattform etc. zur Verfügung steht. Da einer der wesentlichen Anforderungen die Sicherheit war, wurden weiterhin verschiedene Verschlüsselungs- und Authentifizierungsmechanismen und entsprechende Angriffsszenarien untersucht.
Realisiert wurde diese Arbeit mit Hilfe der P2P-Plattform JXTA[1] - auf Java-Basis - und dem Verschlüsselungssystem GnuPG[3]. Das darauffolgende Kapitel beschreibt diese Implementierung ausführlich.
Zum Abschluß wurden noch verschiedene offene Fragen und Verbesserungsmöglichkeiten diskutiert und schließlich eine Auswahl von verwandten oder ähnlichen Projekten vorgestellt.

Kapitel 2

Identifikationssysteme - ein Überblick

Wie schon erwähnt, liegt der Schwerpunkt dieser Arbeit auf der Integration des P2P-Netzes, da aber das Identifikationssystem (ID-System) das Eingabemedium, und damit ein wesentlicher Bestandteil des gesamten Systems ist, sollen im folgenden einige ID-Systeme vorgestellt, und die Vor- und Nachteile diskutiert werden.

2.1 Barcode

Das derzeit verbreitetste System zur Identifikation ist der Strichcode[7], der sich seit Jahren bewährt hat: auf Lebensmitteln, Postpaketen oder auf Läufern beim Marathon. Die wesentlichen Vorteile sind folgende: der Barcode[7] ist sehr billig zu produzieren, nämlich durch einfaches aufdrucken, und ist sehr Robust, je nach Art der Aufbringung. Außerdem gibt es einen weltweiten Barcode-Standard.
Der Barcode hat aber auch entscheidende Nachteile: Er läßt sich problemlos fälschen, nämlich z. B. durch einfaches fotokopieren. Weiterhin benötigt er eine gewisse Menge Platz auf dem Objekt und muß immer sichtbar sein, um gelesen zu werden. Außerdem stellt sich das Problem, daß der 'Speicherplatz' auf den Code selbst beschränkt ist. Aus diesen Gründen ist der Strichcode in vielen Bereichen nur bedingt oder überhaupt nicht einsetzbar: Beispielsweise läßt sich in einem Barcode der Fertigungszustand eines Produktes, der sich während der Produktion natürlich ständig ändert, nicht festhalten. Ein anderes Medium könnte z. B. Auskunft geben, ob das Produkt fertig, halb fertig oder defekt ist. Auch vorstellbar wäre der Einsatz von ID-Systemen in elektronischen Geräten, z. B. Computern, die Auskunft über die einzelnen Bauteile geben können. Bei der Verwendung von Strichcodes müßte das Gerät geöffnet werden und die Teile einzeln ausgelesen werden, bzw. es könnte sich um so kleine Bauteile handeln, daß ein Strichcode keinen Platz darauf findet.

2.2 RFID

Eine Technik die in letzter Zeit in immer neuen Lebensbereichen auftaucht, immer mehr an Bedeutung gewinnt, aber auch oft aufgrund seiner Fähigkeiten kritisch betrachtet wird, ist der RFID-Transponder[8]. Das Kürzel steht für 'Radio Frequency Identification'. Dabei handelt es sich um relativ kleine - oft nur wenige Quadratmilimeter große - Chips, die in der Regel eine ID und einen Speicherplatz für beliebige Einträge anbieten. Aufgrund dieser geringen Abmessungen lassen sich die Tags auf prakratisch jedem Gegenstand, oft unsichtbar platzieren. Es gibt sie in den unterschiedlichsten Varianten die sich im wesentlichen durch Größe, Speicherplatz und Art des Auslesens unterscheiden, wobei das kontaktlose Auslesen, also einfaches vorbeiführen an einem Lesegerät, der verbreitetste Typ ist. Auch hier unterscheiden sich die Chips in Sachen Reichweite. Die Werte liegen derzeit zwischen 5 cm und 2,5 m für passive, d. h. batterielose Transponder.
Ein entscheidender Vorteil - neben dem Preis, der je nach Modell bei wenigen Cent liegt - ist aber sicher die Tatsache, das der Chip nicht direkt mit einer Stromversorgung verbunden sein muß. Seine Energie erhält der RFID-Tag vom Lesegerät: Wird er in seine Reichweite bewegt, bedient er sich des entstehenden Induktionsfeldes, welches ausreicht, um ID und Speicherinhalt an den Leser zu übertragen. Die beschriebenen Merkmale bringen den RFID-Tag aber auch immer wieder in die Kritik, da aufgrund dieser Bauweise der Chip ohne das Wissen vieler Menschen angebracht und ausgelesen werden kann.
Weiterhin gibt es aber Punkte, die gegen einen Einsatz von RFID sprechen. Ein entscheidener Punkt ist, daß es sich bei dem Chip um ein offenes System handelt, auf dem prinzipiell jeder lesen und schreiben kann. Somit ist der RFID-Tag z.B. für Sicherheitkritische Anwendungen, bei denen der Speicherinhalt besonders geschützt sein muß, sicher nicht die richtige Wahl. Ein Nachteil gegenüber den Barcodes ist, daß RFIDs nicht völlig unempfindlich sind gegenüber Umwelteinflüssen, z. B. elektrische oder mechanische.

2.3 Smart-Its/ Smartcard

Um das im vorherigen Abschnitt angesprochene Problem der Sicherheit zu lösen muß man sich etwas aufwendigerer Technik bedienen. Beispielhaft seien hier Smart-Its[10] und die Smartcard[12] aufgeführt. Es handelt sich hierbei um Geräte in der Größe von wenigen Quadratzentimetern, die mit Prozessor, Speicher und Kommunikationsschnittstellen ausgestattet sind und daher selbständig komplexe Rechenoperationen ausführen können. Dadurch bilden sie ein abgeschlossenes System, von dem nicht ohne weiteres Daten ausgelesen werden können. Natürlich bringen diese Geräte auch einen größeren Speicherplatz mit.
Die Nachteile dieser Geräte sind folgende: Zunächst liegt der Preis - auch wenn er nur wenige Euro beträgt - deutlich über dem Preis der RFID-Tags. Ein weiteres Problem ist die Stromversorgung: Die Geräte müssen entweder eine Energiequelle, z. B. Batterie oder Solarzellen, mit sich führen oder an eine angeschlossen werden, wie durch einstecken in einen Kartenleser, um betrieben zu werden. Aus diesen Gründen sollten diese Medien nur in Szenarien zum Einsatz kommen, die diese Komplexität auch wirklich benutzen, z. B. bei sicherheitskritischen Anwendungen.

Kapitel 3

Peer-to-Peer Netze

Allgemein betrachtet definieren sich P2P-Netze folgendermaßen: Beliebig viele, mehr oder weniger gleichberechtigte Applikationen schließen sich - in der Regel auf TCP/IP basierend - dynamisch zu einem verteilten Netz zusammen und bieten in diesem Netz verschiedenste Dienste an oder nehmen diese in Anspruch. Dadurch entsteht ein dezentrales und anonymes Netz aus beliebig vielen Peers.
Die verschiedenen P2P-Ansätze unterscheiden sich durch eine unterschiedlich starke Ausprägung dieser einzelnen Merkmale. Im folgenden soll eine Auswahl verschiedener P2P-Netze vorgestellt, die wesentlichen Unterschiede und die Benutzbarkeit für diese Arbeit aufgezeigt werden.

3.1 Napster/ OpenNap

Napster, bzw. seine Open Source Variante OpenNap[13], ist wohl eines der populärsten P2P-Protokolle. Es zeichnet sich dadurch aus, daß es nicht vollständig dezentral organisiert ist, sondern mit zentralen Servern arbeitet, welche Indexinformationen über die einzelnen Clients halten. Der P2P-Gedanke beschränkt sich hier lediglich auf den Austasch von Infomationen wie z.B. Datei-Downloads, nachdem diese mit Hilfe des Indexes auf dem Server gefunden wurde. Aus diesem Grund ist OpenNap auch nicht für diese Arbeit geeignet, da eine wesentliche Vorgabe die Dezentralität war. Auch eine Anonymität kann nur schwer gewährlistet werden, wenn die Vermittlung über zentrale Server stattfindet.

3.2 Gnutella

Im Gegensatz zum OpenNap-Protokoll, welches eine relativ starke zentrale Komponente besitzt, zeichnet sich Gnutella[14] durch vollständige Dezentralität aus - abgesehen von einem relativ beliebigen initialen Eintrittspunkt, den man i. d. R. benötigt um dem Netz beizutreten. Der komplett dezentrale Ansatz bringt aber verschiedene Probleme mit sich: Im Vergleich zu anderen P2P-Ansätzen sind relativ häufige Verbindungswechsel zu beobachten; Clients kommen und gehen. Als Folge ergibt sich eine praktisch ständige Suche nach neuen Verbindungen. Um Informationen über das Netz zu erhalten werden regelmäßig sogenannte PING- und PONG-Nachrichten gesendet. Untersuchungen[15] haben gezeigt, daß diese Nachrichten über die Hälfte der Netzkommunikation ausmachen können. Daraus ergibt sich nicht nur ein relativ hoher Kommunikations-Overhead, sondern auch daß Suchanfragen relativ lange dauern können. Desweiteren eignet sich Gnutella aus genannten Gründen auch schlecht für kleine Geräte mit wenig Rechenleistung und Bandbreite.

