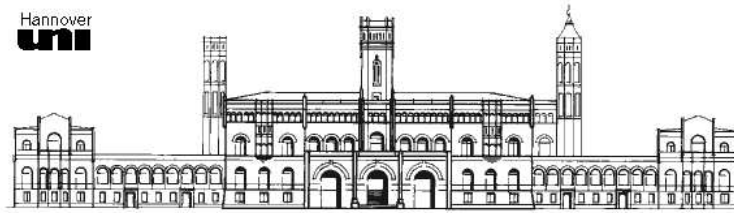


Hannover
uni



UNIVERSITÄT HANNOVER

INSTITUT FÜR INFORMATIONSSYSTEME
FACHGEBIET WISSENSBASIERTE SYSTEME

Diplomarbeit

**Mediation und Indexing in
P2P-basierten
Informationssystemen**

Verfasser : Ingo Brunkhorst

Erstprüfer : Prof. Dr. techn. Dipl.-Ing. Wolfgang Nejdil

Zweitprüfer : Prof. Dr.-Ing. Klaus Jobmann

Betreuer : Dipl.-Ing. Wolf Siberski

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

Hannover, den 13. Januar 2003

Zusammenfassung

Peer-to-Peer-Netzwerke gewinnen immer mehr an Bedeutung. Doch mit der steigenden Anzahl von Knoten und Daten in einem Netz steigt auch der Bedarf an Bandbreite und Rechenleistung. Reine Peer-to-Peer Netzwerke stoßen hier schnell an eine Grenze, die sich durch die Aufgabe der reinen Peer-to-Peer Infrastruktur überwinden läßt. In Super-Peer-Netzwerken existiert eine spezielle Klasse von Knoten, die Super-Peers. Super-Peers verfügen über eine sehr gute Netzanbindung und viel Rechenleistung, die sie den anderen Knoten zur Unterstützung bei Suchanfragen zur Verfügung stellen können. Eine solche Super-Peer Infrastruktur für das Edutella-Netzwerk wurde in dieser Arbeit entwickelt.

Inhaltsverzeichnis

1	Einleitung	3
1.1	Motivation	3
1.2	Software	4
2	Die Edutella Peer-to-Peer Architektur	5
2.1	Die Abfragesprache RDF-QEL	6
2.2	Edutella Common Data and Exchange Model	9
2.3	Query- und Response-Mechanismen in Edutella	10
2.4	Nachteile reiner Peer-to-Peer Architekturen	11
3	Super-Peer Netzwerke und Routing Strategien	13
3.1	Super-Peer/Peer-Indices	15
3.2	Super-Peer/Super-Peer-Indices	19
3.2.1	Das HyperCup Protokoll	20
3.2.2	Die Broadcasting-Topologie	22
3.3	Dynamische Routing-Indices	23
3.3.1	Clustering und Peer-Trading	24
3.4	Mediation zwischen Schemas	25
4	Implementierung	27
4.1	Aufgaben des Peers: Dienste	29
4.2	Initialisierung des Peers	30
4.3	Peerkonfiguration	31
4.3.1	Konfiguration des Provider-Peers	32
4.3.2	Konfiguration des Super-Peers	34
4.4	Topology	36
4.5	Super-Peer-Konfiguration	37
4.5.1	Aufbau der Super-Peer-Topology	37
4.5.2	Suche nach ModuleImplAdvertisement	38
4.5.3	Senden von Informationen: HELO-Message	39
4.6	Aufbau der SP/P-Indices: Der BIND-Service	43
4.6.1	Der BIND-Service des Super-Peers: BindingService	45
4.6.2	Der BIND-Service des Provider-Peers: ClientUplink	46
4.6.3	Verwaltung des SP/P-Indices: Die SpPIndexer-Klasse	47
4.7	Bearbeiten und Weiterleiten von Queries	47
4.7.1	Der QEL-(Edutella Query)-Service	48

4.7.2	Die Provider-Emulation des Super-Peers	50
4.7.3	Der ROUTE-Service	50
4.8	System-Management-Console: CONSOLE-Service	55
5	Die Zukunft des Edutella-Super-Peer-Netzwerkes	56
A	Anhang	59
A.1	API	59
A.2	Online	60
A.3	Services	60
A.4	PeerDescription Schema	62
A.5	SWEBOK-Query	63

Kapitel 1

Einleitung

Peer-to-Peer Netzwerke haben in den letzten Jahren in vielen Bereichen der Informationstechnologie Einzug gehalten. Dazu gehören die Tauschbörsen im Stile von Napster [2] und Gnutella [1], das verteilte Rechnen wie bei SETI@home [4] und Systeme zur Bereitstellung von Lehr- und Lernmaterial. Edutella ist ein im Rahmen des PADLR Projekts entwickelte „learning web infrastructure“ [24] auf Basis einer verteilten Netzwerk-Architektur. Die Abkürzung PADLR steht für „Personalized Access to Distributed Learning Repositories“. Eine kurze Einführung in die Edutella-Technologie findet sich in Kapitel 2. Durch den Einsatz des *Resource Description Frameworks* (RDF) zur Annotation der Materialien werden Möglichkeiten zum Durchsuchen des Datenbestandes ermöglicht, die über das Suchen mit einer einfachen Textzeichenkette hinausgehen. In einem Peer-to-Peer-Netzwerk zum Austausch von Daten und der Identifikation der Dateien über den Dateinamen, ist ein derartiges Suchverfahren ausreichend. Der Dateityp wird über eine Extension am Ende des Dateinamens festgelegt, beispielsweise “.mp3” für MPEG-Audio-kodierte Dateien. Sind alle relevanten Daten im Dateinamen kodiert, ist eine Suche mit einer Zeichenkette oder Teilen davon ausreichend. Ein Dateiname wie *Madonna_MaterialGirl.mp3* ist für einen Filesharing-Dienst zur Identifikation eines Musiktitels hinreichend genau, aber für viele Anwendungen ist diese Art Kodierung ungenügend. Aus *course1.html* ist nicht unbedingt zu erkennen, daß es sich dabei beispielsweise um den ersten Teil eines Mathematikurses für Ingenieure handelt. Metadaten bieten hier eine Möglichkeit, die Inhalte genauer zu beschreiben. Durch die Verwendung standardisierter Elemente lassen sich Dokumente inhaltlich und vor allem maschinenlesbar auszeichnen.

1.1 Motivation

Peer-to-Peer Netzwerke, in denen jeder Knoten mit jedem anderen kommuniziert, haben den Nachteil, daß die Kommunikation mit vielen Partnern gleichzeitig erfolgt und dementsprechend Bandbreite und Rechenleistung benötigt wird. Die von Beverly Yang und Hector Garcia-Molina vorgeschlagene Super-Peer Architektur [32] versucht dieses Problem durch eine Mischung von reiner Peer-to-Peer Architektur und einem Netzwerk von übergeordneten Super-Peers zu

lösen. An jeden dieser Super-Peers können sich "normale" Peers anschließen. Durch die Auswertung der in Edutella verwendeten RDF-basierten Abfragesprache RDF-QEL-i [30] ergibt sich die Möglichkeit, schon im Vorfeld Anfragen nur an die Peers zu senden, die auch eine Antwort liefern können. Im Edutella-Netzwerk sind Provider die Informationsquellen, die auf Anfragen aus dem Netz warten und diese mit Hilfe ihres Metadaten-Repositories zu beantworten versuchen. Die Consumer sind Peers, die auf RDF-QEL-i basierende Anfragen an die Provider stellen, und deren Antworten entgegennehmen. Die in Kapitel 3 beschriebenen Strategien ermöglichen es den Super-Peers, die Anfragen nur an die Provider weiterzuleiten, die durch ihre Metadatenbank in der Lage sind, eine Antwort zu liefern. Ein Peer, der ausschließlich die Titel aller in Europa veröffentlichten Dokumente über die Programmiersprache Java gespeichert hat, wird eine Frage zu einem anderen Themengebiet kaum beantworten können. Der Super-Peer erhält die Informationen über einen Peer während der Anmeldung und verwendet sie zum Aufbau von Routing-Indices. Für die Weiterleitung von Queries an die angeschlossenen Peers eines Super-Peers wird ein Super-Peer/Peer-Index, kurz SP/P-Index (siehe Kapitel 3.1) verwendet. Um das Netz zu strukturieren und zur Optimierung der Suche können auch für die Kommunikation zwischen den Super-Peers Routing-Tabellen eingesetzt werden. Peers, die über Daten zu einem bestimmten Gebiet verfügen, beispielsweise Lehrmaterial über Informatik, können an einem Super-Peer, einem Teilbaum oder Cluster des Super-Peer Netzwerkes versammelt werden (siehe Kapitel 3.3). Die für das Routing zwischen Super-Peers benötigten Super-Peer/Super-Peer-Indices, kurz SP/SP-Indices werden in Kapitel 3.2 beschrieben. Das Konzept der Routing-Indices findet sich in [9] und in [22]. Es gibt weitere Möglichkeiten, die Effizienz der Suche in einem Super-Peer-Netzwerk zu steigern. Durch geeignete Protokolle läßt sich sicherstellen, daß eine Suchanfrage mit einer genau definierten Anzahl von Schritten an alle Super-Peers übermittelt wird. Eines dieser Protokolle, die HyperCup-Architektur [27], wird in Kapitel 3.2.1 kurz beschrieben.

1.2 Software

In Kapitel 4 und Anhang A.3 werden die in dieser Arbeit erstellten Erweiterungen der Edutella-Netzwerk-Software vorgestellt. Das modulare Konzept ermöglicht die einfache Anpassung an weitere Aufgaben. Einige davon werden in Kapitel 5 aufgezeigt. Die erweiterten Peers werden aus Komponenten zusammengesetzt, den Services. Eine Beschreibung der wichtigsten Dienste findet sich in Kapitel 4.1. Neben den Services existiert eine zusätzliche Komponente zum Aufbau der Verbindungen zwischen den Super-Peers (siehe Kapitel 4.4). Der Aufbau eines Peers wird über ein Konfigurationsmodul gesteuert (siehe Kapitel 4.3).

Kapitel 2

Die Edutella Peer-to-Peer Architektur

Dieses Kapitel ist eine kurze Beschreibung der grundlegenden Edutella-Infrastruktur und der Peers, aus denen das Edutella-Netzwerk aufgebaut ist. Edutella basiert auf der Peer-to-Peer Technologie JXTA von Sun [23]. Edutella und JXTA sind in der Programmiersprache JAVA implementiert. Abbildung 2.1 zeigt einen Überblick über die JXTA-Architektur. Eine ausführliche Beschreibung der JXTA-Architektur findet sich in [23].

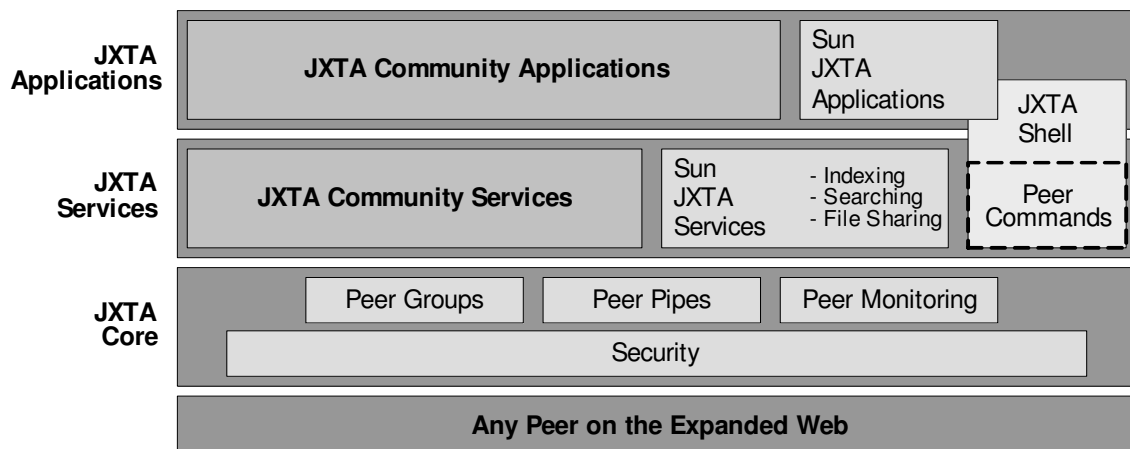


Abbildung 2.1: Der Aufbau der JXTA Architektur

Im Edutella Netzwerk wird zwischen zwei verschiedenen Rollen unterschieden, die von den Peers eingenommen werden können [22]. Die *information provider role* und die *information consumer role*. Eine Sonderaufgabe übernimmt der Rendezvous-Peer. Die drei Typen von Peers sind in Abbildung 2.2 skizziert. Consumer schicken ihre Anfragen an die Provider und nehmen deren Antworten entgegen. Consumer bilden damit die Schnittstelle zum Anwender. Basis für die Kommunikation zwischen Provider und Consumer ist das *Edutella Common Data*

and Exchange Model (ECDM) und die Sprache *RDF-QEL-i*. Der Rendezvous-Peer dient zur Unterstützung der Kommunikation zwischen den Peers, beispielsweise als Relay zur Anbindung von Peers hinter einer Firewall.

Die RDF-Metadaten werden im Provider in einem Repository gespeichert. Metadaten können in Java beispielsweise in einem RDF-Datenmodell aus dem Jena-Projekt [14], oder aber eine separate Datenbank, die nicht auf RDF basiert. Über die *Edutella-Wrapper* [30] kann die in der Sprache *RDF-QEL-i* vorliegende Query in verschiedene Abfragesprachen umgesetzt werden, beispielsweise SQL, dbXML [10], O-Telos [31] und Prolog [11].

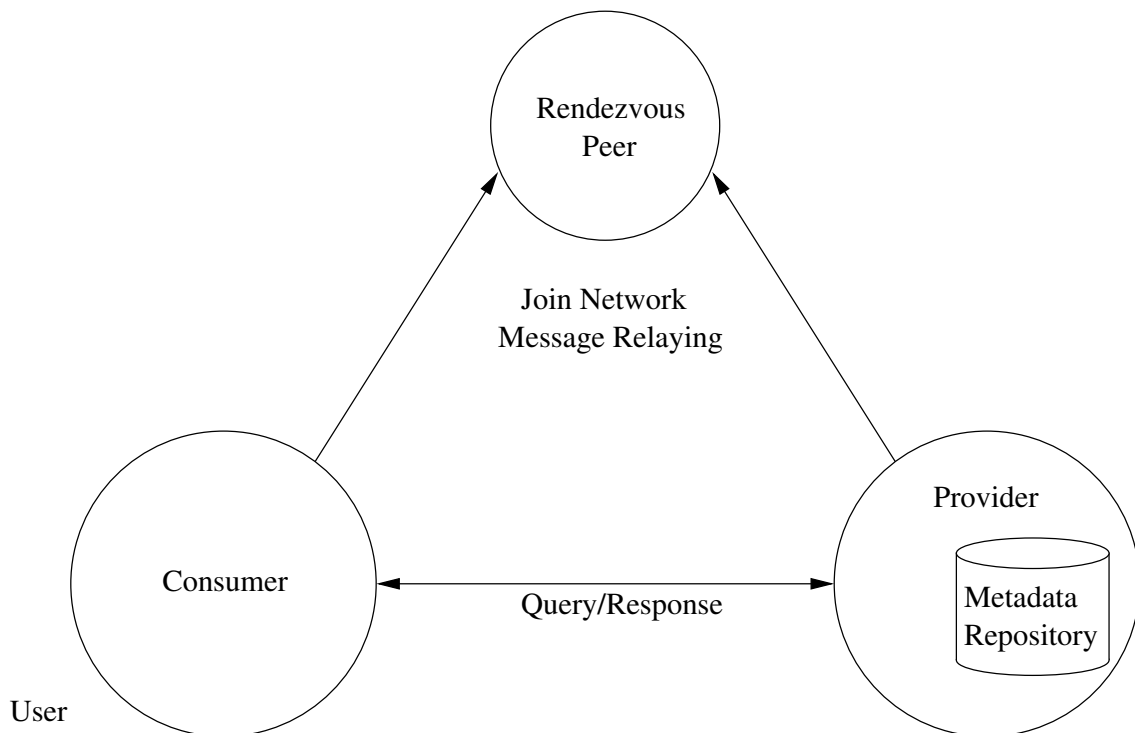


Abbildung 2.2: Provider, Consumer, Rendezvous: Die drei Peer-Varianten im Edutella-Netzwerk

2.1 Die Abfragesprache RDF-QEL

Anfragen an die Provider im Edutella-Netzwerk sind in der *RDF Query Exchange Language (RDF-QEL)* [15] formuliert. Zur Anpassung an die Fähigkeiten der einzelnen Peers sind in *RDF-QEL* verschiedene — sich in der Ausdrucksfähigkeit unterscheidende — Stufen definiert. Eine Query auf der niedrigsten Stufe *RDF-QEL-1 (pure conjunctive queries)* ist direkt als ein RDF-Graph darstellbar. Komplexere Anfragen, die über die Möglichkeiten reinen RDFs hinausgehen, können durch die Stufen *RDF-QEL-2* und höher abgebildet wer-

den. Ein RDF-Graph ist ein gerichteter Graph aus den Statements des RDF-Modells [16]. Statements sind Triple aus den Elementen *Subject*, *Predicate* und *Object*. Sie bilden die Grundbausteine des RDF-Datenmodells. Subject und Object sind Knoten des Graphen, das Predicate bildet die Kante zwischen ihnen. RDF-QEL basiert auf Datalog, einer nicht-prozeduralen Sprache ähnlich Prolog. Eine ausführliche Behandlung der Datalog-Semantik und der verschiedenen Stufen von RDF-QEL findet sich in [30]. Die Grundlage für die nachfolgenden Beispiele bildet die Aufgabe:

Suche nach allen *Objekten*, die einen *Titel* besitzen.

RDF sieht die Verwendung von Schema-Dateien zur Definition von Klassen und Attributen der Metadaten vor. Für beliebige Medien können die *Dublin Core Elements (DC)* [28] zur Auszeichnung verwendet werden. Dublin Core definiert eine Reihe von *Properties* zur Beschreibung von Dokumenten. Dazu gehören *Titel* (title), *Autor* (author), *Sprache* (language) und *Quelle* (source).

Die Query zur Suche nach allen *Resourcen*, die ein Property `dc:title` besitzen, läßt sich wie in Beispiel 2.1 in Datalog schreiben.

Beispiel 2.1 Suche nach allen Titeln (Datalog)

```
resources(X)    :- http://purl.org/dc/elements/1.1/title(X,Y).
? - resources(X)
```

Statements können in folgender Form geschrieben werden: $S(\text{subject}, \text{predicate}, \text{object})$ [13]. Die Namespaces können abgekürzt dargestellt werden, so kann für das Element `http://purl.org/dc/elements/1.1/title` die verkürzte Schreibweise `dc:title` verwendet werden. Neben Dublin Core existieren eine Reihe von weiteren RDF-Schemas. Zu den im Edutella-Umfeld häufig verwendeten gehört das LOM-Schema (*Learning Objects Metadata*)[3]. Für die Darstellung der Queries in RDF wird das Edutella-Schema [15] verwendet. In den RDF-Repositories werden durch diese Query alle Statements gesucht, die als Predicate das Attribut `dc:title` verwenden.

$$\forall X, Y \quad S(X, \text{dc:title}, Y) \quad (2.1)$$

Beispiel 2.2 zeigt die XML-Serialisierung der Query, basierend auf der Formulierung in der Sprache RDF-QEL-3.

Beispiel 2.2 Beispielanfrage zur Suche aller Titel

```

1  <?xml version='1.0' encoding='ISO-8859-1'?>
    <!DOCTYPE rdf:RDF [
      <!ENTITY dc 'http://purl.org/dc/elements/1.1/'>
      <!ENTITY edu 'http://www.edutella.org/edutella#'>
      <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
    ]>

    <rdf:RDF
10     xmlns:dc="&dc;"
        xmlns:edu="&edu;"
        xmlns:rdf="&rdf;"
        xmlns:rdfs="&rdfs;">
        <edu:QEL3Query rdf:about="#TitleQuery">
            <edu:hasQueryLiteral rdf:resource="#Stmt0"/>
            <edu:hasResultType rdf:resource="&edu;TupleResult"/>
        </edu:QEL3Query>
        <edu:Variable rdf:about="#Resource" rdfs:label="Resource"/>
        <edu:Variable rdf:about="#Title" rdfs:label="Title"/>
20     <edu:RDFReifiedStatement rdf:about="#Stmt0">
            <rdf:subject rdf:resource="#Resource"/>
            <rdf:predicate rdf:resource="&dc;title"/>
            <rdf:object rdf:resource="#Title"/>
        </edu:RDFReifiedStatement>
    </rdf:RDF>

```

XML [6] ist eine auf SGML (Standard Generalized Markup Language) basierende Auszeichnungssprache für Textdokumente. SGML selbst ist als ISO-Norm 8879 festgeschrieben. Die RDF-Elemente aus Beispiel 2.2 befinden sich in den Zeilen 14-24. Begrenzt sind sie durch das Startelement `<rdf:RDF...>` in den Zeilen 9-13 und dem Endelement `</rdf:RDF>` in Zeile 25. Die Zeilen 3-6 definieren Abkürzungen für die RDF-Schemas Dublin Core (`dc`), Edutella (`edu`), RDF Syntax (`rdf`) [16] und RDF Schema [7] (`rdfs`).

Zu den Fähigkeiten des Resource Description Frameworks, von denen in dieser Query Gebrauch gemacht wird, gehört *Reification*, die Beschreibung von RDF-Ausdrücken durch spezielle RDF-Elemente (Zeile 20-24). Der gerichtete Graph in Abbildung 2.3 zeigt die Query aus Beispiel 2.2 als RDF-Graph. Der String `file:title_query.rdf` in einigen Knoten ist der Dateiname, unter dem diese Query lokal gespeichert wurde. Zu erkennen ist auch, daß die Ressourcen "Title" und "Resource" Edutella-Variablen sind, ihre Kanten mit dem Predicate `rdf:type` verweisen auf den Knoten `edu:Variable`.

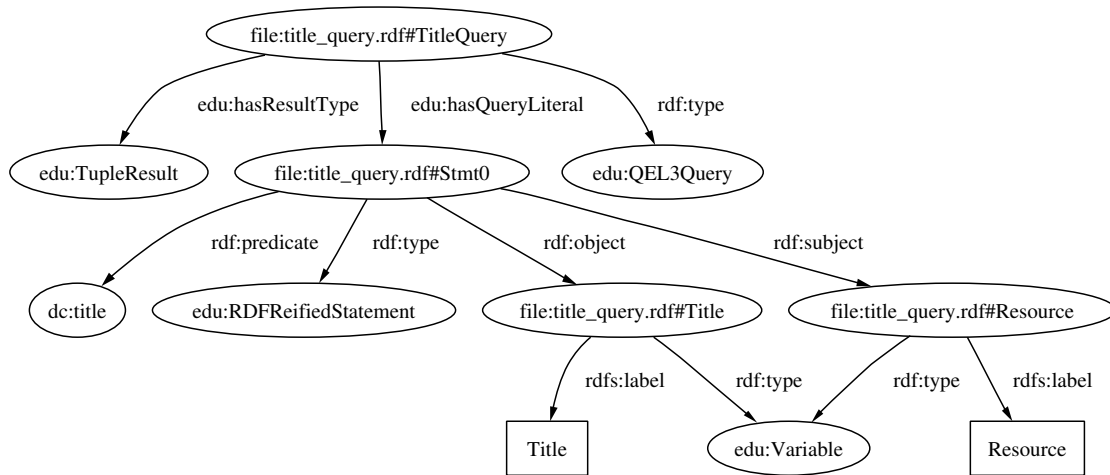


Abbildung 2.3: Query aus Beispiel 2.2 als Graph

Das durch die Reification beschriebene Statement `Stmt0` mit den durch die Attribute `rdf:subject`, `rdf:predicate` und `rdf:object` beschriebenen Elementen entspricht genau der Query aus Beispiel 2.1. `Title` und `Resource` sind die Variablen X und Y .

2.2 Edutella Common Data and Exchange Model

Das *Edutella Common Data and Exchange Model* (ECDM) definiert ein Java-Modell für die Kommunikation zwischen den Peers des Edutella-Netzwerks. Bild 2.2 zeigt das UML Klassendiagramm für die Repräsentation von Anfragen und Antworten.

