JXTA und Poblano: Eine Protokollarchitektur zur Realisierung flexibler P2P-Applikationen

Marco Winter
simawint@stud.informatik.uni-erlangen.de

Kurzzusammenfassung

Aktuelle P2P-Anwendungen veschiedener Entwickler sind sehr ausgereift und stellen viele verschiedene Dienste wie P2P-Chat, Email oder File-Sharing zur Verfügung, ohne jedoch zueinander kompatibel zu sein. Dieses Manko versucht JXTA zu beseitigen, indem es eine allgemeine Systemarchitektur zur Verfügung stellt, die es unterschiedlichen Applikationen erlauben soll, untereinander zu kommunizieren und Daten auszutauschen. Daneben stellt das Teilprojekt Poblano ein Modell zur Verfügung, wodurch es einzelnen Knoten ermöglicht wird, Vertrauen zu anderen Knoten und deren Diensten zu gewinnen, und dementsprechend die Kooperation untereinander zu beeinflussen.

1 JXTA - Eine Protokollarchitektur

1.1 Motivation

Es gibt bereits eine größere Anzahl an P2P-Anwendungen, die beinahe alle möglichen Bereiche an P2P-Dienstleistungen abdecken, z.B. Gnutella für File-Sharing oder den A&M Instant Messenger für P2P-Chat [Gnut][AMim]. Diese und andere Beispiele sind alle durchdachte P2P-Anwendungen, allerdings haben sie einen Nachteil: Jede Anwendung kann nur mit ihresgleichen kommunizieren, und das auch nur in ihrem jeweiligen Bereich, also Chat, File-Sharing, etc. Dies liegt vor allem daran, dass für jede Anwendung eigene proprietäre Kommunikationsprotokolle entwickelt wurden, deren Einsatzgebiete beschränkt und die zueinander inkompatibel sind. Ein Gnutella-Peer zum Beispiel ist nur für den Dateitausch gedacht, und wird wohl nie für Chat verwendet werden. Und damit zu versuchen, einem Napster-Peer Dateien zu entlocken, ist auch ein hoffnungsloses Unterfangen. Genau das ist aber ein Problem, denn falls sich im P2P-Bereich des Internets mehrere inkompatible Systeme etablieren, fällt die Kommunikation untereinander schwer, und es wird unerlässlich werden, wieder eine neue Form des Austauschs zwischen diesen Systemen zu finden. Diesem Problem widmet sich nun JXTA, indem es versucht, einen Standard für die Entwicklung und Kommunikation von P2P-Anwendungen zu definieren [LG01].

1.2 Was ist JXTA?

JXTA bezeichnet keine spezielle Anwendung, sondern vielmehr eine Systemarchitektur ähnlich dem ISO-OSI-Referenzmodell, die großteils auf bestehenden Standards wie XML oder eben TCP/IP aufbaut, und die alle Funktionen bietet, welche von P2P-Anwendungen benötigt werden. Die Idee dahinter ist, dass es möglich sein soll, auf dieser Schnittstelle aufbauend P2P-Applikationen für die unterschiedlichsten Bereiche zu erstellen, z.B. Chat-Rooms, File-Sharing, P2P-E-

Mail, und andere Anwendungen. Darüber hinaus ist es aber theoretisch auch möglich, dass diese Anwendungen, die für unterschiedliche Zwecke konzipiert wurden, auch miteinander kooperieren könnten. Z.B. könnte ein Chat-Programm einfach mit einem Sharing-Modul kombiniert werden, das es erlaubt, jedem Chatter gleich Dateien mit der aktuellen Eingabe zu schicken.

Die Protokolle, die durch JXTA definiert werden, sind nach Definition systemunabhängig und an keine konkrete Programmiersprache gebunden. Die zugrundeliegende Spezifikation beschreibt nur das Verhalten, aber nicht die konkrete Implementierung der Protokolle. Deswegen ist es möglich, JXTA für die unterschiedlichsten Betriebssysteme, Programmiersprachen oder Netzwerke zu implementieren, solange nur die spezifizierten Protokolle richtig umgesetzt werden. In diesem Sinne lässt sich JXTA mit Protokollen wie TCP/IP vergleichen, welches ebenfalls für eine Vielzahl von Plattformen verfügbar ist.