3.3 Freenet

Auch Freenet[16] ist komplett dezentral organisiert, löst aber den bei Gnutella durch ständige Broadcasts verursachten Kommunikations-Overhead durch geschickte Routing- und Suchverfahren. Die wesentliche Philosophie bei der Entwicklung von Freenet ist die Gewährleistung von Anonymität. Zu diesem Zweck werden beispielsweise Inhalte hinter globalen Identifikationsnummern (GUID) verborgen. Folglich läßt sich hier auch nicht einfach nach einem bestimmten Dienst das Netz durchsuchen der evtl. noch von verschiedenen Peers gleichzeitig angeboten wird. Freenet erwartet die Anfrage einer solchen GUID, was die Benutzbarkeit in dieser Arbeit deutlich limitiert.

3.4 DirectConnect

DirectConnect, oder besser bekannt in Form des Clients DC++[17], zeichnet sich dadurch aus, daß es aus zwei Komponenten besteht: Clients und Hubs. Wie der Name schon vermuten läßt haben die Hubs die Aufgabe eines Vermittlers zwischen den einzelnen Clients. Hubs werden nicht dynamisch gefunden, die Adressen werden einfach über Serverlisten, z. B. per Web-Download verteilt. Die Architektur erinnert etwas an die Napster-Server, es gibt aber einen wesentlichen Unterschied dazu: Die Hubs halten keine Informationen, sie dienen lediglich als Knotenpunkte, die Anfragen und Antworten über die einzlenen Clients verteilen. Die Nachteile dieses Systems sind: Zum einen führen überlastete oder zu wenige Hubs unweigerlich zu Überlast-Situationen. Zum anderen bieten die Hubs gute Angriffspunkte und verletzen vor allem die geforderte Dezentralität.

3.5 JXTA

Das Projekt JXTA[1], das 2001 von Sun Microsystems ins Leben gerufen wurde, verfolgt im Gegensatz zu den bisher diskutierten P2P-Ansätzen, ein etwas allgemeineres Ziel. Bei JXTA wurde nicht auf die Entwicklung einer P2P-Applikation Wert gelegt, sondern man möchte eher eine stabile und flexible Programmierplattform inclusive API-Dokumentation für P2P-Anwendungen zur Verfügung stellen. Diese Plattform zeichnet sich aufgrund der Java-Implementation durch eine Unabhängigkeit von Betriebssystem und Rechnerarchitektur aus. JXTA ermöglicht es je nach Bedarf den Grad der Dezentralität beliebig zu bestimmen, indem die einzelnen Peers - je nach Konfiguration - mehr oder weniger Aufgaben im Netz wahrnehmen. Somit bietet sich auch die Möglichkeit z. B. kleinere, mobile Geräte mit wenig Bandbreite in das Netz zu integrieren. Weiterhin bietet die Plattform verschiedene Sicherheits- und Verschlüsselungsmechanismen an. Durch die Möglichkeit, das Netz in verschiedene Gruppen zu unterteilen, sollte es sich auch relativ gut skalieren lassen. Aus diesen Gründen scheint JXTA für diese Arbeit gut geeignet und wurde für die Implementierung verwendet. Eine ausführlichere Beschreibung der P2P-Plattform und der Implementierung wird in den Kapiteln 4 und 6 erfolgen.

Kapitel 4

Die JXTA-Plattform

4.1 Einführung und Überblick

JXTA[1] wurde im Jahr 2001 als Forschungsprojekt von Sun Microsystems gestartet. Im Gegensatz zu vielen anderen P2P-Projekten verfolgte man aber nicht das Ziel P2P-Applikationen wie z.B. für File-Sharing zu entwickeln. Im Vordergrund stand zunächst die Spezifikation von möglichst Plattform- und Geräteunabhängigen P2P-Protokollen. Darauf aufsetzend wurde die JXTA-Plattform entwickelt. Es handelt sich dabei um eine Java-Implementation dieser Protokolle. Diese stellt aber keine fertige Anwendung zur Verfügung, sondern ein Entwicklungstool. Es werden grundlegende Funktionen, wie Verbindungsaufbau, Routing, Nachrichtenaustausch oder die Suche und Publikation von Diensten, in Form von Java-Klassen und API-Dokumentation zur Verfügung gestellt. Auf dieser Basis will man die Entwicklung von beliebigen P2P-Applikationen - wie Dateiaustausch, Chat, verteilte Anwendungen etc. - ermöglichen.

4.2 JXTA Architektur und Komponenten

4.2.1 Architektur

Die JXTA Architektur läßt sich, wie in Abbildung 4.1 zu sehen ist, in drei Schichten unterteilen:

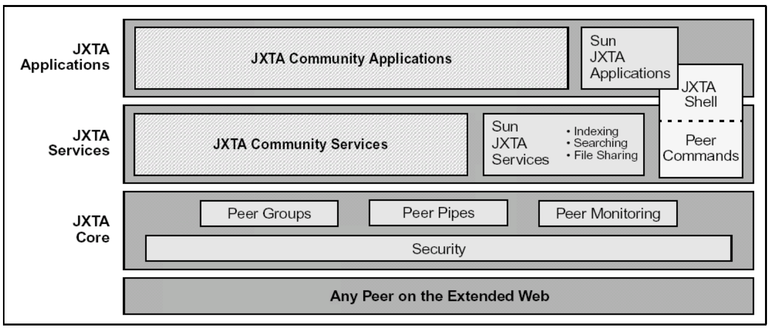

Abbildung 4.1: Die JXTA Architektur

- *Plattform Schicht*
  Die Plattform Schicht, oder auch JXTA Core genannt, beheimatet die grundlegendsten Basisfunktionen, die jede P2P-Anwendung benötigt. Beispiele hierfür sind erzeugen von Peers und Pipes, Transport- und Sicherheitsmechanismen. Diese Schicht stellt daher eine Art Infrastruktur für das P2P-Netz dar.

- *Service Schicht*
  Die Service Schicht setzt auf der JXTA Core auf. Sie bietet Funktionen, die für den Bestand des Netzes zwar nicht zwingend nötig sind, die aber ein sinnvolles Arbeiten arbeiten auf der Plattform erst ermöglichen. Diese sind unter anderem: suchen und auffinden von Diensten oder anderen Resourcen, Identifikations- und Anmeldemechanismen.

- *Anwendungsschicht*
  In der Anwendungsschicht befindet sich der eigentliche P2P-Client, der Auf Basis der JXTA Plattform entwickelt wird, z. B. File-Sharing oder Chat-Anwendungen oder im Fall dieser Studienarbeit ein System zur Auflösung von RFID-Tags.

Die zwei wesentlichen Bestandteile in einem JXTA-Netz sind Peers in unterschiedlichen Varianten und Peer Groups, die, wie Abbildung 4.2 zeigt, völlig unabhänig von dem darunterliegenden physischen Netz existieren. Die Peers sind die einzelnen Clients, die - vor allem zur Zwecken der Skalierung und Authentifizierung - in beliebig viele Peer Groups gegliedert werden können.

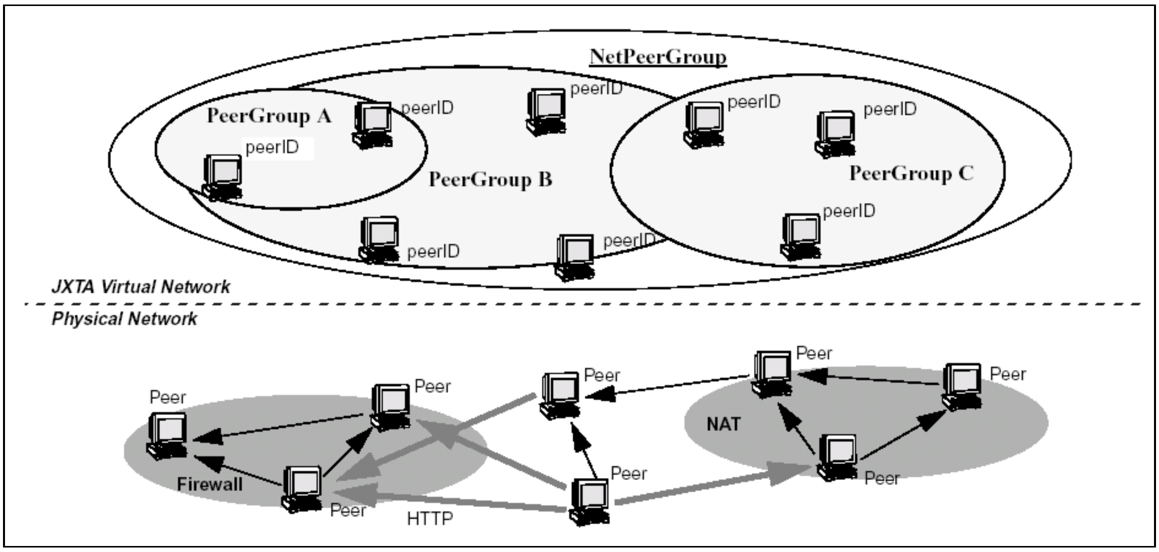

Abbildung 4.2: Das JXTA Netz

Alle Resourcen - wie Peers, Peer Groups, Pipes, Services etc. - in JXTA werden identifiziert mit Hilfe von sog. Advertisements. Dabei handelt es sich um kleine XML-Dokumente, die neben