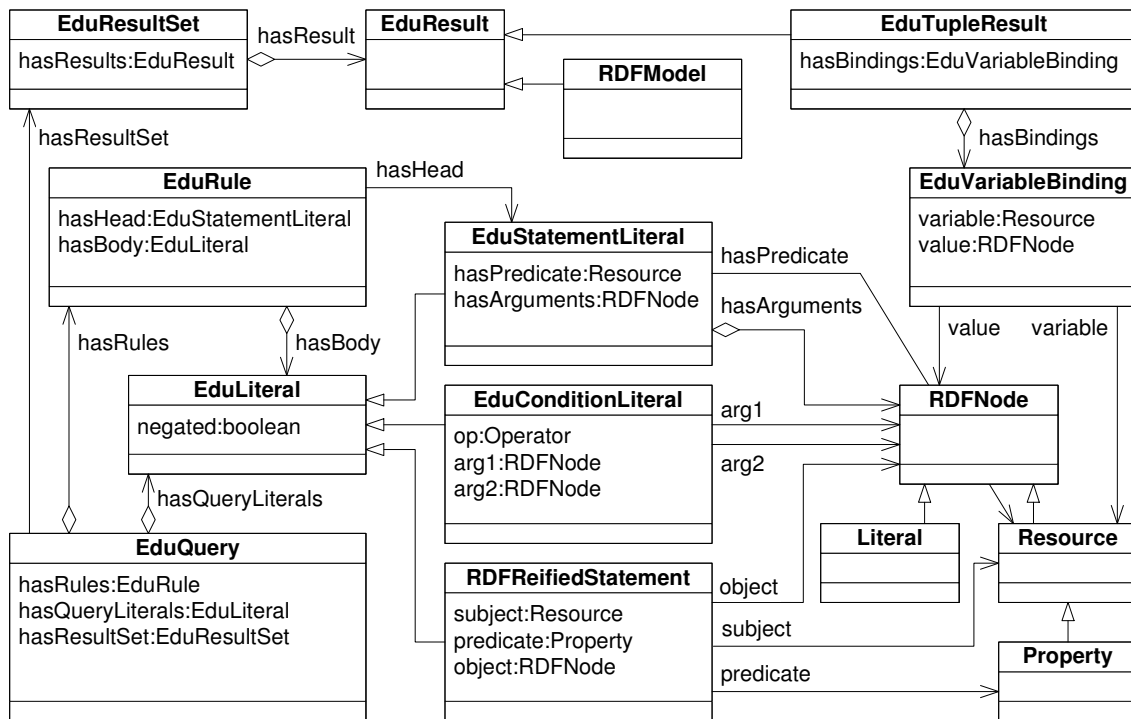


Abbildung 2.4: ECDM: Edutella Common Data and Exchange Model

Eine Query besteht aus Objekten vom Typ `EduQuery`. Aufgebaut sind `EduQuery`-Objekte aus `EduRules` (Regeln) und `EduLiterals` (Literalen). Literale sind durch *Reification* definierte Statements (`RDFReifiedStatements`), oder durch binäre Operatoren verknüpfte Ausdrücke `EduConditionLiterals`. Eine genaue Beschreibung aller Klassen findet sich in [30].

Die Klassen `RDFNode`, `Literal`, `Resource`, `Property` und `RDFModel` entstammen dem RDF-Toolkit Jena [14].

2.3 Query- und Response-Mechanismen in Edutella

Die Kommunikation zwischen den Peers wird über transparente Datenkanäle abgewickelt, den Pipes. Über die Discovery-Services lassen sich die Adressen eines solchen Kanals ermitteln. Dazu muß der bereitstellende Peer seine Pipes im Netz über die Publishing-Services annonciieren. Die dafür benötigte Kommunikation über das jeweilige Netzwerkinterface und die Hardware wird dabei komplett von JXTA erledigt.

Bild 2.5 zeigt die Aktionen eines "information consuming" Peers (Consumer) und eines "information providing" (Provider) Peers. Im ersten Schritt öffnet und veröffentlicht der Provider seine Pipes über die Publishing-Services des JXTA-Netzwerks. Diese werden im zweiten Schritt von einem Consumer über die

Discovery-Services gefunden und identifiziert. Der Consumer öffnet nun einen Kanal zum Provider-Peer und übermittelt seine Anfrage (3. Schritt). Nach der Bearbeitung sendet der Provider die Antwort zum Consumer-Peer zurück. Dazu verwendet er die mit der Anfrage übermittelte Antwortadresse (*reverse path*).

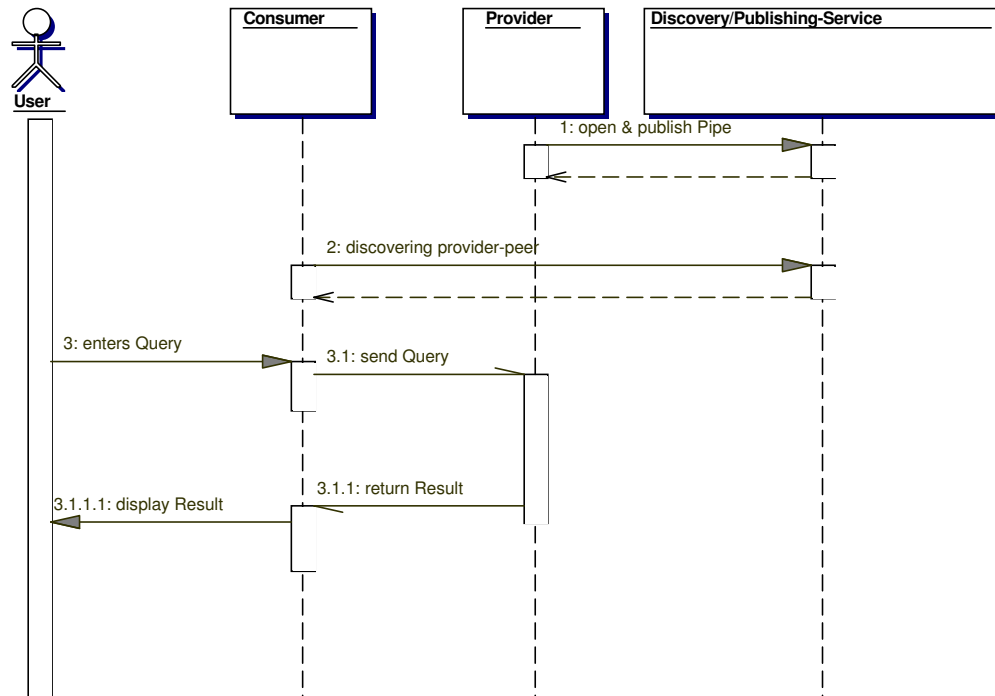


Abbildung 2.5: UML Sequenzdiagramm: Anfrage an Provider

Die in dieser Arbeit entwickelten Peers besitzen alle einen identischen Unterbau, doch jeder Peer kann durch Komponenten erweitert werden, um bestimmte Aufgaben zu erfüllen. So verfügt der Provider-Peer in seiner minimalen Ausbaustufe über einen Dienst, um in RDF-QEL formulierte Anfragen zu beantworten. Details zu den Diensten und ihrer Implementierung finden sich in Kapitel 4.

2.4 Nachteile reiner Peer-to-Peer Architekturen

Alle Peer-to-Peer Architekturen, in denen die Knoten gleichwertig in das Netz integriert sind, skalieren nicht. Grund dafür sind die technischen Einschränkungen einzelner Peers die damit zum Flaschenhals für die Kommunikation im gesamten Netz werden. Als Beispiel ist hier das Gnutella-Netzwerk [1] zu nennen, das im Herbst 2000 einen rasanten Zuwachs an beteiligten Knoten zu verzeichnen hatte.

Es zeigte sich, daß Peers an langsamen Einwählleitungen durch die Suchanfragen überlastet wurden. Siehe dazu [32] und [26]. In anderen Peer-to-Peer Netzwerken wurde die Suche von Dokumenten einem zentralen Index überlassen. Auch diese Systeme skalieren nicht, denn die Rechenleistung und Netzanbindung des zentralen Knotens bestimmt die Leistung einer solchen Suchfunktion. Eine zusätzliche Anforderung an die Suchmaschine ist die Verfügbarkeit, da bei einem Ausfall das Auffinden von Dokumenten systemweit nicht mehr möglich ist. In einem unstrukturierten Netzwerk, zu denen auch die aktuelle Edutella-Implementierung gehört, findet die Platzierung der Peers unabhängig von den in den Provider-Peers gespeicherten Metadaten statt. Für die Suche in einem solchen, sich ad hoc organisierenden Netz werden die gefundenen Knoten in zufälliger Reihenfolge angesprochen. In einem Netz, in denen keine Informationen über die einzelnen Knoten vorhanden sind, gibt es keine andere Möglichkeit der Suche [18].

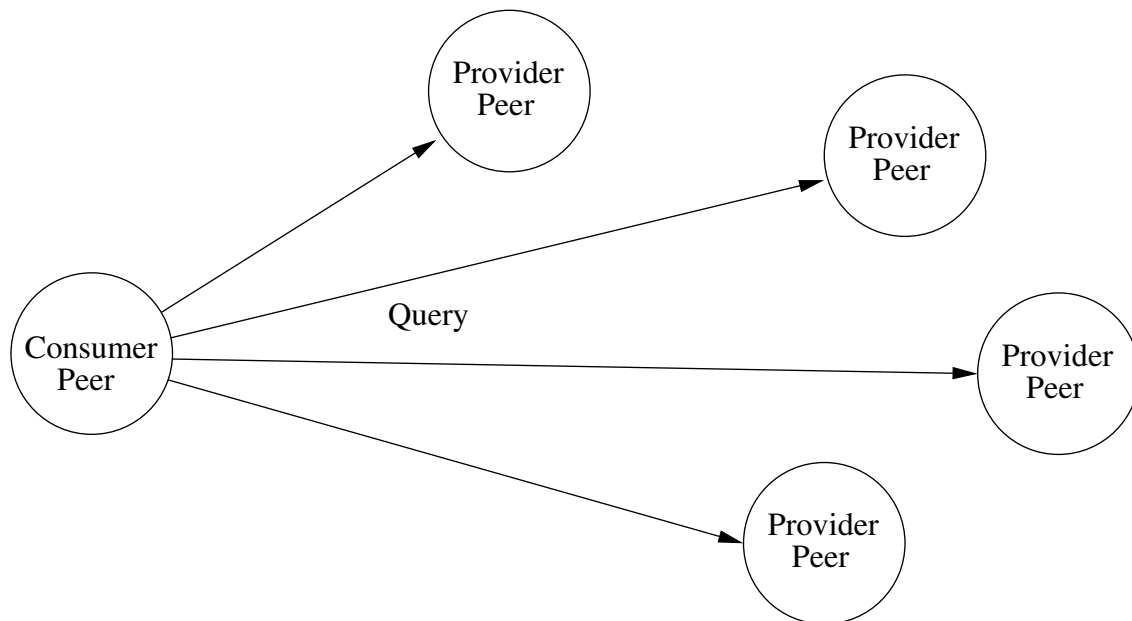


Abbildung 2.6: Abfrage von Peers in einem unstrukturierten Netzwerk

In ihrem Paper "Designing a Super-Peer Network" [32] beschreiben Beverly Yang und Hector Garcia-Molina eine andere Art von Infrastruktur für Peer-to-Peer Netze, um die genannten Einschränkungen zu überwinden. Das folgende Kapitel widmet sich diesen Super-Peer Netzwerken und ihrer Anwendung im Bezug auf das Edutella-Netzwerk und den resultierenden Vorteilen.

Kapitel 3

Super-Peer Netzwerke und Routing Strategien

Ein Super-Peer ist ein spezieller Knoten in einem Peer-to-Peer Netzwerk. Super-Peers sind prinzipiell Knoten mit einer sehr guten und stabilen Netzanbindung und größerer Rechenleistung als normale Peers.

Für eine Gruppe von normalen Peers bilden die Super-Peers eine Art von Server, an die sich die Peers als Client anschließen können. Dabei ist zu berücksichtigen, daß sich ein Client-Peer nur an *einen* Super-Peer anschließen kann.

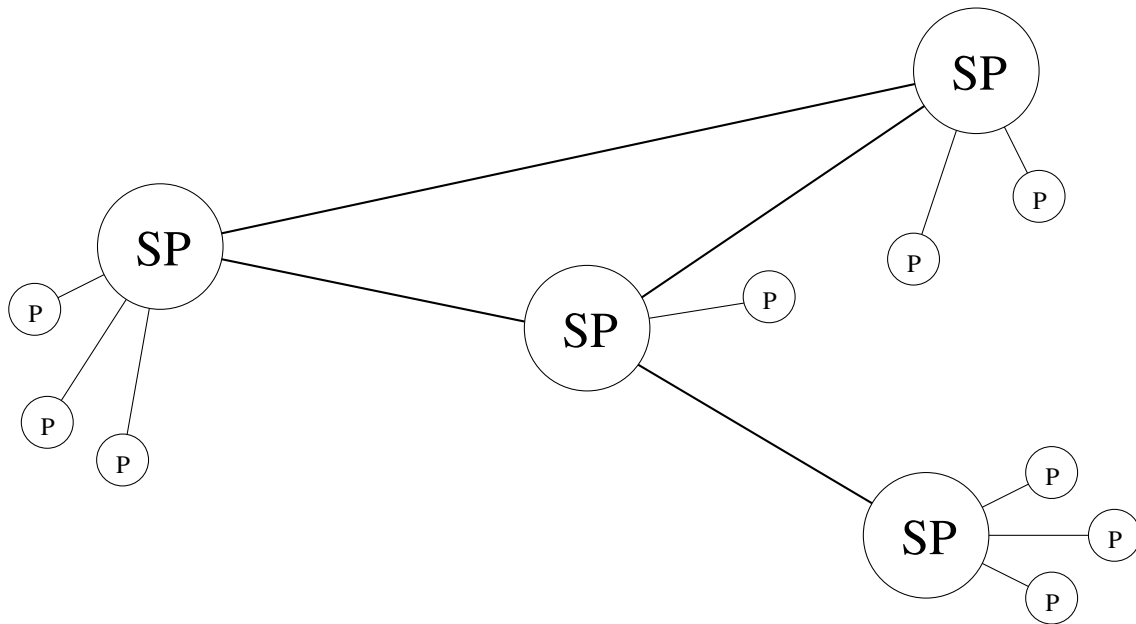


Abbildung 3.1: Eine einfache Super-Peer-Netzwerk-Topology

Die Super-Peers untereinander bilden ein eigenes Peer-to-Peer Netzwerk. Abbildung 3.1 zeigt die Struktur einer einfachen Super-Peer-Topology. Die Kreise mit der Bezeichnung SP repräsentieren die Super-Peers, die mit der Bezeichnung P die einfachen Peers.

Ein Super-Peer und seine Clients werden auch Cluster genannt, dabei ist die Clustergröße die Summe aller angeschlossenen Peers inklusiv des Super-Peers. Abbildung 3.2 verdeutlicht diesen Zusammenhang, die Clustergrenzen sind hier durch punktierte Linien gekennzeichnet.

Ein reines Peer-to-Peer Netzwerk läßt sich auch als ein "degeneriertes" Super-Peer-Netzwerk betrachten, in dem die Größe der Cluster 1 beträgt. Jeder Cluster besteht aus genau einem Peer [32].

In Peer-to-Peer-Tauschbörsen haben die Super-Peers die Aufgabe, Clients bei der Suche nach Dokumenten zu entlasten. Anfragen in solchen Netzen werden an die Super-Peers gesendet. Weitergeleitet werden die Anfragen von den Super-Peers an die angeschlossenen Clients nur, wenn sie die Anfragen nicht selbst beantworten können. Super-Peers in diesen Netzen verfügen über Kopien der in den Clients vorhandenen Datenbanken oder speichern die Anfragen und die dazugehörigen Antworten der Clients (Cache). Die Aufgaben des Super-Peers im Edutella-Netzwerk weichen in einem wichtigen Punkt von der in [32] beschriebenen Definition ab. Der Super-Peer hat keine lokale Kopie der im Provider gespeicherten Metadaten und übernimmt auch nicht die Suche für den Provider-Peer. Im Edutella-Netz werden die Super-Peers verwendet, um Queries nur an die Peers zu verschicken, die durch ihrer Metadaten in der Lage sind, eine Antwort zu liefern. Ein Provider mit einer medizinischen Datenbank wird in den seltensten Fällen Anfragen zur Entwicklung von Hochfrequenzantennen beantworten können. Super-Peers können die Provider thematisch gruppieren (Clustering), so daß sich Peers mit ähnlichen Metadaten in einem Cluster befinden. Dadurch wird Bandbreite gespart, denn Queries müssen nur noch an die Super-Peers geschickt werden, die Peers mit entsprechenden Metadaten angebonden haben. Für einen Edutella-Consumer-Peer ist die Suche in einem Super-Peer Netzwerk nicht von der in einem unstrukturierten Peer-to-Peer Netzes zu unterscheiden. Das Netzwerk aus den Super-Peers verhält sich wie ein "Backbone".

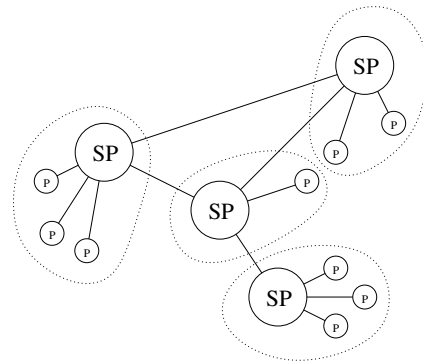


Abbildung 3.2: Clustergrenzen

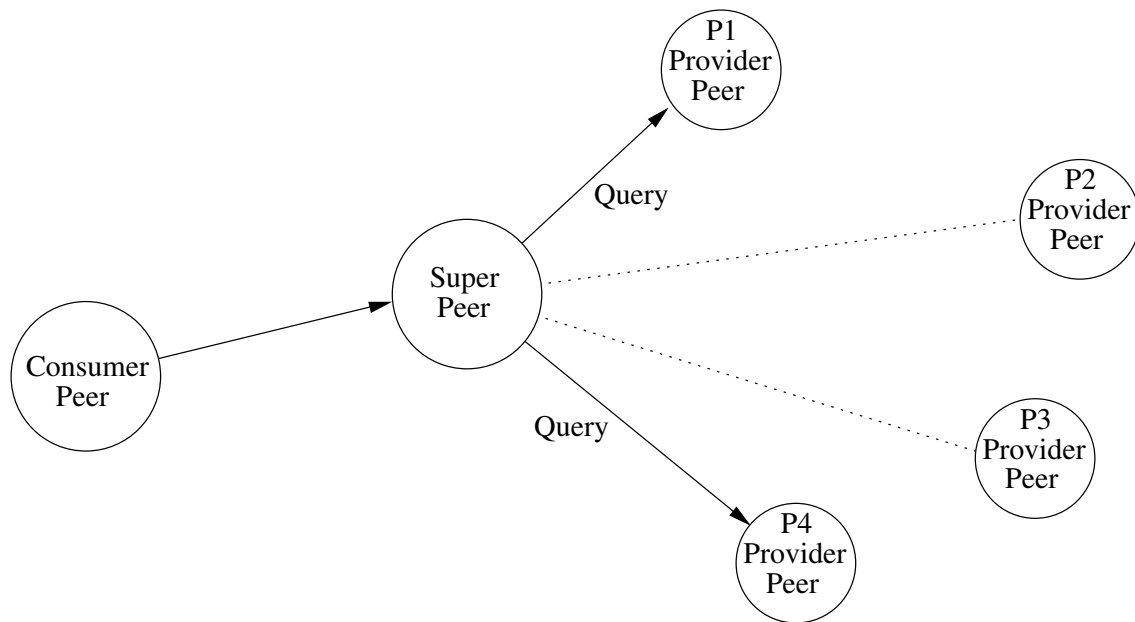


Abbildung 3.3: Super-Peer Netzwerk Topology

Abbildung 3.3 zeigt, wie Anfragen vom Super-Peer an die Provider verteilt werden. Die Provider-Peers P2 und P3 sind zwar angebunden, erhalten aber keine Anfragen vom Super-Peer. Die Mechanismen um zu entscheiden, an welche Peers die Queries weitergeleitet werden, sind in [21] und [22] aufgezeigt. Die nächsten beiden Abschnitten 3.1 und 3.2 befassen sich mit den Indices die zur Weiterleitung von Queries verwendet werden. In Abbildung 3.3 weiß der Super-Peer durch seinen Index, daß nur die Provider-Peers P1 und P4 die Query beantworten können. Beim Routing von Queries in einem Super-Peer-Netzwerk muß zwischen zwei Fällen unterschieden werden. Das Routen von Queries und Responses zwischen den Super-Peers und das anschließende Weiterleiten der Queries vom Super-Peer zu den angeschlossenen Providern.

3.1 Super-Peer/Peer-Indices

Die Super-Peer/Peer-Indices steuern das Query-Routing zu den an den Super-Peer angeschlossenen Provider-Peers.

Ein Provider, der das Peer-to-Peer-Netzwerk betritt, meldet sich bei einem der Super-Peers an. Dies wird im allgemeinen als "Join" bezeichnet. Bei dieser Anmeldung übermittelt er dem Super-Peer eine Beschreibung seiner Fähigkeiten. Beim Verlassen des Netzes meldet sich der Provider bei seinem Super-Peer wieder ab. Der Super-Peer kann so belegte Ressourcen wieder freigeben. Auch für den Fall einer nicht koordinierten Entfernung eines Providers aus dem Netz müssen Maßnahmen getroffen werden.

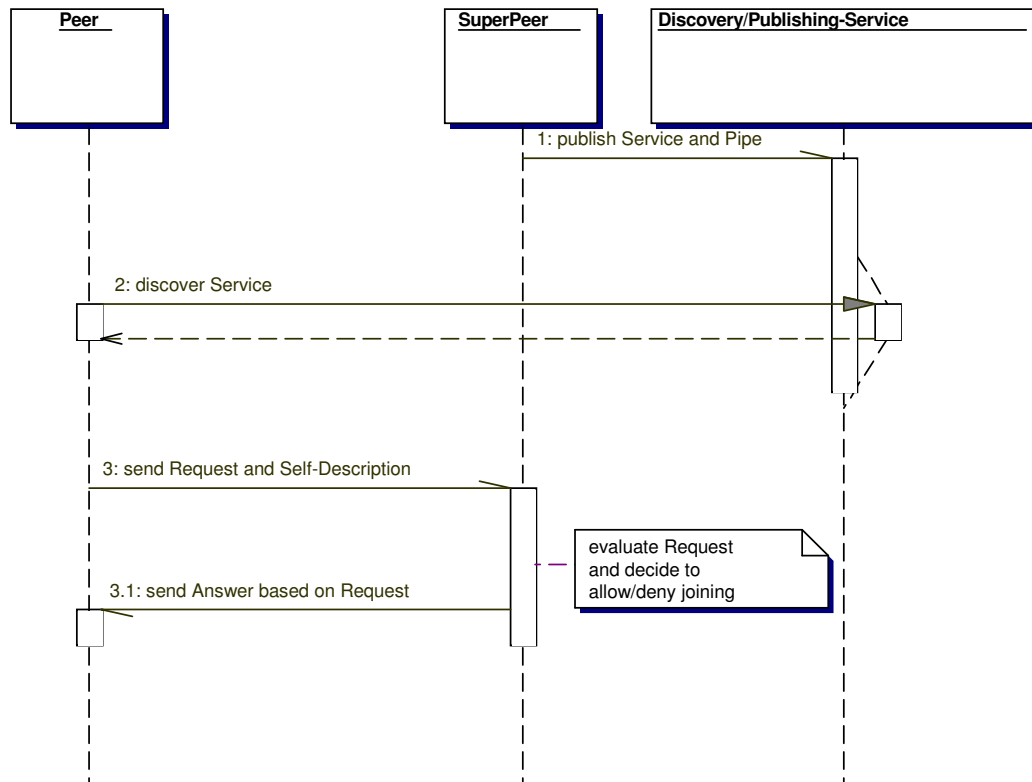


Abbildung 3.4: UML Sequenzdiagramm: Provider meldet sich beim Super-Peer an

Abbildung 3.4 zeigt den vereinfachten Ablauf eines solchen Anmeldevorgangs. In Sequenz 2 versucht der Provider, einen Super-Peer zu finden. Der Provider sendet eine Beschreibung seiner Fähigkeiten an den Super-Peer (Sequenz 3). Dieser kann jetzt entscheiden, ob er den Peer aufnimmt oder zurückweist. Aus der Information in der Peer-Beschreibung des Providers wird der Routing-Index aufgebaut.

Zur Klassifizierung von Properties werden in RDF die RDF-Schema-Definitionen verwendet. Durch die Auswertung dieser Schemas und der Properties können Queries und die Inhalte der Metadaten-Repositories charakterisiert werden.

Die Query aus Beispiel 3.1 ist eine Suche nach deutschsprachigen Dokumenten über Softwaredesign. Eine ähnliche Query befindet ist im Anhang A.5 mit XML-Serialisierung und RDF-Graph abgebildet.

Tabelle 3.1 zeigt die verschiedenen Granularitätsstufen zur Charakterisierung einer Query.

Beispiel 3.1 Query: Dokumente über Softwaredesign

Finde alle Resources in der Sprache (dc:language) *deutsch*, dem Dokumenten-Format (rdf:type) *PDF*, aus dem durch die Klassifikation (Taxonomy lomcls:Taxon) und das Vokabular (Ontology) definierten Bereich *Software-design* (swtont:SWDesign).

lomcls

http://www.imsproject.org/rdf/imsmd_classificationv1p2
LOM-Klassifizierung

swtont

http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf
SWEBOK-Ontologie [5]

Granularity	Query	
<i>Schema</i>	dc (Dublin Core) lom (Learning Object Metadata)	
<i>Property</i>	dc:subject, dc:language, lom:context	
<i>Property Value Range</i>	lomcls:Taxon	swtont:SWDesign
<i>Property Value</i>	dc:type dc:language	“PDF” “de”

Tabelle 3.1: Charakterisierung der Query aus Beispiel 3.1

Auf Schema-Ebenen wird zwischen den verschiedenen Namespaces unterschieden. Auf der höchsten Stufe wird für die Eigenschaft `dc:type` der Wert “PDF” verlangt.

Schema Index Provider, die keine Metadaten aus dem Dublin Core Schema, LOM und der SWT-Ontology besitzen, können Anfragen dazu nicht beantworten. Bei der Weiterleitung der Query müssen diese daher nicht berücksichtigt werden. In Abbildung 3.5 ist der Super-Peer/Peer-Index in einem Super-Peer hervorgehoben dargestellt. In handelt sich um einen Schema-basierten Index, der aus den Beschreibungen der Provider-Peers P1, P2 und P3 ermittelt wurde. Peer 1 enthält Metadaten aus aus dem Dublin Core und LOM-Schema. Peer 2 enthält nur mit Dublin-Core beschriebene Daten, während Peer 3 nur Objekte beschreibt, die nach dem LOM-Standard annotiert wurden.

