Semantisches Routing in dezentralen Netzwerkstrukturen

Diana Tsakadze

Seminar: Semantic Grid

Dozenten: Prof. Steffen Staab
Bernhard Tausch

Inhaltverzeichnis

Abstract

Peer-to-Peer Systeme sind dezentralisierte Netzwerkstrukturen. Sie haben viele Vorteile, wie Robustheit und effizientere Koordination von Ressourcen etc. Dezentralisierung bringt aber auch Probleme mit sich. Die Herausforderung bei P2P Systemen ist es, seine chaotische Natur durch einen strukturierten Ansatz, wie Grid Computing zu ergänzen. Eine weitere Herausforderung ist die Redundanz der Informationen zu vermeiden, die zur Überlastung des Systems führt. Die Realisierung ist durch semantisches Routing möglich. Beispiele dafür sind INGA und Bibster. Die Abfragen werden an selektierten Knoten weitergeleitet, die mehr Kompetenz für diese Abfrage aufweisen als andere.

1. Dezentrale Netzwerksturkturen

1.1 Peer-to-Peer Computing

Die Peer-to-Peer Architektur oder wie man es oft zur Vereinfachung nennt Peer-to-Peer oder abgekürzt P2P ist ein Netzwerkstyp, bei dem jede Komponente des Netzwerkes die gleiche Fähigkeit und Verantwortung hat (eng. Peer=Gleichgestellter). Dadurch unterscheidet es sich von der Client/Server Architektur, in der es ein Server gibt, der einen Dienst anbietet und ein Client, der diesen Dienst nutzt. Im Peer-to-Peer ist jeder Host eines Computernetzwerkes ein Peer und kann gleichzeitig Server und Client sein.

