JXTA: Einführung und Überblick

Robert Becker
rbecker@inf.fu-berlin.de

Franziska Liebsch
fliebsch@inf.fu-berlin.de

Yann-Rudolf Michel
ymichel@inf.fu-berlin.de

Daniel Müller
dmueller@in.fu-berlin.de

16. Juli 2004

Zusammenfassung

Projekt JXTA ist ein von Sun Microsystems ins Leben gerufenes Entwickler-Team, das aus der Notwendigkeit heraus geformt wurde, eine solide und ausgereifte Basis für zukünftige P2P Entwicklungen zu haben. Gegründet wurde es Arpil 2001 unter der Leitung von Bill Joy und Mike Clary, die das Team bis heute führen.

JXTA fungiert mit seinen Protokollen als virtuelles overlay Netzwerk, das auf dem Internet und auf nicht IP - Netzen aufsetzt. Peers können somit unbeeinträchtigt durch Firewalls miteinanander interagieren.

Fehlende Standardisierungen und somit Individuallsungen eines jeden P2P-Technologie Unternehmens haben heute dazu gefhrt, dass Nutzer meistens mehrere parallele P2P-Netze nutzen. JXTA ist der kommende Standard auf den sich alle P2P-Lsungen sttzen und damit Interoperabilitt der verschiedenen Systeme gewhrleisten sollen.

Inhaltsverzeichnis

1 Konzepte und Architektur

In der Einführung werden grundlegende Konzepte und Begriffe des P2P erklärt. Es wird erläutert wie JXTA diese Grundbegriffe in einen programmiersprachenunabhängigen Rahmen fasst und sie spezifiziert. Ausserdem wird eine Motivation für die Entstehung von Projekt JXTA herausgearbeitet. Es wird die Interaktion zwischen den grundlegenden Elementen der JXTA Spezifikation eingeführt. Hinzu kommt eine Veranschaulichung der JXTA Architektur.

1.1 Konzepte

Es wurde erkannt, dass nach der ersten Aufregung der frühen P2P Lösungen wie Napster und Gnutella der Bedarf besteht nach einer gemeinsamen und wohlspezifizierten Sprache für P2P. Genau hier setzt JXTA an, es definiert auf sehr abstrakte Weise eine Reihe von Protokollen und XML-Formaten, die für alle Applikationen, die P2P nutzen möchten, eine ausgereifte und wiederverwendbare Basis sein können. Eine solche gemeinsame Basis macht auch Interoperabilität zwischen völlig verschiedenen P2P Applikationen möglich, etwas das es so vorher nicht gegeben hat.

JXTA leitet sich aus dem englischen juxtapose ab, was soviel wie Nebeneinanderstellung bedeutet. Die Gründer von Projekt JXTA wollen damit ausdrücken, dass ihre Architektur neben der klassichen Client/Server Architektur existieren wird, anstatt diese zu verdrängen.

Die JXTA Spezifikation 1.0 definiert sechs zentrale Protokolle

- Peer Discovery Protocol - erlaubt es Peers, sich untereinander zu finden, ein Vorgang der discovery genannt wird.
- Peer Resolver Protocol - erlaubt es Peers, generische Anfragen zu verschicken und zu empfangen.
- Rendezvous Protocol - wird benutzt, um (bla bla ... Advertisments?)
- Peer Information Protocol - bietet die Möglichkeit generische Informationen über andere Peers abzufragen.
- Pipe Binding Protocol - hat die Funktion, einen virtuellen Kommunikationskanal pipe an einen Kanalendpunkt endpoint zu binden.
- Endpoint Routing Protocol - definiert eine Reihe von Nachrichten, die es möglichen machen, eine Route von einem Quell-Peer zu einem Ziel-Peer aufzubauen.

Die Protokolle werden im folgenden Kapitel ausführlich erläutert.

1.1.1 Peer

Der Peer ist der verarbeitende Grundbaustein eines P2P-Netzes. Ein Peer muss nicht notwendigerweise ein einzelner Computer sein. Ein Peer kann durchaus aus mehreren Computern bestehen, z. B. um eine besonders hohe Last verarbeiten zu können oder weniger fehleranfällig

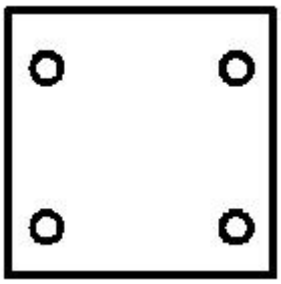

Abbildung 1: Peer mit Endpoints

zu sein. Ein Peer kann auch nur ein „mobile device“ sein, also kein kompletter Desktop-PC, sondern ein Handheld oder PDA. Es ist eine zentrale Anforderung bei JXTA's Entwicklung, es Geräten jeder Art und Gröe zu ermöglichen, dem P2P-Netz beizutreten.

Daraus ergibt sich folgende Definition eines Peers.

Def.: Ein Peer ist eine Einheit in einem P2P-Netz, die in der Lage ist nützliche Arbeit für das P2P-Netz zu verrichten und die Ergebnisse dieser Arbeit zu anderen Peers zu kommunizieren, direkt oder indirekt([9]).

Es gibt drei verschiedene Arten von Peers in Form von Rollen. Ein Peer kann mehrere Rollen annehmen. Die einfachste Art von Peer ist der *Simple Peer*. Der Simple Peer dient als Schnittstelle zum Endbenutzer des P2P-Netzes und bietet ihm eine Benutzeroberfläche, die es ihm ermöglicht *Services* bereit zu stellen und in Anspruch zu nehmen. ber diese Services ergibt sich auch das Konzept von nützlicher Arbeit; sie defnieren, was an Arbeit im P2P-Netz verrichtet wird. Da ein Simple Peer oft durch Netzwerk-Barrieren nur beschränkt mit dem P2P-Netz kommunizieren kann, trägt er am wenigsten Verantwortung im P2P-Netz. Solche Netzwerk-Barrieren können Firewalls sein, aber auch Proxies oder eine Adress-bersetzung wie NAT. JXTA besitzt ausgefeilte Mechanismen zum Durchdringen dieser Barrieren. Ein plötzliches Austreten des Simple Peers aus dem P2P-Netz bringt keine Beeinträchtigung aller mit sich.

Die zweite Art von Peer ist der *Rendezvous Peer*. Der Rendezvous Peer hat die Aufgabe, anderen Peers das Auffinden von Netzwerkressourcen (Services, Advertisments) zu erleichtern. Das Auffinden einer Netwerkressource wird *Discovery* genannt. Die Entlastung des P2P-Netzes erreicht der Rendezvous Peer durch eine Kombination von Caching und Hashing der an ihn gerichteten Anfragen. Gleichzeitg verringert der Rendezvous Peer durch seine netzentlastende Aufgabe die Latenz des gesamten P2P-Netzes, so dass Suchen im P2P-Netz schneller ablaufen und die gesamte Reaktionszeit des P2P-Netzes erhöht wird. Zur besseren Erreichbarkeit befindet sich der Rendezvous Peer

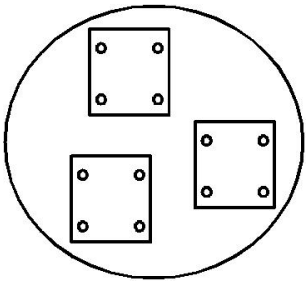

Abbildung 2: Peer Group

im allgemeinen ausserhalb privater Netzwerke im öffentlichen Internet. So ist seine Erreichbarkeit auch durch Netzwerk-Barrieren hindurch gewährleistet.

Die dritte Art von Peer ist der *Router Peer*. Der Router Peer befindet sich an der Grenze von Netzwerken und ist zentrales Element beim Druchbrechen von Netzwerk-Barrieren. Durch den Router Peer wird es erst möglich, z. B. firewall-geschützte Simple Peers in vollem Umfang an der P2P-Kommunikation teilnehmen zu lassen. Der Router Peer stellt anderen Peers eine Liste mit Routing-Informationen bereit. Diese Informationen können andere Peers abfragen, um zu erfahren, wie und auf welchem Weg sie Nachrichten zu bestimmten Peers senden können. Router Peers führen ihre Arbeit durch, indem sie Nachrichten für nicht direkt erreichbare Peers zwischenlagern und sie auf Anfrage des Peers an ihn weiterleiten. Auch der Router Peer verringert die Netzwerk-Last, da nicht immer wieder neue Routen gesucht werden müssen, sondern diese auf den Router Peers gelagert werden und abgefragt werden können.

1.1.2 Peer Groups

Peer Groups (siehe Abbildung 2) bieten dem P2P-Netz eine Möglichkeit zur grundsätzlichen Einteilung und Organisation. Eine Peer Group ist eine Menge von Peers, die sich selbst zu einer Gruppe organisieren, um gemeinsam nützliche Arbeit für die Gruppe zu verrichten.

Was nützliche Arbeit hier bedeutet, definiert jede Gruppe für sich selbst. Nützliche Arbeit kann z. B. das Bereitstellen von Nachrichten-Meldungen sein, das Tauschen von Dateien oder das Bereitstellen eines besonders ausfallresistenten Web-Services. Es kann verschiedene Voraussetzungen zum Beitritt zu einer Peer Group geben. JXTA bietet grundlegende Möglichkeiten zur Peer Authentifizierung (s. Peer Groups and Services). Eine Peer Group kann bestimmte Services nach aussen anbieten und eine Untermenge ihrer Services nur an Mitglieder der Gruppe anbieten.