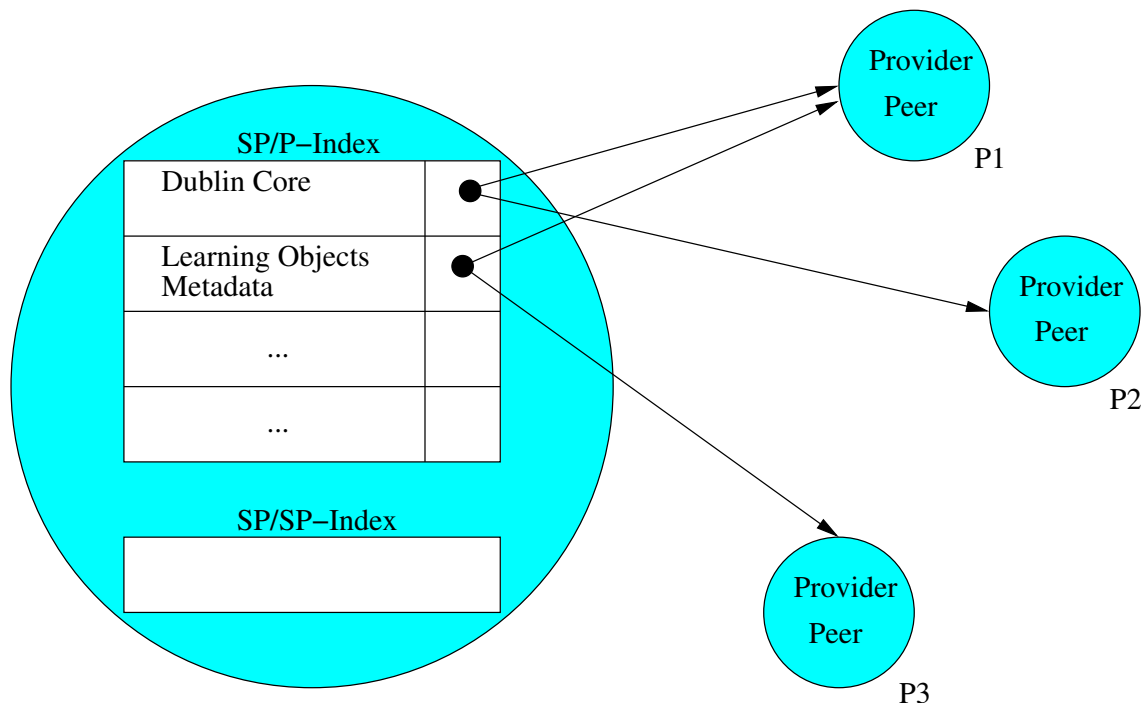


Abbildung 3.5: Super-Peer/Peer-Index

Eine Query wie die aus Beispiel 2.2 enthält nur ein Dublin-Core-Element. Der Index aus Abbildung 3.5 liefert die Provider-Peers P1 und P2 als Ziel für die Weiterleitung der Query.

In RDF existieren Properties für Konzepte wie *Autor* eines Dokuments in verschiedenen Varianten und in mehreren Schemas. Die Umsetzung (Mediation) zwischen unterschiedlichen Schemas wird in Abschnitt 3.4 beschrieben.

Property/Sets of Properties Index Statt den Index auf Basis eines gesamten Schemas zu bilden, wird nur eine Untermenge der Properties eines Schemas ausgewertet. Die Anfrage aus Beispiel 2.2 wird nur an Provider gesendet, die zumindest das RDF-Attribut `dc:title` kennen.

Property Value Range and Property Value Der in diesem Fall verwendete Index unterscheidet das Routingziel durch die Klassifizierung nach `lomcls:Taxon`. Die Query aus Beispiel 3.1 wird nur an die Peers weitergeleitet, die `swtont:SWDesign` (Software design) unterstützen, nicht aber an die mit Metadaten über `swtont:SWConstruct` (Software construction). Durch den hierarchischen Aufbau der SWEBOK [5]-Ontologie gehören auch die Provider mit `swtont:SWDesBasicConcepts` (Software design basic concepts) zu den Routingzielen.

Mit einem klassischen Datenbankindex [22] vergleichbar ist die Methode, Predicate und Inhalt selbst auszuwerten. Queries auf Basis eines solchen Indices werden nur an Provider verschickt, die Objekte mit genau den Elementen annotieren, die bei der Charakterisierung der Query gefunden wurden.

Enthält die Query eine Suche nach deutschsprachigen Objekten, dann wird die Query nur an Provider weitergeleitet, die Objekte mit `dc:title = "de"` beschreiben.

3.2 Super-Peer/Super-Peer-Indices

Die Super-Peer/Super-Peer-Indices bestimmen das Routing zwischen den Super-Peers, den Knoten des Netzwerk-"Backbones".

Analog zu Abbildung 3.5 zeigt die Grafik 3.6 die Komponenten eines Super-Peers. Der SP/SP-Index ist zur Erläuterung hervorgehoben. Im Gegensatz zu den immer auf aktuellem Stand gehaltenen und detaillierten SP/P-Indices enthält dieser Index nur die wichtigsten Informationen zum Auffinden einer Route zu einem Peer. Die Indices werden aus der Summe der Beschreibungen aller an einen Super-Peer angebotenen Provider und der Informationen benachbarter Super-Peers bestimmt. Mit jedem neu aufgenommenen oder entfernten Provider müssen alle SP/SP-Indices aktualisiert werden. Für das Versenden von Aktualisierungen der Indices an alle Super-Peers existiert mit dem HyperCup-Protokoll [27] ein effizienter Algorithmus (siehe 3.2.1).

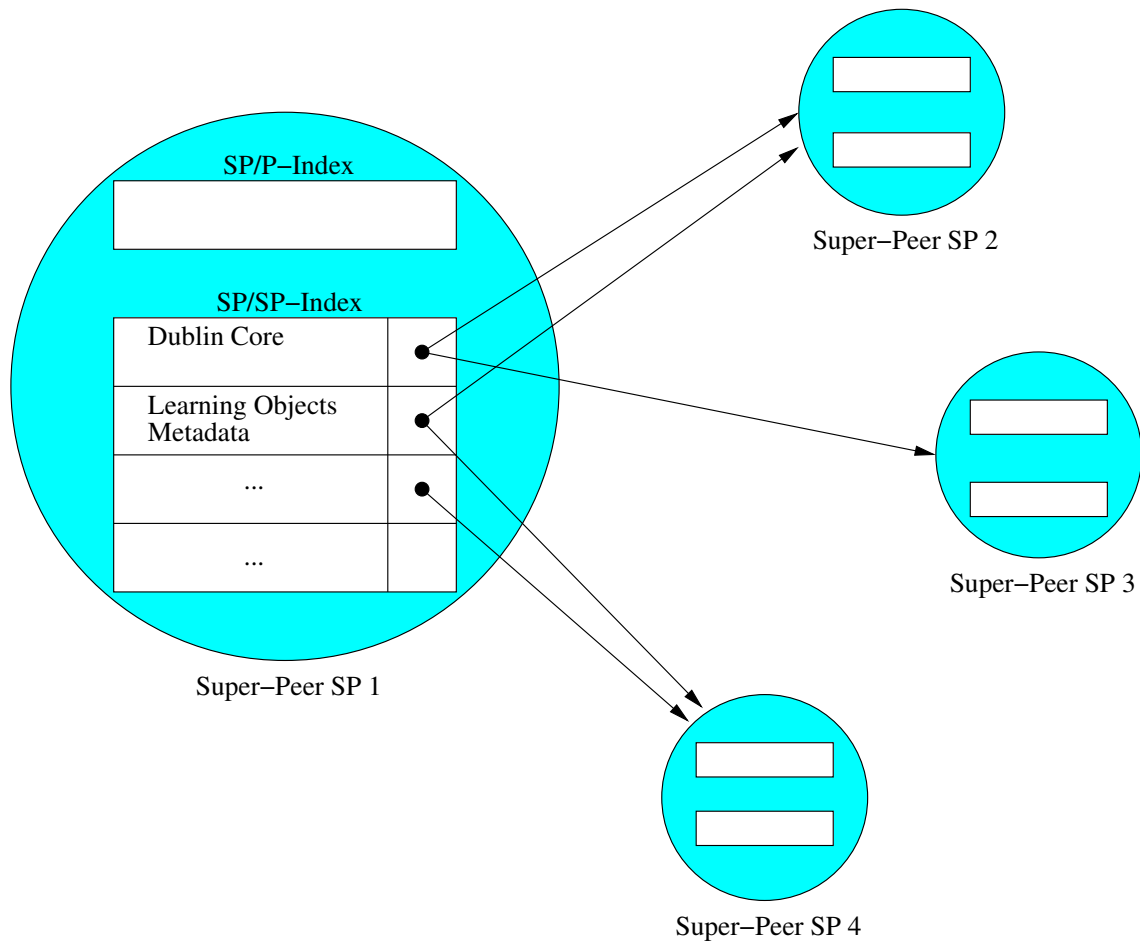


Abbildung 3.6: Super-Peer/Super-Peer-Index

3.2.1 Das HyperCup Protokoll

In [27] wird die auf Caley-Graphen basierende HyperCup-Topology für Peer-to-Peer Netzwerke beschrieben. Nachrichten können damit mit einer minimalen Anzahl an Schritten an alle Knoten verteilt werden. Durch das Verfahren wird sichergestellt, daß jeder Peer die Nachricht genau einmal erhält. Dazu werden die Knoten in einem *Hypercube* angeordnet. Abbildung 3.7 zeigt einen solchen Graphen für ein Netzwerk aus acht Peers. Ein solcher Hypercube ist symmetrisch, keiner der beteiligten Knoten nimmt eine besondere Stellung ein. Es ergibt sich eine automatische Lastverteilung zwischen den Peers ("Load-Balancing").

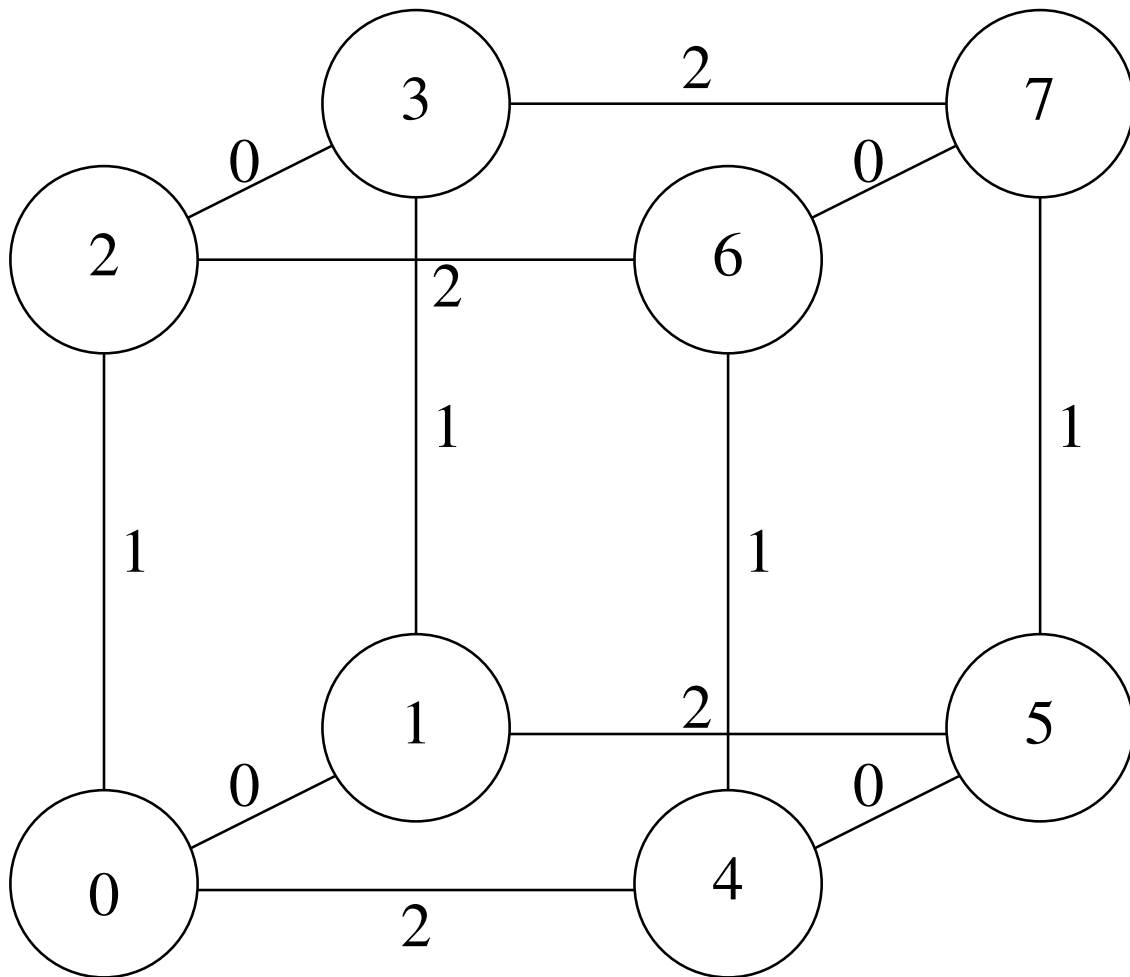


Abbildung 3.7: Hypercup-Topology

Die in Kapitel 4 beschriebene Software trennt strikt zwischen der Super-Peer- und der normalen Peer-Funktionalität. Für die Anbindung einer Super-Peer Topology existieren die notwendigen Schnittstellen durch ein entsprechendes Java-Interface 4.4. In der aktuellen Implementierung wird das HyperCup-Protokoll noch nicht unterstützt.

In einem Super-Peer-Netzwerk mit $N = 8$ Knoten, wie in Abbildung 3.7 dargestellt, kann eine Query mit 3 Schritten (Pfadlänge $\log_2 N$) an alle Peers versendet werden. Dabei werden die Nachrichten entlang der Kanten nach dem in Schema 3.8 dargestellten Verfahren versendet. Ein Peer versendet seine Nachricht nur an den Peer in der nächsthöheren Dimension, gekennzeichnet durch den Index an der jeweiligen Kante.

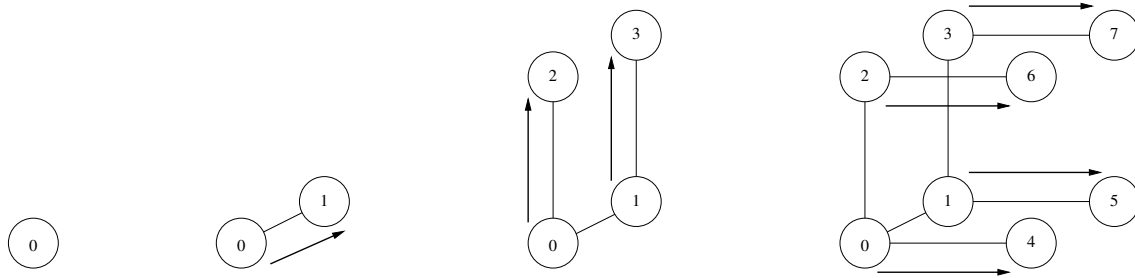


Abbildung 3.8: HyperCup: Ausbreitung der Nachrichten

3.2.2 Die Broadcasting-Topologie

Die in dieser Arbeit verwendete Super-Peer Topology ist eine einfache, flache Broadcast-Architektur.

Ein Super-Peer versendet seine Nachrichten an alle bekannten Super-Peers. Dies ist zwar nicht so effizient wie das HyperCup-Protokoll 3.2.1, aber einfach zu implementieren.

Abbildung 3.9 zeigt den Transport der Nachricht durch das Netz. SuperPeer P1 sendet seine Nachricht gleichzeitig an alle anderen Super-Peers. Das führt bei vielen Peers zu den in Kapitel 2.4 beschriebenen Nachteilen.

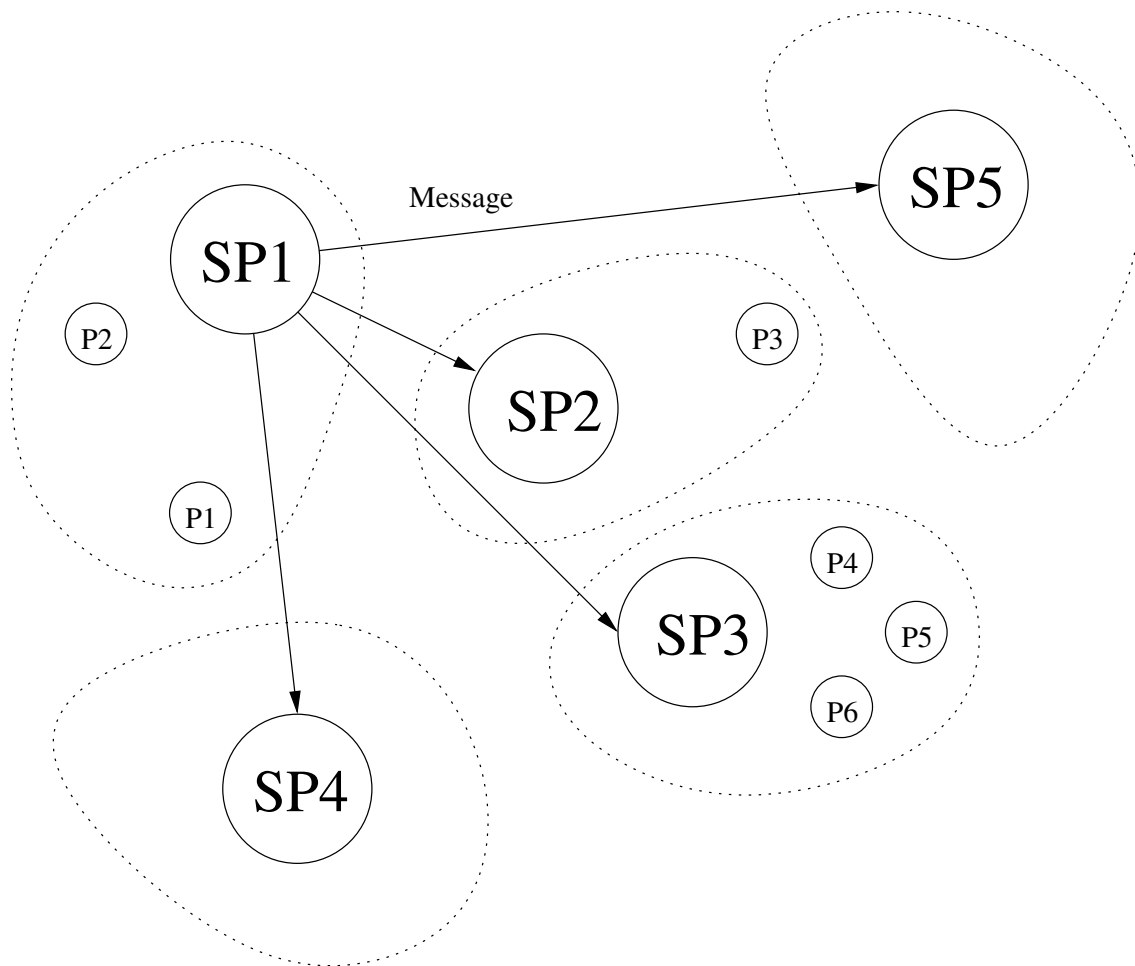


Abbildung 3.9: Messages in der Broadcast-Topology

3.3 Dynamische Routing-Indices

Die in den letzten Abschnitten vorgestellten Arten von Indices basieren auf Informationen, die von den Peers während der Anmeldung im Netz an die Super-Peers übermittelt werden. Es gibt Arten von Indices, die kontinuierlich an die sich ändernden Anforderungen angepasst werden.

Verfahren, wie in [19] beschrieben, dienen dazu, die am häufigsten verwendeten Elemente einer Query zu bestimmen, und diese als Basis für die Routing-Indices zu verwenden. Zu diesen Verfahren gehören beispielsweise *sticky sampling* oder *lossy counting*. Beim *sticky sampling* werden die Elemente in einem Datenstrom abgetastet und daraus ein Index aufgebaut. Dazu kann beispielsweise der Routing-Index dahingehend erweitert werden, daß auch die Häufigkeit des Auftretens eines Elementes gespeichert wird. *Lossy counting* ist ein Verfahren, das ähnlich dem *sticky sampling* arbeitet, aber effizienter bei der Ausnutzung der Ressourcen ist [19].

3.3.1 Clustering und Peer-Trading

Eine weiteres Verwendungsgebiet für Indices ist das bereits am Anfang diese Kapitels kurz angesprochene *Clustering*. Abbildung 3.3.1 zeigt ein Super-Peer Netzwerk mit einem Schema-basiertem Clustering. An die Super-Peers im Dublin-Core Cluster, bezeichnet mit SP1 und SP2, sind nur Provider-Peers mit Metadaten auf Basis des DC-Schemas angebunden. Die Provider an Super-Peer SP3 verwenden die RDF-Schemas DC und LOM, während Super-Peer SP4 nur Peers unterstützt, die LOM-Metadaten verwenden.

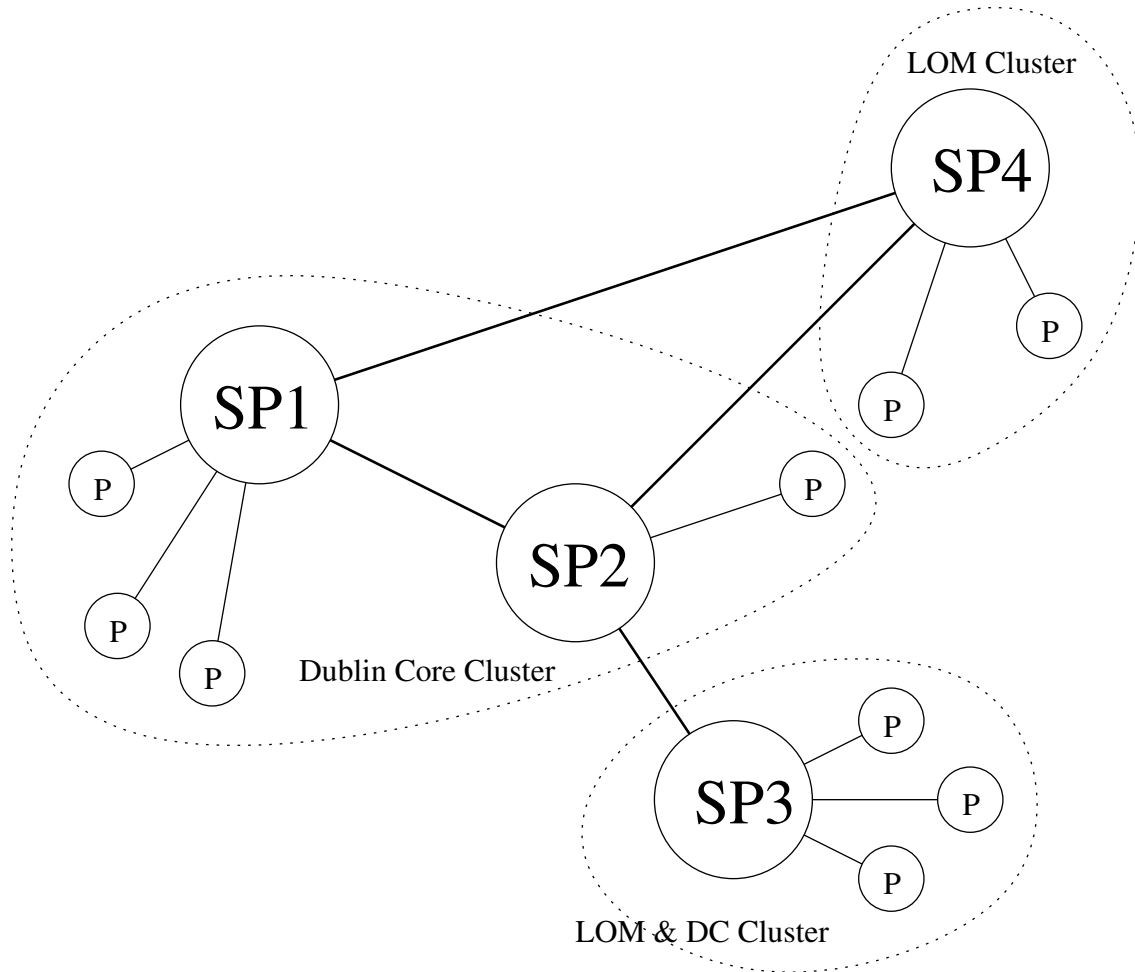


Abbildung 3.10: Schema-basiertes Clustering

Der SP/SP-Index des Super-Peers SP2 könnte die in Tabelle 3.2 gezeigte Form haben.

Dublin Core (DC)	SP1, SP3
Learning Objects Metadata (LOM)	SP3, SP4

Tabelle 3.2: Super-Peer/Super-Peer-Index im Super-Peer SP2 aus Bild 3.3.1

Eine Query, die den Super-Peer SP2 erreicht, und auf dem RDF-Schema Dublin Core beruht, wird nur an die Super-Peers SP1 und SP3 weitergeleitet.