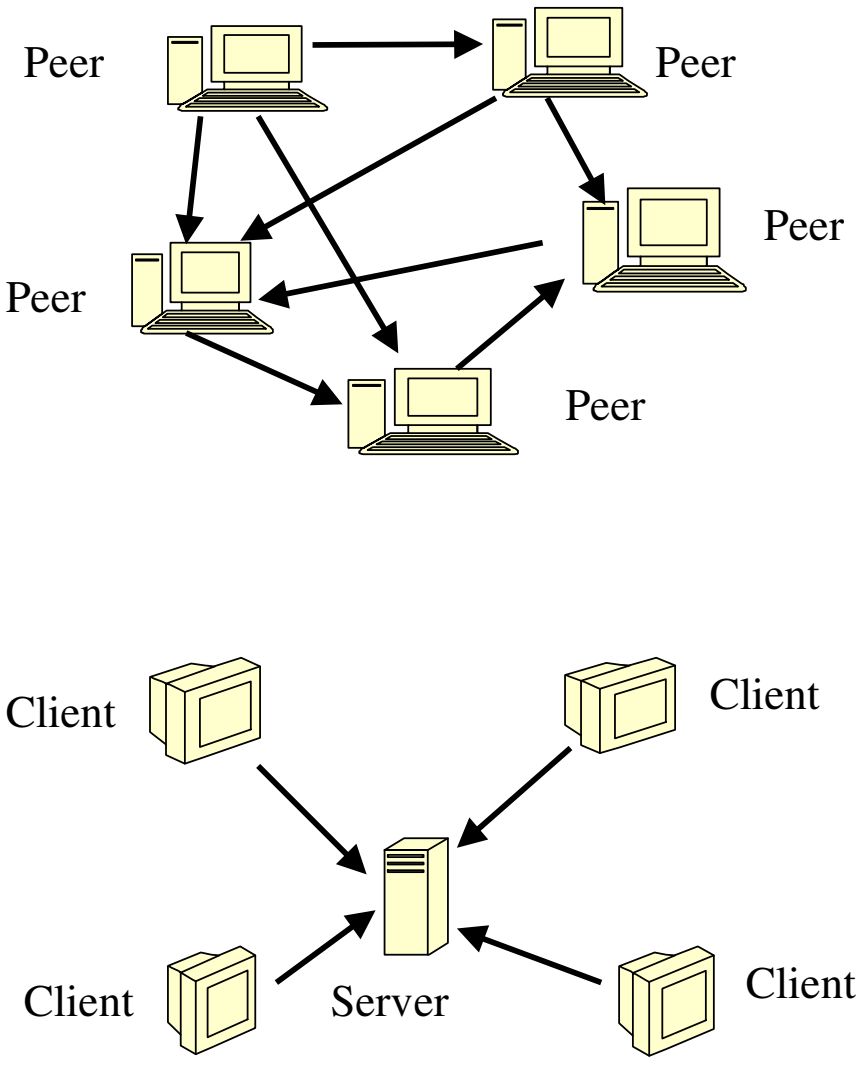

Abb.1 Peer-to-Peer und Client/Server Systeme

In P2P System gibt es keinen zentralen Server, auf dem die Dateien gespeichert sind, sondern sind die Inhalte auf die Computer der gleichberechtigten Benutzer verteilt, also dezentralisiert.

Der Vorteil dieses Prinzips ist, dass zwei Knoten eines Netzwerks direkt miteinander interagieren können, ohne eine zentrale Instanz, die die Kommunikation verzögert oder filtert. Für Rechenperformance und Informationsupdate wird ein Engpaß vermieden. Die Kosten sind geringer, da keine zusätzliche Hardware benötigt wird und Administration und Verwaltung sind einfacher - es werden keine speziellen Kenntnisse über ein Netzwerkbetriebsystem benötigt und jeder Teilnehmer legt selbst fest welche Ressourcen er freigibt, solange er online ist. P2P Systeme sind robust gegen den Aufall einzelner Komponenten und skalierbar in Datenvolumen und in verbundenen Teilen.
Dezentralisierung bringt die oben angeführten Vorteile (Skalierbarkeit, Rechenperformance) doch einige davon nur teilweise. Es fehlt ein kohärentes Schema für die Organisation von Informationsquellen. Die Peers sind unabhängig und koordinieren sich nicht miteinander. Es wird nicht festgehalten, dass ein Knoten eine Anfrage schon bekommen hat und die gleiche Information wird vermehrt verschickt. Das Duplizieren von Information durch das Netzwerk verursacht duplizierte Antworten auf die gleichen Anfrage. Daher wird eine einzelner Server wegen großen Zugriffs überlastet und das skalieren des Netzwerks ist nur eingeschränkt möglich.

1.2 Grid Computing

Grid Computing erlaubt dem Benutzer nicht nur die gemeinsame Nutzung von Dateien, wie beim Peer-to-Peer, sondern auch das Teilen anderer Ressourcen, wie z.B. Rechenleistung etc. Die Grid Architektur kann man konzeptuell in zwei Layer zerlegen: Daten Grid und Rechen Grid.
Das Rechen Grid ist eine virtuelle Maschine und wird durch das Zusammenfügen einer größeren Anzahl unabhängiger Hosts erzeugt. Das Daten Grid ist eine massive zusammengebundene Speicher Infrastruktur, die mit einem Rechen Grid durch High-Speed Netzwerkverbindung verbunden ist. Das Verbindungsnetzwerk auf der Rechengrid Ebene ermöglicht die Kommunikation zwischen den Hosts, während die High-Speed Datenverbindung mit dem Datengrid Layer Datenerreichbarkeit und einen schnellen Informationsaustausch ermöglicht. Eine große Herausforderung ist die chaotische Natur des P2P mit dem strukturierten Ansatz eines Grids zu kombinieren um diese zwei Architekturen zu überbrücken. P2P könnte somit wie ein Data Grid mit breiterer Grid Architektur handeln