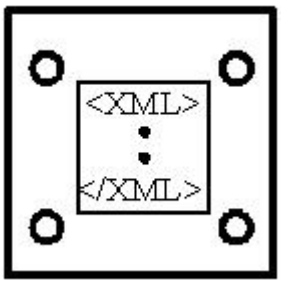

Abbildung 3: Peer mit Advertisment

1.1.3 Services

Services sind die eigentliche Motivation für Peers, sich in eine P2P-Netz zusammenzuschliessen. Wie oben schon erwähnt haben Services in ihrem JXTA-Verständnis ein sehr breites Spektrum. Was auch immer als nützliche Arbeit verstanden wird, kann über Services dem P2P-Netz bereit gestellt oder konsumiert werden. Solch grundlegende Arbeiten wie Suchen oder Indizieren im P2P-Netz gehören zu den *Core Services* der JXTA-Platform und sind schon vorhanden. Weitere Core Services sind z. B. das Auffinden von Routing-Informationen durch das Anfragen an einem Router Peer. Andere komplexere Services können auf Grundlage dieser Core Services realisiert und angeboten werden.

1.1.4 Advertisments

Advertisments (siehe Abbildung 3) dienen zur Beschreibung aller Ressourcen im P2P-Netz. Sofern eine Ressource im P2P-Netz existiert, wird zu ihr eine Beschreibung in Form eines Advertisments bereit gestellt.

Diese Beschreibung der Ressource, sei es nun ein Peer, eine Peer Group, ein Service, eine Pipe oder weitere Ressourcen, muss nicht zwingend auf dem anbietenden Peer gelagert sein. Die Rendezvous Peers haben die Aufgabe, Advertisments zu sammeln, zu verwalten und effektiv bereit zu stellen. Ein Advertisment ist ein XML-Dokument, das die Ressource ausreichend beschreibt. Es gibt keine DTDs oder Schemata für Advertisments, da die beschriebenen Ressourcen sehr vielfältig sein können. Die einzige Anforderung an das XML der Advertisments ist, das es wohlgeformt sein muss, also z. B. alle XML-Tags geschlossen sind.

1.1.5 Pipes und Endpoints

Zur tatsächlichen bertragung von Daten, bei JXTA generisch *Messages* genannt, werden noch zwei weitere Konzepte benötigt, die Pipes und Endpoints. Pipes (siehe Abbildung 4) sind virtuelle, asynchrone

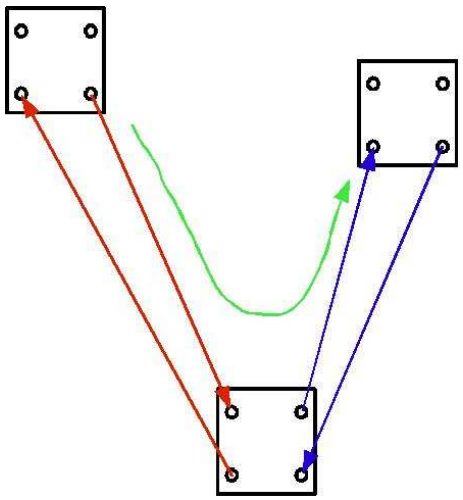

Abbildung 4: Pipes und Peers mit Endpoints

Kommunikationskanäle. Eine Pipe ist immer unidirektional. Eine bidirektionale Verbindung wird über zwei entgegengesetzt gerichtete unidirektionale Pipes erreicht. Eine Pipe ermöglicht den Transport von Messages von einem Endpoint zum nächsten. Eine Pipe kann sich über mehr als zwei Endpoints erstrecken.

Endpoints sind sowohl Quelle als auch Ziel jeder Kommunikation im P2P-Netz. Da alle Kommunikation über generische Messages abgewickelt wird, die über Pipes verschickt werden, ist diese zentrale Forderung garantiert. Ihre Einhaltung ermöglicht erst die von JXTA geforderte Interoperabilität und bietet eine grundlegende Sicherheit der Kommunikation. Ein Enpoint entspricht im allgemeinen einer Netzwerkschnittstelle. Es kann sich aber auch um z. B. eine URL handeln. Bei Router Peers sind die Endpoints oft http-URLs, da die Erreichbarkeit durch http auch aus den restriktivsten Netzwerken heraus fast immer gewährleistet ist.

1.2 Architektur

JXTA's Architektur (siehe Abbildung 5) besteht aus drei Schichten, die aufeinander aufbauen. Die Grundlage bildet die Schicht JXTA Core. In ihr werden die schon beschriebenen essentiellen Sevices bereitgestellt, wie das Verbinden von Endpoints mit Pipes, das Abfragen von Advertisments oder das Anfordern von Peer Status-Informationen. Diese sogenannten Core Services bilden somit die Grundlage für die zweite Schicht, die JXTA Services. Mit Hilfe der Core Services realisiert die JXTA Services Schicht komplexere Services wie z. B. JXTA Search, JXTA's eingebaute Suchfunktion. Auch einface Datei-Transfer-Services werden von der zweiten Schicht bereitgestellt. Die zweite Schicht wird

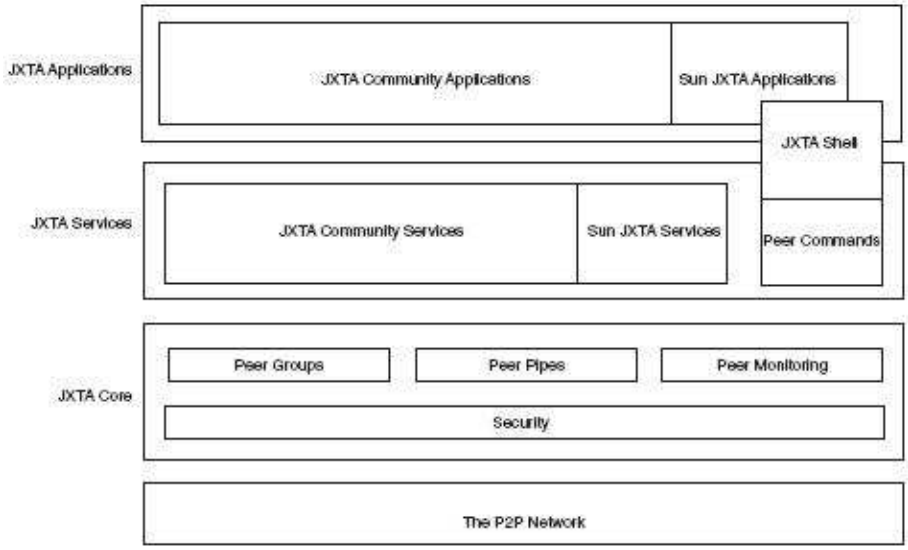

Abbildung 5: JXTA Architektur

sowohl von den Sun-Entwicklern als auch von der JXTA Community entwickelt und gepflegt. Mit Hilfe ihrer komplexen Services ist es dann möglich, in der dritten Schicht die Applikationen zu entwickeln, die JXTA als Grundlage für ihre P2P-Funktionalität nutzen. Zwischen JXTA Services und JXTA Applications gibt es noch die von Sun entwickelte JXTA Shell. Die JXTA Shell ist eine minimale konsolenbasierte Benutzerschnittstelle zu den JXTA Services. Mit ihr lassen sich leicht die Primitive der JXTA Kommunikation wie Pipes und Messages und deren Interatkion demonstrieren. Applikationen werden wieder sowohl von Sun als auch von der Community entwickelt. Beide Etnwickler stellen schon ausgereifte Applikationen zur sofortigen Verwendung bereit. Sie reichen vom bekannten P2P-Chat-Programm ala ICQ bis zu P2P-basierter Groupware.

Im folgenden Kapitel werden jetzt die sechs zentralen Protokolle von JXTA vorgestellt.

2 JXTA Protokolle

Im Folgenden werden die JXTA - Protokolle im Einzelnen beschrieben. Sie dienen dazu den einzelnen peers den Kontakt miteinander zu ermöglichen, miteinander zu kommunizieren und die *P2P* Applikationen zu steuern. Mit Hilfe der Protokolle ist es einfacher *P2P* Anwendungen zu realisieren.

2.1 Das Peer Discovery Protocol (PDP)

Dieses Protokoll beschreibt, wie die einzelnen peers andere peers, Gruppen, Services oder Kanäle finden können bzw. dürfen. Es handelt sich also um einen Suchmechanismus, um Informationen über das *P2P* Netzwerk zu sammeln.

Im Folgenden wird das Nachrichtenformat dieses Protokolls näher beschrieben.

Das Peer Discovery Protocol beinhaltet zwei Nachrichtenformate:

- Ein Anfrageformat, um advertisements zu finden (Discovery Query Message)
- Ein Antwortformat, um auf eine Anfrage zu antworten (Discovery Response Message)

Damit ein Peer einen anderen Peer finden kann, schickt er also eine Discovery Query Message an alle ihm bekannten Peers und Rendezvous Peers.

Wenn ein normaler Peer diese Nachricht erhält, sucht er in seinem lokalem Cache nach passenden Advertisements. Wenn er ein solches findet, sendet er eine Discovery Response Message direkt zu dem Peer, der die Anfrage stellte.

Wenn ein Rendezvous Peer die Anfrage erhält, behandelt er die Anfrage und kann eine Discovery Response Message mit dem Advertisements aus seinem Cache an den Absender der Anfrage zurück schicken. Auerdem schickt er die Nachricht an alle ihm bekannten normalen Peers weiter, welche dann wieder direkt an den Absender der Anfrage antworten.