Aus Gründen der besseren Lastverteilung und Organisation ist es möglich, das Super-Peer-Netzwerk auch in eine solche Struktur zu zwingen. Die Super-Peers können veranlaßt werden, nur noch Provider aufzunehmen, die bestimmte RDF-Schemas verwenden. Damit läßt sich eine Struktur wie in Abbildung 3.3.1 aufbauen. Peers mit Metadaten auf Basis von Dublic-Core werden von einem Super-Peer, der LOM-Provider verwaltet (SP4), abgewiesen. Auf Basis seiner SP/SP-Indices kann der Super-Peer den DC-Provider an einen anderen Super-Peer (SP1) verweisen. Stellt ein Super-Peer im laufenden Betrieb fest, daß er sich seiner Kapazitätsgrenze nähert, kann er einige seiner Peers an andere Super-Peers abgeben (*Peer-Trading*). Auch in diesem Fall müssen die Indices im gesamten Netz aktualisiert werden.

3.4 Mediation zwischen Schemas

Ein Super-Peer hat durch die Informationen, die der Provider bei seiner Anmeldung übermittelt, einen Überblick über die von ihm bereitgestellten Metadaten. Basis der Weiterleitung einer Query an einen Peer bilden die in Kapitel 3.1 beschriebenen Indices. Es existieren aber Möglichkeiten, eine Query nicht nur an einen exakt auf die entsprechende Query passenden Peer zu schicken, sondern die Lösung aus den Antworten mehrerer Peers zu kombinieren. Grundlage für die Routing-Indices aus Kapitel 3.1 sind beispielsweise die RDF-Schemas. Für die Umsetzung von Queries zwischen verschiedenen Schemas existieren mit *Query Correspondence Assertions (QCA)* [17] und *model correspondences (MoCA)* [8] flexible Algorithmen [22], die aus dem Umfeld der Mediator-basierten Informationssysteme (MBIS) stammen.

Um Mediation benutzen zu können, werden *Korrespondenzen* gebildet. Für Vorlesungsdokumente wäre folgendes Beispiel 3.2 denkbar.

Beispiel 3.2 Beispiele für Korrespondenzen zwischen RDF-Schemas

- | | | | |
|----|---------------------------|---|--------------------------------|
| | lectures:identfier | = | dc:title |
| 1. | lectures:language | = | dc:lang |
| | lectures:subject | = | dc:subject |
| | lectures:identfier | = | lom:general.identfier |
| 2. | lectures:language | = | lom:general.language |
| | lectures:subject | = | lom:educational.context |
-

Damit stehen Abbildungen aus dem abstrakten Schema `lectures` sowohl nach Dublin Core (`dc`), als auch nach Learning Objects Metadata (`lom`) zur Verfügung. Aus den Korrespondenzen aus Beispiel 3.2 lassen sich *Views* auf die in den Peers gespeicherten Metadaten generieren.

Beispiel 3.3 Beispiele für Views auf RDF-Schemas

1. **lecturesViewDC**(`lectures:identifizier`, `lectures:language`, `lectures:subject`)
 ← **DC**(`dc:title`, `dc:language`, `dc:subject`)
 2. **lecturesViewDC**(`lectures:identifizier`, `lectures:language`, `lectures:context`)
 ← **LOM**(`lom:general.identifizier`, `lom:general.language`, `lom:educational.context`)
-

Die Tabelle in Beispiel 3.4 beschreibt, welche Attribute des Super-Peer-basierten Schemas `lectures` von den angeschlossenen Peers verarbeitet werden kann.

Beispiel 3.4 Abbildung der Attribute aus dem `lecture`-Schema auf die RDF-Schemas `DC` und `LOM`

1. **lectures**(`lectures:identifizier`, `lectures:language`, `lectures:context`, -)
 ← **lecturesViewDC**(`dc:title`, `dc:language`, `dc:subject`)
 2. **lectures**(`lectures:identifizier`, `lectures:language`, -, `lectures:context`)
 ← **lecturesViewLOM**(`lectures:identifizier`, `lectures:language`, `lectures:context`)
-

Durch die Kombination der Korrespondenzen aus Beispiel 3.2, `refbsp:view`, 3.3 und 3.4 ergibt sich die Übersetzungsvorschrift wie in Beispiel 3.5 dargestellt. Peer1 verwendet das Dublin-Core (`dc`) Schema für seine Metadaten, Peer2 Learning Object Metadata (`lom`).

Beispiel 3.5 Resultierende Korrespondenzen für die Umsetzung auf die RDF-Schemas in den Peers P1 und P2

Peer1:Correspondence1

- lectures**(`lectures:identifizier`, `lectures:language`, `lectures:context`, -)
 ← **lecturesViewDC**(`dc:title`, `dc:language`, `dc:subject`)
 ← **DC**(`dc:title`, `dc:language`, `dc:subject`)

Peer2:Correspondence2

- lectures**(`lectures:identifizier`, `lectures:language`, -, `lectures:context`)
 ← **lecturesViewLOM**(`lectures:identifizier`, `lectures:language`, `lectures:context`)
 ← **LOM**(`lom:general.identifizier`, `lom:general.language`, `lom:educational.context`)
-

Kapitel 4

Implementierung

In diesem Kapitel werden die Erweiterungen der Edutella Infrastruktur beschrieben. Die Edutella-Peers aus [30] und [15] sind seit ihrer Entstehung auf unterschiedliche Art weiterentwickelt worden und verwenden nur einen kleinen Teil gemeinsamen Codes. Die existierenden Peers, beispielsweise der Provider-Peer, wurden speziell für ihre zentrale Aufgabe entworfen, das Beantworten von Queries. Ziel war die Erarbeitung einer Architektur, die als Unterbau für die Peers und Super-Peers verwendet werden kann. Abbildung 4.1 stellt den alten Edutella-Provider den Komponenten der neuen Architektur gegenüber. Der als *Database Backend* gekennzeichnete Block stellt die vom Provider übernommene Anbindung des Repositories für Metadaten dar. Aus dem *Edutella Query Service* wurde einer der Dienste (*Peer Services*), hier mit "QEL" bezeichnet, aus denen die neuen Peers zusammengesetzt sind. Am *Edutella Common Data and Exchange Model* (ECDM) und dem Format der im Edutella-Netz verwendeten Nachrichten wurden keine Änderungen vorgenommen. Das System ist weiterhin kompatibel mit den alten Peers. Anzumerken ist, daß die Consumer einen Super-Peer nicht als solchen erkennen können.

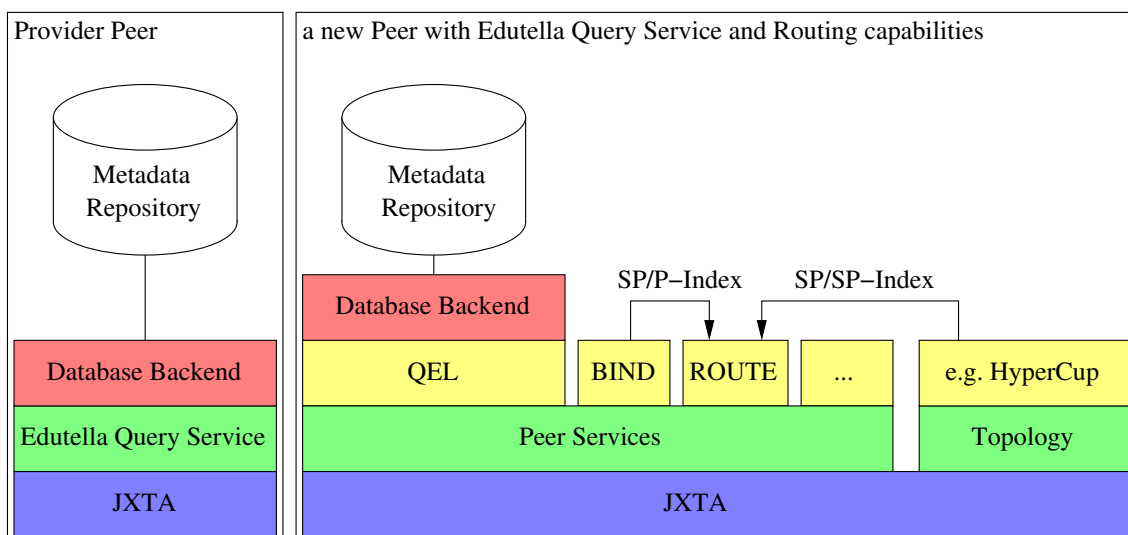


Abbildung 4.1: Die alte und neue Architektur im Überblick

Die in dieser Arbeit entstandene Implementierung erweitert die Edutella-Infrastruktur mit neuen Elementen. Zentrale Komponente der neuen Peers ist das Singleton `PeerServiceRegistry`. Durch das Singleton wird sichergestellt, daß nur eine Instanz existiert und ein einfacher globaler Zugriff möglich ist [12].

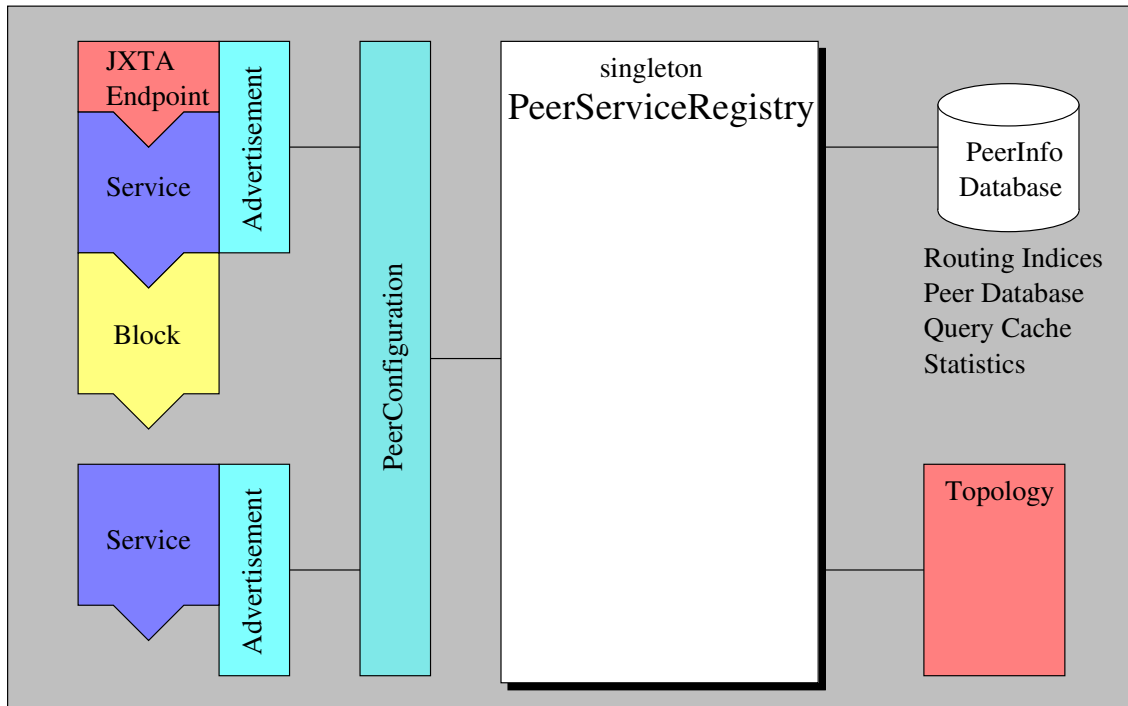


Abbildung 4.2: Überblick über die Super-Peer Architektur

Generische Bausteine und dazugehörigen Konfigurationsobjekte dienen zum Aufbau komplexer Peers. Die Elemente des Peers kommunizieren durch Event-Objekte miteinander. Ähnlich den Event-Interfaces der grafischen Benutzeroberfläche von Java (Swing) werden die Event-Objekte durch eine Methode des `EventListener`-Interfaces übergeben. Abbildung 4.2 zeigt einen Überblick über die Hauptklassen der neuen Architektur. Die Aufbau der Services werden durch die `PeerConfiguration`-Klasse gesteuert. Die vom JXTA-System empfangenen Nachrichten werden an die `Service`-Klasse übermittelt. Die Verkettung der Services wird durch die `PeerConfiguration` festgelegt, kann aber auch durch den Service selbst erfolgen (siehe Kapitel 4.3).

Zu jedem Peer gehört neben der zentralen Singleton-Klasse `PeerServiceRegistry` die `PeerInfo`-Datenbank für die in Kapitel 3 beschriebenen Indices.

Die `Topology`-Klasse repräsentiert die Struktur zwischen den Peers. Wie in Abbildung 4.1 dargestellt, ist die `Topology` vom eigentlichen Peer getrennt. Auf diese Weise können unterschiedliche Topologien eingebunden werden. Diese Trennung ist die Voraussetzung für die Anbindung an die separat entwickelte HyperCup-Architektur [27].

Die Peers werden über das `PeerConfiguration`-Objekt konfiguriert. Der Aufbau der Konfigurationsdatei wird in Kapitel 4.3 erläutert. Der folgende Abschnitt 4.1

befaßt sich mit den aus dem Provider-Peer nachgebildeten und den neu implementierten Diensten. Kapitel 4.6 beschreibt die Dienste zum Aufbau und zur Verwaltung der für das Routing benötigten Indices. Das Routing wird in Kapitel 4.7 behandelt.

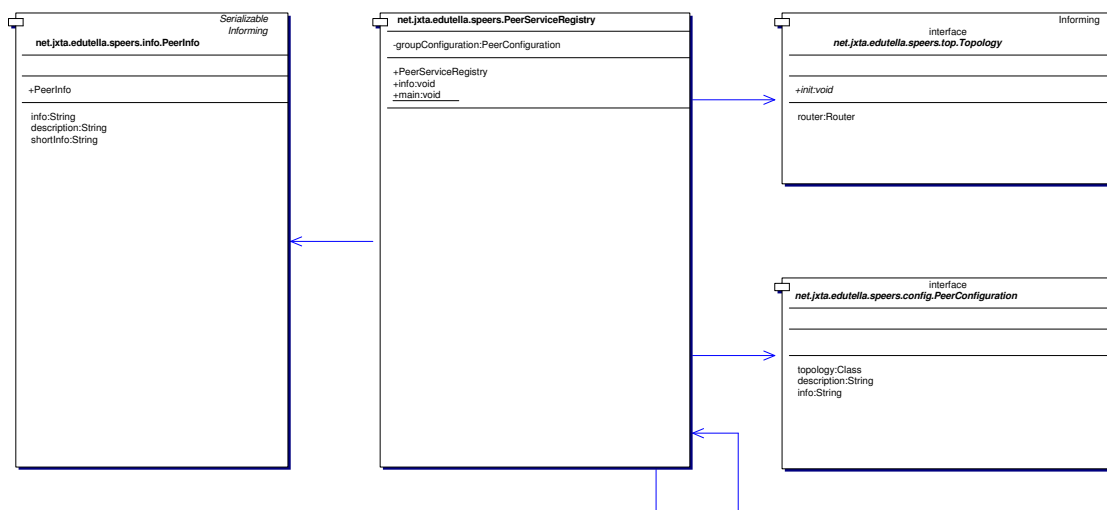


Abbildung 4.3: UML Klassendiagramm: Die Basisklassen eines Peers

4.1 Aufgaben des Peers: Dienste

Klasse: `net.jxta.edutella.peers.pool.Service`

Jeder Service eines Peers wird über eine kurze Zeichenkette identifiziert. Zu den bisher implementierten Service-Typen gehören “**QEL**”, “**MEL**”, “**BIND**”, “**ROUTE**” und “**CONSOLE**”. Eine Übersicht aller Dienste befindet sich in Tabelle A.1 im Anhang.

QEL ist der *Edutella Query Service* der Edutella-Provider, MEL der Provider-Service für Modifikationen an RDF-Repositories [20]. Der QEL-Service wird in 4.7.1 beschrieben.

Zu den neu hinzugekommenen Diensten gehören BIND, für die Anmeldung der Provider-Peers, ROUTE für das Routen von Queries zwischen Super-Peers und CONSOLE, eine Schnittstelle zur Administratoren über einen Telnet-Client [25]. Der ROUTE-Service wird in Abschnitt 4.7.3, die System-Management-Console in 4.8 erläutert.

Dienste sind als Finite-State-Machine realisiert. Der Java-Thread des Services wird mit der `wait()`-Methode geblockt und bei Empfang eines Events fortgesetzt (`notify()`).

Services, die über die JXTA-Methoden kommunizieren, haben eine assoziierte `ServiceAdvertisement`-Klasse. Das Advertisement enthält Deklarationen und Informationen zum Aufbau der entsprechenden Verbindungen. Peers, die Dienste

im Netz anbieten, verwenden das Advertisement um ihre Pipes eindeutig zu benennen. Ein Peer auf der Suche nach einem Service benutzt das Advertisement um die Services der anderen Peers schneller zu finden.

Die QELAdvertisement-Klasse mit den Informationen über den Edutella Query Service ist aus der EduConstant-Klasse des alten Edutella-Systems entstanden. Zur einfacheren Identifizierung ist die Servicekennung dem Klassennamen Advertisement vorangestellt.

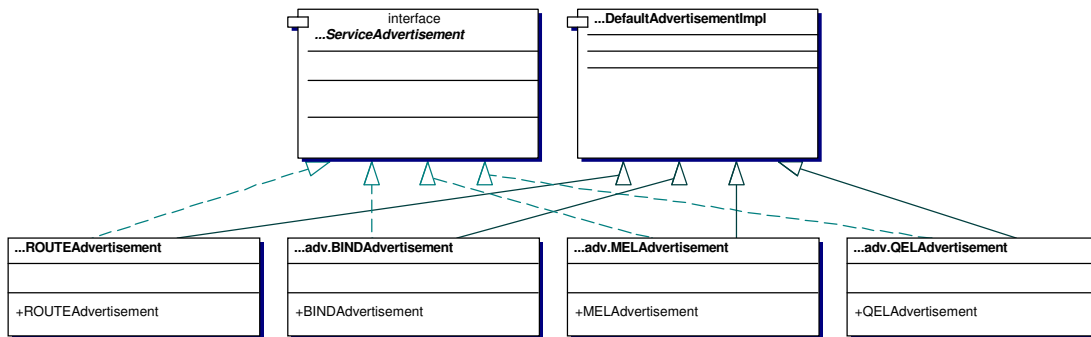


Abbildung 4.4: UML Klassendiagramm: net.jxta.edutella.peers.adv

Eine Ausnahme bildet die SuperPeerAdvertisement-Klasse. Diese enthält die Definitionen für den Aufbau der Broadcasting-Topology des Super-Peer Netzwerks.

4.2 Initialisierung des Peers

Die main()-Methode befindet sich in der Klasse PeerServiceRegistry. Diese initialisiert das PeerServiceRegistry-Singleton. Im nächsten Schritt wird die beim Start als Argument übergebene PeerConfiguration instanziiert und zum Aufbau der Dienste verwendet.

Abbildung 4.5 zeigt die einzelnen Schritte der Initialisierung.

Einzelne Elemente der PeerServiceRegistry werden beim ersten Zugriff darauf initialisiert. Dazu gehören die PeerInfo und die NetPeerGroup für die JXTA-Anmeldung.

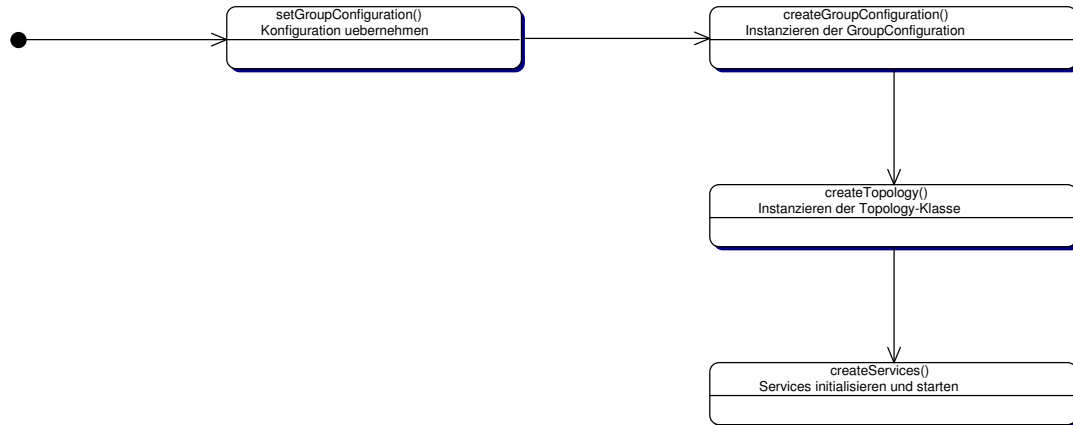


Abbildung 4.5: UML Zustandsdiagramm: Hochfahren eines Peers

4.3 Peerkonfiguration

Interface `net.jxta.edutella.speers.config.PeerConfiguration`

Über die Konfigurationsobjekte des Typs `PeerConfiguration` werden aus den einzelnen Serviceklassen die Funktionen des Peers aufgebaut. Die für den Aufbau des Netzwerks benötigten Super-Peers werden durch die `SuperPeer`-Klasse, die Provider-Peers durch die `ProviderPeer`-Klasse definiert. Die Consumer-Peers können durch die Emulation des *Edutella Query Service* ohne Änderungen weiterverwendet werden (siehe Abschnitt 4.7.2).

Die von einem Peer verwendeten Services werden als Zeichenkette im Array `services` hinterlegt. Für jeden dieser Services kann im `advertisements`-Array ein `ServiceAdvertisement` angegeben werden. In `endpoints` wird die Klasse festgelegt, die Events und Messages für den jeweiligen Service empfängt. Wie in Abbildung 4.2 zu sehen, kann ein Service aus der Verkettung mehrerer Blöcke bestehen. Zusätzliche Klassen können in `childs` definiert und über Einträge in `chains` bei den anderen Klassen registriert werden. Im nächsten Abschnitt wird das Verfahren am Beispiel des Provider-Peers erklärt.

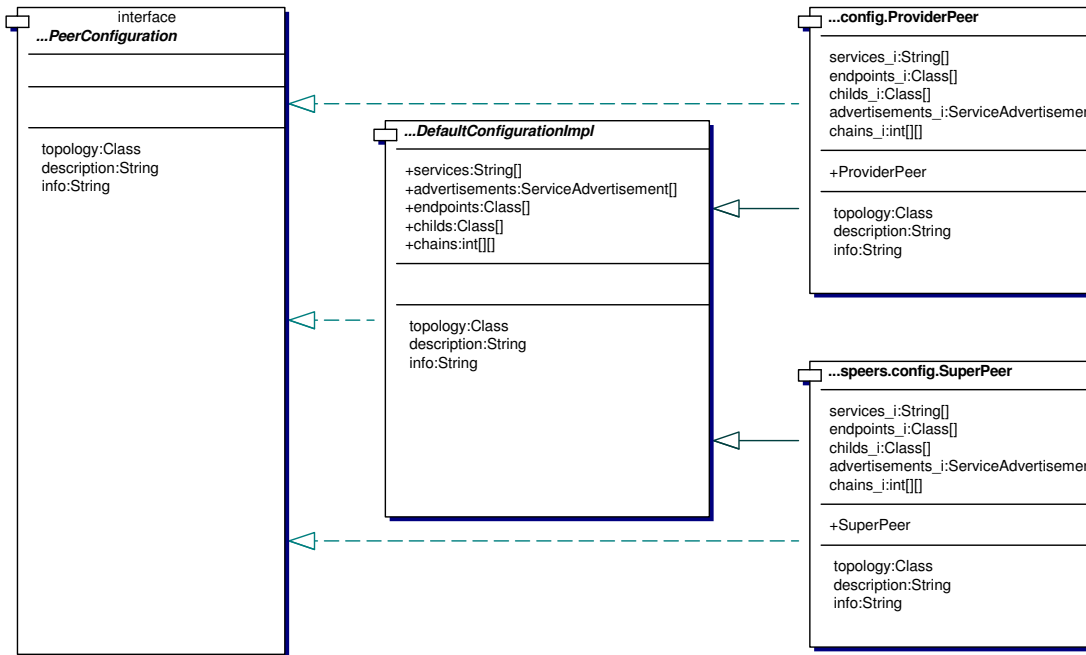


Abbildung 4.6: UML Klassendiagramm: PeerConfiguration

4.3.1 Konfiguration des Provider-Peers

Klasse: `net.jxta.edutella.speers.config.ProviderPeer`

Das Codebeispiel 4.1 zeigt die Elemente aus der `ProviderPeer`-Klasse, die zum Aufbau der Provider-Peers verwendet werden.

Beispiel 4.1 Ausschnitt aus `ProviderPeer.java`

```

1  public class ProviderPeer extends DefaultGroupImpl
    implements PeerConfiguration {

    String[] services_i = { "BIND", "QEL", "CONSOLE", "RDV" };
    Class[] endpoints_i = { ClientUplink.class, ProviderService.class,
                          SMConsole.class, RendezvousConnector.class };

    Class[] childs_i = { ProviderPool.class };

10  ServiceAdvertisement[] advertisements_i = { new BINDAdvertisement(),
                                             new QELAdvertisement() };

    int[][] chains_i = { {0, -1}, {1, 0}, {2, -1}, {3, -1} };

    public ProviderPeer() {

        services = services_i;
        endpoints = endpoints_i;
        childs = childs_i;
20  advertisements = advertisements_i;
        chains = chains_i;
    }

    public Class getTopology() {
        return(ClientPeer.class);
    }

    [...]
30 }
  
```

Die vom Provider-Peer verwendeten Dienste werden im Array in Zeile 8 definiert. **BIND** wird für das Anmelden des Providers bei einem Super-Peer benötigt, **QEL** ist der Dienst für das Empfangen und Bearbeiten von Queries. Über den **CONSOLE**-Dienst können Kommandos an den Provider gesendet werden. **RDV** ist ein Hilfsdienst zum Anmelden im JXTA-System.