Namen, Beschreibungen etc., vor allem eine UUID (universal unique identifier) enthählt, die die Resource im Netz eindeutig identifiziert. Die UUIDs werden per Zufallsgenerator erzeugt und der Resource zugewiesen. Die weiteren Einträge in den Advertisements können je nach Typ - z. B. Peer- oder PipeAdvertisement - in Bedeutung und Umfang variieren. Jedoch hat jede Resource immer genau ein Advertisement und jedes Advertisement hat zumindest eine UUID. Diese IDs müssen aber zum Auffinden eines Dienstes nicht zwingend bekannt sein. Sie dienen vor allem der Unterscheidbarkeit einzelnen Advertisements.
Mit Hilfe des 'Discovery Service' können all diese Advertisements im den einzelnen Peer Groups gesucht und anschießend darauf zugegriffen werden. Abbildung 4.3 zeigt ein Beispiel eines PipeAdvertisements.

```
 <!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta=''http://jxta.org''>
<Id>urn:jxta:uuid-596162616162614A757874614D507284FBC4E5D498AA0919F6624002804</Id>
<Type> JxtaUnicast </Type>
<Name> PipeExample </Name>
</jxta:PipeAdvertisement>
```

Abbildung 4.3: Beispiel eines PipeAdvertisements

4.2.2 Die JXTA Komponenten

Der folgende Abschnitt wird die wichtigsten und für diese Studienarbeit relevanten Komponenten der JXTA-Plattform vorstellen.

Peer

Das Peer ist wohl die wichtigste Komponente und repräsentiert wie schon erwähnt die Applikation im Netz. Es gibt unterschiedliche Ausprägungen eines Peers, die sich in ihrem Funktionsumfang wesentlich unterscheiden:

full-featured edge peer: Das 'full-featured edge peer' ist das gebräuchlichste aller Peers. Es kann Nachrichten senden und empfangen, advertisements in einem Cache halten. Aber es leitet keine Nachrichten zwischen zwei Peers weiter. Sofern es kann, beantwortet dieses Peer eine Anfrage oder verwirft sie. Aus diesem Grund wird eine weitere Instanz benötigt, die diese Aufgabe übernehmen kann: die rendezvous- und relay peers. Jedes Peer muß mindestens zu einem oder auch mehreren rendezvous und relay verbunden sein, oder muß selber eines sein.

minimal edge peer: Wie der Name schon vermuten läßt, ist das 'minimal edge peer' das mit dem geringsten Funktionsumfang. Es kann im wesentlichen nur Nachrichten senden und empfangen, hält aber z. B. keine Cache-Informationen. Es ist gedacht für kleine Geräte, z.B. PDAs oder Mobiltelefone, die mit einer geringen Bandbreite auskommen müssen.

rendezvous peer: Das 'rendezvous peer' arbeitet zunächst mal wie ein 'full-featured edge peer', es hat aber eine zentrale Bedeutung: Jede Peer Group benötigt mindestens ein rendevous peer. Da es in jedem JXTA-Netz mindestens eine Gruppe geben muß, muß es in jedem Netz mindestens ein rendezvous peer geben, eine Beschränkung nach oben gibt es nicht. So können im Extremfall alle Peers als rendezvous arbeiten. Die Aufgabe der rendezvous peers ist hauptsächlich die Vermittlung. Sie beantworten Nachrichten und Anfragen oder leiten zwischen den Peers weiter, halten Listen über Peers die zu ihm verbunden sind, und Listen von weiteren Rendezvous Peers. Auserdem halten sie ausführlichere Caches um Nachrichten gezielter weiterleiten zu können.

relay peer: Relay Peers haben im wesentlichen zwei Aufgaben: Zum einen sind Relays für das Routing im Netz verantwortlich. Sie halten Listen über Routen zu den einzelnen Peers und suchen im Netz nach neuen Routen, falls eine angefragte Route noch nicht bekannt ist. Des weiteren dienen Relays als Vermittler für Rechner, die hinter einer Firewall sitzen.

Peer Group

Mit Peer Groups lassen sich die einzelnen Peers zu einer Gruppe zusammenfassen. Peers können in diesen Gruppen Dienste Anbieten oder nach solchen suchen und können in beliebig vielen Gruppen Mitglied sein. Peers können auch selbst beliebig viele Gruppen erzeugen. Es muß lediglich sichergestellt sein, daß in jeder Gruppe auch ein Rendezvous Peer existiert. Somit bietet sich z. B. die Möglichkeit, Services in verschiende Klassen zu gliedern.
Einer Gruppe kommt jedoch eine spezielle Bedeutung zu: die '*NetPeerGroup*'. In dieser Gruppe ist jedes Peer nach dem Start der Applikation automatisch Mitglied. Hier können damit Dienste für das gesamte Netz angeboten werden, bzw. das Konzept der Gruppen kann komplett ausgelassen werden, indem man nur auf der NetPeerGroup operiert. Da diese Gruppe immer vorhanden ist, ist sie, abgesehen von der Null ID, die einzige Resource die keine zufällig erzeugte UUID hat, sondern über die ID 'NetPeerGroup' angesprochen wird.

Pipes und Messages

Pipes sind der Kommunikationsmechanismus der es in JXTA den Peers ermöglicht beliebige Informationen untereinander auszutauschen. Es gibt Pipes in unterschiedlichen Ausführungen: Point-to-Point für 1:1-Verbindungen zwischen genau zwei Peers, Propagate Pipes für 1:n-Verbindungen von einem zu mehreren Peers, Secure Unicast Pipes, und dies als unidirektionale oder Bidirektionale Pipes.
Relativ unabhängig vom Typ der Pipe erfolgt die Benutzung: ein Peer publiziert ein Pipe-Advertisement und wartet auf eine Verbindung. ein anderes Peer sucht z. B. mit Hilfe eines Discovery Service dieses Advertisement und baut die Verbindung auf. Darüber können dann JXTA-Messages ausgetauscht werden.

Modules und Services

Module sind in JXTA eine Abstraktion, die es ermöglichen beliebige Arten von Funktionalitäten im Netz anzubieten. Die häufigste - und auch für diese Arbeit relevante - Nutzung, sind Services. Ein Service wird in JXTA wie alle anderen Resourcen auch mit einem Advertisement verbreitet und über eine ModuleSpecID eindeutig identifiziert. Einzige Besonderheit ist, daß sich in diesem ModuleSpecAdvertisement gleich ein PipeAdvertisement einbetten läßt. So kann man zum entsprechenden Peer gleich eine Verbindung herstellen um evtl. Daten empfangen.

Discovery Service

Obwohl der Discovery Service letztendlich nur eine von vielen Funktionalitäten ist, die JXTA anbietet, soll er trotzdem hier kurz vorgestellt werden, da er in jeder Anwendung eine wichtige Rolle spielt. Der Discovery Service dient der Behandlung, also zum Auffinden oder Publizieren, von Advertisements und ist somit das Werkzeug um sich im P2P-Netz zu bewegen. Publiziert wird mit Hilfe der remotePublish-Funktionen und den entsprechenden Advertisements als Paramenter. Für die Suche stehen die getRemoteAdvertisements-Funktionen zur Verfügung, die es erlauben die einzelnen PeerGroups nach Pipes, Peers, Services etc. unter Angabe von Namen, IDs, Beschreibungen oder ähnliches, zu durchsuchen. Der Discovery Service hat dabei aber keinerlei Einfluß auf die Struktur des Netzes, es handelt sich lediglich um eine Funktion die jedes Peer zur Suche benutzen kann.

4.3 JXTA Konfiguration

Beim ersten Start einer Applikation, die auf der JXTA-Plattform basiert müssen zunächst einige Einstellungen vorgenommen werden. Diese werden dann in dem Unterverzeichnis '.jxta' abgelegt, in dem sich unter anderem auch alle Caches, Advertisements, etc. befinden.
Die Konfiguration wird vorgenommen mit Hilfe eines Dialogfensters, welches automatisch beim ersten Start erscheint (bzw. immer dann wenn die Konfiguration unvollständig ist). Später wird nur noch das Login abgefragt. Neben Benutzernamen und Paßwort werden hier vor allem das Verhalten des Peers im Netz bestimmt. Zunächst lassen sich hier die zu verwendenden Verbindungsarten - TCP, HTTP und entsprechende Eingangs- und Ausgangsports - definieren. Der wesentliche Punkt ist aber die Auswahl des Peer-Typs. Hier muß festgelegt werden, ob das Peer als Rendezvous oder Relay Peer arbeiten soll, andernfalls muß die IP-Adresse/Port eines (oder auch mehrerer) Rendezvous-/ Relay Peers angegeben werden.