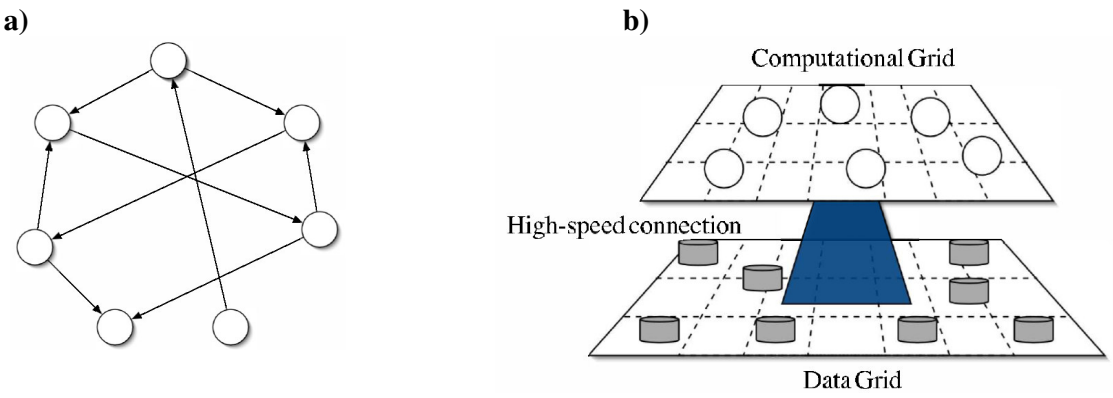

Abb.2 a) Peer-to-Peer System b) Rechen- und Datengrid

und die kollaborative Natur des Grids kann die Funktionalität des P2P ergänzen. Ein Versuch eine solche hybride Architektur zu erschaffen ist Pyragrid. Das Ziel von PG ist es die P2P Funktionalität als eine Infrastruktur sowohl für Rechen- als auch für Datengridlayer zu benutzen

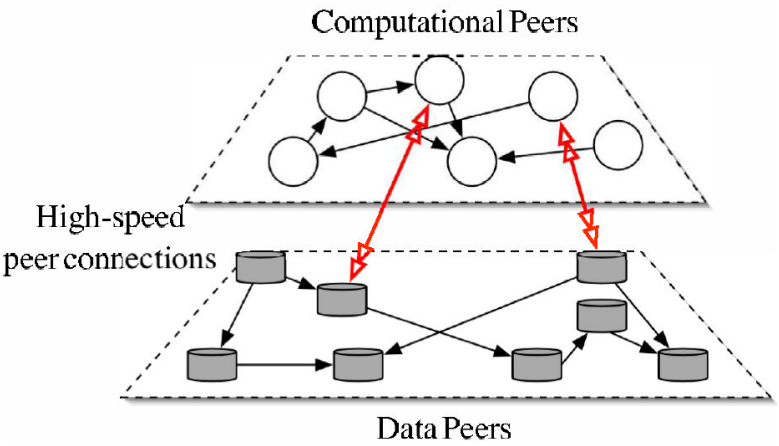

Abb.3 PyraGrid Architektur

2. Semantisches Routing

Probleme des P2P, die in Kapitel 1.1 aufgezählt sind, wie fehlende Koordination zwischen Peers, Duplizieren von Informationen und Überlastung einzelner Knoten durch zu großen Zugriff sind auch Herausforderungen. Durch semantisches Routing kann man ein Query zu dem best geeigneten Knoten schicken und dadurch die Anzahl der Messages halbieren, wie es bei INGA (Interest-based Node Grouping Architecture) der Fall ist. Das Problem ist das Finden des besten Peers für ein bestimmtes Query in dem Netzwerk. Die Peers koordinieren miteinander für ein effizientes Routing der Anfrage mit ausschließlich lokalem Wissen anhand des adaptiven Shortcut-basierten Overlay. Dabei stellt sich die Frage nach der effektivsten Routingstrategie in einem semantischen Peer-to-Peer Netzwerk, wenn nur lokales Wissen über andere Knoten vorhanden ist.
INGA ist von Werken im Bereich der sozialen Netzwerke inspiriert. Die Wichtigkeit von Shortcuts für den Informationsfluss in dem menschliche Netzwerk wurde von Forschern erkannt. Menschen mit lokalem wissen können Nachrichten effektiv steuern und übertragen indem sie in ihrer Umgebung algorithmische Prozeduren ausführen.
Menschen, die eine Antwort auf ihre Frage suchen können zwischen drei Typen von Personen innerhalb eines sozialen Netzwerkes wählen.