Die Variable `services` in Zeile 14 definiert die Advertisement-Klassen für die Dienste BIND und QEL. Die Zuordnung ist durch die Position innerhalb des Arrays festgelegt.

Mit dem Array `endpoints` in Zeile 9 werden die Klassen definiert, die Events und Messages für den zugehörigen Service empfangen. Messages stammen aus dem JXTA-System und werden über eine Pipe empfangen, Events können von anderen Diensten des Peers stammen.

Über `childs` in Zeile 12 werden zusätzliche Klassen definiert, die von den Services verwendet werden können. Die Einträge im `chains`-Array verketteten die Endpoint-Klasse mit der Child-Klasse. Der zweite Eintrag $(1, 0)$ verkettet die QEL-Klasse `ProviderService` mit dem `ProviderPool`. Der `ProviderService` empfängt die Queries aus dem Netz, der `ProviderPool` bearbeitet die Queries. Die Queries werden über das Event-Interface weitergeleitet.

Abbildung 4.7 zeigt den schematischen Aufbau des Provider-Peers. In der `PeerInfo`-Datenbank werden nur die vom Provider entdeckten Super-Peers gespeichert. Das Metadaten-Repository ist davon getrennt. Das letzte Glied in der QEL-Servicekette ist der `RequestProcessor`, der zur Anbindung der Datenbank die `ProviderConnection`-Klasse des Original-Provider-Peers verwendet.

Der `RequestProcessor` und die `SMConsoleConnection` wird nicht durch die `PeerConfiguration` gestartet, sondern von den Services selbst.

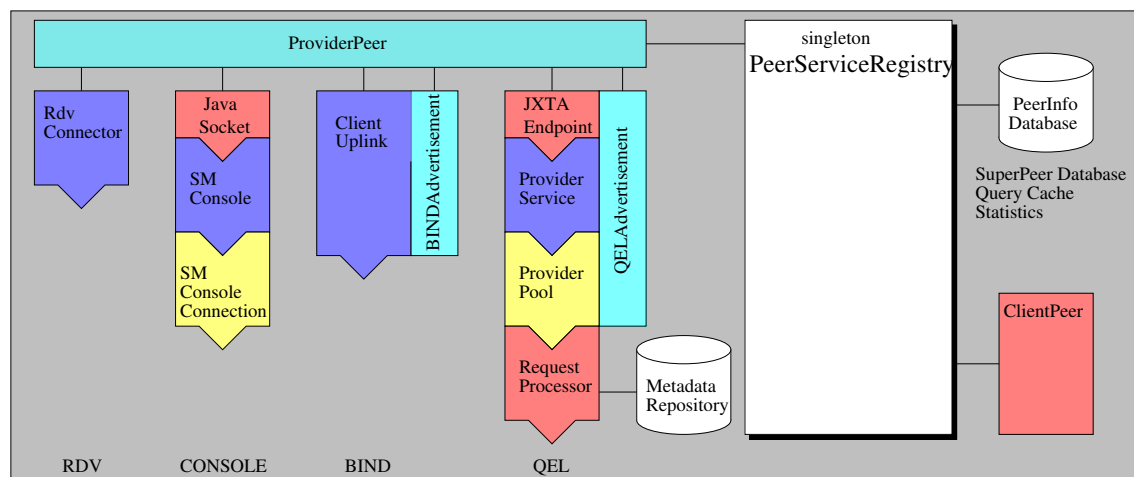


Abbildung 4.7: Provider-Peer-Konfiguration

Die Dienste des Provider-Peers werden im Zusammenhang mit den Komponenten des Super-Peers in Abschnitt 4.5 beschrieben.

4.3.2 Konfiguration des Super-Peers

Klasse: `net.jxta.edutella.speers.config.SuperPeer`

Für den Super-Peer wird eine größere Zahl von Diensten benötigt. In der Klasse `SuperPeerGroup` wird analog zur Klasse `ProviderPeer` die Konfiguration des Super-Peers definiert. Abbildung 4.8 zeigt den Aufbau eines Super-Peers. Die aus dem Java-Quellcode übernommenen Array-Definition sind in Codebeispiel 4.2 abgebildet.

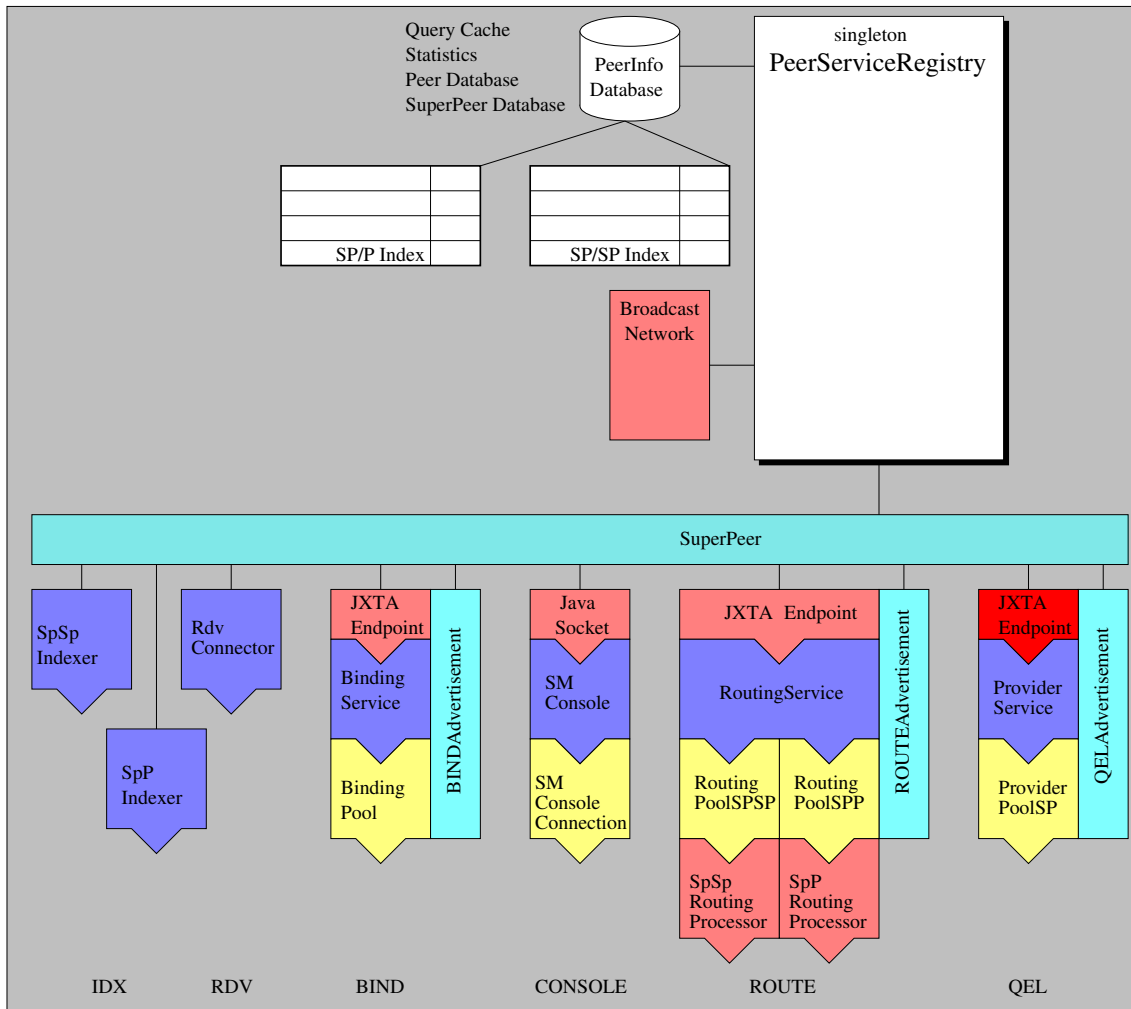


Abbildung 4.8: Super-Peer-Konfiguration

Beispiel 4.2 Ausschnitt aus SuperPeerGroup.java

```

1  String[] services_i = { "BIND", "ROUTE", "CONSOLE", "RDV", "QEL", "UTIL" };
   Class[] endpoints_i = { BindingService.class,
                          RoutingService.class,
                          SMConsole.class,
                          RendezvousConnector.class,
                          ProviderService.class,
                          SpSpIndexer.class };

10  Class[] childs_i = { BindingPool.class,
                       RoutingPoolSPP.class,
                       RoutingPoolSPSP.class,
                       ProviderPoolSP.class };

   SPAdvertisement[] advertisements_i = { new BINDAdvertisement(),
                                         new ROUTEAdvertisement(),
                                         null,
                                         null,
                                         new QELAdvertisement(),
                                         null };

20  int[][] chains_i = { {0, 0}, {1, 1}, {1, 2}, {2, -1}, {3, -1},
                       {4, 3}, {5, -1} };
   }

```

In der Abbildung 4.9 ist die Verkettung der Klassen graphisch dargestellt. Die Dienste **“CONSOLE”**, **“RDV”** und **“UTIL”** verwenden keine JXTA-Dienste und benötigen daher keine ServiceAdvertisements. Die einzelnen Komponenten eines Super-Peers werden im Detail in Abschnitt 4.5 beschrieben.

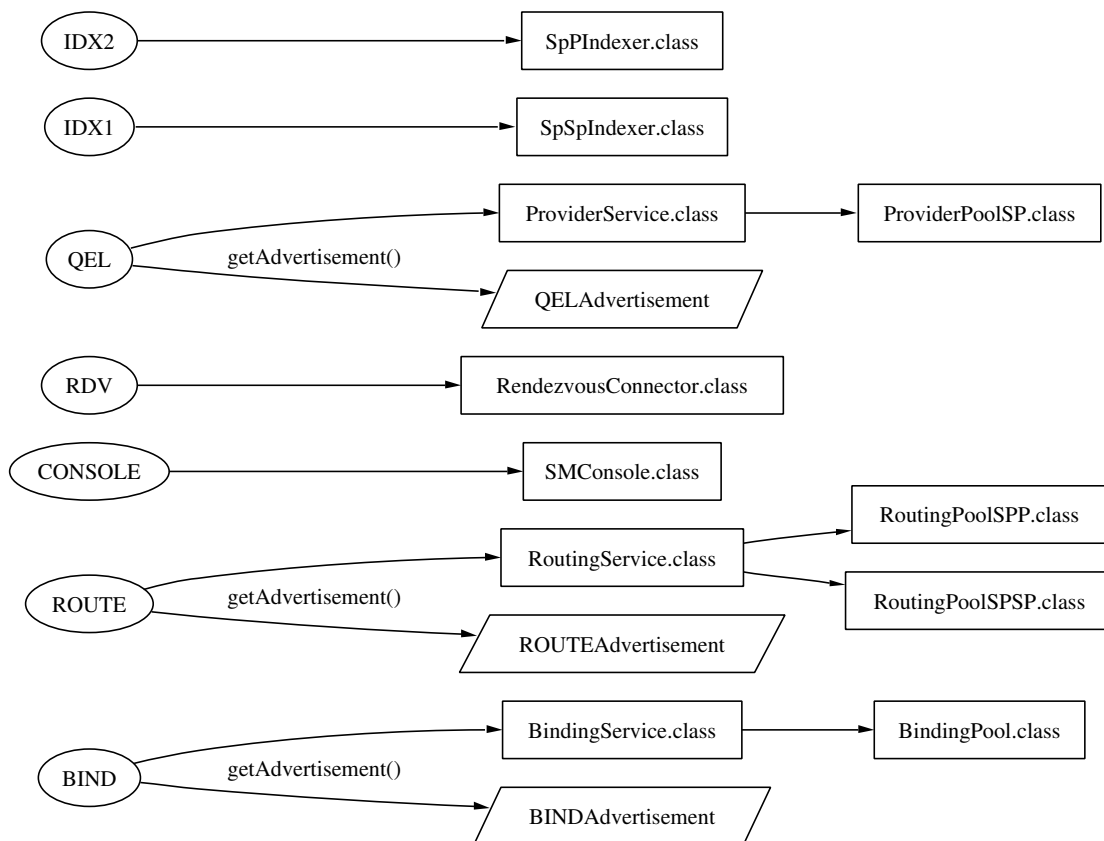


Abbildung 4.9: Super-Peer-Konfiguration

4.4 Topology

Interface: `net.jxta.edutella.speers.top.Topology`

implementierende Klassen: `net.jxta.edutella.speers.top.ClientPeer`

Klasse für einfache Peers (Provider)

`net.jxta.edutella.broadcast.BroadcastNetwork]`

Topology-Klasse für Super-Peers, mit einem einfachen Algorithmus zur Verteilung von Nachrichten.

In jeder `PeerConfiguration` ist festgelegt, wie ein Peer in das Netz eingebunden wird. Für den Provider-Peers wird die `ClientPeer`-Klasse verwendet. Die Provider organisieren sich nicht selbst zu einem strukturierten Netz. Aus diesem Grund enthält die `ClientPeer`-Klasse keine Funktionalität. Die Anbindung in das Netz wird vom BIND-Service durchgeführt.

Die Super-Peers bilden ein strukturiertes Netz, beispielsweise einen Hypercube, wie in [27] und Kapitel 3.2.1 beschrieben. In dieser Arbeit wird eine einfache, flache Architektur verwendet (siehe Kapitel 3.2.2). Die Algorithmen zum Aufbau des Netzes sind in der Klasse `BroadcastNetwork` implementiert.

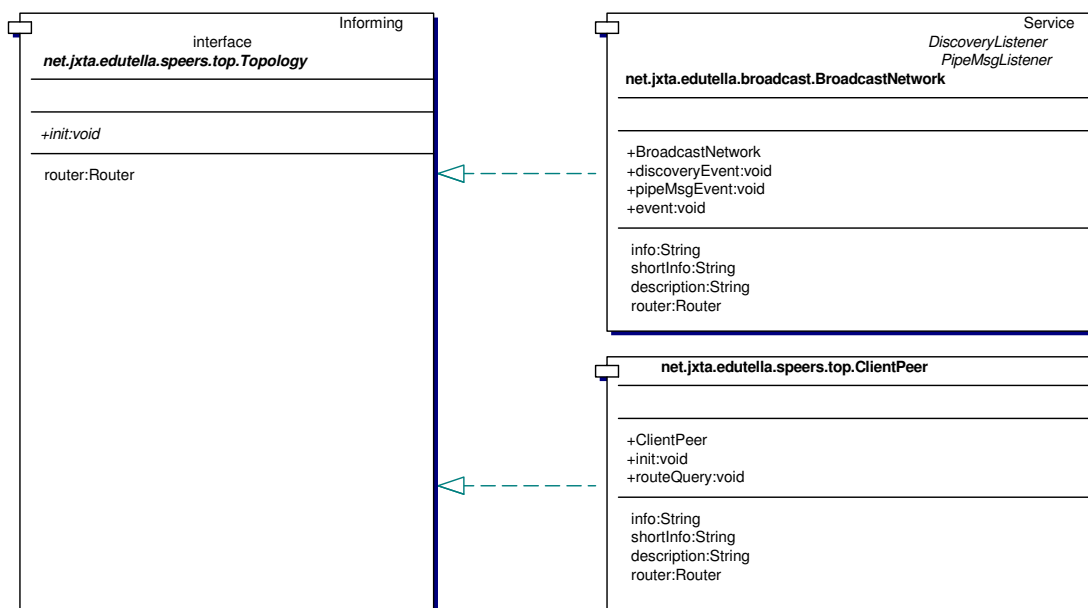


Abbildung 4.10: UML Klassendiagramm: Topology

Zur Broadcast-Topology gehört die Advertisement-Klasse `SuperPeerAdvertisement`.

4.5 Super-Peer-Konfiguration

PeerConfiguration: SuperPeer

Topology: BroadcastNetwork

Services: BIND, CONSOLE, ROUTE, QEL, RDV, IDX

Wie in Abbildung 4.8 gezeigt, gehören in einem Super-Peer die drei Dienste "BIND", "ROUTE" und die Provider-Emulation "QEL" zu den entscheidenden Elementen. Die Topology-Klasse steuert die Kommunikation der Super-Peers untereinander.

4.5.1 Aufbau der Super-Peer-Topology

Die in dieser Arbeit implementierte Topology zur Organisation des Super-Peer-Netzwerkes ist die eines unstrukturierten Peer-to-Peer Netzwerkes. Wie in Abbildung 4.11 abgebildet hat jeder Super-Peer eine Verbindung zu allen anderen Super-Peers. Für die Kommunikation zwischen den Super-Peers wird die Definition aus der SuperPeerAdvertisement verwendet. Jeder Super-Peer erzeugt eine JXTA-Pipe zur Kommunikation und annonciert die Adresse über die JXTA-Publishing-Services (siehe Codebeispiel 4.3).

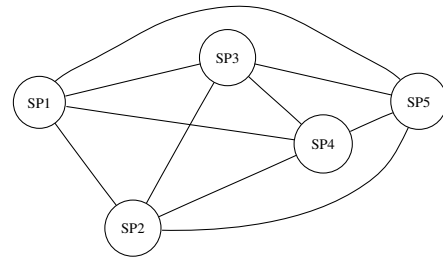


Abbildung 4.11: Topology

Beispiel 4.3 Ausschnitt aus BroadcastNetwork.java

```
[...]
adv = new SuperPeerAdvertisement();

pipeAdvertisement = adv.getPipeAdvertisement();
peerGroup = adv.getPeerGroup();

pipeService = peerGroup.getPipeService();
inpipe = pipeService.createInputPipe(pipeAdvertisement, this);
Publisher.publishService(adv);

[...]
```

Die einzelnen Zustände der BroadcastNetwork-Klasse sind in Abbildung 4.12 dargestellt.

Mit der Publisher-Klasse wird das PipeAdvertisements im Netz veröffentlicht. Andere Super-Peers können es über das JXTA-System suchen und zum Aufbau der SP/SP-Indices verwenden (siehe nächsten Abschnitt).

Beispiel 4.4 Advertisement eines Super-Peers: ModuleImplAdvertisement

```

1  <?xml version="1.0"?>
    <!DOCTYPE jxta:MIA>
    <jxta:MIA xmlns:jxta="http://jxta.org">
        <MSID>
            urn:jxta:uuid-3DF03B4A7C344041AFB38988CBDEE79F0193AF331B7F4FEAA0F68DF7595810FF06
        </MSID>
        <Code>
            class net.jxta.edutella.broadcast.BroadcastNetwork
        </Code>
        <PURI>
            http://www.learninglab.de/~brunkhor/
        </PURI>
        <Prov>
            Edutella Project
        </Prov>
        <Desc>
            This is the Module Implementation for the SuperPeer Broadcast Network Topology
        </Desc>
        <Parm>
            <SuperPeerPipeID>
                urn:jxta:uuid-02C5023ECBBB4053B1CD9FF44A4D77ABED3716114BD8449CB7FD91C6CCA05E3E04
            </SuperPeerPipeID>
            <SuperPeerID>
                urn:jxta:uuid-59616261646162614A78746150325033BDE738BE20C64C1CA30F9A4079BBA04703
            </SuperPeerID>
            <SuperPeerName>
                peer@damokles
            </SuperPeerName>
            <SuperPeerHostname>
                damokles.sonnenstein.org
            </SuperPeerHostname>
        </Parm>
    </jxta:MIA>

```

Die `getRemoteAdvertisement()`-Methode sucht alle Advertisements ((ADV), Zeile 4). Beschränkt wird die Suche auf Advertisements, deren `ModuleSpecID` (MSID) mit der MSID des `SuperPeerAdvertisements` übereinstimmt.

Die MSID steht in den Zeilen 6-8 des `ModuleImplAdvertisements` (siehe Beispiel 4.4 und in den Zeilen 15-18 der Klasse `SuperPeerAdvertisement` (siehe Beispiel 4.6)).

Im `Parm`-Block des gefundenen `ModuleImplAdvertisements` (Beispiel 4.4, Zeilen 21-34), sind zusätzliche Informationen untergebracht. Diese Daten werden nicht vom JXTA-System ausgewertet und müssen durch die Super-Peer-Software analysiert werden. Aus der `SuperPeerPipeID` und der `SuperPeerID` wird die Adresse der Pipe des gefundenen Super-Peers ermittelt. Zusätzlich ist der Name und der FQDN (Full Qualified Domain Name) des Super-Peers kodiert.

4.5.3 Senden von Informationen: HELO-Message

Die zweite Quelle von Informationen sind Nachrichten, die ein Super-Peer an alle neu entdeckten Super-Peers sendet. Neben den Adressen für die Services (QEL, ROUTE) des Super-Peers werden damit Informationen zum Aufbau der SP/SP-Indices übermittelt.

Jeder Super-Peer erzeugt eine Beschreibung über die an ihn angebotenen Provider und die verwendeten Routing-Indices. Repräsentiert werden die Informationen durch die `SuperPeerDescription`-Klasse. Zur Darstellung der Informationen wird das Resource Description Format (RDF) verwendet. Das dazu neu definierte RDF-Schema befindet sich im Anhang A.4. Über die Properties des `PeerDe-`

Beispiel 4.6 SuperPeerAdvertisement.java

```
1 package net.jxta.edutella.broadcast.adv;

    [...]

    midescription =
        "This is the Module Implementation for the SuperPeer " +
        "Broadcast Network Topology";
    miclass =
        net.jxta.edutella.broadcast.BroadcastNetwork.class.toString();
10 miprovider =
    "Edutella Project";
    miurl =
        "http://www.learninglab.de/~brunkhor/";

    msname =
        "SuperPeer Broadcast Network Topology";
    msid = "urn:jxta:uuid-" +
        "3DF03B4A7C344041AFB38988CBDEE79F0" +
        "193AF331B7F4FEAA0F68DF7595810FF06";
20

    msdescription =
        "This is the Module Spec Description for the " +
        "SuperPeer Broadcast Network";
    mscreator =
        "Edutella Project";
    mcid =
        "urn:jxta:uuid-3DF03B4A7C344041AFB38988CBDEE79F05";

    mcname =
30     "Edutella SuperPeer Broadcast Network";
    mcdescription =
        "This ist the Module Class Description for the " +
        "SuperPeer Broadcast Network";

    gid =
        "urn:jxta:uuid-02C5023ECBBB4053B1CD9FF44A4D77AB02";
    gname =
        "Edutella SuperPeer Broadcast Network Group";
    gdescription =
40     "This is the PeerGroup for the Broadcast Network SuperPeers";

    type =
        "SuperPeer";
}

    [...]
```

scription (PD) Schemas können Informationen über die Fähigkeiten und Eigenschaften eines Peers in einem RDF-Datenmodell abgebildet werden. In der aktuellen Implementierung der Broadcast-Network-Topology werden keine SP/SP-Indices verwendet, daher werden in der SuperPeerDescription gegenwärtig keine Metadaten-Informationen verwaltet. Mit der SuperPeerDescription und den Informationen des ModuleImplAdvertisements aus Abschnitt 4.5.2 wird ein `Msg`-Objekt vom Typ `SuperPeerHEL0` erzeugt. Diese wird nach dem Broadcast-Verfahren an den neu gefundenen Super-Peer versendet.

Beispiel 4.7 Ausschnitt aus `BroadcastNetwork.java`

```
[...]

SuperPeerHEL0 sph = new SuperPeerHEL0();
sph.setResponsePipeAdvertisement(pipeAdvertisement);
sph.setResponsePeerID(peerGroup.getPeerID());
sph.setResponsePeerGroupID(peerGroup.getPeerGroupID());

[...]

Message message = sph.toMessage();
Vector peers = new Vector(1);
peers.addElement(peerGroup.getPeerID());
PipeAdvertisement padv = spd.getPipeAdvertisement();

try {
    PeerGroupID pgid = spd.getPeerGroupID();
    PeerGroup peerGroup = SuperPeer.SUPERPEER.getPeerGroup(pgid);
    PipeService pipeSvc = peerGroup.getPipeService();
    OutputPipe outputPipe = pipeSvc.createOutputPipe(padv,
                                                    peers.elements(),
                                                    EduConstant.PIPE_CREATION_TIMEOUT);

    outputPipe.send(message);
    outputPipe.close();

} catch (IOException ioe) {

    [...]
```

Die Variable `spd` ist die aus dem `ModuleImplAdvertisement` generierte `SuperPeerDescription` des Ziel-Super-Peers. Alle Beschreibungen werden lokal in einer Datenbank abgelegt. Wird die `SuperPeerHEL0`-Message eines anderen Super-Peers empfangen, werden die Einträge in der Datenbank aktualisiert. Über die Template-Methode [12] der Klasse `Msg` kann das `SuperPeerHEL0`-Objekt und das `SuperPeerDescription`-Objekt nach der Übermittlung zurückgewonnen werden.

Für jede über die Pipe empfangene Nachricht wird vom `BroadcastNetwork` ein `Event` generiert und an die `EventListener` in der `PeerServiceRegistry` gesendet.

Ausgewertet wird das Event von den Klassen des IDX-Services, SpSpIndexer und SpPIndexer.

Erzeugen des SP/SP-Routing-Indices: Die SpSpIndexer-Klasse

Der SpSpIndexer-Service erzeugt die für das Routing von Queries zwischen den Super-Peers erforderlichen SP/SP-Indices. Nach jeder empfangenen SuperPeerHELLO-Message wird der Routing-Index aktualisiert. In der aktuellen Implementierung wird nur die SuperPeerDescription gespeichert. Diese Informationen werden in einem Hashtable in der zum Peer gehörenden PeerInfo abgelegt. Die Aktualisierung der Tabelle erfolgt *asynchron*, eingehende Events werden in einer Warteschlange eingetragen und von einem Thread im Hintergrund abgearbeitet.