2 JXTA - Konzepte, Protokolle, Architektur

2.1 Konzepte in JXTA

JXTA hat sein eigenes Systemmodell, um Vorgänge zwischen den einzelnen Teilnehmern des P2P-Netzes zu beschreiben. Dabei spielen vor allem folgende Begriffe und Konzepte eine Rolle:

- Eine ID ist ein eindeutiger globaler Bezeichner im System. Jede Komponente besitzt einen solchen Bezeichner, damit sie eindeutig im Netz identifiziert werden kann, ganz egal, wie das physikalische Netz darunter aussieht.
- Ein Advertisement ist eine Art "Visitenkarte", die für jede Ressource oder Komponente im Netz vorhanden ist. Jede Komponente besitzt meist ihr eigenes Advertisement und kann es an andere Einheiten weiterschicken, um so bekannter zu werden und Informationen über ihre Anforderungen weitergeben zu können. Gleichzeitig kann sie Advertisements anderer Komponenten empfangen und verarbeiten. Advertisements sind in XML geschrieben, um beliebig erweiterbar zu sein (siehe Abb. 2.1).
- Messages sind die Nachrichten, die im Netz verschickt werden. Messages sind keine Advertisements, können aber solche beinhalten. Sie benutzen vielmehr ein binäres Datagrammformat, um sowohl binäre als auch Textinhalte transportieren zu können (sogenannte Codats). Darüber hinaus können sie auch Metadaten speichern, die z.B. den Absender eindeutig identifizieren oder zum Entschlüsseln der Nachricht gedacht sind. Da JXTA auf beliebigen Plattformen eingesetzt werden könnte, sind die darunterliegenden Nachrichtenkanäle entsprechend flexibel realisiert, um auf unzuverlässigen, uni-direktionalen Verbindungen eingesetzt werden zu können.
- Der Peer ist die kleinste Einheit im System und stellt den eigentlichen Teilnehmer in einem P2P-System dar, der mit anderen Teilnehmern im System kommuniziert. Ein Peer kann entweder ein kompletter Rechner, ein Prozess, oder ein Benutzer sein, solange er bzw. sein digitales Pendant in der Lage ist, die vorgeschriebenen Protokolle zu realisieren.
- Mehrere Peers lassen sich zu Peer Groups zusammenfassen. Wann oder warum das passieren soll, bleibt dem Benutzer überlassen, nur das "wie" sowie das Finden von Peergruppen sind definiert. Peers schließen sich meist dann zu Gruppen zusammen, wenn sie gemeinsame Interessen haben und entsprechend verstärkt Nachrichten austauschen möchten. Ein

Peer kann Mitglied beliebig vieler Gruppen sein, solange er die Protokolle und Verhaltensregeln implementiert, die von jeder Gruppe gefordert werden. Die Angabe, welche Voraussetzungen er zu erfüllen hat, findet er im Advertisement jeder Gruppe. Ein wichtiger Punkt dabei ist, dass jeder Peer standardmäßig der sog. *NetPeerGroup* angehört, also auch deren Protokolle verstehen können muss.

- Zuletzt gibt es noch sogenannte Pipes. Diese stellen virtuelle, uni-direktionale Kanäle zwischen zwei oder mehreren Peers dar; Dabei wird zwischen Ein- und Ausgängen von Pipes unterschieden. Eine Pipe kennt grundsätzlich zwei verschiedene Kommunikationsmethoden: Die Point-to-Point Pipe realisiert eine 1-zu-1-Verbindung zwischen zwei Knoten, während die Propagation Pipe einen Eingang, aber mehrere Ausgänge besitzt und so für Broadcastanwendungen gedacht ist.

```
<?xml version=”1.0”?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta=”http://jxta.org”>
 <GID> urn:jxta:jxta-NetGroup</GID>
 <MSID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206</MSID>
 <Name>NetPeerGroup</Name>
 <Desc>NetPeerGroup by default</Desc>
</jxta:PGA>
```