Kapitel 5

Sicherheit und Authentifizierung

Einer der wesentlichen Punkte dieser Arbeit ist das Problem der sicheren Übertragung und der Gewährleistung der Authentizität der Daten. Die Wichtigkeit dieser Fragestellung ergibt sich aus der P2P-Architektur. Da in einem solchen dezentralen Netz prinzipiell jeder teilnehmen, Informationen verbreiten und abrufen kann, ist ein Peer-to-Peer-Netz, bedingt durch seinen losen Zusammenschluß, zunächst eine unsichere und nicht vertrauenswürdige Domäne. Dieser Umstand muß daher mit Hilfe eines kryptographischen Verfahrens ausgeglichen werden. Das folgende Kapitel diskutiert die unterschiedlichen Möglichkeiten zur Sicherung und verschiedene Angriffspunkte im P2P-Netz.

5.1 symmetrische vs. asymmetrische Verfahren

Bei der Benutzung eines Verschlüsselungsverfahrens stellt sich zunächst die Frage, ob man auf ein symmetrisches oder ein asymmetrisches Verfahren zurückgreift. Symmetrische Verfahren zeichnen sich dadurch aus, daß sie mit lediglich einem Schlüssel arbeiten, der sowohl zum chiffrieren, als auch zum dechiffrieren verwendet wird. Asymmetrische Verfahren hingegen benutzen zwei Schlüssel, einen öffentlichen zum Verschlüsseln und einen geheimen zum Entschlüsseln, werden daher auch als Public Key Verfahren bezeichnet.
Prinzipiell gibt es von beiden Typen sichere Verfahren, wobei die symmetrischen Verfahren in Sachen Geschwindigkeit und Schlüssellänge leichte Vorteile[6] haben. Für den Anwendungsfall in dieser Arbeit bietet sich aber aus verschiedenen Gründen das Public Key Verfahren an: Zum einen wird das Risiko des Schlüsselmisbrauchs bei einem asymmetrischen Verfahren minimiert. Da hier nur die Public Keys an die anfragenden Peers (Enquirer) verteilt werden. Mit einem Schlüssel, der in falsche Hände gerät, können lediglich Nachrichten verschickt, aber keine Nachrichten entschlüsselt werden. Ein symmetrischer Schlüssel hingegen kann immer beides, ver- und entschlüsseln. Außerdem erfüllt das Public Key Verfahren eine wichtige Funktion bei der Authentifizierung. Dieses Thema wir noch ausführlich im nächsten Abschnitt behandelt.
Völlig unabhängig vom gewählten Verfahren muß immer darauf geachtet werden, daß Schlüssel von ausreichender Länge gewählt werden, damit sich nicht einfach per 'Brute Force' durch probieren aller Möglichkeiten Schlüssel und Nachricht erraten lassen.

5.2 Authentifizierung

Trotz Verschlüsselung der Daten gibt es aber immer noch Möglichkeiten für unauthorisierte Eingriffe: Es muß nicht nur sichergestellt werden, daß eine Nachricht nur befugte Personen lesen, sondern auch, daß der Inhalt der authentisch - also korrekt, unverfälscht und vom richtigen Absender zum richtigen Empfänger gelangt. Bei einem reinen Public Key Verfahren - also keine Signaturen, ein Schlüssel geheim, einer öffentlich - gibt es verschiedene Szenarien für solche Angriffe:

- Wenn die Public Keys tatsächlich öffentlich sind, könnte einfach ein man-in-the-middle Angriff durchgeführt werden: ein dritter setzt sich zwischen die beiden Kommunikationspartner, fängt deren Nachrichten ab und verschickt stattdessen jeweils eigene Nachrichten. Enquirer und Resolver wissen so nur, daß sie Nachrichten erhalten, die sie entschlüsseln können, sie wissen aber nicht, von wem sie tatsächlich stammen.

- Wenn Nachrichten nicht eindeutig sind, sondern z. B. gleiche Anfragen wieder das gleiche chriffrierte Paket ergeben, könnte ein dritter die Nachricht mithören, speichern und später für eine eigene Anfrage verwenden.

- Wenn die gleichen Nachrichten mehrfach auftreten können, ist es desweiteren auch einfacher möglich, Rückschlüße auf den Inhalt zu ziehen, ohne die Schlüssel zu kennen.

Um dies zu vermeiden wird bei der Anfrage eines Serivces, wie auch Abbildung 5.1 nochmals veranschaulicht, folgendermaßen vorgegangen:

- Um zu vermeiden, daß beliebige Personen verschlüsselte Nachrichten verschicken können, sind die Public Keys nicht öffentlich zugänglich. Es wird davon ausgegangen, daß die Schüssel vorher auf einem sicheren Weg zwischen den Kommunikationspartnern ausgetauscht wurden.

- Eine mit dem pulbic key des Resolvers verschlüsselte Anfrage enthält außer dem gewünschten Service und der RFID eine zufällig erzeugte SessionID, die nur einmalig für diese Anfrage verwendet wird. Dadurch wird sichergestellt, alle versendeten Nachrichten eindeutig bzgl. der Zeit sind und somit inhaltlich identische Nachrichten trotzdem immer unterschiedlich aussehen. Außerdem versieht der Sender die Nachricht mit Hilfe des Private Keys mit einer Signatur.

- Der Resolver, also das Peer, welches den gewünschten Service anbietet und dem Enquirer antwortet, entschlüsselt diese Anfrage mir seinem privaten Schlüssel und prüft zunächst ob ID und Serivce zusammen in seinem Servicefile verzeichnet sind. Außerdem prüft er die Signatur mit Hilfe des Public Keys.
  Schließlich generiert der Resolver eine (verschlüsselte) Anwort, die außer des geforderten Services wieder die empfangene SessionID enthält. Dadurch wird sichergestellt, daß die Antwort eine unmittelbare Reaktion auf die vorher gestellte Anfrage ist.

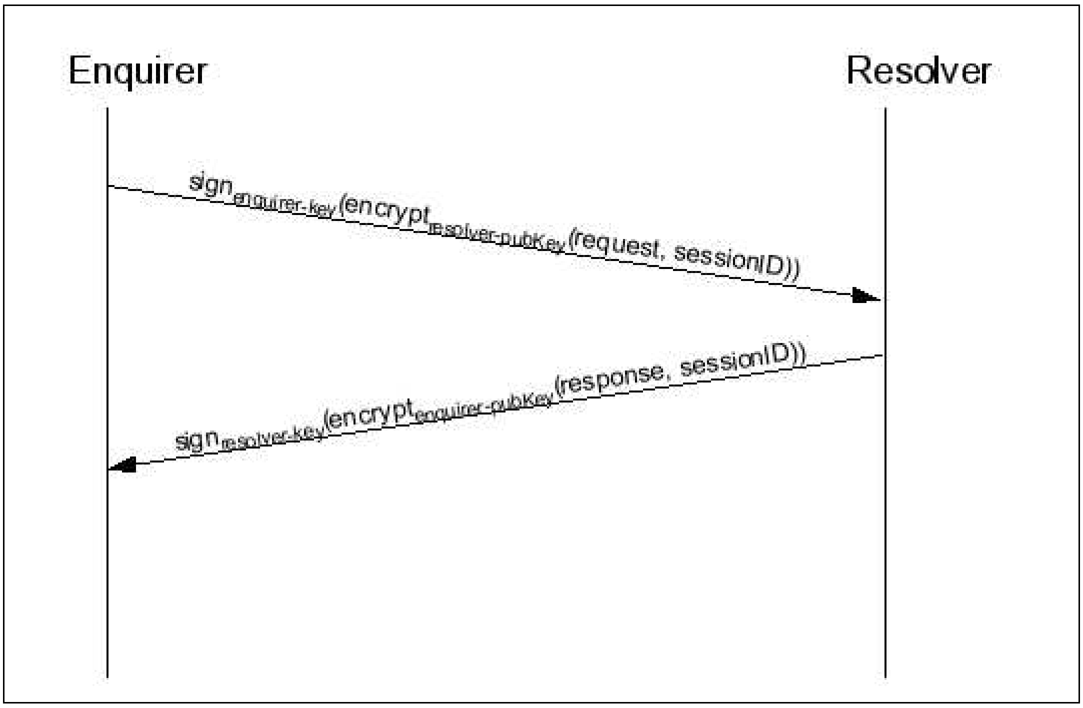

Abbildung 5.1: Der Ablauf einer Anfrage und der entsprechenden Antwort

Kapitel 6

Die Implementierung

Das folgende Kapitel beschreibt die konkrete Implementierung der P2P-Anwendung, die Architektur und Datenstrukturen und die verwendeten Hilfsmittel.