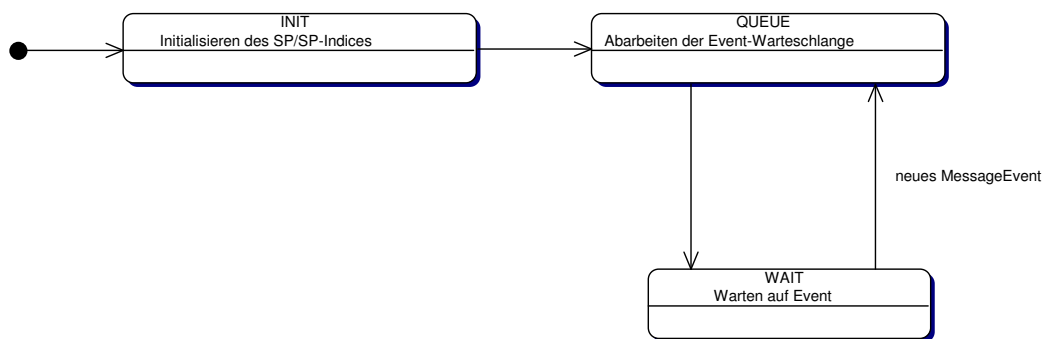


Abbildung 4.13: UML Zustandsdiagramm: Zustände des SpSpIndexer-Dienstes

Die XML-Serialisierung in Beispiel 4.8 zeigt die SuperPeerDescription eines Superpeers, die aus einer SuperPeerHELLO-Message extrahiert wurde.

Beispiel 4.8 Beschreibung eines SuperPeers: SuperPeerDescription (gekürzt)

```

1 <rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:NS0='http://www.learninglab.de/~brunkhor/rdf/peerDesc#'
  >
  <rdf:Description rdf:about=''>
    <NS0:routerPeerGroupID>urn:jxta:uuid-73AB3A3FD53E48248FE08133CD3E795602</NS0:routerPeerGroupID>
    <NS0:routerPeerID>urn:jxta:uuid-596162616461...AC0E756A837703</NS0:routerPeerID>
    <NS0:routerPeerName>peer@menegroth</NS0:routerPeerName>
    <NS0:routerPipeAdvertisement><?xml version='1.0'>&#xA;&#xA;&lt;
  10 [...]
    :PipeAdvertisement&#xA;</NS0:routerPipeAdvertisement>
    <NS0:providerPeerGroupID>urn:jxta:uuid-73AB3A3FD53E48248...FE795602</NS0:providerPeerGroupID>
    <NS0:providerPeerID>urn:jxta:uuid-596162616461...04BB08DBEAC0E756A837703</NS0:providerPeerID>
    <NS0:providerPipeAdvertisement><?xml version='1.0'>&#xA;&#xA;&lt;
    [...]
    :PipeAdvertisement&#xA;</NS0:providerPipeAdvertisement>
  </rdf:Description>
</rdf:RDF>
  
```

4.6 Aufbau der SP/P-Indices: Der BIND-Service

Dieser Abschnitt beschreibt die Implementierung des BIND-Services. Die Implementierung für den Super-Peer befindet sich in den Klassen `BindingService`, `BindingPool` und `BindingRequestProcessor`. Für den Provider-Peer wird nur die `ClientUplink`-Klasse verwendet. Dieser Service ist die Umsetzung der in Kapitel 3.1 vorgestellten Mechanismen zum Aufbau des SP/P-Indices.

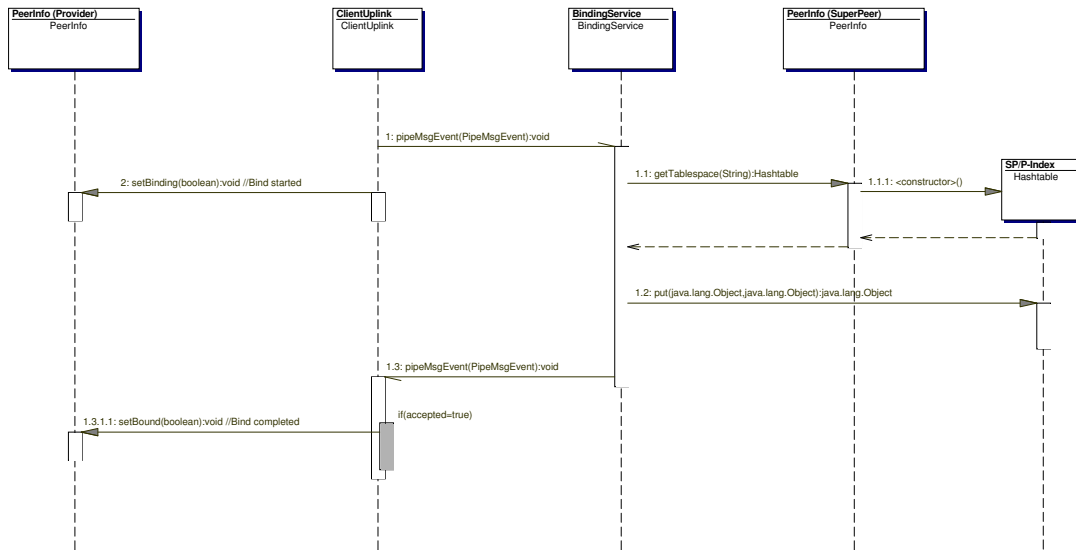


Abbildung 4.14: UML Sequenzdiagramm: Kommunikation zwischen Provider und Super-Peer

Abbildung 4.14 zeigt den Ablauf der Anmeldung eines Provider-Peers bei einem Super-Peer. Ähnlich wie die `BroadcastNetwork`-Klasse des Super-Peers versucht, andere Super-Peers zu entdecken, verwendet die `ClientUplink`-Klasse das `BINDAdvertisement` zum Auffinden des `BIND-Service`s der Super-Peers. Ist ein Super-Peer gefunden, so sendet der Provider eine Beschreibung seiner Fähigkeiten an den Super-Peer. Der Super-Peer nimmt die Beschreibung des Providers in seine Datenbank auf und sendet eine positive Bestätigung zurück. Trennt der Provider seine Verbindung zum Super-Peer, werden seine Einträge aus der Datenbank und den Routing-Indices entfernt.

Beispiel 4.9 Ausschnitt aus BINDAdvertisement.java

```
1  public class BINDAdvertisement extends DefaultAdvertisementImpl
    implements SPAdvertisement {

    public static final String MESSAGE_TYPE_BIND = "BIND-Message";

    public BINDAdvertisement() {
        midescription =
            "This is the Module Implementation " +
            "for the SuperPeer Binding Service";
10     miclass =
            net.jxta.edutella.speers.manager.ClientUplink.class.toString();
        miprovider =
            "Edutella Project (myself: Ingo Brunkhorst)";
        miurl =
            "http://edutella.jxta.org/";

        msname =
            "Edutella BIND Service";
        msid =
20         "urn:jxta:uuid-" +
            "9490667657004C58B51B876E7FCDD73E0" +
            "6DE44616BB94015B18989EA2CB79F2106";
        msdescription =
            "Service specification for Binding to SuperPeers";
        mscreator =
            "Edutella Project";

        mcid =
30         "urn:jxta:uuid-" +
            "9490667657004C58B51B876E7FCDD73E05";

        mcname =
            "Edutella BIND Service";
        mcdescription =
            "This service allows Binding Peers to SuperPeers";

        gid =
            EduConstant.EDUTELLA_PROVIDER_GID;
        gname =
40         EduConstant.EDUTELLA_PROVIDER_GNAME;
        gdescription =
            EduConstant.EDUTELLA_PROVIDER_GDESCRIPTION;

        type =
            "BIND";
    }
} // BINDAdvertisement
```

4.6.1 Der BIND-Service des Super-Peers: BindingService

Klasse: `net.jxta.edutella.speers.bind.BindingService`

Die `BindingService`-Klasse hat die in 4.15 abgebildeten Zustände, und verhält sich ähnlich der `BroadcastNetwork`-Klasse. Statt des `SuperPeerAdvertisement`s werden die Definitionen aus dem `BINDAdvertisement` verwendet.

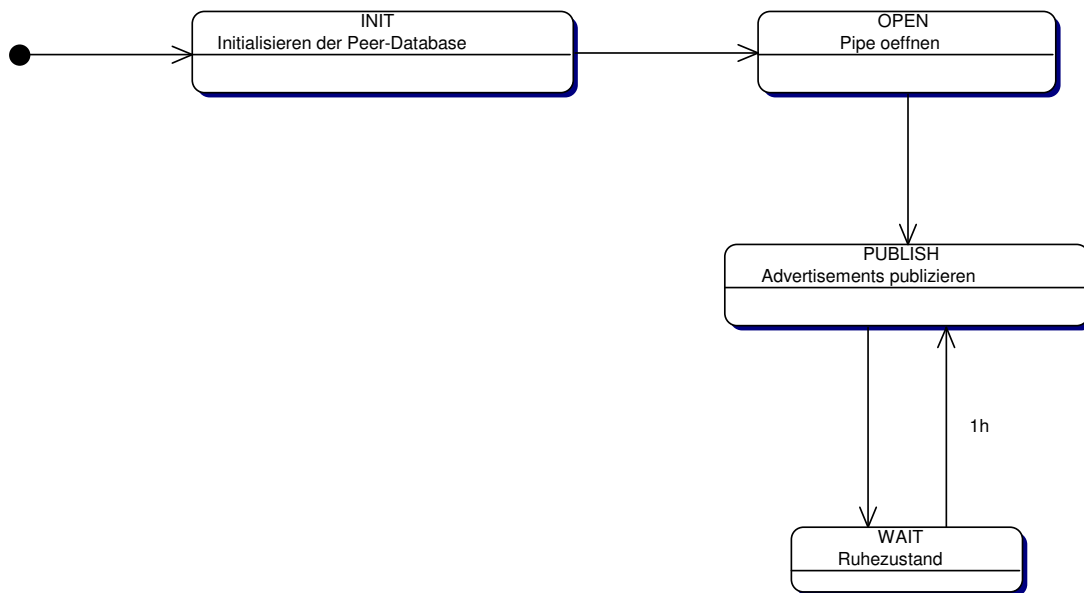


Abbildung 4.15: UML Zustandsdiagramm: Zustände des `BindingService`-Dienstes

Nach der Initialisierung der Peer-Tabellen in der `PeerInfo` wird im Zustand `OPEN` die Pipe für die Anfragen der Provider-Peers geöffnet. Damit die Provider die Pipe finden können, wird sie im Netz regelmäßig versendet (`PUBLISH`). Eine von einem Provider empfangene Nachricht wird über die `Msg.fromString()`-Factory dekodiert und als `MessageEvent` an die registrierten `EventListener` weitergeleitet, in diesem Fall der `BindingPool`. Hier wird für jeden Event ein wartender `BindingRequestProcessor` aufgeweckt, der die Anfrage bearbeitet. Die einzige Bedingung für die Aufnahme eines Provider-Peers ist die Anzahl der bereits angemeldeten Peers. Es wird in jedem Fall eine `BINDAnswer`-Message erzeugt und an die aus der empfangenen `BINDRequest`-Message dekodierten Adresse zurückgesendet.

Der Parameter `Status` in der Message hat den Wert "true", wenn in der Datenbank weniger Peers vorhanden sind als die Variable `MAXPEERS` zulässt. Damit wird dem Provider-Peer die Anbindung an diesen Super-Peer signalisiert. Die `BINDRequest`-Message wird zusätzlich über den `PeerServiceRegistry`-Singleton als `MessageEvent` an den `SpPIndexer` zum Aufbau des SP/P-Indices gesendet (siehe 3.1).

4.6.2 Der BIND-Service des Provider-Peers: ClientUplink

Klasse: `net.jxta.edutella.speers.bind.BindingService`

Den ClientUplink-Service verwendet der Provider-Peer, um nach Super-Peers zu suchen. Abbildung 4.16 zeigt die einzelnen Zustände dieses Dienstes. Nach der Initialisierung werden im Netz Advertisements des BIND-Services gesucht. Die Suche wird an die `GlobalDiscoveryListener`-Klasse delegiert (siehe Codebeispiel 4.10). Gefundene Advertisements werden über das `GlobalDiscoveryConsumer`-Interface zurückgemeldet und der wartende Thread des ClientUplink wird wieder aufgeweckt.

Beispiel 4.10 Suche nach Advertisements

```
GlobalDiscoveryListener gdl = GlobalDiscoveryListener.GLOBALDISCOVERYLISTENER;
gdl.registerMeFor(ModuleImplAdvertisement.class, this);
gdl.forcedDiscovery(null, DiscoveryService.ADV,
    "MSID", msid.toString(), 5, peerGroup);
```

Gesucht wird das `ModuleImplAdvertisement` (siehe Kapitel 4.5.2) des BIND-Services, welches über die eindeutige `ModuleSpecID` identifiziert wird. Für jedes gefundene Advertisement wird eine `PeerDescription` erzeugt und in einem Hashtable abgelegt. Ist der Provider noch nicht angemeldet, dann wird Kontakt mit dem neu gefundenen Super-Peer aufgenommen. Schlägt die Anmeldung fehl, so wird in der `PeerDescription` ein Zähler erhöht. Der Super-Peer wird dadurch an das Ende der Warteschlange gesetzt.

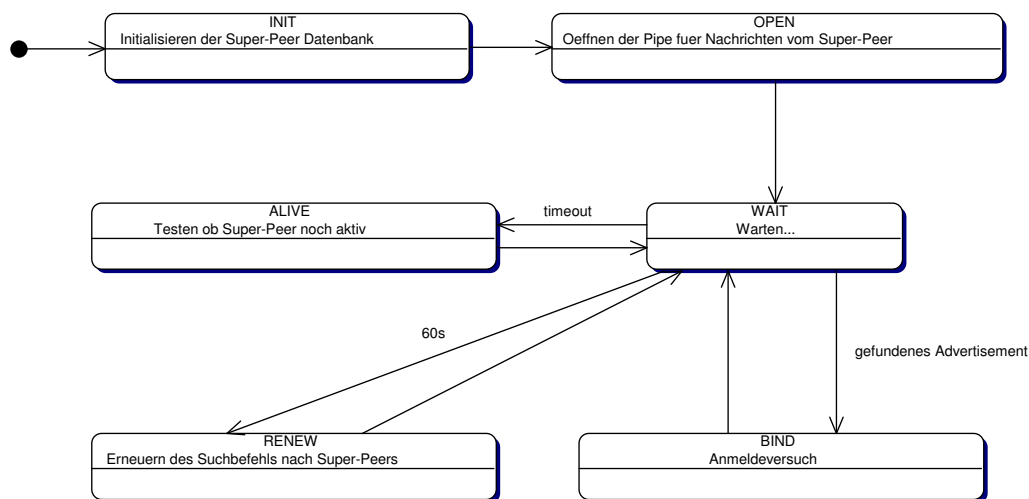


Abbildung 4.16: UML Zustandsdiagramm: Zustände des ClientUplink-Dienstes

ProviderDescription

Damit der Super-Peer die Routing-Indices erzeugen kann, benötigt er die Beschreibung der Fähigkeiten des Provider-Peers. Aufgebaut wird diese Beschreibung aus den im Provider-Peer vorhandenen Informationen. In der aktuellen Implementierung beschränkt sich die Informationsgewinnung auf das Einlesen und Auswerten einer RDF-Datei. Von jedem Property, das als Predicate in einem der RDF-Statements auftaucht, wird der Namespace zur Charakterisierung der Metadaten herangezogen (siehe Kapitel 3.1).

Die ProviderDescription ist wie die SuperPeerDescription (siehe Beispiel 4.8) ein RDF-Datenmodell auf Basis des PeerDescription-Schemas aus Anhang A.4. Für einen Provider, der die Beispieldatenbank *Künstliche Intelligenz I* aus dem Edutella-Projekt verwendet, sieht die RDF-Beschreibung wie in Beispiel 4.11 aus.

Beispiel 4.11 Beschreibung eines ProviderPeers: ProviderDescription (gekürzt)

```

1 <rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:NS0='http://www.learninglab.de/~brunkhor/rdf/peerDesc#'
  >
  <rdf:Description rdf:about='urn:jxta:uuid-59616261646162614A78...E20C64C1CA30F9A4079BBA04703'>
    <NS0:providerPipeAdvertisement>&lt;?xml version='1.0'>
      [...]
    </NS0:providerPipeAdvertisement>
    <NS0:providerPeerID>urn:jxta:uuid-59616261646162614A78746150...BBA04703</NS0:providerPeerID>
    <NS0:providerPeerGroupID>urn:jxta:uuid-73AB3A3FD53E48248FE08...602</NS0:providerPeerGroupID>
    <NS0:hasNamespace>http://www.w3.org/1999/02/22-rdf-syntax-ns#</NS0:hasNamespace>
    <NS0:hasNamespace>http://purl.org/dc/elements/1.1/</NS0:hasNamespace>
    <NS0:hasNamespace>http://telemann.kbs.uni-hannover.de:3333/olr/olr_v9#</NS0:hasNamespace>
    <NS0:hasNamespace>http://dublincore.org/2000/03/13/dcq#</NS0:hasNamespace>
  </rdf:Description>
</rdf:RDF>

```

Neben der Adresse der Pipe für QEL-Queries (ProviderPipeAdvertisement) sind durch das Property hasNamespace die vier RDF-Schemas <http://www.w3.org/1999/02/22-rdf-syntax-ns#>, <http://purl.org/dc/elements/1.1/>, http://telemann.kbs.uni-hannover.de:3333/olr/olr_v9# und <http://dublincore.org/2000/03/13/dcq#> definiert.

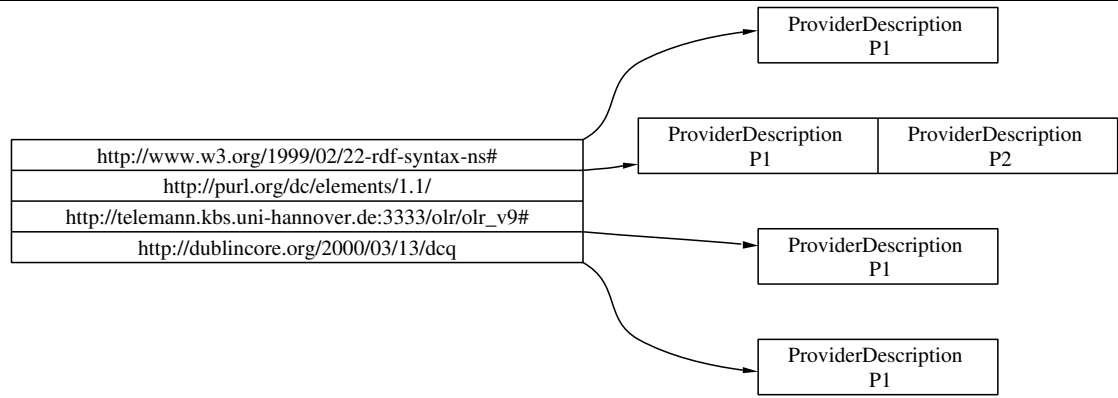
4.6.3 Verwaltung des SP/P-Indices: Die SpPIndexer-Klasse

Der SpPIndexer baut aus den BINDRequest-Messages und den damit übertragenen ProviderDescriptions den SP/P-Index auf. Dieser wird in einem Hashtable im PeerInfo des Super-Peers abgelegt.

Mit der ProviderDescription des Providers P1 aus Beispiel 4.11 und einem Provider P2 der nur das Dublin-Core-Schema <http://purl.org/dc/elements/> kennt, ergibt sich der SP/P-Index 4.12.

4.7 Bearbeiten und Weiterleiten von Queries

In diesem Abschnitt werden die Details der Implementierung vorgestellt, die zur Bearbeitung von Queries benötigt werden. Dazu gehört der Zugriff des Provider-

Beispiel 4.12 SP/P-Index nach Anmeldung der Provider P1 und P2 aus 4.6.2

Peers auf ein Repository mit Metadaten, und das Routen von Queries durch das aus den Super-Peers gebildete “Backbone“-Netzwerk. Wie in Kapitel 3 beschrieben, werden dafür Routing-Indices benötigt. Das Erzeugen der SP/SP-Routing-Indices wurde in Kapitel 4.5.1 beschrieben. In Kapitel 4.6 folgt das Verfahren zur Erstellung des noch fehlenden SP/P-Indices aus den in ermittelten Provider-Beschreibungen. In Kapitel 4.7.3 wird Anwendung der Routing-Indices und Routing-Tabellen erläutert. Neben dem eigentlichen QEL-Service, der nur in einem Provider-Peer Verwendung findet, existiert mit der Provider-Emulation im Super-Peer die Möglichkeit, die Super-Peers als Ziel für Anfragen zu verwenden. Die Emulation erlaubt die Verwendung der existierenden Consumer-Peers zur Abfrage der Super-Peers.

4.7.1 Der QEL-(Edutella Query)-Service

Klasse: `net.jxta.edutella.peers.provider.ProviderService`

Die `ProviderService`-, `ProviderPool`- und `RequestProcessor`-Klasse bilden die Fähigkeiten des Provider-Peers aus dem `provider`-Package (`net.jxta.edutella.provider` des Edutella-Projektes nach [15]). In der aktuellen Implementierung ist die Verwendung der `RDQLProvider`-Klasse fest in den `RequestProcessor` einkodiert. Die Metadatenbank wird aus einer RDF-Datei gelesen und in einem Java-Modell aus dem Jena-Toolkit abgelegt [14].

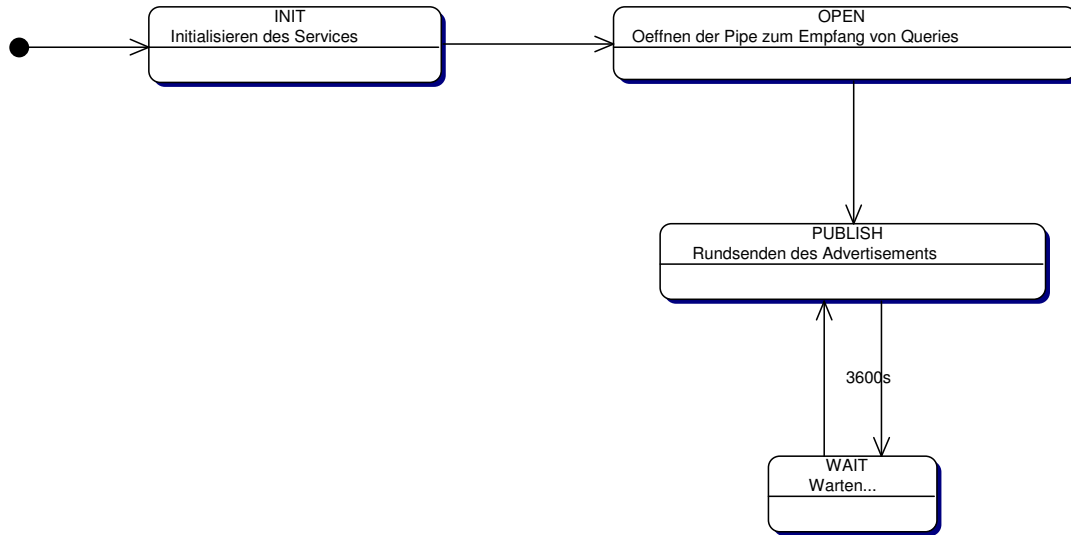


Abbildung 4.17: UML Zustandsdiagramm: ProviderService-Dienst

Wie in Abbildung 4.17 zu sehen, wird nach dem Öffnen der Pipe für eingehende Queries das dazugehörige Advertisement im JXTA-Netzwerk veröffentlicht. In regelmäßigen Abständen wird die Veröffentlichung des Advertisements erneuert. Eingehende Messages werden vom JXTA-System an die `pipeMsgEvent()`-Methode des `ProviderService` gesendet. Die `ProviderService`-Klasse verpackt die Nachricht in ein `MessageEvent` und sendet es an den `ProviderPool`. Der `ProviderPool` weckt für jede Nachricht einen wartenden `RequestProcessor` auf, der die Message bearbeitet und die Antwort über das Netzwerk an den fragenden Peer zurücksendet.

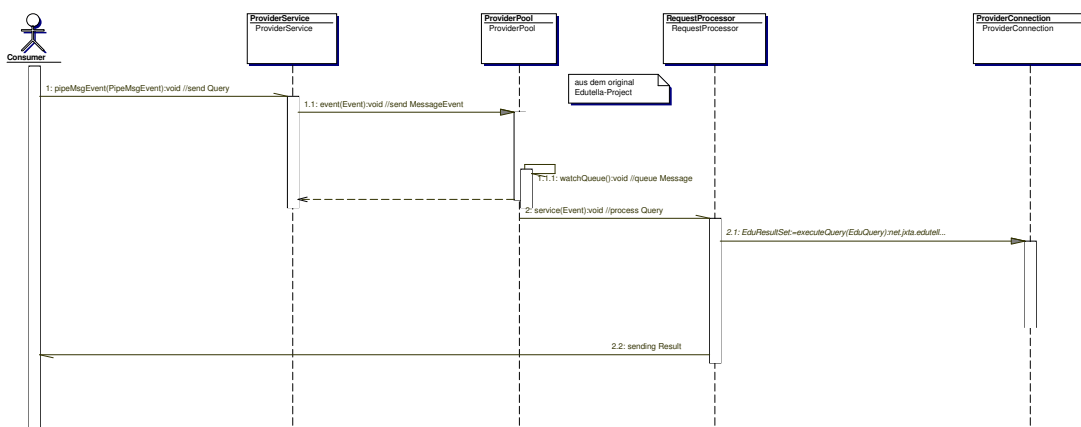


Abbildung 4.18: UML Sequenzdiagramm: Edutella Query Service

4.7.2 Die Provider-Emulation des Super-Peers

Klasse: `net.jxta.edutella.speers.provider.ProviderPoolSP`

Der Super-Peer empfängt Queries ebenfalls über die `ProviderService`-Klasse. Er reicht diese an den `ProviderPoolSP` weiter. Die Query wird mit einer eindeutigen ID versehen und zusammen mit einer Beschreibung des Consumers in einer Datenbank des `PeerInfo`-Objektes gespeichert. Aus der Query wird ein `MessageEvent` generiert und an den im nächsten Abschnitt 4.7.3 beschriebenen `Routing-Service` des Super-Peers gesendet. Mit der ID können die vom `Routing-Service` empfangenen Antworten an den passenden Consumer zurückgeschickt werden.