Abb. 2.1 Ein Beispiel für ein Peer Group Advertisement

2.2 Die Protokolle von JXTA

Wie erwähnt, geben Peergruppen in ihren Advertisements stets auch die Dienste und Protokolle an, welche die teilnehmenden Knoten implementieren müssen. Da alle Knoten von Haus aus der *NetPeerGroup* angehören, müssen damit auch alle deren Protokolle verstehen können. Genau diese Protokolle stellen die Basis von JXTA dar, da sie den primären "Wortschatz" und die grundlegenden Fähigkeiten aller JXTA-Peers repräsentieren. Aufbauend auf den Konzepten der vorhergehenden Abschnitte definiert JXTA nun die folgenden (Basis-)Protokolle:

- Das Peer Endpoint Protocol (PEP) wird verwendet, um einen möglichen Pfad zwischen zwei Peers zu finden. Befinden sich beide Knoten im selben Subnetz, so kann die Nachricht einfach per Broadcast übertragen werden, ansonsten werden spezielle Router, die sogenannten "Relay Peers", verwendet, welche die Nachricht entsprechend weiterleiten sollen. Diese verwenden dabei das PEP, um den passenden Weg durch das Netz zu finden. Sollte sich der Pfad dynamisch ändern, kann mittels PEP der aktuelle Weg ermittelt werden.
- Das Rendezvous Protocol (RVP) realisiert eine Art Nachrichtenverteiler, den sogenannten Rendezvous Peer. Falls eine Nachricht einen Rendezvous Peer erreicht, wird sie an alle Knoten weiterverteilt, die in der Liste des Rendezvous Peers stehen. Dienstleistende Knoten können sich mittels RVP bei diesen Rendezvous Peers anmelden, um so schneller erreichbar zu sein und besser ihre Dienste anbieten zu können. Gleichzeitig sind diese Peers für alle der beste Weg, andere Knoten zu finden.

- Das Peer Resolver Protocol (PRP) stellt einen allgemeinen Weg dar, um Anfragen an andere Knoten zu richten, und Antworten darauf zu empfangen. Dabei werden alle notwendigen Konvertierungen und Hintergrundaufgaben durch das Protokoll automatisch durchgeführt, wie z.B. Adressauflösung mittels DNS oder Zugriffe mittels NFS. Die Anfragen werden durch sogenannte "Handler" bearbeitet, wobei jeder Anfragetyp einem bestimmten Handler zugeordnet ist. Das Protokoll erlaubt die Verbreitung der Anfragen auf mehrere "Handler" auf mehreren Knoten, wobei jeder Knoten nur die Handler implementieren muss, deren Anfragen er auch beantworten will.
- Das Peer Discovery Protocol (PDP) wird verwendet, um eigene Advertisements zu verbreiten sowie fremde Advertisements (für Peers, Peergruppen, Pipes, etc.) zu suchen und zu finden. Es benutzt dabei das PRP, um die nötigen Anfragen zu verbreiten.
- Durch das Peer Information Protocol (PIP) lassen sich Zustandsinformationen über andere Knoten sammeln, wie z.B. allgemeiner Zustand, Auslastung, Betriebszeit, Ausfallrate, etc.
- Das Pipe Binding Protocol (PBP) regelt den Aufbau eines virtuellen Kanals zwischen zwei oder mehreren Knoten. Es wird verwendet, um die Enden einer Pipe mit den jeweiligen Endpunkten der Knoten zu verbinden.

Die oben genannten Protokolle stellen nur minimale Anforderungen an die zugrundeliegende Hardware: Sie gehen von uni-direktionale Verbindungen aus, bei denen Nachrichten verloren gehen dürfen; Ebenso gibt es keine oberen Zeitschranken für Nachrichten im System. Diese einfachen Voraussetzungen sind nötig, um JXTA auf so vielen Plattformen wie möglich lauffähig zu halten.

Obwohl diese Protokolle das Basissystem für jeden JXTA-Peer darstellen, müssen nicht alle implementiert werden. Sollte ein Knoten z.B. niemals Informationen über andere Knoten benötigen und auch selbst nie eigene Statusinformationen preisgeben wollen, muss er auch kein Modul für das PIP bereitstellen. Desweiteren kann jeder Entwickler für seine Peergruppe weitere Protokolle entwerfen oder die vorhandenen ergänzen, z.B. ein angepasstes PDP oder ein verbessertes PRP. Es ist allerdings vorteilhaft für jeden Knoten, wenn er die wichtigsten Basisprotokolle beherrscht. So lässt es sich schließlich besser mit anderen Knoten kommunizieren.