Im Folgenden werden die zwei Nachrichtenformate beschrieben.

2.1.1 Die Discovery Query Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:DiscoveryQuery>
    <Type>...</Type>
    <Threshold>...</Threshold>
    <PeerAdv>...</PeerAdv>
    <Attr>...</Attr>
    <Value>...</Value>
  </jxta:DiscoveryQuery>
```

Die Werte der Elemente können wie folgt aussehen:

Type ist ein integer, wobei 0 für eine Anfrage an peer advertisements, 1 eine Anfrage für peer Gruppen advertisements, 2 für eine Anfrage an irgendein anderen Typen des advertisements steht.

Theshold repräsentiert die maximale Anzahl der advertisements, welche von einem peer beantwortet werden soll. - *optional* -

PeerAdv enthält das peer advertisement für diesen peer - *optional* -

Attr repräsentiert den Namen des angeforderten Elements - *optional* -

Value beinhaltet den Wert des Elements, welches als Antwort zurückgegeben werden soll - *optional* -

2.1.2 Die Discovery Response Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:DiscoveryResponse>
    <Type>...</Type>
    <Count>...</Count>
    <PeerAdv>...</PeerAdv>
    <Attr>...</Attr>
    <Value>...</Value>
    <Response Expiration="expiration time">...</Response>
  </jxta:DiscoveryResponse>
```

Die Werte der Elemente können wie folgt aussehen:

Type ist ein integer, wobei 0 für eine Antwort an peer advertisements, 1 eine Antwort für peer Gruppen advertisements, 2 für eine Antwort an irgendein anderen Typen des advertisements steht.

Count enthält die Anzahl der Antwortelemente in einer Nachricht - *optional* -

PeerAdv enthält das peer advertisement der Antwort des peers zu der ursprünglichen Discovery Query Message - *optional* -

Attr repräsentiert den Namen des angeforderten Elements - *optional* -

Value beinhaltet den Wert des Elements, welches als Antwort zurückgegeben werden soll - *optional* -

2.2 Das Peer Resolver Protocol (PRP)

Hierbei handelt es sich um ein Protokoll, welches benutzt wird um die Nachrichten die über das Peer Discovery Protocol geschickt werden in ein allgemeines Format umzuwandeln, um sie dann zu entfernten Peers zu schicken.

Es definiert, dass die Nachrichten im *request / response* Format verschickt werden.

Das Peer Resolver Protocol beinhaltet deshalb ebenfalls zwei Nachrichtenformate:

- Ein Nachrichtenformat, um Anfragen zu senden (Resolver Query Message)
- Ein Nachrichtenformat, um auf Anfragen zu antworten (Resolver Response Message)

Der Ablauf zwischen den Peers einer Gruppe sieht also wie folgt aus: Ein Peer sendet eine Resolver Query Message an alle ihm bekannten Peers und Rendezvous Peers.

Wenn ein Rendezvous Peer solch eine Anfrage erhält, sucht er bei sich nach einem registrierten Resolver Handler. Dieser überprüft, ob es sich um eine gültige Anfrage handelt, wenn nicht, wird die Anfrage nicht weiter propagiert. Wenn sie gültig ist, wird eine Resolver Response Message an den Absender der Anfrage geschickt. Zusätzlich wird die Anfrage an alle ihm bekannten Peers weiter gesendet.

Wenn nun ein normaler Peer eine solche Anfrage erhält, sucht er ebenfalls nach einem registrierten Resolver Handler. Wenn er einen passenden gefunden hat, wird die von ihm erstellte Resolver Response Message an den Absender der Anfrage geschickt.

2.2.1 Die Resolver Query Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:ResolverQuery xmlns:jaxta= "http://jxta.org">
    <HandlerName>...</HandlerName>
    <Credential>...</Credential>
    <QueryID>...</QueryID>
    <SrcPeerID>...</SrcPeerId>
    <Query>...</Query>
  </jxta:ResolverQuery>
```

Die Werte der Elemente können wie folgt aussehen:

HandlerName enthält einen eindeutigen Name des Resolver service auf dem Zielpeer, welcher aufgerufen wird, um die Anfrage zu bearbeiten.

Credential enthält ein Authetifizierungstoken, welches den Quellpeer und seine Authetifizierung für die Anfrage an die Gruppe beinhaltet. - *optional* -

QueryID stellt eine eindeutige ID für die Anfrage dar. - *optional* -

SrcPeerID beinhaltet die Id des Senders der Anfrage.

Query beinhaltet die Anfrage, welche an die anderen peers geschickt wird.

2.2.2 Die Resolver Response Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:ResolverResponse xmlns:jxta= "http://jxta.org">
    <HandlerName>...</HandlerName>
    <Credential>...</Credential>
    <QueryID>...</QueryID>
    <Response>...</Response>
  </jxta:ResolverResponse>
```

Die Werte der Elemente können wie folgt aussehen:

HandlerName enthält einen eindeutigen Name des Resolver service auf dem Zielpeer, welcher aufgerufen wird, um die Antwort zu bearbeiten.

Credential enthält ein Authetifizierungstoken, welches den Quellpeer und seine Authetifizierung für die Antwort an die Gruppe beinhaltet. - *optional* -

QueryID stellt eine eindeutige ID für die Antwort dar. - *optional* -

Response beinhaltet die Antwort, welche an die anderen peers geschickt wird.

2.3 Das Rendezvous Protocol (RP)

Das Rendezvous Protocol ist verantwortlich dafür, dass Nachrichten in den JXTA Gruppen propagiert werden können. Es bildet die Grundlage für das Peer Resolver Protocol im letzten Kapitel. Bevor ein Peer einen Rendezvous Peer benutzen kann, um Anfragen über ihn zu senden, muss er sich zuerst mit ihm verbinden und eine Lease bitten. Dazu sind drei Nachrichtenformate definiert:

- Ein Nachrichtenformat, um eine Verbindungslease an einen Rendezvous peer zu stellen.
  (Lease Request Message)
- Ein Nachrichtenformat, um auf eine Verbindungslease von einem Rendezvous peer zuzustimmen und die Länge der lease zu erhalten
  (Lease Granted Message)
- Ein Nachrichtenformat, um die Verbindung zu einem Rendezvous peer abzubrechen.
  (Lease Cancel Message)

Damit ein Peer als Rendezvouz Peer agieren kann, muss er mittels Rendezvousanzeige seine Leistungsfähigkeit im Netzwerk veröffentlichen:

```
<?xml version="1.0"?>
  <jxta:RdvAdvertisement xmlns:jxta="http://jxta.org">
    <RdvGroupId>...</RdvGroupId>
    <RdvPeerId>...</RdvPeerId>
    <Name>...</Name>
  </jxta:RdvAdvertisement>
```

Die Werte der Elemente können wie folgt aussehen:

RdvGroupId enthält die ID der peer Gruppe, bei der der peer seinen Rendezvous Service anbietet.

RdvPeerId enthält die ID von dem peer, der den Rendezvous Service in der Gruppe anbietet

Name ein symbolischer Name für das Rendezvous peer - *optional* -

Um sich nun mit einem Rendezvous Peer verbinden zu können, sendet ein Peer eine Lease Request Message an einen Rendezvouz Peer.

Wenn dieser diese Nachricht erhält, entscheidet er, ob er dem Peer eine Lease geben will und sendet in diesem Fall eine Lease Granted Message an den besagten Peer.

Wenn der Peer die Lease Granted Message erhält, kann er den Rendezvous Peer benutzen um Nachrichten über ihn an das Netzwerk zu senden.

Wenn ein Rendezvouz Peer nun eine Nachricht erhält, überprüft er als erstes die Herkunft der Nachricht, das heit, ob der Absender der Nachricht eine Lease von ihm zugeteilt bekommen hat. Wenn dies der Fall ist, propagiert er die Nachricht an alle ihm bekannten Peers, welche eine Verbindungslease von ihm erhalten haben (multicast / broadcast).

2.3.1 Die Lease Request Message

Wenn ein peer ein Rendezvous Advertisement gefunden hat, kann er sich mit dem Rendezvous peer verbinden und eine lease anfragen, deshalb beinhaltet die Nachricht nur ein Element: *jxta:Connect*, um eine lease von einen Redezvous peer zu erhalten.

2.3.2 Die Lease Granted Message

Wenn der Rendezvous peer der Anfrage auf eine lease zustimmt, wird der anfragende peer in die Menge der verbundenen peers aufgenommen. Der Rendezvous peer antwortet mit der Lease Grant Message auf die Anfrage. Diese Nachricht enthält folgende Elemente:

```
jxta:RdvAdvReply    : enth\"alt das peer Advertisement des Rendezvous
                      peers, der der lease zustimmte.
jxta:ConnectedPeer  : enth\"alt die peer ID des Rendezvous peers, der
                      der lease zustimmte.
jxta:ConnectedLease : Stringrepr\"asentation der lease time in
                      Millisekunden.
```

2.3.3 Die Lease Cancel Message

Wenn ein peer den Rendezvous peer nicht mehr benutzen möchte, kann er seine Verbindungslease beenden und sich selbst aus der Menge der verbundenen peers entfernen. Diese Nachricht enthält nur ein Element: *jxta:Disconnect*.