Content Provider – er hat früher auf diese Frage mit Erfolg geantwortet.
Recommender – hat sich früher mit einer ähnlichen Frage beschäftigt. Es wird angenommen, dass er einen geeigneten Content Provider kennt.
Bootstrapper - hat ein gutes soziales Netzwerk in mehreren Bereichen.

Diese Annahme wird auch im INGA-Model angewendet. Peers kontrollieren von welchen anderen Peers sie oft erfolgreiche Antworten auf ihre Abfrage bekommen, welche Peers stellen ähnliche Fragen, welche Peers viele Dokumente anbieten oder welche früher wie viele Fragen gestellt haben. Diese Information über die anderen Peers wird lokal in einem Shortcut gespeichert. Jeder Shortcut repräsentiert ein zusätzliches Link (on top of the default network layer) im Peer-to-Peer System.

2.1 Basis Infrastruktur

Das Teilen von Information, suchen und veröffentlichen von Ressourcen sind „Standard“ Peer-to-Peer Funktionalitäten von INGA. Was aber INGA von den anderen Peer-to-Peer Systemen unterscheidet ist, dass hier Peers miteinander kooperieren um einen „guten“ Peer für eine Abfrage zu finden. Jeder Peer speichert ***Qeries x Peers*** Paare in einem **Shortcut Index** nach einer erfolgreichen Abfrage. Er speichert diese „egoistisch“ und behält nur die Shortcuts, die für ihn selbst interessant sind

2.1.1 Default Netzwerk Komponente

Die Hauptaufgaben dieser Komponente sind:
Die Netzwerkverbindung zu den anderen Peers aufrecht zu erhalten – jeder Peer kann eine physikalische Verbindung zu den andren Peers aufbauen.
Eindeutige Peer-Identifizierer (PID) – angewendetes JXTA-Konzept der virtuellen Adressierung ermöglicht es ein Peer zu identifizieren auch wenn dieser eine dynamische IP-Adresse benutzt.

2.1.2 Lokale Datenbanken

Die Hauptaufgaben dieser Komponente sind:
Inhalte des Benutzers veröffentlichen - Der lokale Knoten speichert und veröffentlicht Information. In der Implementation von INGA sind es RDF(S) Statements. Ein Statement beschreibt Daten - z.B. (WWW2005 *InstanceOf* WWWConf) oder konzeptionelle Information - z.B. (WWWConf *subClassO*f Conferences)
Durchführen der Abfrage und liefern der Ergebnisse –

2.1.3 Shortcut Management Komponente

Die Hauptaufgaben dieser Komponente sind:
Lernen und Speichern von Shortcuts – für jede erfolgreiche Abfrage (mindestens mit einem Ergebnis) extrahiert ein Peer die Information über die Peers, die diese Query beantwortet oder weitergeleitet haben und speichert diese in dem lokalen Shortcut Index.
Empfehlen von Top-k Peers für die Abfrage – ein Peer wählt Top-k geeignete Peers zu denen die Abfrage (eigene oder eines remote Peer) weitergeleitet wird. Die Wahl des Peers ist von dem Ähnlichkeits-Maß zwischen der Abfrage und lokal gespeicherten Shortcuts abhängig.

2.1.4 Query Message und Result Message

Die Query Message beinhaltet
Das Query selbst, bootstrapping Fähigkeit des abfragenden Peers, die im bootstrapping Index gespeichert oder aktualisiert wird. Eindeutige ID von Peers, die diese Abfrage schon bekommen haben, um das Duplizieren von Query Messages zu vermeiden. Die QueryID- eine eindeutige Identifikationsnummer für jedes Query, um das wiederholte Antworten auf dasselbe Query zu vermeiden. QueryID's werden durch einen Zufallsgenerator erzeugt.
Die Result Message besteht aus den oben aufgezählten Parametern und beinhaltet zusätzlich die Antwort auf das Query.