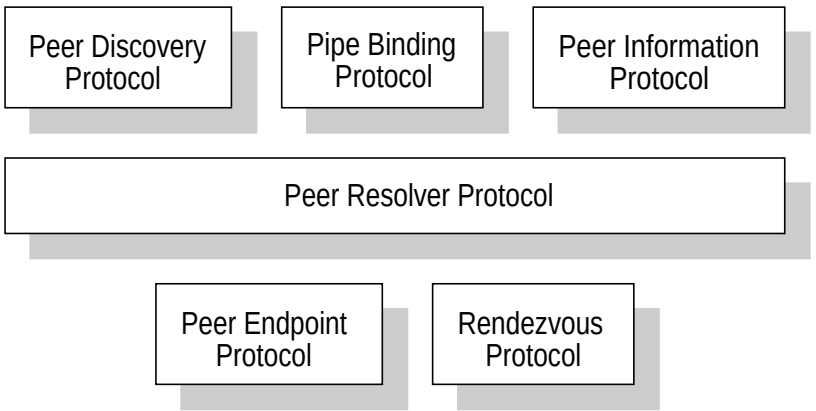


Abb. 2.2 Die Protokollhierarchie von JXTA

2.3 Die Architektur von JXTA

JXTA definiert, aufbauend auf den gerade besprochenen Protokollen, eine Softwarearchitektur, die wie folgt dargestellt werden kann:

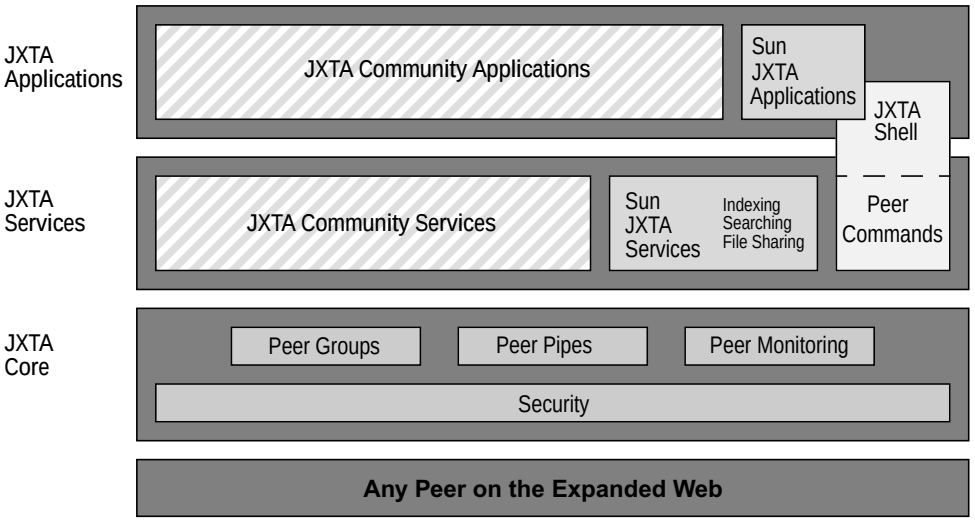


Abb. 2.3 Das P2P-Softwaremodell von JXTA

- Auf der untersten Ebene befindet sich der Kern von JXTA. Hier sind die besprochenen Protokolle und Konzepte realisiert, auf denen JXTA aufbaut, u.a. eben Peers, Peergruppen, Pipes, usw. Somit kann man den Kern gleichzeitig als eigentliches JXTA-System betrachten. Man beachte, dass, um die Flexibilität so groß wie möglich zu halten, auf dieser Ebene ausschließlich grundlegende Mechanismen implementiert werden. Die Anwendung dieser Mechanismen, v.a. derjenigen, welche für die Sicherheit relevant sind, ist Aufgabe der darüberliegenden Schichten.
- Die nächste Ebene stellt die Diensteebene dar. Hier werden, aufbauend auf den Konzepten und Protokollen von JXTA, komplexere Dienste zur Verfügung gestellt, z.B. eine Suchfunktion, ein komplexer Authentifizierungsmechanismus oder ein File-Sharing-Dienst. Hierzu wurden bereits einige Implementierungen von Sun und der JXTA Community geliefert, jedoch können von der Entwicklergemeinde immer noch weitere hinzugefügt werden, wie z.B. ein Sicherheits- oder Verschlüsselungsdienst.
- Die oberste Ebene schließlich ist die Schicht der jeweiligen Anwendung (z.B. ein Chat-Modul oder ein File-Sharing-Peer). Diese kann sowohl auf Dienste als auch auf die Kernschicht zurückgreifen, um ihre Funktion zu realisieren. Auch hier gibt es bereits eine kleine Anzahl von Anwendungen, die natürlich noch weiter anwachsen soll.

2.3.1 Die JXTA-Shell

Eine Besonderheit von JXTA ist eine spezielle Anwendung, die sog. JXTA-Shell. Angelehnt an Shells in UNIX oder Linux, ist die JXTA-Shell ein zeilenorientierter Kommandointerpreter, der Entwicklern und Neueinsteigern ein einfaches Interface in den JXTA-Kern bietet.

3 Poblano - ein Vertrauensmodell für JXTA

3.1 Motivation

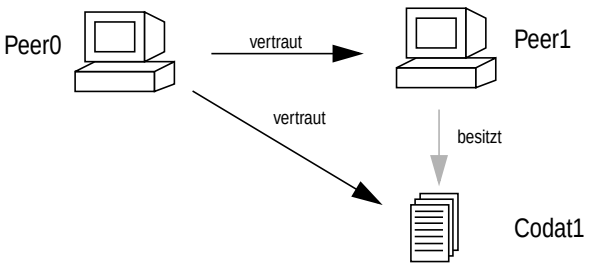
Das Projekt Poblano, ein Teilprojekt von JXTA, versucht ein allgemeines Vertrauensmodell aufzubauen, auf dessen Basis es möglich sein soll, Knoten sowohl nach Vertrauen, als auch nach Nützlichkeit einzustufen, und so einen Knoten im Hinblick auf weitere Kooperation möglichst genau einzuschätzen [CY01].

Bisherige Vertrauenskonzepte wie bei Publius oder Freehaven bestimmen das Vertrauen in einen Peer durch Werte wie Leistung, Ehrlichkeit oder Zuverlässigkeit [Publ][FrHav]. Dies ist sinnvoll, um den Knoten selbst zu bewerten, liefert jedoch keine Aussage über die Qualität seiner Dienste bzw. Daten. Poblano geht deswegen noch einen Schritt weiter und bewertet neben dem Knoten selbst auch das, was ihn für andere Peer eigentlich interessant macht, also die Services, die er zur Verfügung stellt.

3.2 Darstellung von Vertrauensverhältnissen

Wie erwähnt, werden unter Poblano sowohl Peers als auch deren angebotene Codats (“Code und Daten”) bewertet, und zwar nach Vertrauen und Nützlichkeit. Um die Nützlichkeit bewerten zu können, müssen die Interessen des bewertenden Knoten mit einbezogen werden. Dies geschieht durch die Verwendung von Schlüsselwörtern, welche die Interessen darstellen. Zur Veranschaulichung soll das folgende Beispiel dienen:

Peer0 sucht Daten zu einen bestimmten Begriff und stellt u.a. eine Anfrage bei Peer1. Peer1 besitzt Codat1, welches passende Daten zu dem Begriff enthält, und welches er an Peer0 übermittelt. Peer0 berechnet die “Zufriedenheit” mit dem Codat mit Hilfe des Benutzers. Aus dem Vertrauen, das er für Codat1 berechnet hat, bestimmt er dann sein Vertrauen fur Peer1.



Jeder Peer kennt drei unterschiedliche Vertrauensklassen, aus denen das endgültige Vertrauen bestimmt wird:

- CodatConfidence stellt die Zufriedenheit mit dem jeweiligen Codat dar und enthält u.a. auch Wertungen des Benutzers über das Codat.
- PeerConfidence stellt die Zufriedenheit mit einem Peer dar und wird aus den CodatConfidences aller Codats berechnet, die der Peer besitzt.
- Risk ist die klassische Komponente, in der Werte wie Leistung oder Verfügbarkeit des Peers verzeichnet sind.