2.4 Das Peer Information Protocol (PIP)

Das Peer Information Protocol erlaubt einem peer den Status eines anderen peers zu erfragen.Es handelt sich um ien optionales Protokoll, welches erlaubt einen entfernten peer zu beaufsichtigen und somit Informationen über seinen Status zu erlangen, welches zum Beispiel bei *filesharing* Applikationen wichtig ist, um einen entfernten Peer effizienter nutzen zu können.

Das Peer Information Protocol beinhaltet zwei Nachrichtenformate:

- Ein Nachrichtenformat, um den Status eines entfernten peers anzufragen
  (Peer Info Query Message)
- Ein Nachrichtenformat, um den Status eines peers an andere peers weiterzugeben
  (Peer Info Response Message)

Wenn ein Peer eine Peer Info Query Message an einen speziellen Peer sendet, überprüft dieser ob die Ziel Peer ID (targetPid) auf die lokale Peer ID passt, also ob er wirklich der Empfänger der Message sein soll. Wenn sie passt, antwortet der befragte Peer mit einer Peer Info Response Message.

2.4.1 Peer Info Query Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:PeerInfoQueryMessage xmlns:jxta= "http://jxta.org">
    <sourcePid>...</sourcePid>
    <targetPid>...</targetPid>
    <request>...</request>
  </jxta:PeerInfoQueryMessage>
```

Die Werte der Elemente können wie folgt aussehen:

sourcePid enthält die ID des peer, der Statusinformationen von einem entfernten peer anfragt.

targetPid enthält die ID des entfernten peers, von dem Statusinformationen angefordert wurden.

request ein String, der die Statusinformationen beschreibt, die von dem entfernten peer kommen sollen.- *optional* -

2.4.2 Peer Info Response Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:PeerInfoResponseMessage xmlns:jxta= "http://jxta.org">
    <sourcePid>...</sourcePid>
    <targetPid>...</targetPid>
    <uptime>...</uptime>
    <timestamp>...</timestamp>
    <response>...</response>
```

```
    <traffic>
      <lastIncomingMessageAt>...</lastIncomingMessageAt>
      <lastOutcomingMessageAt>...</lastOutcomingMessageAt>
      <in>
        <transport endptaddr="...">...</transport>
      </in>
      <out>
        <transport endptaddr="...">...</transport>
      </out>
    </traffic>
  </jxta:PeerInfoResponseMessage>
```

Die Werte der Elemente können wie folgt aussehen:

sourcePid enthält die ID des peer, der Statusinformationen von einem entfernten peer anfragt.

targetPid enthält die ID des entfernten peers, von dem Statusinformationen angefordert wurden.

uptime Zeit in Millisekunden die vergangen sind, seit der peer dem Netzwerk beigetreten ist.- *optional* -

timestamp Zeit in Millisekunden, die der peer gebraucht hat um die Antwort mit den Statusinformationen zu generieren. - *optional* -

response enthält einen String, welcher die Statusinformationen spezifiziert, welche als Antwort zurückgegeben werden. - *optional* -

traffic enthält die Netzwerkbelastung des peers. - *optional* -

- **lastIncomingMessageAt** Zeit in Millisekunden, seit eine Nachricht beim Endpunkt des peers angekommen ist. - *optional* -
- **lastOutcomingMessageAt** Zeit in Millisekunden, seit eine Nachricht beim Endpunkt des peers ausgegangen ist. - *optional* -
- **in** kann null oder mehrere transport - Elemente beinhalten, beschreibt den upstream von endpunkt des peers. - *optional* -
- **transport** enthält die Anzahl der Bytes, die dieser Endpunkt verarbeitet hat. - *optional* -
- **out** kann null oder mehrere transport - Elemente enthalten und enthält Details über den downstream des Endpunkts.

2.5 Das Pipe Binding Protocol (PBP)

Pipes weden benutzt, um Daten zu einem entfernten Peer zu senden und welche von ihm zu empfangen. Damit Pipes benutzt werden können, müssen sie zuerst an einen physikalischen Kanalendpunkt gebunden werden. Dieses Prozess beschreibt dieses Protokoll. Dazu beinhaltet es zwei Nachrichtenformate:

- Ein Nachrichtenformat, um Anfragen an einen entfernten Peer zu senden, wenn er an eine Pipe gebunden ist, die auf die Pipe ID des Absenders der Anfrage passt.
  (Pipe Binding Query Message)
- Ein Nachrichtenformat, um Antworten auf die Anfrage zu senden.
  (Pipe Binding Answer Message)

Eine Pipe wird mittels Pipe Advertisement beschrieben:

```
<?xml version="1.0"?>
  <jxta:PipeAdvertisement>
    <Id>...</Id>
    <Type>...</Type>
    <Name>...</Name>
  </jxta:PipeAdvertisement>
```

Die Werte der Elemente können wie folgt aussehen:

Id enthält die eindeutige ID des Kanals

Type enthält eine Beschreibung des Typs der Verbindung

Name enthält einen sysmbolischen Namen für den Kanal. - *optional* -

In JXTA sind Pipes unidirektional und asynchron. Damit wird Peers erlaubt ohne eine Art von Zustandssynchronisation miteinander zu interagieren.

Um Nachrichten über Pipes schicken zu können, muss ein Peer also zuerst eine input Pipe von einem Pipe Advertisement erstellen und kann dann auf Nachrichten warten.

Damit ein zweiter Peer Nachrichten an den Peer schicken kann, muss er eine output Pipe für das gleiche Pipe Advertisement erstellen. Dazu sendet er eine Pipe Binding Query Message an alle ihm bekannten Peers und Rendezvous Peers.

Wenn der erste Peer eine solche Nachricht erhält, überprüft er in seinem Speicher von Pipes ob er eine passende findet. Wenn dem so ist, sendet er eine Pipe Binding Answer Message mit seinem Peer Advertisement zurück.

Diese erhält dann der zweite Peer und extrahiert die Endpunktinformationen aus dem Peer Advertisement. Mit diesen

Informationen kann er nun eine output Pipe erstellen und Nachrichten an den ersten Peer senden.

Es gibt natürlich auch die Möglichkeit der bidirektional Kommunikation der Peers. Dazu braucht man lediglich zwei Pipes auf beiden Seiten, eine um Daten zu senden und eine um Daten zu empfangen.

2.5.1 Die Pipe Binding Query Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:PipeResolver>
    <MsgType>Query</MsgType>
    <PipeId>...</PipeId>
    <Type>...</Type>
    <Cached>...</Cached>
    <Peer>...</Peer>
  </jxta:PipeResolver>
```

Die Werte der Elemente können wie folgt aussehen:

MsgType soll den Typ der Pipe Binding Message beinhaltet, in diesem Fall ist Query hartkodiert.

PipeId enthält die ID des Kanals, von dem der angefragte peer versucht, die peer ID aufzulösen.

Type enthält den Typ des Kanals

Cached spezifiziert, ob der entfernte peer, der eine Anfrage erhalten hat, seinen lokalen Cache von getrennten Kanälen benutzen kann, um die Anfrage zu bearbeiten. - *optional* -

Peer enthält die peer ID des peers, der auf die Anfrage antworten soll. - *optional* -

2.5.2 Die Pipe Binding Answer Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:PipeResolver>
    <MsgType>Answer</MsgType>
    <PipeId>...</PipeId>
    <Type>...</Type>
    <Peer>...</Peer>
    <Found>false</Found>
    <PeerAdv>...</PeerAdv>
  </jxta:PipeResolver>
```

Die Werte der Elemente können wie folgt aussehen:

MsgType soll den Typ der Pipe Binding Message beinhaltet, in diesem Fall ist Answer hartkodiert.

Found bedeutet, ob ein passende peer ID gefunden wurde.- *optional* -

PeerAdv enthält das peer Advertisement des peers, welcher die passende peer ID hat. - *optional* -

Die übrigen Werte der Elemente entsprechen denen der *Pipe Binding Query Message*.

2.6 Das Endpoint Routing Protokol (ERP)

Um Daten über das Netzwerk an Peers zu schicken, die nicht direkt miteinander verbunden sind, zum Beispiel wegen unterschiedlicher Netzwerkprotokolle oder Firewalls, werden Endpoint Router gebraucht, um die Nachrichten weiter zu leiten. Das Endpoint Routing Protocol beinhaltet drei Nachrichtenformate:

- Ein Nachrichtenformat, um eine Menge von geordneten Peers festzusetzen, welche benutzt werden, um Nachrichten an eine gegebene Endpoint Adresse zu senden.
  (Route Query Message)
- Ein Nachrichtenformat, um an eine Menge von geordneten Peers zu antworten.
  (Route Response Message)
- Ein Nachrichtenformat, welches die Informationen erhält, um eine Nachricht zu ihrem Ziel zu bringen.
  (Endpoint Router Message)

Wenn ein Peer nicht direkt mit einem anderen Peer vebunden ist, sendet er eine Route Query Message an alle ihm bekannten Peers und Rendezvouz Peers.

Wenn ein Rendezvouz Peer den Weg zu dem gesuchten Peer kennt, sendet er eine Route Response Message zu dem Peer, welcher die Anfrage stellte.

Dieser fügt dann seiner Nachricht die Endpoint Router Message hinzu und sendet die Nachricht an die erste Endpoint Adresse in den erhalten Routeninfomationen.

Wenn nun ein Peer diese Nachricht erhält, sendet er die Nachricht weiter an den nächsten in der Endpoint Router Message, auer er ist genau der Peer, für den die Nachricht bestimmt war.