6.1 Architektur

Der grundlegende Aufbau des Netzes und seiner Komponenten zeigt Abblidung 6.1. Grundlage ist das P2P-Netz bestehend, aus beliebig vielen P2P-Clients. Diese Anwendungen können fungieren als Anfrage-Client (Enquirer), als Dienste-Anbieter (Resolver) oder auch beide Funktionen gleichzeitig erfüllen. Ausgangspunkt für jede Aktion ist immer ein Lesegerät, z. B. ein RFID-Leser. Die von ihm ausgelesenen Daten (1. ID und 2. der String, der den Service identifiziert) werden direkt an den P2P-Client übergeben. Wichtig dabei ist, das diese Verbindung zwischen Leser und Client als sicher betrachtet wird und daher die Daten auf diesem Weg nicht verschlüsselt werden. Erst die jetzt folgende Anfrage an das P2P-Netz durch den Enquirer gilt als unsicher und wird mit entsprechenden Verschlüsselungs- und Authentifizierungstechniken behandelt. Alle als Resolver arbeitenden P2P-Clients beantworten diese Anfrage, falls sie diesen Service anbieten. Die beim Enquirer ankommenden Ergebnisse werden nach dem FIFO-Prinzip abgearbeitet, um darunter das richtige Ergebnis auszuwählen. Wurde der Service von einem Resolver authentisch aufgelöst, so wird einem Ausgabegerät die Antwort übergeben. Auch diese Verbindung wird wieder als sicher betrachtet. Als Ausgabegeräte kann man sich unterschiedlichste Medien vorstellen, die evtl. auch mit dem Lesegerät kombiniert sein können: Displays, die einen Text/ Webseite etc. anzeigen oder akustische/ optische Signale, die z.B. eine Türöffnung anzeigen.

6.2 verwendete Tools und Umgebung

Die Anwendung wurde in dieser Studienarbeit zunächst für Desktop-Rechner entwickelt. Sicherlich wäre auch ein Einsatz auf mobilen Geräten - z. B. PDAs - sinnvoll, ist aber nicht Bestandteil dieser Studienarbeit. Ansonsten wurde die Applikation weitestgehend Plattform unabhängig entwickelt. Die wesentlichsten Anforderungen sind eine aktuelle Java-Laufzeitumgebung und GnuPG. Diese Voraussetzungen sollten alle derzeit gängigen Systeme, nämlich Windows, Linux

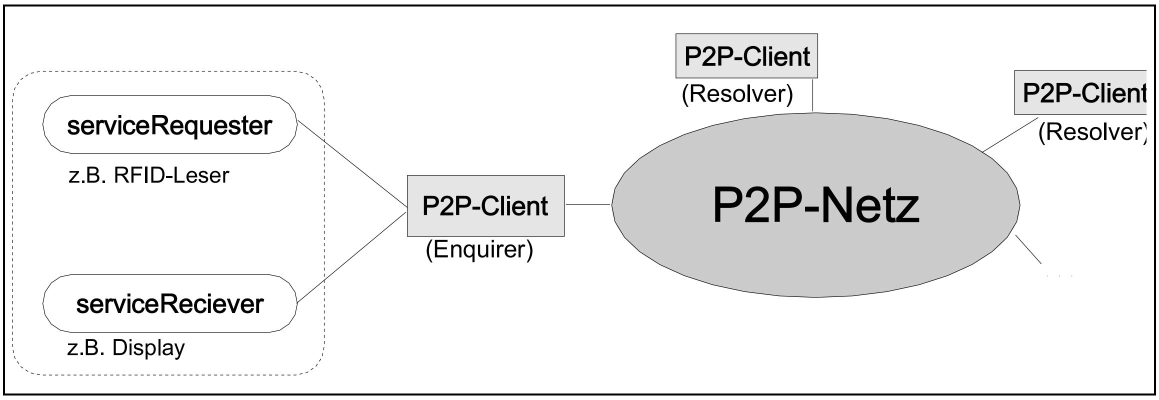

Abbildung 6.1: Die Architektur: serviceRequester fragt per RFID einen Dienst an, serviceReciever empfängt die Antwort.

und Mac OS X, erfüllen. Getestet wurde die Anwendung unter Windows 2000 und Linux (Kernel 2.4.18).
Auf die Entwicklung einer graphischen Benutzeroberfläche wurde zunächst verzichtet.

Für die Entwicklung wurden die folgenden Hilfsmittel verwendet:

- Sun Java SDK 1.3.x
- JXTA Plattform Version 2.1.00 (Release 06-08-2003)
  Im einzelnen werden davon die Pakete jxta.jar, jxtasecurity.jar, log4j.jar, cryptix-asn1.jar, cryptix32.jar, jxtaptls.jar benötigt. Es empfielt sich einfach das aktuelle 'JXTA Demonstration GUI' (instantp2p) zu benutzen, da darin alle benötigen Pakete in der korrekten Version enthalten sind.
- GnuPG Version 1.2.2
  GNU Privacy Guard - ein frei verfügbares Verschlüsselungstool, das auf OpenPGP[4] basiert.

Da die Anwendung nicht für ein spezielles Kartenlesegerät und Identifizierungssystem entwickelt werden sollte, mußte zunächst eine Testumgebung geschaffen werden. Zu diesem Zweck wurden die Java-Applikationen 'serviceReqeuster' und 'serviceReciever'. Sie simulieren das Szenario eines Kartenlesegerätes und eines Ausgabemediums, die mit dem P2P-Netz (über UDP) kommunizieren. In der Praxis könnte beides natürlich auch zu einem Gerät zusammengefaßt werden.
Der serviceRequester erwartet dabei eine Eingabe der Form serviceName#KartenID#publicKey, die er unverändert per UDP an die P2P-Applikation übergibt. Der serviceReciever wartet einfach auf einen Antwortstring und gibt diesen aus bzw. eine Fehlermeldung bei fehlgeschlagener Anfrage.

6.3 Struktur und Funktionen

Der Anwendung liegt die in Abbildung 6.2 gezeigte Klassenstruktur zu Grunde. Nicht berücksichtigt sind die im vorherigen Abschnitt beschriebenen Klassen serviceRequester und serviceReciever.

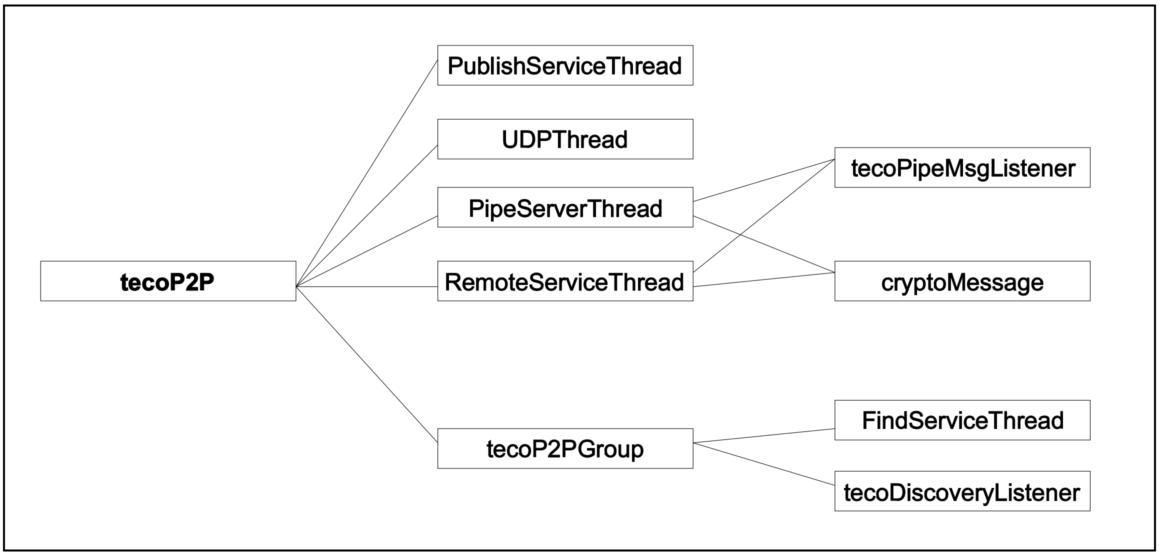

Abbildung 6.2: Die Struktur der P2P-Applikation

Im folgenden werden nun die wichtigsten Bereiche des Programms und ihre Funktionen beschrieben.

Programmstart

Wie Abbildung 6.2 zeigt, ist 'tecoP2P' die Hauptklasse, mit der auch die Anwendung gestartet wird. Während des Starts werden schon eine Reihe wichtiger Aufgaben erledigt:

- *tecoP2P.ini:* Vor allen anderen Aktionen wird ein Konfigurations-File gelesen, in dem der Benutzer Dinge wie UDP-Ports oder Dateinamen einstellen kann.
- *Anmeldung:* Zunächst wird sich dann der Client am P2P-Netz entsprechend der JXTA-Konfiguration (siehe oben) anmelden. Dazu wird das vorher festgelegte Login/ Paßwort abgefragt.
- *Pipe:* In einem ersten Schritt wird geprüft, ob ein PipeAdvertisement für dieses Peer existiert. Falls nicht, wird eines erzeugt, welches von nun an für alle Pipe-Aktionen auf diesem Peer verwendet wird. Es ist wichtig immer wieder das gleiche PipeAdvertisement zu verwenden, da dieses mit den Services publiziert wird und darüber alle Service-Anfragen kommen. Anschließend startet das Peer seinen Pipeserver-Thread, der auf Anfragen an

diese Pipe wartet und evtl. beantwortet. Dieser Pipeserver ist nur relevant, wenn das Peer auch als Resolver arbeitet, also auch eigene Dienste anbieten will.