Die Idee hinter diesen Klassen ist, dass der Benutzer die Daten eines Peers mittels CodatConfidence selbst bewerten kann (das ist einfacher, als den ganzen Peer an sich zu bewerten), und die restlichen Klassen automatisch berechnet werden: PeerConfidence wird aus CodatConfidence berechnet, und Risk wird vom unterliegenden System z.B. mittels PIP bestimmt.

Das folgende Szenario soll veranschaulichen, wie diese Vertrauensklassen konkret benutzt werden, um Vertrauen zwischen Knoten zu realisieren.

3.3 Implementation und konkrete Anwendung: Suchen nach Daten

3.3.1 Grober Ablauf

Jeder Peer besitzt Tabellen von verschiedenen Tabellentypen, um sein Vertrauen berechnen zu können: U. a. eine CodatConfidence-Tabelle für jede Peergruppe, der er angehört, und in der zu jedem Schlüsselwort eine Liste von passenden Codats steht; sowie eine PeerConfidence-Tabelle pro Peergruppe, die zu jedem Peer das Vertrauen bzw. die Nützlichkeit bzgl. eines Wortes eingetragen ist.

Eine Suche nach Daten zu einem bestimmten Schlüsselwort läuft wie folgt ab:

(1) Das Wort wird in der CodatConfidence-Tabelle gesucht. Falls es ein passendes Codat gibt, das auch lokal vorhanden ist, dann ist die Suche erfolgreich beendet. Falls nicht, so folgt Schritt 2.

(2) Das Wort wird in der PeerConfidence-Tabelle gesucht. Falls es Peers mit ausreichendem Vertrauen gibt, wird eine Anfrage an diese gestellt. Die angesprochenen Peers führen nun ihrerseits die Schritte 1 und 2 durch, bis ein Peer passende lokale Codats findet. Dieser informiert den Initiator und beendet seine Suche.

(3) Nachdem das Codat empfangen und verwertet wurde, werden nun die Tabellen aktualisiert. Der Popularitätszähler des Codats beim “Provider” wird erhöht, die CodatConfidence des Codats und darauf folgend die PeerConfidence des Providers wird auf Seiten des Initiators neu berechnet (mit Hilfe des Benutzers). Die neu errechnete CodatConfidence wird dem Provider als Feedback geliefert, so dass dieser seine Tabellen ebenfalls aktualisieren kann.

3.3.2 Formeln für die Berechnungen

Für die Aktualisierungen der Tabellen in Schritt 3 sind einige Berechnungen nötig. Dabei wird für die Zahlendarstellung der einzelnen Vertrauenswerte eine Werteskala im Bereich -1, 0, 1, 2, 3, 4 benutzt, mit folgender Zuordnung: *-1= Misstrauen, 0 = Ignorieren, 1 = geringes Vertrauen, 2 = durchschnittliches Vertrauen, 3 = gutes Vertrauen, 4 = volles Vertrauen*.

Als Beispiele für nötige Berechnungen werden die folgenden zwei Situationen betrachtet:

- Der Initiator berechnet die PeerConfidence des Providers neu. Logischerweise ist die Zufriedenheit mit dem Knoten mit der Zufriedenheit der von ihm zur Verfügung gestellten Codats verbunden, ebenso spielt der alte Wert des Providers eine Rolle. Deswegen wird der neue Wert einfach aus der Mittelung des alten Wertes und dem Mittelwert aller CodatConfidences gebildet:

$$newPeerConf = \left(oldPeerConf + \frac{1}{|K|} \sum_{a \in K} CodatConf_{Provider} \right) / 2$$

K = Anzahl an CodatConfidences des Providers

- Der Provider aktualisiert seine CodatConfidence durch das Feedback des Initiators. Der neue Wert des gerade verwendeten Codats ergibt sich aus dem alten Wert, ebenso spielt das Feedback des Initiators eine Rolle, aber es muss auch noch die Meinung berücksichtigt werden, die der Provider vom Initiator hat. Die zuständige Formel sieht deswegen wie folgt aus:

$$newCodatConf = \left(oldCodatConf + feedback \times \frac{PeerConf_{Initiator}}{4} \right) / 2$$