2.6.1 Route Query Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:EndpointRouter>
    <Type>RouteQuery</Type>
    <DestPeer>...</DestPeer>
    <RoutingPeerAdv>...</RoutingPeerAdv>
  </jxta:EndpointRouter>
```

Die Werte der Elemente können wie folgt aussehen:

Type soll den Typ der Endpoint Router Message beinhalten, in diesem Fall ist RouteQuery hartkodiert.

DestPeer ist der Ziel Peer - *optional* -

RoutingPeerAdv enthält das Peer Advertisement des peers, welcher die Routeinformationen angefordert hat. - *optional* -

2.6.2 Route Response Message

Die Nachricht hat folgendes Format:

```
<?xml version="1.0" encoding="UTF-8"?>
  <jxta:EndpointRouter>
    <Version>2</Version>
    <Type>RouteResponse</Type>
    <DestPeerIdTag>...</DestPeerIdTag>
    <RoutingPeerIdTag>...</RoutingPeerIdTag>
    <NbOfHops>...</NbOfHops>
    <RoutingPeerAdvTag>...</RoutingPeerAdvTag>
    <GatewayForward>...</GatewayForward>
   </jxta:EndpointRouter>
```

Die Werte der Elemente können wie folgt aussehen:

Version enthält die Version des Endpoint Routing Protocols, zur Zeit: 2

Type soll den Typ der Endpoint Router Message beinhalten, in diesem Fall ist RouteResponse hartkodiert.

DestPeerIdTag enthält die Endpunktadresse des Ziel Peers- *optional* -

RoutingPeerIdTag enthält die Endpunktadresse des Peers, der die Routeinformationen bereitgestellt hat. - *optional* -

NbOfHops enthält die Anzahl der Adressen bis zum Ziel Peer. - *optional* -

RoutingPeerAdvTag enthält das Peer Advertisement des Peers, der die Routeinformationen angefragt hat. - *optional* -

GatewayForward enthält die Endpunktadresse eines Peers entlang der Route, können mehrere Elemente in dieser Nachricht enthalten sein.- *optional* -

2.6.3 Endpoint Router Message

Die Nachricht hat folgendes Format:

```
<jxta:JxtaEndpointRouter>
  <jxta:Src>...</jxta:Src>
  <jxta:Dest>...</jxta:Dest>
  <jxta:Last>...</jxta:Last>
  <jxta:NBOH>...</jxta:NBOH>
  <jxta:GatewayForward>...</jxta:GatewayForward>
  <jxta:GatewayReverse>...</jxta:GatewayReverse>
</jxta:JxtaEndpointRouter>
```

Die Werte der Elemente können wie folgt aussehen:

Src enthält die Endpunktadresse des verantwortlichen Peers, der die Nachricht gesendet hat

Dest enthält die Endpunktadresse des Ziel Peers.

Last enthält die Endpunktadresse des Vorgängers in der Liste der Route. - *optional* -

NBOH enthält die Anzahl der Adressen bis zum Peer, der die Anfrage stellte. - *optional* -

GatewayForward enthält die Endpunktadresse eines Peers entlang der Route zum Ziel Peer, es können mehrere Elemente hiervon in dieser Nachricht enthalten sein. - *optional* -

GatewayReverse enthält die Endpunktadresse des Peers entlang der Route zurück zum Peer, der die Anfrage stellte, es können mehrere Elemente hiervon in dieser Nachricht enthalten sein.- *optional* -

3 JXTA im Einsatz

JXTA ist, wie bereits aus 1.2 bekannt, in drei Schichten aufgebaut: Core, Services und Applications (siehe Abbildung 5). Einige wichtige Komponenten und deren Zusammenspiel sollen hier vorgestellt werden. Ferner soll der Aspekt der Sicherheit hier überblicksweise dargestellt werden.

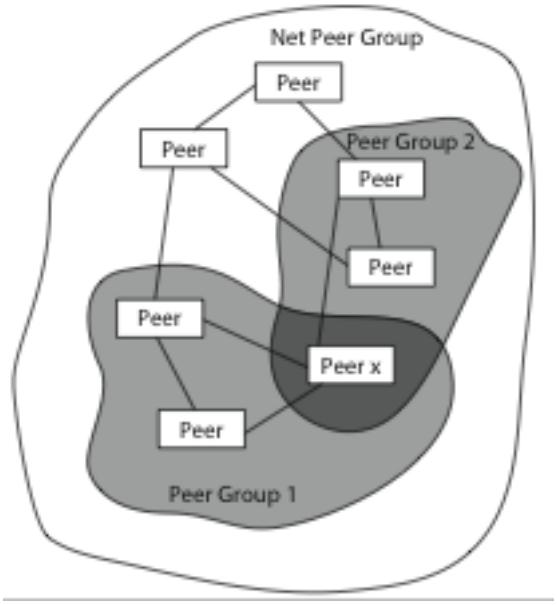

Abbildung 6: Peer Groups und ihre Mitglieder (Quelle: [5])

3.1 Peer Groups und Services

Ein besonderer Teil des Cores besteht aus den sogenannten *Peer Groups*, die es ermöglichen das gesamte Netzwerk in logische "Regionen" einzuteilen. Eine Peer Group teilt alle ihre Ressourcen und somit alle ihre Dienste miteinander. Das hat zur Folge, daß jedes Mitglied einer Gruppe alle *Services* dieser Gruppe zur Verfügung stellt. Ein Peer kann Mitglied in mehreren Gruppen sein, was Abbildung 6 zu sehen ist. Alle hier zu sehenden Peers sind Mitglied der Peer Group "Net Peer Group". Einige Peers gehören auch noch anderen Gruppen an, z.B. "Peer x" gehört außerdem noch den Gruppen "Peer Group 1" und "Peer Group 2" an. [8] gibt außer dem Ordnungsgedanken noch folgende Motivationen für den Nutzen von Peer Groups an:

- Es können sichere Zonen mit Peer Groups eingerichtet werden, d.h. Services bzw. Ressourcen können durch den sog. *Membership Service*[1] kontrolliert werden.
- Peer Groups bilden einen Namensraum, d.h. Dienste, Ressourcen und Peers müssen lediglich dafür sorgen, dass IDs bzw. Namen lediglich innerhalb der Gruppe eindeutig sind.

[1]Mit dem Membership Service kann bestimmt werden, welche Peers in einer Gruppe teilnehmen dürfen.

- Innerhalb einer Gruppe kann "Monitoring" stattfinden, d.h. die Ressourcennutzung kann ermittelt und bei kostenpflichtigen Angeboten einfach berechnet werden.

Der gesamte Peer Group Mechanismus ist bei den existierenden Anwendungen als Service implementiert. Um diesen nutzen zu können, mus ein entsprechendes *Module Implementation Advertisement* konfiguriert werden. Ist dies erfolgt, so kann eine Net Peer Group instanziiert werden und der Peer kann mit dem Netzwerk kommunizieren.

Das Elter aller Gruppen ist die sog. *World Peer Group*, die eine besondere *Peer Group ID* besitzt. Diese Gruppe definiert die grundlegenden Fähigkeiten eines Peers wie z.B. dessen Services, die Protokoll Implementierung und die vom Peer für das Netzwerk bereitgestellten Dienste. Obwohl nun jeder Peer zu dieser Gruppe gehört und diese Gruppe auch die Implementierung des Peer-Protokolls definiert, kann sie nicht für die eigentliche P2P Kommunikation benutzt werden. Die Gruppe dient lediglich als Schablone (Template) für das Erzeugen der *Net Peer Group* Instanz. Diese Gruppe ist die Standardgruppe, die es nun auch wirklich allen Peers ermöglicht miteinander zu kommunizieren.

Während die *World Peer Group* die Basisfunktionalität definiert, kann die *Net Peer Group* weitere Eigenschaften über einen Peer festlegen[2]. Obwohl alle Peers zu eben einer solchen Gruppe gehören ist es nicht zwingend für alle die selbe. Beispielsweise kann eine Firma eine Anwendung innerhalb ihrer "eigenen" Peer Group definieren und nutzen. Andere Peers können jedoch nicht ohne weiteres Mitglied der Gruppe werden.

Eine weitere Eigenschaft einer Peer Group ist deren Lebensdauer, die beim instanziieren und der bekanntgabe der Gruppe im Netzwerk festgelegt wird. Aktiv zerstört werden kann eine Gruppe nicht. Sie kann lediglich dadurch aufgelöst werden, daß keine Mitglieder mehr in ihr existieren oder aber die Cache Zeit über das Advertisement der Gruppe abläuft.

Nach dem nun die Net Peer Group instaziiert wurde werden die Dienste / Services einer Gruppe gestartet. In der minimalen Net Peer Gruppe sind das die folgenden Standarddienste:

- Discover: Dienst für das *Peer Discovery Protocol* (siehe 2.1)
- Resolver: Dienst für das *Peer Resolver Protocol* (siehe 2.2)
- Rendezvous: Dienst für das *Rendezvous Protocol* (siehe 2.3)
- Peer Info: Dienst für das *Peer Information Protocol* (siehe 2.4)

[2] vergl. Vererbungsprinzip objektorientierter Programmiersprachen

- Binding: Dienst für das *Pipe Binding Protocol* (siehe 2.5)
- Endpoint: Dienst für das *Endpoint Routing Protokol* (siehe 2.6)

Es können jedoch auch weitere Dienste zur Verfügung gestellt werden[3]. In der Referenzimplementierung sind das z.B. folgende:

- Search: Dienst für die Bereitstellung von Suchalgorithmen
- Indexing: Dienst zur Indexierung von in der Gruppe vorhandenen Informationen
- Membership: Dienst zu Authentifizierung von Mitgliedern einer Gruppe zur Nutzung bestimmter Dienste

Nach dem Starten aller Dienste ist der Peer bereit mit dem Netzwerk zu komminuzieren. Es können nun weitere Gruppen eingerichtet werden um das Netz, wie oben beschrieben, weiter ein- bzw. zu unterteilen oder aber neue Peers der Gruppe beitreten.

3.2 Sicherheit

Beim Einsatz von P2P Netzwerken muss man sich auch intensiv mit dem Thema Sicherheit befassen. Zum einen, weil man evtl. nicht möchte, daß vertrauliche Daten im Netzwerk von dritten eingesehen können, zum anderen, weil man nur vertauenswürdigen Peers in seine Gruppe Eintritt ermöglichen möchte. In JXTA gibt es mehrere Ansätze, diese Probleme durch geeignete Sicherheitsmaßnahmen zu eliminieren.

3.2.1 Transport Layer Security (TLS)

Transport Layer Security bezeichnet die gesicherte Übertragung von Daten über ein geeignetes Protokoll, z.B. TCP/IP. Hierbei wurde beim Entwurf der JXTA Spezifikation der Standard der *Internet Engineering Task Force (IETF) Networking Group* übernommen. Die Aufgabe einer veschlüsselten, zuverlässige Verbindung zwischen zwei Peers stellt hierzu folgende Anforderungen:

- Interoperabilität: Unterschiedliche Programmierer sollen einfach im Stande sein, das entsprechende Protokoll zu implementieren, ohne den Code der anderen Peer-Seite zu kennen
- Erweiterbarkeit: Das Protokoll bildet ein Framework, das es ermöglicht auch nachträglich noch weitere Verschlüsselungsalgorithmen einzubinden.

[3]Eigene Services können durch Implementierung eines *Service* Interface erfolgen.

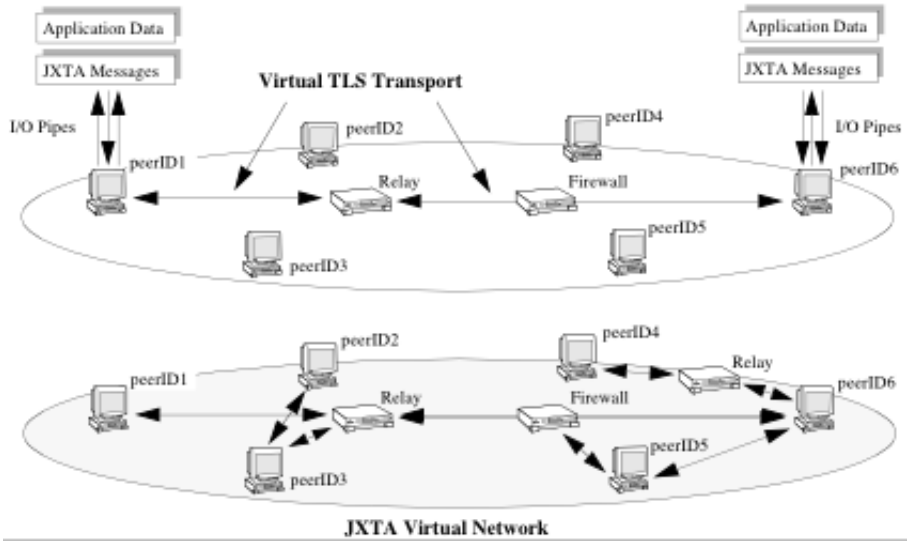

Abbildung 7: Mitglieder eines JXTA Netzwerks sind durch Firewalls und ausgewiesene Gateways miteinander verbunden. Auf einer weniger abstrakten Ebene verbinden virtuelle TLS Verbindungen die Peers, die über Pipes Nachrichten austauschen. Um die Leistungsfähigkeit zu erhöhen, teilen sich mehrere Pipes zwischen zwei Peers immer eine Verbindung. Innerhalb dieser Vebindung wird die existierende *Public Key Infrastructure* ebenfalls gemeinsam benutzt. Wenn das zugrunde liegende Transportprotokoll auf TCP/IP basiert, so nutzt die virtuelle TLS Verbindung ebenfalls TCP/IP.(Quelle: [1])

- Relative Produktivität: Kryptografische Algorithmen neigen dazu besonders CPU intensiv zu sein. Daher soll das Protokoll die Verbindungen auf ein notwendiges Minimum reduzieren.

TLS besteht aus zwei Schichten: dem *TLS Record Protocol* und dem *TLS Handshake Protocol*. Auf der untersten Ebene befindet sich das TLS Record Protokoll (oberhalb der eigentlichen Transportschicht, z.B. TCP/IP), welches die Verbindungssicherheit in zwei Aspekten unterstützt:

- *Privatsphäre*: Es wird symetrische Verschlüsselung benutzt, wie sie auch für Datenverschlüsselung benutzt wird (z.B. DES, RC4). Die Schlüssel für jede Verbindung sind einmalig und werden während der Verbindungsaufnahme ausgetauscht (z.B. TLS Handshake Protocol).