2.2 Semantische Ähnlichkeits- Funktion

Der Routing Algorithmus nutzt die Ähnlichkeit zwischen dem Query und dem lokal gespeicherten Query abhängigen Shortcut. Die Ähnlichkeitsmetrik ist oft Bereichspezifisch und hängt von der Semantik der Abfrage und der benutzten Ontologie ab. In der Implementierung von INGA ist es die Ähnlichkeitsmetrik für die Themenhierarchie. Die Themen (Topic), die die gleiche Taxonomie haben können miteinander verglichen werden. Die Ähnlichkeitsfunktion $sim : Q \times \Phi \rightarrow [1;0]$ zwischen einem Query $q \in Q$ und Schortcut $\phi \in \Phi$ ist folgendermaßen definiert:

$$sim(q,\phi) = \begin{cases} e^{-\alpha l} * \dfrac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{wenn } q \neq \phi \\ 1 & \text{Sonst} \end{cases}$$

Wobei ist l die Länge des kürzesten Pfades im Graph zwischen q und ϕ ist, wenn sie SubTopic Beziehung aufweisen. h ist die minimale Tiefe von q oder ϕ im Baum. $\alpha \geq 0, \beta \geq 0$ sind Skalierungsparameter.

2.3 Queryabhängige Shortcuts

2.3.1 Content Provider Shortcuts

Die oben genannte Shortcut Management Komponente verwaltet shortcut Indizes für Shortcuts, die vom Thema der Abfrage abhängig sind. Es unterscheiden sich Content Provider Shortcuts – für Peers, die geeignete Inhalte für ein bestimmtes Query anbieten und Recommender Shortcuts- für die Peers, die sich früher mit einer ähnlichen Frage beschäftigt haben. Das Verwalten von Shortcut Index basiert auf der Prinzip der Interessen- basierten Lokalität, die wie folgt lautet: Wenn ein Content Provider Peer ein Teil der Inhalte hat, die für den lokalen Peer interessant sind, dann ist es sehr wahrscheinlich, dass andere seiner Dokumente ebenfalls interessant sind. Wenn eine Abfrage einen Peer erreicht, wird der Shortcut Index aktualisiert- entweder ein neuer Shortcut zur Liste zugefügt, falls die zugelassene maximale Größe noch nicht erreicht ist oder ein älterer Shortcut damit ersetzt. Ein Shortcut hat die folgende Form ϕ(Topic, PID, QueryHits,C). Das Topic ist das Thema der Abfrage, PID- die eindeutige Identifizierungsnummer des Peers, QueryHits- die Anzahl der Antwort-Statements und C- der Shortcut Typ für Content Provider Shortcut. Die Peers, die mindestens ein Statement zurückliefern, werden zur Content Provider Shortcut Liste hinzugefügt. Auf Abb.4 kann man sich diesen Vorgang verdeutlichen. Peer 2 sendet ein Query Message an die anderen Peers ohne Auswahl - flooding (Shortcut Index Liste ist noch leer). Das Thema für diese Abfrage ist education/uml. Der lokale Peer bekommt von Peer3 und Peer5 23 bzw. 20 Statements zurück. Der lokale Peer erzeugt zwei Content Provider Shortcuts für Peer3 und Peer5. Tabelle 1 zeigt die Struktur der Content Provider Shortcut Liste vom lokalen Peer.