- *Peer Groups:* Jetzt wird geprüft, ob dieses Peer eventuell Gruppen erzeugt hat. Falls ja, werden diese Gruppen im JXTA-Netz publiziert. Diese Funktion ist momentan eher nebensächlich, da die aktuelle Implementierung nur auf der NetPeerGroup funktioniert.
- *Services:* Falls dieses Peer irgendwelche Services anbietet, werden diese auf der entsprechenden Peer Group, momentan ausschließlich der NetPeerGroup, publiziert. Dabei muß beachtet werden, daß Services in JXTA eine bestimme (aber festlegbare) Lebenszeit haben. Aus diesem Grund müssen die Adverisements der Services (ModuleSpecAdvertisements) in regelmäßigen Intervallen wieder aufgefrischt werden, um anzuzeigen, daß der Client noch aktiv ist. Zu diesem Zweck wurde diese Aufgabe in einen separaten Thread (PublishServiceThread) ausgelagert. Der diese Auffrischung ca. alle 2 Stunden wiederholt.
- *UDP-Thread:* Zum Schluß wird noch der UDP-Thread gestartet. Er ist für die Kommunikation mit dem serviceRequester und -Reciever zuständig.

Nach dieser Startprozedur steht dem Benutzer ein textbasiertes Auswahlmenü mit allen ausführbaren Funktinonen zur Verfügung. Die wichtigsten werden im folgenden beschrieben.

Service erstellen

Eine wesentliche Funktion der Applikation ist das Erstellen und Publizieren eines Services. Prinzipiell ist vorgesehen, Services in Form von Zeichenketten oder Dateien anzubieten. Da derzeit lediglich die Zeichenketten-Variante vollständig implementiert ist, wird sich die Beschreibung auf diese beschränken.
Services werden in JXTA von drei Advertisements beschrieben: ModuleClassAdvertisement, ModuleSpecAdvertisement und ModuleImplAdvertisement. Es sollen hier aber nur die zwei wichtigsten kurz erläutert werden: ModuleClassAdvertisement ist eine Art Oberklasse für die Services, diese sind dort alle registrsiert. In der Implementierung wird es letztlich nur benutzt um eine ModuleSpecID zu erzeugen. Das ModuleSpecAdvertisement ist das wichtigste der drei, es definiert den eigentlichen Service. Nach dem wird auch gesucht, um einen Service zu finden. Es enthält Namen, ID etc. und v. a. ein PipeAdvertisement, um mit dem Peer zu kommunizieren und den Service abzufragen.

Die Erstellung eines funktionierenden Services erfolgt in zwei Schritten:

Zunächst wird das entsprechende ModuleSpecAdvertisement erzeugt und publiziert. Dazu muß der Benutzer einen Namen, eine Gruppe, in der der Service publiziert werden soll und den eigentlichen Service in Form eines Strings angeben. Dieser String wird dann in einer speziellen Service-Datei abgelegt, wie im nächsten Abschnitt erläutert wird. Der Name kann dabei beliebig - d.h. auch mehrfach benutzt - sein, da die richtige Variante später mit Hilfe des RFID-Tags und des Public Keys eindeutig ausgewählt werden. Als Gruppe sollte in der derzeitigen Implementierung ausschließlich die laufende Nr. 0 - die NetPeerGroup - ausgewählt werden. Nach dieser

Prozedur wird der Service ab sofort bei jedem Programmstart automatisch wieder publiziert.
Im zweiten Schritt müssen die RFID-Tags für den Service registriert werden. Dies geschieht momentan durch manuelle Eingabe der ID. Diese IDs werden dann ebenfalls in das Service-File geschrieben. Danach kann dieser Service von den erfassten RFIDs benutzt werden - sofern der entsprechende Public Key bekannt ist. Die Verteilung dieser Public Keys ist jedoch nicht Bestandteil dieser Studienarbeit.

Service anfragen

Eine andere wichtige Komponente der Applikation ist natürlich die Anfrage eines Services. Wie schon weiter oben beschrieben, geschieht dies über die Hilfsprogramme 'serviceRequester' und 'serviceReciever', die einen RFID-Leser und ein Ausgabegerät simulieren. So erhält also die Enquirer über einen UDP-Port eine Anfrage in Form von Servicenamen, RFID und einem zu benutzenden Public Key Namen.

Die Anwendung führt nach dem Erhalt dieser Informationen folgende Schritte aus:

- zuerst werden auf allen vorhandenen Peer Groups suchen initiiert, die nach dem ModuleSpecAdvertisements mit dem Servicenamen suchen. Dazu wird für jede Gruppe ein FindServiceThread gestartet um die Suche parallel durchzuführen. Die Suche wird mehrfach durchgeführt und per Timeout beendet. Die nach und nach eintreffenden Ergebnisse werden der Reihe nach mit nachfolgenden Schritten abgearbeitet. Auch hier wird für jede Anfrage ein RemoteServiceThread gestartet, um lange Wartezeiten zu vermeiden.

- Das PipeAdvertisement wird aus dem ModuleSpecAdvertisement extrahiert und es wird eine bidirektionale Pipe zu dem entfernten Peer aufgebaut.

- Es wird mit Hilfe der cryptoMessage Klasse eine verschlüsselte Nachricht erstellt. Diese besteht aus dem Namen des lokalen Public Keys, der ModuleSpecID des Dienstes, RFID und einer Session ID. Diese Session ID wird für jede Anfrage per Zufallsgenerator neu erzeugt. Verschlüsselt wird die Nachricht mit dem öffentlichen Schlüssel des Remote Peers; der Schlüssel muß vorher bekannt sein.
  Als Hilfsmittel zur Verschlüsselung wird GnuPG verwendet.

- Die Nachricht wird über die geöffnete Pipe an das andere Peer verschickt. Dieses erhält die Anfrage über den beim Programmstart aufgerufenen PipeServer. Ist das Peer im Besitz des passenden Private Keys, kann es die Nachricht entschlüsseln und weiter verarbeiten, ansonsten wird sie verworfen und der Server geht wieder in Wartestellung.

- Jetzt wird auf dem Remote Peer in der Service-Datei, auf die der nächste Abschnitt eingehen wird, geprüft, ob die übergebene KartenID für den angefragten Service - identifiziert durch die ModuleSpecID - registriert ist.

- Ist dies der Fall, so wird mit dem mitgeschickten Schlüsselnamen - der Schlüssel muß bekannt sein - eine Antwort generiert, ebenfalls wieder mit der cryptoMessage-Klasse. Die Antwort enthält den Service-String, die KartenID und wieder die erhaltene SessionID.

- Das immer noch auf die Antwort wartende lokale Peer empfängt die Nachricht und dechiffriert sie mit dem privaten Schlüssel. Jetzt wird noch die SessionID und KartenID mit den ursprünglich verschickten verglichen. Bei Übereinstimmung wird dem serviceReciever der Service-String als Antwort übergeben.

In Abbildung 6.3 wird das oben beschriebene Szenario noch einmal veranschaulicht.