- *Zuverlässigkeit*: Die übertragenen Daten sollen sicher beim Partner ankommen. Dies wird durch Hash-Verfahren unterstützt.

Das TLS Record Protocol wird benutzt, um Protokolle auf höherer Ebene einzukapseln. Eines dieser Protokolle ist nun das TLS Hanshake Protocol, welches beiden Kommunikationspartnern ermöglicht sich gegenseitig zu authentifizieren, ein geeignetes Veschlüsselungsprotokoll zu vereinbaren und hierfür notwendige Schlüssel auszutauschen noch bevor die Anwendung selber Daten austauscht. Die drei wesentlichen Bestandteile des TLS Handshake Protocol sind:

- Die Identität eines Peers kann durch asymetrische- oder öffentliche Schlüssel bestätigt werden. Diese Authentifizierung ist nicht zwingend erforderlich, wird jedoch zumindest für einen der Peers meist gefordert.
- Die Übertragung des *shared secret* ist sicher, d.h. für ein abhorchen der Verbindung nicht brauchbar.
- Die eigentliche Verbindung ist sicher, denn selbst wenn ein vermeintlicher Angreifer sich in die Verbindung zwischen zwei Peers einklinken möchte (*man in the middle*), so bleibt dies für die Peers nicht unentdeckt.

3.2.2 Public Key Infrastructure (PKI)

Public Key Umgebungen ermöglichen das verschlüsseln von Informationen oder ganzen Verbindungen durch den Austausch von Schlüsseln. Hierbei hat jeder Teilnehmer einen öffentlichen (public) und einen privaten (private) Schlüssel. Wollen nun zwei Partner auf Basis von PKI Informationen austauschen, so geben

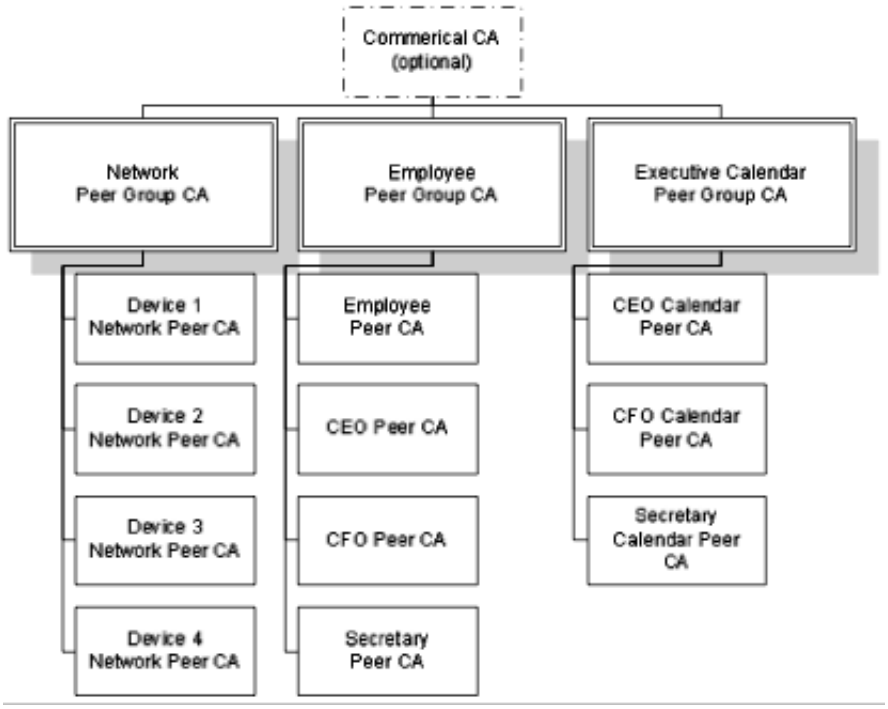

Abbildung 8: Hierarchiebildung durch *Certificate Authorities* (Quelle: [6])

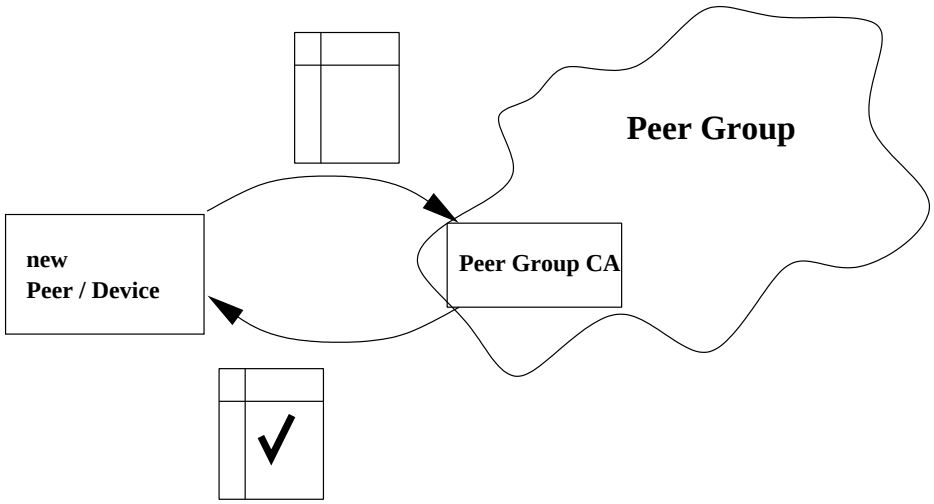

Abbildung 9: Peer Zertifizierungs Prozess

sie dem jeweils anderen Partner ihren öffentlichen Schlüssel. Dieser wird benutzt, um die Daten zu verschlüsseln, die sich nur mit dem privaten Schlüssel wieder entschlüsseln lassen. Somit ist immer gewähleistet, daß die Information nur demjenigen Partner zugänglich gemacht wird, für den sie auch bestimmt war. Andererseits können die Schlüssel nicht nur zum ver-/entschlüsseln, sondern auch zum signieren, d.h. sozusagen zum unterschreiben genutzt werden. Die Authentizität der signierten Daten kann unter Zuhilfenahme des öffentlichen Schlüssels gesichert werden. Hierbei kann ferner festgestellt werden, ob irgendwe sonst auf dem Weg die Daten manipuliert hat.

Für die Schlüsselbestätigung werden hierzu meist X.509 Zertifikate eingesetzt. diese können entweder von einer "großen" CA (z.B. VeriSign) oder aber einer eigenen CA bestätigt werden. Im P2P-Bereich ist es naheliegend, daß z.B. in einer größeren Organisationsform Hierarchien der CAs gebildet werden um die Vertrauenswürdigkeit von Informationen gleichermaßen zu hierarchiesieren (siehe Abbildung 8). Genauso naheliegend ist es aber auch, daß Peers selber als CA innerhalb ihrer Gruppe fungieren.

Will nun ein Peer Mitglied einer Gruppe werden, so sendet er ein *Certificate Sign Request* an einen ausgezeichneten Peer, den PeerGroup CA aus der gewünschten Peer Group (siehe Abbildung 9). Dieses CSR beinhaltet die Identität des Peers sowie seine gewünschten Privilegien die er in der Gruppe haben möchte. Der *Peer Group CA* authorisiert den neuen Peer, indem er ihm

das signierte Certificat zurücksendet. Der neue Peer nutzt dieses signierte Zertifikat nun für die weitere Kommunikation in der Gruppe quasi als "Mitglidsausweis".

3.2.3 Vertrauen als Sicherheitsbasis

Vertrauen ist die Basis auf der Sicherheitsumgebungen geschaffen werden. Vertrauensverhältnisse müssen aufgebaut werden um das Risiko der Netzwerkkommunikation zu reduzieren. In lose zusammenhängenden ad-hoc Netzwerken ist das Risiko ungleich höher, als in zentralen Netzwerken hinter denen eine feste Organisationsstruktur steht. Unbekannte Peers begegnen sich und wollen sicher Daten austauschen. In [6] wird hierzu *Poblano*, ein Model des "guten Rufs" vorgeschlagen. Die Vertrauenswürdigkeit eines Peers hängt dabei von seinem Ruf ab, den er bei anderen Peers hat. Außerdem wird die Qualität und Quantität dessen Services sowie das zu erwartende Risiko sowie die Verfügbarkeit des Peers bewertet (z.B. "verteilt der Peer Viren?"). Diese Vertrauenswerte eines jeden Peers oder Inhalt des Netzwerkes werden durch die Beziehungen der Peers untereinander oder von Peers zu Inhalten bestimmt. Beispielsweise wird ein Mitglied mit wichtigen und vielfach nachgefragten Informationen in Ruf und Ansehen aufgewertet. Werden seine Informationen allerdings nicht mehr so häufig nachgefragt, so sinkt sein Ruf wieder ab[4].

4 Verwandte Arbeiten

Im folgenden Teil werden zu JXTA verwandte Arbeiten behandelt. Arbeiten die JXTA sehr nahe stehen, mit denen JXTA verwechselt bzw. manchmal synonymhaft genannt wird, aber in einem anderen Kontext als JXTA zu verstehen sind. Zudem Arbeiten, auf die JXTA aufbaut, von denen JXTA Ideen, Funktionalitäten übernommen hat. Zuletzt noch konkurrierende Arbeiten, andere Peer-to-Peer-Lösungen die auf den Markt drängen, bzw. sich teilweise schon etabliert haben.

4.1 Nahestehende Arbeiten

Eine Technologie die hier zu nennen wäre, ist JINI. JINI ist eine Technologie, die meistens mit JXTA verwechselt wird. Der

[4] vergl. Popularität/Ansehen im zwischenmenschlichen Umgang

Unterschied zwischen JXTA und JINI ist vielleicht vergleichbar mit dem Unterschied zwischen Wireless LAN und Bluetooth. Wo Wireless LAN eher darauf abzielt LANs zu verbinden, LANs an WANs anzubinden und sich perfekt als Transportmedium für jegliche Internetanwendung eignet, ist Bluetooth eher auf das Anbinden lokaler Geräte spezialisiert, wie zum Beispiel Drucker, Scanner, WebCam, etc. Gleicher Unterschied gilt nun auch für JXTA und JINI. JXTA ermöglicht das Erstellen eines Peer-to-Peer-Netzwerkes in LANs sowie WANs, stellt die notwendigen Protokolle bereit und ist dabei besonders gut für das Internet geeignet. JINI hingegen ist darauf spezialisiert in einem lokalen Netzwerk Geräte zu finden, die im Bedarfsfall angesprochen werden können. Drucker, Scanner, WebCam, etc. würden in dem Fall nicht nur ihre Anwesenheit kundtun, sondern über JINI Informationen austauschen, wo benötigte Treiber und Tools zu finden sind, wie sie zu installieren sind und sogleich eine Schnittstelle bereitstellen. JINI hält sich dabei größtenteils an die Grenzen eines Netzwerks, da es meistens wenig Sinn macht Geräte außerhalb eines Netzwerkes zu finden. Wird es jedoch explizit verlangt, bietet auch JINI diese Möglichkeit.
Ein Punkt der meistens Auslöser für Verwirrung ist, ist, wie auch bei Blueooth und Wireless LAN, die Möglichkeit mit JINI Funktionen zu nutzen, die auch JXTA anbietet. Zum Beispiel ist es mit Bluetooth durchaus realisierbar mit genügenden Zugangsknoten ein großes LAN aufzuziehen, dieses ans Internet anzuschliessen und somit Funktionen zu bieten, die auch Wireless LAN bieten würde (siehe www.lesswire.de). Ebenso ist es mit JINI möglich in einem lokalen Umfeld Chat-Dienste zu nutzen und diese sogar falls gewünscht und so bereitgestellt auf andere Netzwerke auszudehnen. Damit wäre ein Peer-to-Peer-Chat möglich wie ihn auch JXTA bereitstellen würde. In den meisten Fällen gibt es allerdings nur eine sinnvolle Wahl für die Realisierung einer Transportverbindung, da beide Lösungen auf unterschiedliche Bereiche abgestimmt sind und eine Sinnverfremdung meistens nur zu Performance- bzw. Bandbreiten-Einbußen führt.(Quelle:[2])

4.2 Grundlegende Arbeiten

In diesem Teil wollen wir Technologien besprechen auf die JXTA zurückgreift, bzw. es empfiehlt auf diese zurückzugreifen. Wir behandeln dazu das “Händler-Bote-Prinzip“ und das der Web Dienste.

Das “Händler-Bote-Prinzip“ zeigt drei verschiedene Wege auf, wie ein Kunde mit einem Händler kommunizieren kann. Im Mittelpunkt steht dabei im besonderen immer die Funktion des Boten. Anhand eines Beispiels kann man die drei Arten sehr gut unterscheiden.
Das Beispiel sei ein normaler Einkaufsvorgang mit Einkaufliste.

- Beim ersten Typ würde der Bote unsere Einkaufsliste nehmen und von Geschäft zu Geschäft laufen und uns alles besorgen was wir benötigen.
- Beim zweiten Typ würde der Bote nur jeweils in einem Geschäft existieren. D.h. wir würden die jeweiligen Produkte einem Geschäft mitteilen und der Bote des Geschäftes würde uns die vorrätigen Produkte liefern, danach würden wir zum nächsten Geschäft gehen und dort die restlichen Produkte dem dortigen Boten mitteilen. Usw.