Beispiel 4.13 Erzeugen einer eindeutigen ID zur Identifizierung der Query

```
public String calculateHash(OldQELQuery oqelq) {
    log.debug(".calculateHash()");
    String a = oqelq.getResponsePeerID().toString();
    String b = oqelq.getResponseNumber();
    String id = a + ":" + b;
    return(id);
}
```

Gebildet wird die ID aus der `PeerID` des Consumer-Peers und der ID der Query.

4.7.3 Der ROUTE-Service

Der `RoutingService` ist eine Kopie des `ProviderService`. Die Unterschiede liegen in der unterschiedlichen Adresse der Pipe, in der die Queries empfangen werden. Die Nachrichten werden an zwei `EventListener` weitergereicht. Für das auf den SP/P-Indices basierende Routing der Queries an die Provider wird die `RoutingPoolSPP`-Klasse und der `SpPRoutingProcessor` verwendet. Das Routing zwischen den SuperPeers wird über die `RoutingPoolSPSP`-Klasse und den `SpSpRoutingProcessor` abgewickelt.

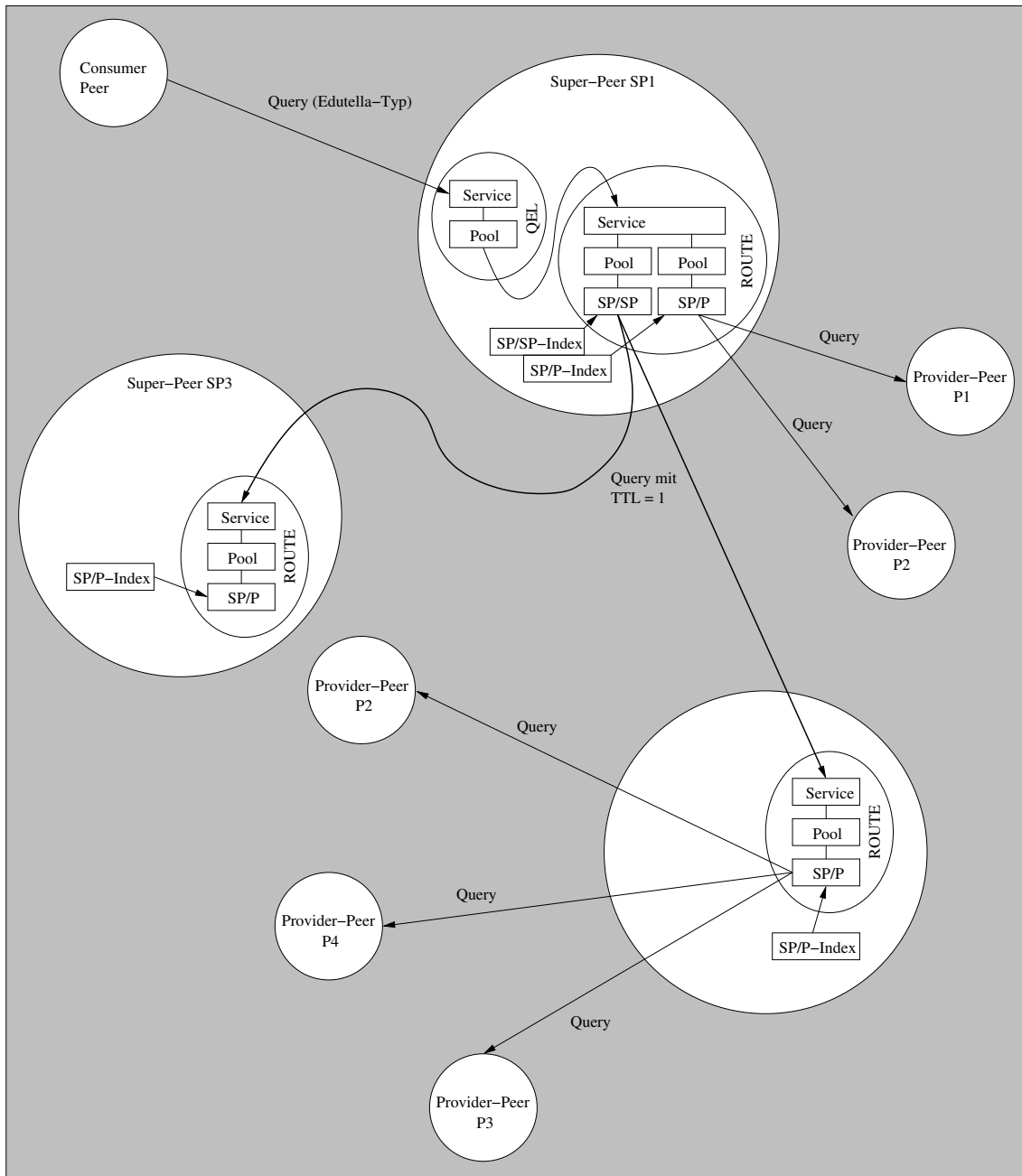


Abbildung 4.19: Routing von Queries im Super-Peer-Netzwerk

Abbildung 4.19 zeigt den Transport einer Query durch das Netz. Empfängt ein Super-Peer eine Query, dann leitet er sie über den `RoutingPoolSPP` an die angeschlossenen Provider weiter. Der `RoutingPoolSPP` erzeugt mit Hilfe des `SpPRouters` und der vom `SpPIndexer` generierten Routing-Indices eine Routing-Tabelle. Diese Tabelle besteht aus den `ProviderDescriptions` der abzufragenden Provider und einem zusätzlichen Statusfeld. Für jeden Eintrag in dieser Tabelle wird ein `SpPRoutingProcessor` gestartet, der die Verbindung zu einem der Provider in der Tabelle aufbaut. Da die Anzahl der Provider in der Tabelle bekannt ist, kann dem aufrufenden Super-Peer die zu erwartende Anzahl der Ant-

worten übermittelt werden. Für die Übermittlung der Anzahl der zu erwartenden Antworten wird die erste der verwendeten SpPRoutingProcessoren verwendet. Vor dem Versenden wird die Query umgeschrieben, damit die Antworten an den richtigen Super-Peer zurückgesendet werden. (siehe Abbildung 4.20). Der SpPRoutingProcessor wartet auf die Antwort des entsprechenden Providers und leitet sie zum gespeicherten Absender zurück.

Der Pfad durch das Netz ist in Abbildung 4.20 zu erkennen. In jedem Super-Peer wird die Query umgeschrieben, damit der folgende Peer sie an die richtige Adresse zurücksendet. Bei der Antwort wird nur die Zieladresse angepaßt. Der Super-Peer SP1 transformiert die Antwort zusätzlich wieder in das vom Consumer verstandene Format zurück.

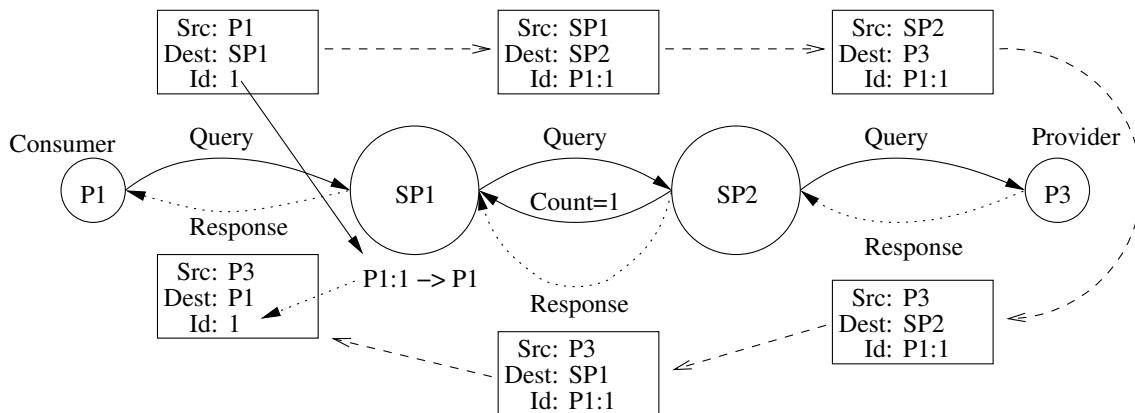


Abbildung 4.20: Der Pfad durch das Netz

Für den Aufbau der RoutingTable ist der Router zuständig, wobei das Format der Tabelle für SP/P-Routing und SP/SP-Routing identisch ist (siehe Abbildung 4.21). Über den RoutingEntryIterator kann auf die einzelnen Einträge der RoutingTable zugegriffen werden. Das Statusfeld ist in der aktuellen Implementierung immer auf RoutingEntry.RELAY gesetzt.

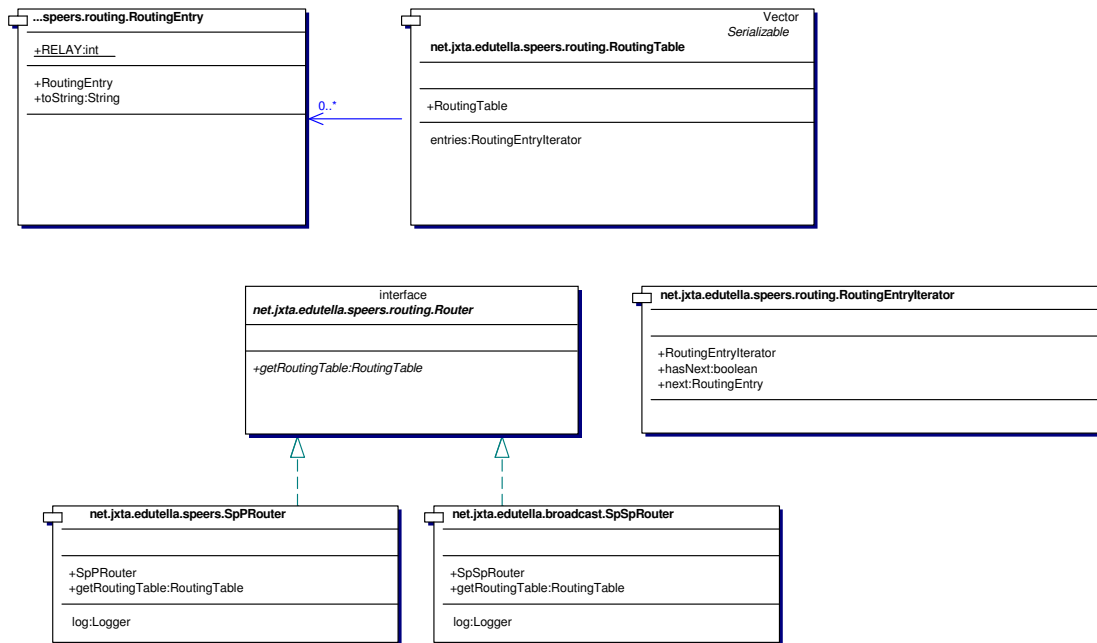


Abbildung 4.21: UML Klassendiagramm: Routing

Routing-Entscheidungen: Charakterisieren der Queries.

Die für den Aufbau der `RoutingTable` zuständige Klasse benötigt neben dem SP/P-Index auch das aus der Query erzeugte `EduQuery`-Objekt. Die `QueryCharakterizer`-Klasse analysiert die Elemente der Query und vergleicht sie mit den Einträgen des Routing-Indices. Anhand der Charakterisierung der Query entscheidet der Router, welche Einträge des SP/P-Indices in die Routing-Tabelle übernommen werden.

Eine `EduQuery` besteht aus Literalen und Rules (siehe Abbildung 2.2. Rules bestehen aus Head und Body, der Body wiederum ist ein Literal. Die `analyzeQuery()`-Methode liefert eine Aggregation von `DescriptionElements` zurück, die den Inhalt der Query beschreiben. Das einzige in der aktuellen Implementierung vorhandene Komponente ist die Analyse der Query hinsichtlich verwendeter RDF-Schemas. Die Elemente der Charakterisierung können direkt als Keys zur Selektion der Elemente des Routing-Indices verwendet werden.

Die Query aus Beispiel 2.2 und Abbildung 2.3 enthält ein einziges Statement mit zwei Variablen und dem Predicate `dc:title`. Die daraus erzeugte Charakterisierung enthält ein Element mit dem Namespace des Dublin-Core Schemas `http://purl.org/dc/elements/1.1/` als Inhalt. Aus dem Routing-Index des Beispiels 4.12 wird das zweite Element selektiert. Die resultierende Routing-Tabelle enthält die Provider P1 und P2 als Ziele.

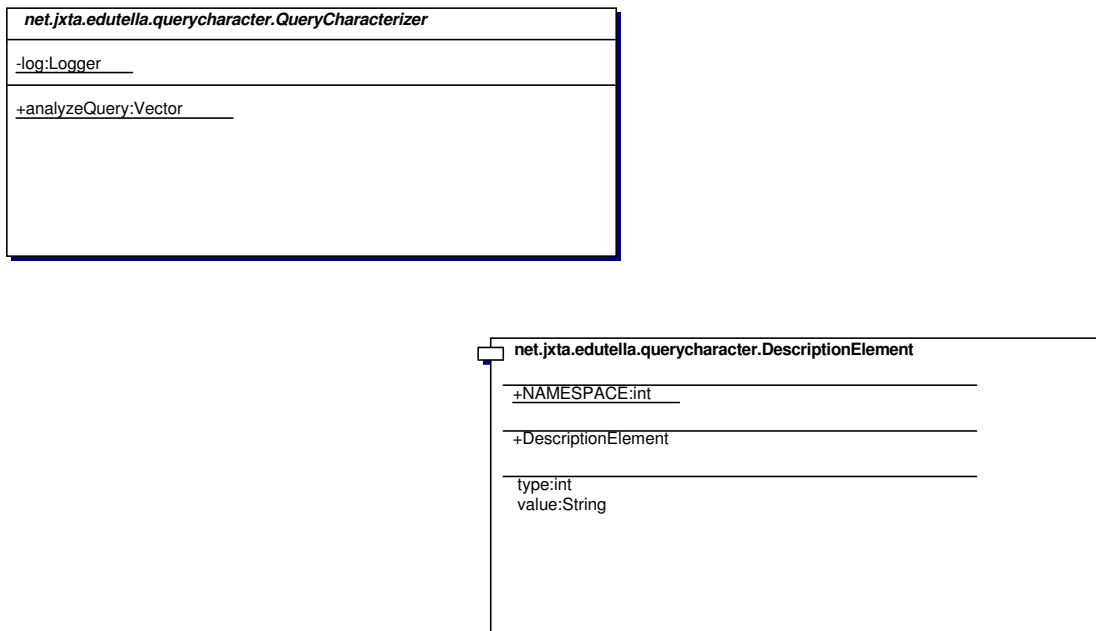


Abbildung 4.22: UML Klassendiagramm: Charakterisierung einer EduQuery

Die Weiterleitung der Queries zwischen den Super-Peers wird von der `RoutingPoolSPSP`-Klasse übernommen. In der aktuellen Implementierung wird nur die Broadcast-Topology aus Kapitel 3.2.2 eingesetzt. Eine Query wird von einem Super-Peer direkt an alle anderen Super-Peers weitergeleitet. Um Schleifen zu vermeiden, wird in der Query ein TTL-Feld (Time to Life) verwendet. Durch die Initialisierung der TTL auf 1 wird sichergestellt, daß jeder Super-Peer die Query nur einmal erhält.

Jeder Super-Peer, der eine Query erhält, vermindert den Wert des TTL-Feldes um eins, und leitet die Query nur an andere Super-Peers weiter, wenn die Lebensdauer noch nicht abgelaufen ist ($TTL > 0$).

Der `RoutingPoolSPSP` ermittelt den zuständigen Router über die `getRouter()`-Methode der Topology-Klasse.

Für die `BroadcastNetwork-Topology` ist das der `SpSpRouter`, der den vom `SpSpIndexer` erzeugten SP/SP-Index zum Aufbau der Routing-Tabelle verwendet.

Für jeden Super-Peer wird ein `SpSpRoutingProcessor` gestartet, der die Query weiterleitet und die Responses entgegennimmt. Zusätzlich wertet er die `ProviderCount-Message` des `SpPRoutingProcessors` auf der Gegenseite aus, und beendet die Bearbeitung, wenn alle Antworten eingetroffen sind.

4.8 System-Management-Console: CONSOLE-Service

Für den Betreiber eines Super-Peers ist es hilfreich, den Zustand seines Peers jederzeit überwachen und auch aktiv in den Ablauf eingreifen zu können. Der "CONSOLE"-Service erlaubt den Zugriff auf den Super-Peer mit einem Telnet-Client [25]. Über die Console können Informationen zu einzelnen Diensten abgefragt werden und der Peer heruntergefahren oder vom Netz getrennt werden. Der Inhalt der Routing-Indices und Peer-Datenbank kann angezeigt und beobachtet werden. Eintreffende Queries oder neue Peers werden dem Administrator angezeigt.

Kapitel 5

Die Zukunft des Edutella-Super-Peer-Netzwerkes

Die in dieser Arbeit erstellte Software ist als Basis für den Aufbau eines Super-Peer-Netzwerkes gedacht. Der erste Schritt zur Umsetzung der in [22] beschriebenen Ideen ist damit getan. Doch zum jetzigen Zeitpunkt ist nur ein kleiner Teil der Möglichkeiten ausgeschöpft, die durch den Einsatz von Routing-Indices machbar sind. Durch die Super-Peer/Peer-Indices werden Queries nur noch an die Provider gesendet, deren RDF-Metadaten dem Charakter der Query entsprechen. Die Auswertung der Query und die Granularität des SP/P-Indices beschränkt sich in der aktuellen Implementierung nur auf RDF-Schemas. Durch eine genauere Analyse der Metadaten lassen sich die Anfragen noch selektiver auf die Provider-Peers verteilen und dadurch läßt sich Bandbreite sparen. Ein Schritt ist die Verbesserung der zum Aufbau des SP/SP-Indices verwendeten Verfahren. Schnittpunkt zwischen dem Routing-Service der Super-Peers und der Topology des Super-Peer-Netzwerkes sind die aus den SP/SP-Indices erzeugten Routing-Tabellen.

obachtungen sendet er über das `EventInterface` an die `Indexer-Services` um die `Routing-Indices` anzupassen. Realisiert man die `PeerInfo-Datenbank` als `RDF-Datenmodell`, so werden Informationen über die Struktur des Netzes als `Metadaten` verfügbar. Über den `Edutella Query Service` können Peers damit Information über den Aufbau des Netzes ermitteln. Um die `Performance` des `Super-Peer-Netzwerks` zu bestimmen, müssen Messungen im Netz vorgenommen werden. Über eine `Management-Schnittstelle` kann die `Auslastung` der Peers und die `Laufzeit` von `Queries` ermittelt werden. Der in `Abbildung 5.2` in die Kette eingefügte `Frequency Counter` ermittelt die `Häufigkeit` der in `Anfragen` und `Antworten` auftretenden `RDF-Elemente`. Die daraus generierten `Statistiken` können genutzt werden, um herauszufinden, welche `Queries` von besonderer Bedeutung sind und optimiert werden sollten. Die Aufgaben eines `Super-Peers` sind vielfältig. Neben der `Entlastung` des Netzwerks sollen `Super-Peers` durch ihre hohe `Verfügbarkeit` das Netz stabilisieren. Das `HyperCup-Protokoll` [27] verwendet spezielle Verfahren, um die `Positionen` im `Hypercube` mit Peers zu besetzen. Mit der `Anbindung` der `HyperCup-Architektur` an die in dieser Arbeit entwickelte `Super-Peer-Architektur` entsteht ein gut skalierendes `Peer-to-Peer Netzwerk`.

Anhang A

Anhang

A.1 API

`net.jxta.edutella.querycharacter`

Klassen zur Charakterisierung von Edutella Queries. Zerlegt die Queries in einzelne Elemente.

`net.jxta.edutella.broadcast`

Einfache Super-Peer-Topology. Nachrichten werden direkt an alle bekannten Super-Peers gesendet.

`adv`

Zur Topology gehörende Service-Advertisements.

`routing`

Routing-Hilfsklassen für die Topology.

`net.jxta.edutella.speers`

Enthält die Hauptkomponenten der neuen Peerarchitektur.

`adv`

Service-Advertisements

`config`

Peer-Configuration

`event`

Events werden für der Kommunikation zwischen den Bausteinen der Peers verwendet.

`messages`

Messages können durch JXTA-Pipes zu anderen Peers versendet werden.

`document`

RDF-Modelle zur Definition der Fähigkeiten und Eigenschaften der Peers.

- info
Datenbank und Indices.
- provider
Klassen für den Edutella Query Service (QEL)
- routing
Implementierung des Routing-Services (ROUTE).
- manager
System Management Console
- bind
Protokoll zur Anbindung der Provider-Peers an einen Super-Peer.
- top
Interface-Deklaration für die Topology-Klassen.

A.2 Online

Die API-Dokumentation und die Quellen stehen unter der URL

<http://www.learninglab.de/~brunkhor/superpeers>

zum Download bereit.

A.3 Services

Name		Appearance	Advertisement	Description
QEL	RDF-QEL-i Query Service	Provider	QELAdvertisement	Service for receiving RDF-QEL Queries from Consumer-Peers.
MEL	RDF Modification Service	Provider	MELAdvertisement	This Service is used for remote modification of a peers metadata repository.
BIND	Binding-Service	all Peers	BINDAdvertisement	This service is used by the Provider to connect itself to a super-peer.
ROUTE	Routing-Service	Super-Peer	ROUTEAdvertisement	This is a service used for routing queries between super-peers based on routing-indices.
CONSOLE	Management Console	all Peers	no JXTA	The Management-Console is used for retrieving information from a peer and modifying peers parameters.
IDX1	SP/SP-Indexer	Super-Peer	no JXTA	The SP/SP-Indexer creates the routing-index for routing messages between the super-peers. Creation of this index is based on information provided by the topology class
IDX2	SP/S-Indexer	Super-Peer	no JXTA	Based on information from the Binding-Service this subsystem creates the routing-index for super-peer/peer connections.

Tabelle A.1: Services

A.4 PeerDescription Schema

```

1  <?xml version="1.0"?>
    <!DOCTYPE rdf:RDF [
      <!ENTITY rdfsns 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
      <!ENTITY rdfsns 'http://www.w3.org/2000/01/rdf-schema#'>
      <!ENTITY peerdesc 'http://server3.learninglab.uni-hannover.de/~brunkhor/rdf/peerDesc/'>
    ]>

    <rdf:RDF xmlns:rdf="&rdfsns;"
            xmlns:rdfs="&rdfsns;"
10         xmlns:pd="&peerdesc;">

      <rdf:Description rdf:about="&peerdesc;">
        <dc:title xml:lang="en-US">RDF Schema for describing the abilities of the Edutella (super)peers</dc:title>
        <dc:publisher xml:lang="en-US">Learning Lab Lower Saxony (L3S)</dc:publisher>
        <dc:description xml:lang="en-US">Description of the Edutella superpeer abilities (Draft)</dc:description>
        <dc:language xml:lang="en-US">English</dc:language>
        <dcterms:issued>2002-12-16</dcterms:issued>
        <dcterms:modified>2002-12-16</dcterms:modified>
        <dc:source rdf:resource="http://server3.learninglab.uni-hannover.de/~brunkhor/rdf/peerDesc/">
20 </rdf:Description>

      <rdf:Property rdf:about = "&peerdesc;peername">
        <rdfs:label xml:lang="en-US">PeerName</rdfs:label>
        <rdfs:comment xml:lang="en-US">The name given to the Peer/Superpeer.</rdfs:comment>
        <dc:description xml:lang="en-US">The name of the Peer</dc:description>
        <rdfs:isDefinedBy rdf:resource="&peerdesc;"/>
        <dcterms:issued>2002-12-16</dcterms:issued>
      </rdf:Property>

30 <rdf:Property rdf:about="&peerdesc;hasNamespace">
      <rdfs:label xml:lang="en-US">hasNamespace</rdfs:label>
      <rdfs:comment xml:lang="en-US">Peer knows of this Schema</rdfs:comment>
      <dc:description xml:lang="en-US">A RDF-Schema the Peer knows about</dc:description>
      <rdfs:isDefinedBy rdf:resource="&peerdesc;"/>
      <dcterms:issued>2002-12-16</dcterms:issued>
    </rdf:Property>

40 </rdf:RDF>

```

A.5 SWEBOK-Query

Beispiel A.1 ist die XML-Serialisierung einer Suchanfrage nach deutschsprachigen Dokumenten über Softwaredesign. Erstellt wurde die Query mit dem SWEBOK-Consumer-Peer [5] aus dem Edutella-Projekt [30].