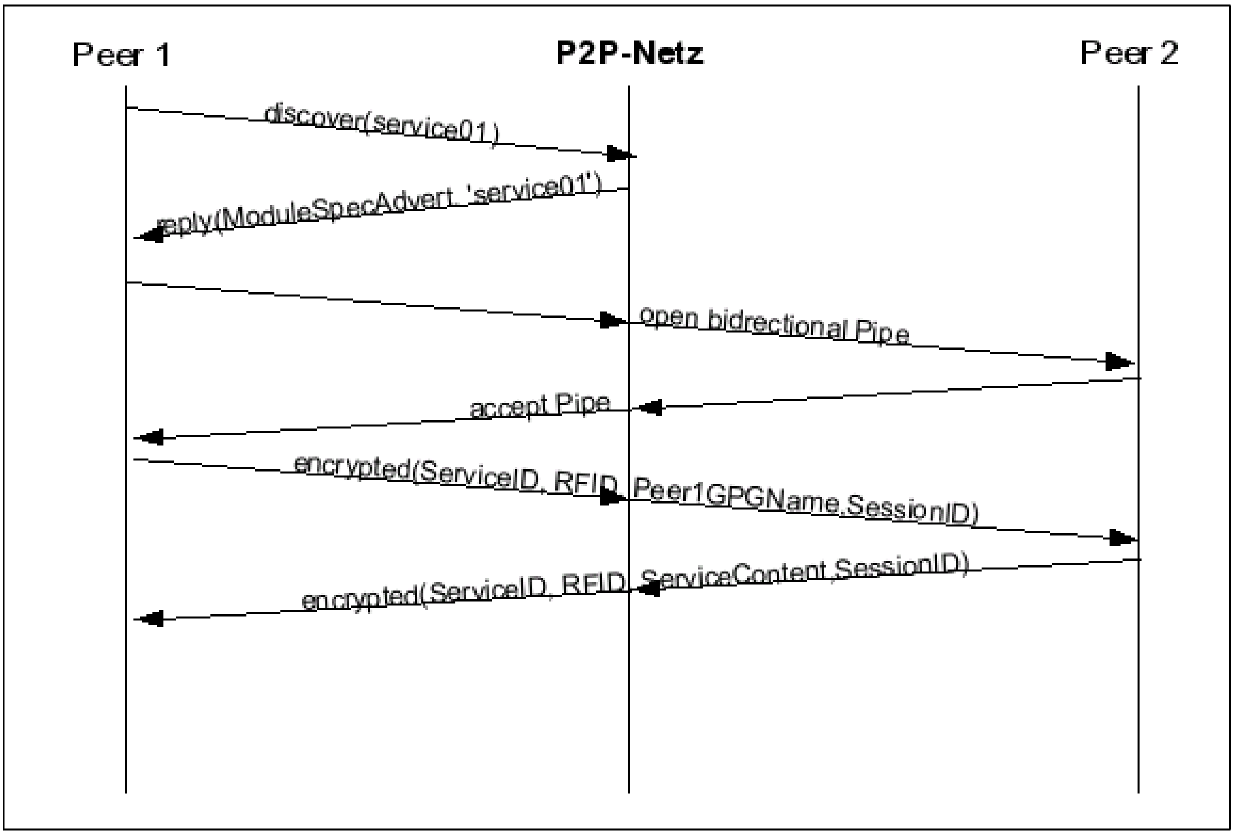

Abbildung 6.3: Anfrage eines Services

6.4 Daten

Die P2P-Anwendung verwendet im Verlauf der Benutzung verschiedene Dateien und Datenstrukturen, die im folgenden beschrieben werden. Im einzelnen sind das die Konfigurationsdatei (tecoP2P.ini), die PipeAdvertisement-Datei (pipe.adv), eine Servicedatei und eine Gruppendatei.

- In der Konfigurationsdatei (tecoP2P.ini) lassen sich der lokale GPG-Schlüssel, verschiedene Dateinamen und die UDP-Ports festlegen. Abbildung 6.4 zeigt eine Beispiel-Konfiguration mit allen derzeit verfügbaren Parametern. Für die Zukunft ist es sicher sinnvoll, weitere

```
 #set the name of the local GPG Key
localgpgkey = myKey
#the filenames
servicefile = services.dat
groupfile = groups.dat
#the UDP ports for communication with serviceRequester
and serviceReciever
INPORT = 1111;
OUTPORT=1112;
MYPORT=1113;
```

Abbildung 6.4: Beispiel einer tecoP2P.ini

Parameter, z. B. Timeouts, konfigurierbar zu machen. Die funktionsweise orientiert sich am weit verbreiteten Schema für Config-Dateien:
Variablenname = Wert
'#' zeigt eine Kommentarzeile an.

- In der Datei pipe.adv wird das PipeAdvertisement des Peers gespeichert. Wie bereits oben erläutert, wird es beim ersten Programmstart erzeugt und dann immer wieder verwendet.
- In der Gruppendatei - groups.ini oder entsprechend der Konfiguration in tecoP2P.ini - werden die Peer Groups im XML-Format abgelegt, die vom lokalen Peer erzeugt wurden um sie bei jedem Programmstart zu publizierem. Da die Gruppenfunktion derzeit nicht vollständig implementiert ist, soll darauf nicht weiter eingegangen werden.
- Eine entscheidende Rolle spielt die Servicedatei: services.ini oder wie in tecoP2P.ini festgelegt. Hier werden die vom Peer erzeugten Services inclusive der zugehörigen RFIDs - ebenfalls im XML-Format - gespeichert. Abbildung 6.5 zeigt ein solches Beispiel. Zu sehen sind die ModuleSpecID im SID-Tag, danach Name und Gruppe in der der Service publiziert wird. Der string-Tag wird bei Anfrage als Dienst zurückgegeben. In den ID-Tags können beliebig viele RFIDs gespeichert werden, die diesen Service benutzen dürfen.

```
.<?xml version=''1.0''?>
.<!DOCTYPE SERVICES>
.  <SERVICES>
.    <service>
.      <SID>
.        urn:jxta:uuid-B7F91A7E4BE44BA0AC10A8F2A3BF40DFE083D017760C43BABE1ABC9C1E35611506
.      </SID>
.      <name> serivce01 </name>
.      <groupName> NetPeerGroup </groupName>
.      <string>
.        Hello World, I'm the content of the service01
.      </string>
.      <ID> 123 </ID>
.      <ID> 456 </ID>
.    </service>
.    <service>
.      ...
.    </service>
.  </SERVICES>
```

Abbildung 6.5: Beispiel einer Servicedatei

Kapitel 7

Offene Fragen und Ausblick

Da diese Studienarbeit die gesamte Thematik sicher nicht erschöpfend behandeln konnte, bleiben einige Fragen unbeantwortet und einige Punkte verbesserungswürdig. Das folgende Kapitel soll einige dieser Punkte aufzeigen, mögliche Probleme und Lösungsansätze diskutieren.

7.1 Verteilung GPG-Schlüssel

Wie bereits zu Anfang erwähnt ein derzeit noch unbehandeltes Problem die Verteilung der Schlüssel. Im wesentlichen geht es hier um die Verteilung der Public Keys, die nicht komplett öffentlich zugänglich sein sollen, sondern nur den einzelnen berechtigten Enquirern bekannt sein dürfen. Die Nutzung des Peer-to-Peer Netzes scheint hier schwierig zu sein, da sich hier immer das Vertrauensproblem im Netz ergibt. Eine Variante wäre die Nutzung einer zentralen vertrauenswürdigen Instanz die die Schlüssel verteilt. Hier ergibt sich allerdings wieder das Problem der Überwachung und die evtl. fehlende Anonymität. Da diese Zentrale dann weiß, welcher Enquirer welche Dienste in Anspruch nehmen kann. Ein weiteres Problem ergibt sich aber auch nach der Verteilung Schlüssel: Prinzipiell kann ein Enquirer die erhaltenen Schlüssel auch an andere - eigentlich nicht authorisierte Personen - weitergeben. Dies ist insbesondere dann ein Problem, wenn das System z. B. für kostenpflichtige oder sicherheitskritische Dienste eingesetzt werden soll.

7.2 Performance und Skalierung

Das System reagiert in der aktuellen Version nach ca. 4-6 Sekunden mit einer Antwort. Diese Zeitverzögerung ist für viele Anwendungen nicht zufriedenstellend. Man stelle sich eine Tür vor. die sich erst nach 5 Sekunden Wartezeit öffnet. Daher sollte nach verschiedenen Möglichkeiten gesucht werden diese Zeit zu verkürzen.
Ein erster Erfolg kann durch Verbesserung einzelner Such- und Kommunikationsmethoden erzielt werden. Beispielsweise könnten Pipes offen gelassen werden in der Hoffnung, daß die nächste Anfrage an das selbe Peer geht, oder lokale Caches eingerichtet werden, die häufige Anfragen zwischenspeichern.
Eine weitere Hilfe können auch JXTA Peer Groups, vor allem in sehr großen Netzen sein. Mit

Hilfe dieser Gruppen könnten die Informationen im Netz gegliedert werden. Der Enquirer würde nicht das gesamte P2P-Netz nach einem Service befragen, sondern nur gezielt eine Peer Group, deren ID auf dem RFID gespeichert oder auch dem Enquirer fest vorgegeben sein kann. Somit könnte der Umfang der Suche erheblich eingeschränkt werden.

7.3 Benutzerfreundlichkeit

Die vorliegende Implementierung bietet momentan noch keine einfache und unkomplizierte Benutzung des Systems. Es ist lediglich eine Demonstration der Funktionalitäten. So wäre zunächst eine grafische Benutzeroberfläche wünschenswert, um Konfiguration, anlegen und publizieren von Services etc. zu erleichtern.
Desweiteren gestaltet sich die Eingabe der RFIDs derzeit sehr umständlich, da zunächst ein Service erzeugt werden und anschließend jede einzelne ID manuell eingegeben werden muß. Man stelle sich hier die Erfassung von Lebensmitteln im Supermarkt vor. Eine automatische Eingabe über einen Kartenleser, evtl. soar mit gleichzeitigem beschreiben des Mediums mit dem Service, würde hier sicher von Vorteil sein.

7.4 Portierung

Derzeit liegt die Anwendung als Java-Implementierung für PCs vor. Für dieses System kommen aber auch Anwendungsbereiche in Frage, an denen kein Desktoprechner verfügbar ist, sondern auf kleine, mobile Geräte zurückgegriffen werden muß. Ein vorstellbares Szenario wäre ein großes Warenlager, in dem alle eingelagerten Objekte, aber auch verschiedene Lagerbereiche - also Hallen, Räume, Regale etc. - mit RFIDs ausgestatttet sind. Eine Person, die auf der Suche nach einem bestimmten Objekt im Lager ist, könnte mit Hilfe eines mobilen Gerätes die Transponder auslesen und mit den erhaltenen Informationen - z. B. Inhalt von Kisten, Standortinformationen etc. - den Gegenstand leicht finden. Dies erfordert die Portierung der Applikation auf entsprechende mobile Plattformen. JXTA bietet hierzu schon verschiedene Hilfestellungen an, z. B. die minimal-edge-peers oder die JXTA Plattform für die Java Micro Edition.