3.4 Der Kooperations-Schwellwert

In den letzten Abschnitten wurde gezeigt, wie die beiden Komponenten CodatConfidence und PeerConfidence berechnet werden. Diese beiden Komponenten sowie die Risk-Komponente werden nun dazu hergenommen, um den sogenannten Kooperations-Schwellwert einzuführen. Dieser legt letztendlich fest, ob ein bestimmter Peer als vertrauenswürdig genug eingestuft wird, um mit ihm zu kooperieren.

Der Schwellwert selbst wird berechnet aus allen drei Vertrauenskomponenten, sowie einem Wichtigkeitswert. Dieser Wert gibt an, wie wichtig es dem Benutzer ist, mit diesem Peer kooperieren zu können. Auf diese Weise kann der Benutzer direkt den Schwellwert beeinflussen. Die Werte für PeerConfidence und CodatConfidence werden wie gehabt in Werten -1, 0, 1, 2, 3, 4 kodiert; Der Risk-Wert dagegen in 0, 1, 2, 3 oder 4, wobei 0 kleinstes und 4 größtes Risiko bedeutet.

Der Schwellwert wird durch folgende Ungleichung angegeben, die wahr sein muss, damit Kooperation erlaubt wird:

$$PeerConf \times Importance > \frac{Risk_{Peer}}{\frac{1}{|K|} \sum_{a \in K} CodatConf_{Peer}}$$

3.5 Zertifikate in Poblano

Zertifikate spielen auch unter JXTA eine grosse Rolle, vor allem wenn es darum geht, sichere Peergruppen zu realisieren, denen sich Peers nur durch eine gültige Authentifizierung anschliessen dürfen, oder wenn man bestimmte Codats nur privilegierten Peers zugänglich machen möchte. Poblano unterstützt dabei ein weites Spektrum an Zertifikaten: Von einfachen, kostenlosen, selbstsignierten Zertifikaten über beglaubigte Zertifikate bis hin zu solchen, die von öffentlichen Certificate Authorities (CA) gegen Bezahlung signiert werden, ist alles unter Poblano verwendbar.

Das Vertrauen in Zertifikate wird ähnlich dem vorhergehenden Beispiel berechnet. Jeder Peer besitzt dafür leicht abgeänderte Tabellen: Eine CodatConfidence-Tabelle, in der alle bekannten Zertifikate samt deren Wertungen gespeichert sind, sowie eine PeerConfidence-Tabelle, die zu jedem bekannten Peer dessen Bewertung als Besitzer und als Fürsprecher von Zertifikaten beinhaltet. So kann ein Peer sowohl über seine eigenen als auch über seine empfohlenen bzw. unterstützten Zertifikate eingeschätzt werden. Die Bewertung der eigenen Zertifikate geschieht abhängig von der Sicherheit der Signierung , z.B. “1” für selbstsignierte oder “4” für CA-signierte Zertifikate.

4 Literaturverzeichnis

PJ-OIC01. *Project JXTA: An Open, Innovative Collaboration*, Sun Microsystems, Inc., 2001

PJ-GS01. *Project JXTA: Getting Started*, Sun Microsystems, Inc., 2001

LG01. Li Gong, *Project JXTA: A Technology Overview*, Sun Microsystems, Inc., 2001

TAD+02. Bernard Traversat, Mohamed Abdelaziz, Mike Duigou, Jean-Christophe Hugly, Eric Pouyoul, Bill Yeager, *Project JXTA Virtual Network*, Sun Microsystems, Inc., 2001

JXTA. Project JXTA, *www.jxta.org*

Gnut. Gnutella, *www.gnutella.com*

AMim. A&M Instant Messenger, *www.amsoftwaredesign.com/instantmessenger.asp*

CY01. Rita Chen, William Yeager, *Poblano - A Distributed Trust Model for Peer-to-Peer Networks*, Sun Microsystems, Inc., 2001

Publ. Publius, *cs1.cs.nyu.edu/~waldman/publius/publius.html*

FrHav. FreeHaven, *www.freehaven.net*

.