- Beim dritten Typ wäre der Händler selbst der Bote. Der anstatt, daß wir zu ihm gehen, er selbst kommen und uns seine Ware anbieten würde.

JXTA lehnt, wie viele Peer-to-Peer-Systeme, an diesen Prinzipien an. Prinzipiell wäre es möglich eine JXTA-Implementierung für jedes Prinzip zu erstellen, jedoch stützen sich viele JXTA-Implementierungen auf das zweite Prinzip, da dieses für die meisten Anwendungen am Optimalsten ist. Das jeweils zu bevorzugende Prinzip hängt dabei von der Anzahl der unterschiedlichen Produkte ab, sowie von der insgesamten Menge jedes einzelnen Produktes und wie groß der Bedarf jedes Einzelnen ist. So gibt es auch Anwendungsfälle und JXTA Projekte wo das dritte Prinzip eher passend ist und folglich auch bevorzugt wird. Das erste Prinzip wird seltener umgesetzt, da es im Allgemeinen zu schwer zu kontrollieren ist. Dies gilt für Sicherheit und Kontrolle wie für Rechenpower. Da der Code für einen Boten auf einem fremden Rechner arbeitet, ist es schwer Sicherheit für den fremden Rechner zu garantieren. Außerdem könnten solche Rechner schnell überlastet werden, wenn viele Boten anfangen auf ihnen zu arbeiten. Um ein JXTA-System zu implementieren, können diese Prinzipien sehr hilfreich für die Findung der richtigen Struktur sein. Welches Prinzip im Endeffekt genutzt wird, muss im Einzelfall entschieden werden und hängt nicht unerheblich von der Struktur der zu nutzenden Daten sowie der Struktur des zu nutzenden Netzwerkes ab.

Web Dienste stellen das Konzept dar, das es ermöglicht über HTTP einen Dienst auf einem Server zu nutzen. Dabei kann der Nutzer sowohl ein Klient als auch ein anderer Server sein. Das klassischste Beispiel wäre wohl ein User der über seinen Browser einen HTML-Dienst auf einem Server nutzt.
Web Dienste sind deshalb so interessant für JXTA, weil man sie sehr gut in JXTA einbinden kann. Sie nutzen normalerweise HTTP als Transportprotokoll und kommen so durch viele Firewalls - sind somit auch auf schwierigen Verbindungswegen, wie sie häufig bei Peer-to-Peer-Netzen vorkommen, noch einsetzbar. Ein naheliegendes Beispiel für einen interessanten Web Dienst wäre zum Beispiel ein Kreditkarten-Authentifikations-Dienst, der besonders kommerziellen Anwendungen sehr dienlich wäre.(Quelle:[2])

4.3 Konkurierrende Arbeiten

In diesem Abschnitt möchten wir insbesondere auf drei Technologien eingehen. Dies sind Gnutella, Freenet und SETI@home.
Gnutella ist nach dem Napster Boom und der langsamen Sperrung des Napster-Netzwerkes bekannt geworden. Ursprünglich von AOL initiiert, sollte Gnutella ein Filesharing-Tool werden, das auf reinem Peer-to-Peer beruht. Dabei spielte vor allem die Legalität und damit maximale Dezentralität eine große Rolle. Die Suche nach Daten wurde durch Breitensuche realisiert. Diese Breitensuche hat nach einiger Zeit und gestiegener Popularität von Gnutella solch einen Verkehr im Gnutella-Netzwerk ausgelöst, daß Suchanfragen nicht mehr beantwortet werden konnten und das Gnutella-Netzwerk komplett blockiert war.
Freenet ist auch ein Filesharing Programm, jedoch eines das sich hauptsächlich auf Dokumente zum Schutz vor Zensur konzentriert. Freenet hat die Suche im Peer-to-Peer-Netz durch Tiefensuche realisiert und zudem in jedem Knoten einen Cache für bereits erfolgte Suchen angelegt. So wurde das Netz im Grossen und Ganzen skalierbar und ein Kollaps wie bei Gnutella wurde vermieden.
SETI@home ist ein Projekt das Berechnungen astronomischer Kontexte an ein Peer-to-Peer-Netz übergibt. Die Hauptaufgabe dieses Peer-to-Peer-Netzes ist somit kein Filesharing sondern Verteiltes Rechnen. Dabei stehen SETI@home 26 TeraFLOPs/sec zur Verfügung. SETI@home ist kein reines Peer-to-Peer-System im eigentlichen Sinne, da es einen Server gibt der die zu berechnenden Daten überträgt und verteilt. Würde man jedoch den

Server nur als ein weiteres Peer sehen, wäre ein Peer-to-Peer-Netz gegeben.
Alle drei Systeme unterscheiden sich zu JXTA in dem Punkt, daß sie spezialisiert sind auf einen bestimmten Anwendungsfall. Sie nutzen bestimmte festgelegte Protokolle (wie z.B. Gnutella: HTTP) und beschränken sich auf gewisse Inhalte (wie z.B. Freenet: Dokumente). Zudem nutzen alle nur einen Aspekt eines Peer-to-Peer-Netzwerkes (Filesharing, Verteiltes Rechnen). JXTA bleibt in diesen Dingen offen und überlässt die Wahl der konkreten Implementierung, wodurch später zwar Anwendungen wie obige entstehen könnten, jedoch durch die gemeinsame JXTA Basis Interoperabilität gewährleistet wäre.(Quelle:[2], [4])

5 Ausblick & Wertung

In diesem Kapitel soll etwas über die Zukunft von JXTA gesagt werden, die kommenden Möglichkeiten, die sich JXTA bieten, sowie die Hürden, die es noch nehmen müsste um ein bedeutender zukünftiger Standard zu werden.

JXTA könnte eine der bedeutendsten Zukunftstechnologien der nächsten Jahre werden. Dies dürfte nicht nur an der allgemeinen Beliebtheit von Peer-to-Peer-Systemen liegen, sondern vielmehr auch an der Notwendigkeit einer Standardisierung von Peer-to-Peer-Protokollen.
Peer-to-Peer-Systeme werden derzeit in den verschiedensten Softwarebereichen eingesetzt. Angefangen bei Chat-Systemen über Filesharing-Systeme bis zum Verteilten Speichern bzw. Verteilten Rechnen. Dabei steht einer effizienten gemeinsamen Nutzung eines Peer-to-Peer-Netzes meistens nur die Kompatibilität der verschiedenen Programme entgegen. So kann es passieren, daß ein Nutzer, falls er ein Chat-Programm und ein Filesharing-Programm gleichzeitig nutzt, zwei unterschiedliche Peer-to-Peer-Netze nutzt und diese mit unterschiedlichen Protokollen bedient. Dadurch entsteht ein Overhead an Bandbreiten-Nutzung und Rechenleistung, da Discovery Protokolle, Pipes, Advertisements etc. doppelt erzeugt werden müssen. JXTA führt an diesem Punkt die Anwendungen zusammen. Basieren alle installierten Peer-to-Peer-Anwendungen auf JXTA, kann jede Anwendung auf dasselbe Netz zurückgreifen, bereits entdeckte Peers nutzen, eventuell sogar bereits bestehende Pipes nutzen, bzw. bekannte Endpunkte ansprechen.([3])
Die Kehrseite einer Standardisierung, nämlich die Limitierung

der eventuellen Möglichkeiten, umgeht JXTA dabei elegant, indem es vieles in der Spezifikation offen lässt und damit die Breite der Möglichkeiten der jeweiligen Implementierung überläßt. JXTA versucht sich so unabhängig wie möglich weder an eine Plattform noch an eine Programmiersprache zu binden. So ist JXTA, obwohl es von Sun Microsystems initiiert wurde, die bisher einzige vollständige Implementierung nur in JAVA existiert und es zudem noch mit einem 'J' anfängt, keine JAVA-Domäne. Gerade um diesen Verdacht abzustreifen und eine schnellere, weitere Verbreitung von JXTA zu ermöglichen, wird in der JXTA-Gemeinde (unter www.jxta.org) an diversen Referenz-Implementierungen in unterschiedlichen Programmiersprachen (wie zum Beispiel: Perl, C++ und diversen Programmiersprachen für mobile Endgeräte) gearbeitet.([7]) Diese werden auch nötig sein, wenn JXTA wirklich den Anspruch hat der Peer-to-Peer-Standard künftiger Systeme zu werden. Ohne Plattform- und Programmiersprachenübergreifende JXTA-Implementierungen besteht die Gefahr, daß JXTA nur ein theoretisches Konstrukt bleibt und irgendwann durch eine einseitig unterstützte Industrie-Lösung ersetzt wird.

Literatur

[1] Security and project jxta. Technical report, SUN Microsystems, 2004.

[2] Navaneeth Krishnan Juan Carlos Soto Daniel Brookshier, Darren Govoni. *JXTA: Java P2P Programming*. Sams Publishing, 2002.

[3] Michael Dimitrov. Peer-to-peer technologie. April 2003. http://lrb.cs.uni-dortmund.de/ hildebra/Seminare/Presentations/modinf/p2p.pdf/.

[4] Bruno Habegger. Lieber nehmen als geben. November 2003. http://www.pctip.ch/library/pdf/2003/11/1128Taus.pdf/.

[5] Carsten Habicht. Seminar verteilte systeme - ausarbeitung zum thema "jxta services". July 2003.

[6] CTO Jeffrey Eric Altman. Pki security for jxta overlay networks. Technical report, IAM Consulting, Inc., 2003.

[7] Project JXTA. Javaone papers. Technical report, 2002. http://www.jxta.org/JavaOne/JavaOne2002/.

[8] JXTA Protocol Specifications Project. Project jxta v2.0: Protocol specification. Technical report, April 2003. http://spec.jxta.org/.

[9] Brendon J. Wilson. *JXTA*. New Riders, 2002.