Beispiel A.1 RDF-QEL-i Suche nach Dokumenten über Softwaredesign

```

1  <?xml version='1.0'?>
    <rdf:RDF
      xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
      xmlns:edu='http://www.edutella.org/edutella#'
      xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
      <edu:Variable rdf:about='#Language'
        rdfs:label='Language'/>
      <edu:QEL3Query rdf:about='#genQuery'>
        <edu:hasResultType rdf:resource='http://www.edutella.org/edutella#TupleResult'/>
10    <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st0'>
        <rdf:subject rdf:resource='#CourseItem'
          rdf:type='http://www.edutella.org/edutella#Variable'
          rdfs:label='CourseItem'/>
        <rdf:predicate rdf:resource='http://purl.org/dc/elements/1.1/title'/>
        <rdf:object rdf:resource='#Title'
          rdf:type='http://www.edutella.org/edutella#Variable'
          rdfs:label='Title'/>
      </edu:RDFReifiedStatement>
20    </edu:hasQueryLiteral>
      <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st1'>
        <rdf:subject rdf:resource='#CourseItem'/>
        <rdf:predicate rdf:resource='http://www.w3.org/1999/02/22-rdf-syntax-ns#type'/>
        <rdf:object rdf:resource='#Type'
          rdf:type='http://www.edutella.org/edutella#Variable'
          rdfs:label='Type'/>
      </edu:RDFReifiedStatement>
30    </edu:hasQueryLiteral>
      <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st2'>
        <rdf:subject rdf:resource='#CourseItem'/>
        <rdf:predicate rdf:resource='http://www.imsproject.org/rdf/imsmd_classificationvip2#Taxon'/>
        <rdf:object rdf:resource='#Topic'
          rdf:type='http://www.edutella.org/edutella#Variable'
          rdfs:label='Topic'/>
      </edu:RDFReifiedStatement>
40    </edu:hasQueryLiteral>
      <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st3'>
        <rdf:subject rdf:resource='#CourseItem'/>
        <rdf:predicate rdf:resource='http://www.w3.org/1999/02/22-rdf-syntax-ns#type'/>
        <rdf:object rdf:resource='http://telemann.kbs.uni-hannover.de:3333/olr/olr_v9#PDF'/>
      </edu:RDFReifiedStatement>
50    </edu:hasQueryLiteral>
      <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st4'>
        <rdf:subject rdf:resource='#CourseItem'/>
        <rdf:predicate rdf:resource='http://purl.org/dc/elements/1.1/language'/>
        <rdf:object rdf:resource='http://www.kbs.uni-hannover.de/~brase/lang.rdf#de'/>
      </edu:RDFReifiedStatement>
60    </edu:hasQueryLiteral>
      <edu:hasQueryLiteral>
      <edu:RDFReifiedStatement rdf:about='#st5'>
        <rdf:subject rdf:resource='#CourseItem'/>
        <rdf:predicate rdf:resource='http://www.imsproject.org/rdf/imsmd_classificationvip2#Taxon'/>
        <rdf:object rdf:resource='http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf#SWDesign'/>
      </edu:RDFReifiedStatement>
    </edu:hasQueryLiteral>
  </edu:QEL3Query>
</rdf:RDF>

```

Durch die Reification ist die Serialisierung der Query komplex, nach Umwandlung in *Reified Statements* hat die Query die in Abbildung A.1 dargestellte vereinfachte Form.

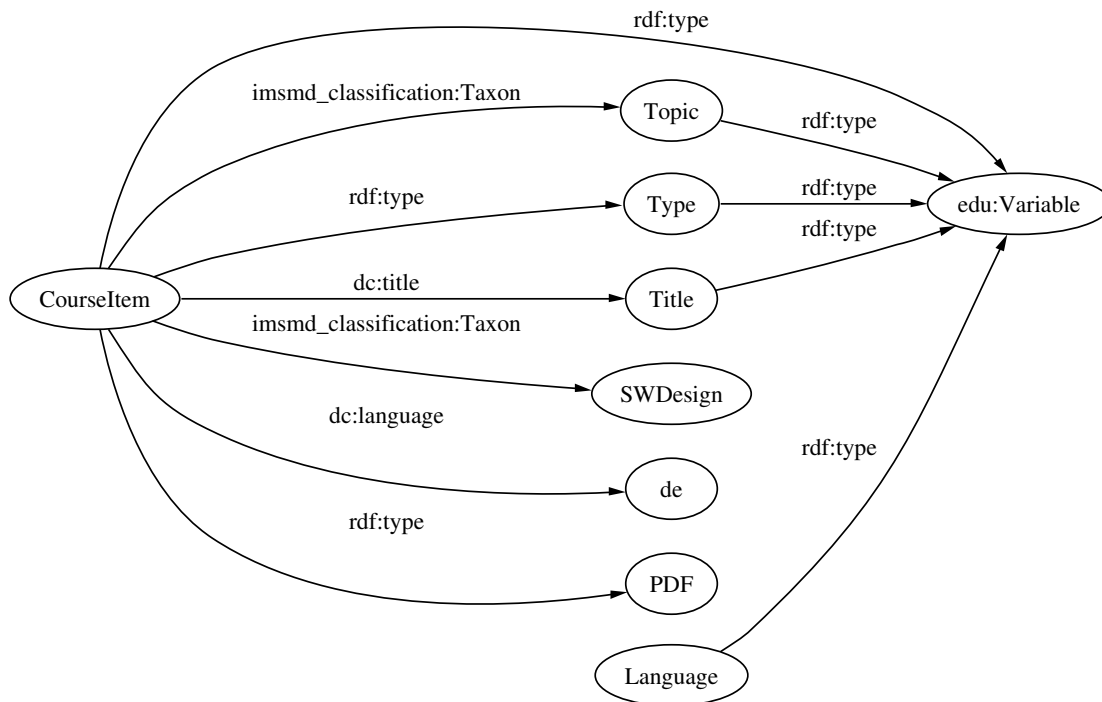


Abbildung A.1: RDF-Graph: Vereinfachung der Query aus A.1

Gesucht wird ein *CourseItem* aus dem Bereich (Taxonomy) *SWDesign*, in der Sprache (*dc:language*) *deutsch*, und dem Dokumenten-Format (*rdf:type*) (*PDF*)

Literaturverzeichnis

- [1] Gnutella. <http://www.gnutella.com/>.
- [2] Napster (site is closed). <http://www.napster.com/>.
- [3] Rdf binding of lom metadata. <http://kmr.nada.kth.se/el/ims/metadata.html>.
- [4] Seti@home. <http://setiathome.berkeley.edu/>.
- [5] BOURQUE, P., DUPUIS, R., AND ABRAN, A. The guide to the software engineering body of knowledge. *IEEE Software* 16, 6 (November/December 1999), 35–44. <http://www.swebok.org/>.
- [6] BRAY, T., PAOLI, J., AND SPERBERG-MCQUEEN (EDS), C. M. “Extensible Markup Language (XML) 1.0 (2nd Edition)”. W3C Recommendation, 2000.
- [7] BRICKLEY, D., AND GUHA, R., Eds. *RDF Schema Specification 1.0* (<http://www.w3.org/TR/rdf-nt>, März 2000), vol. RDF, Resource Description Framework, World Wide Web Consortium, W3C. Candidate Recommendation.
- [8] BUSSE, S. *Modellkorrespondenzen für die kontinuierliche Entwicklung mediatorbasierter Informationssysteme*. PhD thesis, Technische Universität Berlin, 2002. Dissertation.
- [9] CRESPO, A., AND GARCIA-MOLINA, H. Routing indices for peer-to-peer systems. In *Proceedings International Conference on Distributed Computing Systems* (July 2002).
- [10] DBXML GROUP. dbxml native database. <http://www.dbXML.org/>.
- [11] DOLOG, P., GAVRILOAIE, R., NEJDL, W., AND BRASE, J. Integrating adaptive hypermedia techniques and open rdf-based environments.
- [12] GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. *Design Patterns*. Addison-Wesley, 1995. Elements of Reusable Object-Oriented Software.
- [13] HAYES, P., Ed. *RDF Model Theory* (<http://www.w3.org/TR/rdf-nt>, September 2001), vol. RDF, Resource Description Framework, World Wide Web Consortium, W3C. Work in progress.

- [14] HP SEMANTIC WEB ACTIVITY. Jena toolkit. Tech. rep., HP Labs. <http://www.hpl.hp.com/semweb/javadoc/index.html>.
- [15] L3S AND KNOWLEDGE BASED SYSTEMS, INSTITUTE AIFB. *EDUTELLA: Searching and Annotating Resources within a RDF-bases P2P Network* (University Hannover, University Karlsruhe, Germany, March 2002), ACM.
- [16] LASSILA, O., AND SWICK, R., Eds. *RDF Model and Syntax Specification* (<http://www.w3.org/TR/REC-rdf-syntax>, Februar 1999), vol. RDF, Resource Description Framework, World Wide Web Consortium, W3C. Work in progress, RECOMMENDATION.
- [17] LESER, U. *Query Planning in Mediator Based Information Systems*. PhD thesis, Technische Universität Berlin, 2002. Dissertation.
- [18] LV, Q., RATNASAMY, S., AND SHENKER, S. Can heterogeneity make gnutella scalable?
- [19] MANKU, G., AND MOTWANI, R. Approximate frequency counts over data streams, 2002.
- [20] NEJDL, W., SIBERSKI, W., SIMON, B., AND TANE, J. Towards a modification exchange language for distributed rdf repositories. Tech. rep., Learning Lab Lower Saxony, Wirtschaftsuniversität Wien, Universität Karlsruhe, 2002.
- [21] NEJDL, W., SIBERSKI, W., WOLPERS, M., AND SCHMITZ, C. Routing and clustering in schema-based super peer networks.
- [22] NEJDL, W., WOLPERS, M., SIBERSKI, W., SCHMITZ, C., SCHLOSSER, M., BRUNKHORST, I., AND LÖSER, A. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks.
- [23] ON, R. T. Project jxta: An open, innovative collaboration.
- [24] *Organisation und Nutzung von verteilten Inhalten und Lehrmaterialien* (November 2001). <http://www.learninglab.de/padlr/>.
- [25] POSTEL, J., AND REYNOLDS, J. Request for comments: 854 (telnet protocol specification).
- [26] RITTER, J. Why gnutella can't scale. Tech. rep., Darkridge, Inc., 2001.
- [27] SCHLOSSER, M., SINTEK, M., DECKER, S., AND NEJDL, W. HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In *International Workshop on Agents and Peer-to-Peer Computing* (Bologna, Italy, July 2002).
- [28] WEBTEAM AT DUBLINCORE.ORG. Rdf schema declaration for the dublin core element set 1.1. Tech. rep., Dublin Core Metadata Initiative (DCM), August 2001.

-
- [29] WIEDERHOLD, G. Mediators in the architecture of future information systems. In *Readings in Agents*, M. N. Huhns and M. P. Singh, Eds. Morgan Kaufmann, San Francisco, CA, USA, 1997, pp. 185–196.
- [30] WOLF, B. Peer-to-peer network for distributed learning repositories. Diplomarbeit, Universität Hannover, Dezember 2001.
- [31] WOLPERS, M., BRUNKHORST, I., AND NEJDL, W. An o-telos provider peer for the rdf-based edutella.
- [32] YANG, B., AND GARCIA-MOLINA, H. Designing a super-peer network. <http://dbpubs.stanford.edu:8090/pub/2002-13>, 2002.

Abbildungsverzeichnis

2.1	Der Aufbau der JXTA Architektur	5
2.2	Provider, Consumer, Rendezvous: Die drei Peer-Varianten im Edutella-Netzwerk	6
2.3	Query aus Beispiel 2.2 als Graph	9
2.4	ECDM: Edutella Common Data and Exchange Model	10
2.5	UML Sequenzdiagramm: Anfrage an Provider	11
2.6	Abfrage von Peers in einem unstrukturierten Netzwerk	12
3.1	Eine einfache Super-Peer-Netzwerk-Topology	13
3.2	Clustergrenzen	14
3.3	Super-Peer Netzwerk Topology	15
3.4	UML Sequenzdiagramm: Provider meldet sich beim Super-Peer an	16
3.5	Super-Peer/Peer-Index	18
3.6	Super-Peer/Super-Peer-Index	20
3.7	Hypercup-Topology	21
3.8	Hypercup: Ausbreitung der Nachrichten	22
3.9	Messages in der Broadcast-Topology	23
3.10	Schema-basiertes Clustering	24
4.1	Die alte und neue Architektur im Überblick	27
4.2	Überblick über die Super-Peer Architektur	28
4.3	UML Klassendiagramm: Die Basisklassen eines Peers	29
4.4	UML Klassendiagramm: net.jxta.edutella.peers.adv	30
4.5	UML Zustandsdiagramm: Hochfahren eines Peers	31
4.6	UML Klassendiagramm: PeerConfiguration	32
4.7	Provider-Peer-Konfiguration	33
4.8	Super-Peer-Konfiguration	34
4.9	Super-Peer-Konfiguration	35
4.10	UML Klassendiagramm: Topology	36
4.11	Topology	37
4.12	UML Zustandsdiagramm: Zustände des BroadcastNetwork-Dienstes	38
4.13	UML Zustandsdiagramm: Zustände des SpSpIndexer-Dienstes	42
4.14	UML Sequenzdiagramm: Kommunikation zwischen Provider und Super-Peer	43
4.15	UML Zustandsdiagramm: Zustände des BindingService-Dienstes	45
4.16	UML Zustandsdiagramm: Zustände des ClientUplink-Dienstes	46

4.17 UML Zustandsdiagramm: ProviderService-Dienst	49
4.18 UML Sequenzdiagramm: Edutella Query Service	49
4.19 Routing von Queries im Super-Peer-Netzwerk	51
4.20 Der Pfad durch das Netz	52
4.21 UML Klassendiagramm: Routing	53
4.22 UML Klassendiagramm: Charakterisierung einer EduQuery	54
5.1 Routing-Tabellen: Vermittler zwischen Topology und Routing-Service	57
5.2 Verketteten neuer Module	57
A.1 RDF-Graph: Vereinfachung der Query aus A.1	64

Tabellenverzeichnis

3.1	Charakterisierung der Query aus Beispiel 3.1	17
3.2	Super-Peer/Super-Peer-Index im Super-Peer SP2 aus Bild 3.3.1	25
A.1	Services	61

Beispiele

2.1	Suche nach allen Titeln (Datalog)	7
2.2	Beispielanfrage zur Suche aller Titel	8
3.1	Query: Dokumente über Softwaredesign	17
3.2	Beispiele für Korrespondenzen zwischen RDF-Schemas	25
3.3	Beispiele für Views auf RDF-Schemas	26
3.4	Abbildung der Attribute aus dem lecture-Schema auf die RDF-Schemas DC und LOM	26
3.5	Resultierende Korrespondenzen für die Umsetzung auf die RDF-Schemas in den Peers P1 und P2	26
4.1	Ausschnitt aus ProviderPeer.java	32
4.2	Ausschnitt aus SuperPeerGroup.java	35
4.3	Ausschnitt aus BroadcastNetwork.java	37
4.5	Suche nach Advertisements	38
4.4	Advertisement eines Super-Peers: ModuleImplAdvertisement	39
4.6	SuperPeerAdvertisement.java	40
4.7	Ausschnitt aus BroadcastNetwork.java	41
4.8	Beschreibung eines SuperPeers: SuperPeerDescription (gekürzt)	42
4.9	Ausschnitt aus BINDAdvertisement.java	44
4.10	Suche nach Advertisements	46
4.11	Beschreibung eines ProviderPeers: ProviderDescription (gekürzt)	47
4.12	SP/P-Index nach Anmeldung der Provider P1 und P2 aus 4.6.2	48
4.13	Erzeugen einer eindeutigen ID zur Identifizierung der Query	50
A.1	RDF-QEL-i Suche nach Dokumenten über Softwaredesign	63

Nomenclature

Backbone 1. The high-traffic-density connectivity portion of any communications network. (188) 2. In packet-switched networks, a primary forward-direction path traced sequentially through two or more major relay or switching stations. Note: In packet-switched networks, a backbone consists primarily of switches and interswitch trunks. *Quelle: Telecommunications - Glossary of Telecommunication Terms*

Cluster Eine Gruppe von Peers und Super-Peers, die ähnliche Metadaten verwalten oder räumlich nah zusammenliegen.

Consumer Ein Peer im Edutella-Netzwerk. Sucht Provider und sendet vom Benutzer oder einem Programm erzeugte Anfragen an einen der gefundenen Provider.

ECDM Edutella Common Data and Exchange Model, Basis für den Datenaustausch im Edutella-Netzwerk.

Firewall Eine Firewall ist Software auf einem sogenannten "Gateway"-Rechner, der die internen Ressourcen eines Netzwerks schützt.

JXTA Eine Peer-to-Peer Infrastruktur entwickelt von SUN

Metadaten Metadaten sind Daten **über** Daten. In Umfeld dieser Arbeit sind Metadaten die Beschreibung von Lehr- und Lernmaterial mit Attributen wie beispielsweise Autor, Titel, Kommentar, Kurzbeschreibung und Rechten.

MPEG Motion Picture Experts Group. Der MPEG-Standard bietet qualitativ hochwertige Datenkompression von Audio- und Videodaten.

Ontology Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. The agents sharing a vocabulary need not share a knowledge base; each knows things the other does not, and an agent that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary.

PADLR Personalized Access to Distributed Learning Repositories, Organisation und Nutzung von verteilten Inhalten und Lehrmaterialien.

PDF Adobe Portable Document Format

Peer Ein einzelner Knoten in einem Peer-to-Peer Netz

Peer-to-Peer Eine dezentralisierte Architektur für Netzwerke. Daten und Rechenlast werden auf alle Knoten des Netzwerks verteilt.

Pipes In JXTA sind Pipes transparente Kommunikationskanäle, über die Nachrichten ausgetauscht werden können.

Provider Ein Peer im Edutella-Netzwerk. Beantwortet Anfragen in RDF-QEL aus einer RDF-Datenbasis heraus. Es existieren Anbindungen für verschiedene Datenbanksysteme und Abfragesprachen (SQL, Prolog, O-Telos, RDQL, dbXML)

RDF-Graph Ein RDF-Graph ist ein gerichteter Graph, der aus den Statements des RDF-Modells gebildet wird. *Subject* und *Object* bilden die Knoten, das *Predicate* die Kante zwischen ihnen.

RDF-QEL-i Query Exchange Language. Eine auf Datalog basierende Abfragesprache für RDF Metadaten.

Super-Peer Ein spezieller Knoten, aus denen das Super-Peer-Netzwerk aufgebaut wird.

SWEBOK Software Engineering Body of Knowledge, Klassifizierung von Themen aus dem Gebiet der Softwareentwicklung.

Taxonomy The classification of organisms in an ordered system that indicates natural relationships

XML eXtended Markup Language, Auszeichnungssprache für Textdokumente aller Art

Index

- ad hoc, 12
- Advertisement, 30
- advertisements, 31
- Anbindung, 27
- asynchron*, 42

- Backbone, **14**, 48
- Backbones, 19
- Bandbreite, 14
- Beschreibung, 16
- BIND, 29
- BindingPool, 43
- BindingRequestProcessor, 43
- BindingService, **45**
- BindingService, 43
- BINDRequest, 47
- Broadcast, **22**
- BroadcastNetwork, 36, 45
- BroadcastNetwork, 54

- Cache, 14
- Caley, 20
- chains, 31
- childs, 31
- Client, **13**
- ClientPeer, 36
- Clients, 14
- ClientUplink, 43, 46
- Cluster, 4, **14**
- Clustergröße, 14
- Clustering, 14
- Clustering*, 24
- CONSOLE, 29
- Consumer, 4, 5, 10, 50

- Datalog, 7
- Dateinamen, 3
- Datenbank, 6
- Datenkanäle
 - transparente, 10
- Datenstrom, 23
- dbXML, 6
- DC, 24
- DC, 26
- dc, 26
- dc, 8, 17, 26
- dc:language, 17
- dc:subject, 17
- dc:title, 18, 19
- dc:type, 17
- deutsch*, 17
- Dienst, 11
- Dienste, 29
- Discovery-Services, 10
- Dublin Core, 7, 8

- ECDM, 6, **9**, 27
- edu, 8
- EduConditionLiterals, 10
- EduConstant, 30
- EduLiterals, 10
- EduQuery, 10
- EduRules, 10
- Edutella, 3, **5**, 8, 27, 28
- Edutella Query Service*, 27, 31
- endpoints, 31
- Event, 28
- Event, 45
- EventListener, 50
- Extension, 3

- Finite-State-Machine, **29**
- Firewall, 6

- getRemoteAdvertisement(), 39
- getRouter(), 54
- Gnutella, 3, 11
- Granularität, 16

- Graph
 - gerichtete, 8
- Hardware, 10
- Hashtable, 46, 47
- Hashtable, 42
- Hauptklassen, 28
- Hochfrequenzantennen, 14
- Hypercube*, 20
- HyperCup, 19, **20**, 28
- ID, 50
- Implementierung, 47
- Index, 15
 - zentralen, 12
- Indices, 28
- Informationen, 12
- infrastructure, 3
- Infrastruktur, 12
- inhaltlich, 3
- Jena, 6, 10
- Join, **15**
- JXTA, **5**, 10
- kompatibel, 27
- Konfigurationsobjekte, 28
- Korrespondenzen*, 25
- Lastverteilung, 20, 25
- learning, 3
- lecture, 26
- lectures, 26
- Literal, 10
- LOM, 17, 24
- LOM, 26
- lom, 26
- lom, 17, 26
- lom:context, 17
- lomcls:Taxon, 17, 18
- lossy counting*, 23
- main(), 30
- maschinenlesbar, 3
- MAXPEERS, 45
- Mediation, **25**
- Medien, 7
- MEL, 29
- MessageEvent, 50
- Metadaten, 3, 14
- ModuleImplAdvertisement, 41
- ModuleImplAdvertisements, 39
- ModuleSpecID, 39
- MPEG, 3
- Msg, 41
- Namespaces, 7, 17
- Napster, 3
- NetPeerGroup, 30
- Netzwerk
 - unstrukturiert, 12
- Netzwerkinterface, 10
- notify(), 29
- O-Telos, 6
- Object*, 7
- Ontology, 17, **17**
- P, 13
- PADLR, 3
- Parm, 39
- PDF*, 17
- Peer-to-Peer, 3, 11
 - degeneriertes, 14
 - reines, 14
 - unstrukturierten, 14
- PeerConfiguration, 28, 31
- PeerInfo, 28, 30, 42, 50
- PeerServiceRegistry, 28, 30
- pipeMsgEvent(), 49
- Pipes, **10**
- Platzierung, 12
- Predicate, 47
- Predicate*, 7
- Prolog, 6, 7
- Property, 7, 18, 47
- Property, 10
- Provider, 4, **5**, 6, 10, 15, 27
- ProviderConnection, 33
- ProviderCount, 54
- ProviderDescription, 47
- ProviderDescriptions, 51
- ProviderPeer, 31, 32
- ProviderPipeAdvertisement, 47
- ProviderPoolSP, 50

- ProviderService, 49, 50
- Publishing-Services, 10

- QEL, 27, 29
- Queries, 4, 47

- RDF, **3**, 7
- rdf, 8
- RDF Schema, 8
- RDF Syntax, 8
- RDF-Graph, **7**
- RDF-QEL, **6**
- RDF-QEL-i, 4
- RDF-Schema, 8, 16
- rdf:object, 9
- rdf:predicate, 9
- rdf:subject, 9
- rdf:type, 17
- RDFModel, 10
- RDFNode, 10
- RDFReifiedStatements, 10
- rdfs, 8
- RDQLProvider, 48
- Reification, 63
- Reihenfolge, 12
- Rendezvous, 6
- Repositories, 16, 27
- Repository, 6, 48
- RequestProcessor, 33, 48
- Resource, 10
- ROUTE, 29
- Routen, 15
- Routing, 4, 15, 29
- Routing-Index, 16
- RoutingPoolSPP, 50, 51
- RoutingPoolSPSP, 50, 54
- RoutingService, 50

- Schema, **7**
- Schema Index, **17**
- Server, 13
- ServiceAdvertisement, 29, 31
- Services, 4, 28
- services, 31, 33
- Sets of Properties, 18
- Singleton, **28**
- skalieren, 11, 12

- SMConsoleConnection, 33
- Softwaredesign*, 17
- SP, 13
- SP/P, 43
- SP/P-Index, 4, 15
- SP/P-Indices, 50
- SP/SP-Index, 4, 19, 24
- spd, 41
- SpPIndexer, 47
- SpPIndexer, 51
- SpPRouters, 51
- SpPRoutingProcessor, 50
- SpPRoutingProcessors, 54
- SpSpIndexer, 54
- SpSpRouter, 54
- SpSpRoutingProcessor, 50, 54
- SQL, 6
- Statements, 7, **7**
- Status, 45
- sticky sampling*, 23
- Strategien, 4
- Stufen, 6
- Subject*, 7
- Suche, 12, 14
- Suchmaschine, 12
- Sun, 5
- Super-Peer, 3, 12, **13**, 15, 27
- Super-Peer/Peer-Indices, **15**
- SuperPeer, 31
- SuperPeerAdvertisement, 30, 38
- SuperPeerAdvertisements, 39
- SuperPeerDescription, 41, 42, 47
- SuperPeerHEL0, 41, 42
- SuperPeerID, 39
- SuperPeerPipeID, 39
- SWEBOK, 17, **63**
- Swing, 28
- swtont:SWDesBasicConcepts, 18
- swtont:SWDesign, 17

- Tauschbörsen, 14
- Taxonomy, **17**, **64**
- Template, 41
- Textzeichenkette, 3
- Thread, 29
- Topology, 13

Broadcasting, 30
Topology, 28
Triple, 7
TTL, 54

Verfügbarkeit, 12
Views, 26
Vorteile, 12

`wait()`, 29
web, 3
Weiterleiten, 15

XML, 16