Kapitel 8

Ähnliche und verwandte Projekte

Aufgrund der immer größer werdenden Bedeutung von RFID- und ähnlichen Systemen gibt es bereits einige andere Ansätze zur Auflösung von RFIDs. Exemplarisch sollen im Folgenden drei Systeme kurz vorgestellt und Unterschiede und Gemeinsamkeiten aufgezeigt werden.

8.1 AutoID-Center

Das AutoID-Center[18] ist wohl momentan das bekannteste System für RFID-Auflösung, verfolgt aber einen völlig anderen Ansatz. AutoID-Center nutzt keine P2P-Technik, sondern basiert auf einem klassischen hierarchischen Client-Server-Modell. Das hauptsächliche Einsatzgebiet für dieses System liegt in der Warenhaltung, -auslieferung und -bestellung.
Die Kernstücke dabei bilden die Savant-Applikation, der Object Name Service (ONS) Server und der Physical Markup Language (PML) Server. Ausgangspunkt für jede Aktion ist dabei immer Savant. Das Programm bekommt dabei in der Regel einen Elektronischen Produkt Code (EPC) - gespeichert auf dem RFID von z. B. einer Lebensmittelverpackung - vom Kartenleser übergeben. Mit diesem EPC stellt Savant eine Anfrage an den Object Name Service. Diesen könnte man in etwa mit dem Domain Name Service (DNS) vergleichen, welcher im Internet für Adressauflösung zuständig ist. Der ONS-Server sucht anhand des übergebenen EPC einen weiteren Server, dem PML-Server, der Informationen über das entsprechende Produkt liefert. Bei der "Physical Markup Language" handelt es sich um eine, auf XML basierende, Sprache zur Beschreibung von physischen Objekten. Diese PML-Daten werden an die anfragende Savant-Applikation zurückgeliefert.
Dieser Lösungsansatz bringt aber ein Problem mit sich. Es sind durch diese hierarchische Struktur zentrale Instanzen nötig, die das Netz kontrollieren und überwachen können. Dies hat zur Folge, daß es schwierig wird, sich anonym durch die Welt zu bewegen, sobald man einen RFID-Tag an oder bei sich trägt. Genauso können auch nicht beliebig Dienste angeboten werden, sie müssen immer am ONS-Server registriert sein. Andererseits gibt es beim hierarchischen Ansatz kein Vertrauensproblem. Im Gegensatz zum P2P-Netz, das grundsätzlich als vertrauensunwürdig angesehen werden muß, kann man sich darauf verlassen, daß zentrale Server authentische Daten zurückliefern, ähnlich wie sich jeder Internetbenutzer darauf verläßt, daß seine Domains vom DNS-Server korrekt aufgelöst werden.

8.2 Stockway/ WWAI

Hinter dem Kürzel WWAI[20] verbirgt sich der Begriff World Wide Article Information Protocol. Dabei handelt es sich um ein XML-basiertes Protokoll, entwickelt von der finnischen Firma Stockway[20], das auf TCP/IP aufsetzt. Genau wie diese Studienarbeit setzt Stockway auf ein dezentrales Netzwerk. Die grundlegende Idee war, z. B. verschiedene Unternehmen in der Fertigungskette eines Produkts mit Hilfe von WWAI miteinander zu verbinden, um beispielsweise Informationen über Standort oder Zustand eines Produkts zu verteilen. Auch Stockway verwendet dabei keine zentralen Indexlisten oder Ähnliches. Gemeinsamkeiten finden sich auch bei der Wahl des Identifkationssystems: Es wird kein spezielles System vorausgesetzt, es ist lediglich eine eindeutige ID nötig. Favorisiert wird allerdings auch hier RFID.
Es gibt aber zwei wesentliche Unterschiede: ein wesentlicher Aspekt bei Stockway ist offensichtlich die Lokation von Gegenständen oder den Informationen dazu, was gegen eine Vorgabe dieser Studienarbeit verstoßen würde: die Anonymität. Der andere Punkt ist die Zuordnung von Informationen: WWAI hält für jede ID ein eigene Information vor. Der in dieser Arbeit beschriebene Ansatz ist jedoch etwas allgemeiner und arbeitet mit zunächst unabhängigen Services, bei denen die IDs registriert werden müssen. Im Extremfall könnte natürlich auch hier jede ID einen eigenen Service haben. Umgekehrt wäre es aber mit dem Stockway-Modell recht umständlich, ein und die selbe Information, wie z.B. die Zusammensetzung eines Lebensmittels, für viele verschiedene IDs anzubieten.

8.3 Cooltown

Cooltown[21] ist ein Forschungsprojekt von HP, das versucht Internettechnologie, mobile und drahtlose Geräte miteinander zu kombinieren. Dabei setzt Cooltown zunächst auf der HTTP-Schicht auf unter der Annahme, daß in Zukunft Web-Technologie überall verfügbar sein wird. Ziel ist es dabei einerseits, unterschiedlichste mobile Geräte drahtlos mit Informationen zu versorgen und andererseits diese Geräte in einen lokalen Kontext zu bringen, d. h. beispielsweise Interaktion mit lokalen Geräten und Gegenständen der realen Welt zu ermöglichen. Zu diesem Zweck muß die Umgebung mit verschiedenen Tags zur Identifikation ausgestattet sein, die dann entweder von Cooltown aufgelöst werden (indirect sensing) oder die selbst Informationen über ihr Objekt liefern können (direct sensing). In der ursprünglichen Cooltown-Version wurde dazu eine URL von sog. Beacon Tags als Identifizierung benutzt. Diese lieferte dann die virtuelle Presenz des Objektes aus dem Internet. Das System wurde aber erweitert und funktioniert mit beliebigen Systemen wie RFID oder Barcode.
Schon an der Tatsache, daß die Auflösung mit Hilfe von URLs erfolgt und die zurückgelieferten Informationen letztendlich Webseiten sind, läßt sich erkennen, daß Cooltown keinen P2P-Ansatz benutzt. Ähnlichkeiten zu dieser Studienarbeit zeigen sich hingegen in der Resourcenzuordnung: Resourcen - oder in dieser Arbeit 'Services' genannt - können einer, oder auch beliebig vielen IDs zugeordnet werden.

Literaturverzeichnis

[1] JXTA Project - Peer-to-Peer-Plattform
http : //www.jxta.org

[2] Einführung, Dokumentation und Programmierhinweise zu JXTA
http : //www.jxta.org/docs/JxtaProgGuide_v2.pdf

[3] Open Source Public Key Verschlüsselung - GNU Privacy Guard
http : //www.gnupg.org/

[4] OpenPGP - der offene Verschlüsselungs-Standard
http : //www.openpgp.org

[5] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, RFC-Standard 2440: OpenPGP Message Format
http : //www.ietf.org/rfc/rfc2440.txt

[6] F. Poschadel: Diplomarbeit: Einsatzmöglichkeiten kryptographischer Methoden zur Signatur und Verschlüsselung im Krankenhaus
http : //www.iti.cs.tu − bs.de/ waetjen/Diplomarbeiten/Poschadel.pdf

[7] Barcode
http : //www.barcode − 1.com

[8] RFID
http : //www.rfid.org

[9] RFID Basics Primer
http : //www.aimglobal.org/technologies/rfid/resources/RFIDPrimer.pdf

[10] Smart-Its
http : //smart − its.teco.edu

[11] M. Beigl, T. Zimmer, A. Krohn, Ch. Decker, Ph. Robinson: Smart-Its - Communication and Sensing Technology for UbiComp Environments
http : //www.teco.edu/˜michael/publication/tech$_r$*1.pdf*

[12] Smartcard
http : //www.smartcard.co.uk

[13] Napster/ OpenNap
http : //opennap.sourceforge.net/

[14] Gnutella Protokoll
http : //www9.limewire.com/developer/gnutella$_p$*rotocol*$_0$*.4.pdf*

[15] K. Sripanidkulchai, Carnegie Mellon University: The popularity of Gnutella queries and its implications on scalability
http : //www − 2.cs.cmu.edu/ kunwadee/research/p2p/gnutella.html

[16] The Freenet Project
http : //www.freenetproject.org

[17] Direct Connect
http : //dcplusplus.sourceforge.net/

[18] Auto-ID Center
http : //www.autoidcenter.org

[19] S. Sarma, D. Brock, K. Ashton: The Networked Physical World - Proposals for Engineering The Next Generation of Computing, Commerce and Automatic Identification
http : //www.autoidcenter.org/publishedresearch/MIT − AUTOID − WH − 001.pdf

[20] Stockway/ WWAI
http : //www.stockway.fi/wwai.html

[21] Cooltown
http : //www.cooltown.com

[22] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra :People, Places, Things: Web Presence for the Real World
http : //www.cooltown.com/dev/wpapers/jbartonppt.pdf