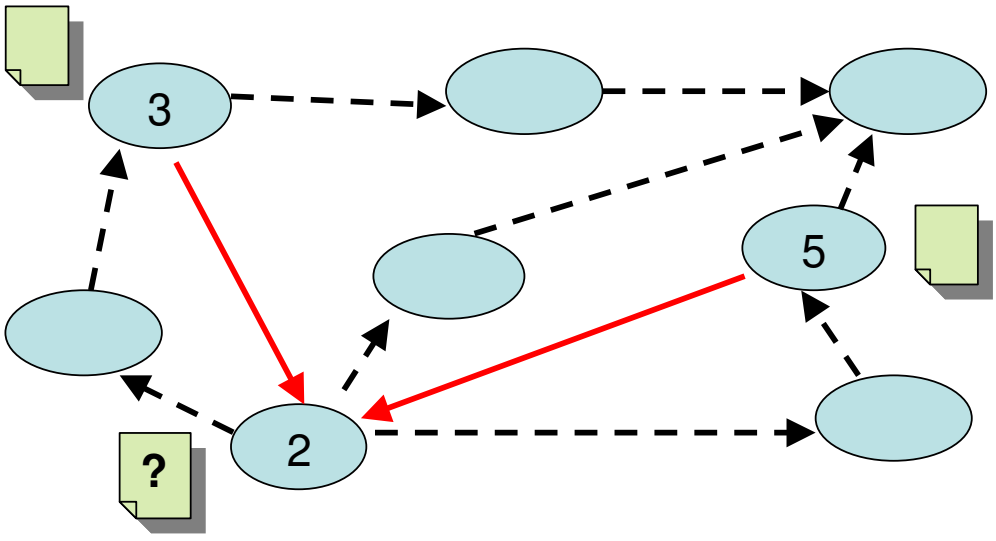

Abb.4 Erzeugung von Content Provider Shortcuts

Topic	PID	QueryHits	Type	Utility
/Education/UML	3	23	C	1
/Education/UML	5	20	C	1

Tabelle 1: Queryabhängige Shortcut Index von Peer2

2.3.2 Recommender Peer Shortcuts

Wenn kein Content Provider Peer bestimmt werden kann, dann wird in der lokalen Recommender Shortcut Liste gesucht und die Abfrage zum ‚besten' Recommender geschickt. Der Recommender Peer ist ein aktiver Peer, mit hoher Überlappung der Anfragen des lokalen Peers. Dadurch wird die Trefferwahrscheinlichkeit im Vergleich zum normalen flooding erhöht. Die Shortcut Management Komponente ist dafür zuständig die Recommender Shortcut Liste aktuell zu halten. Das Entdecken eines Recommender Peers verläuft auf aktive oder passive Art.

Aktives Entdecken des Recommender Peers – wenn die Statements als Anwort auf das Query zurück geliefert werden, schaut der lokale Peer nach dem letzten Peer in dem Message Pfad, also nach dem Peer, der diese Query zu dem Content Provider weitergeleitet hat. Dieser Peer wird als ein Recommender Peer in Shortcut Liste gespeichert.

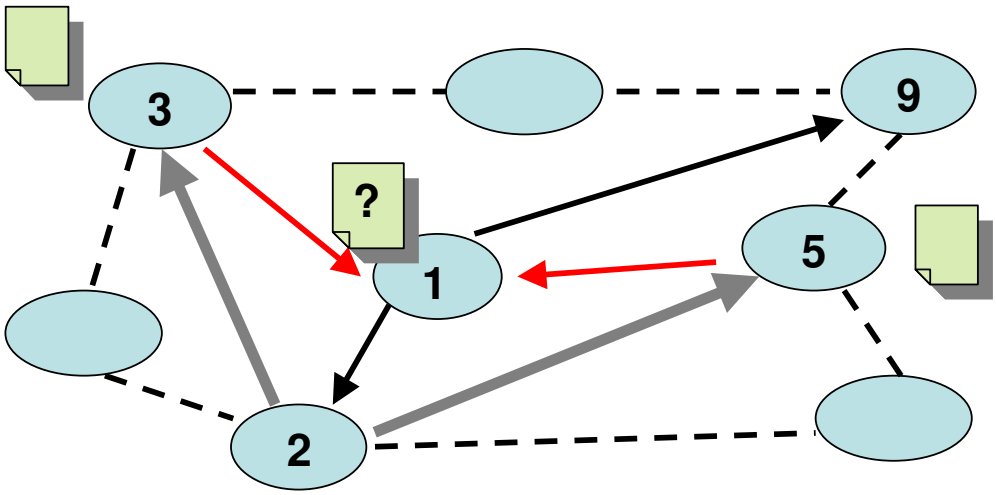

Abb.5. Erzeugung des Recommender Shortcut

Topic	PID	QueryHits	Type	Utility
/Education/UML	3	23	C	1
/Education/UML	5	20	C	1
/Education/UML	2	43	R	1

Tabelle 2: Queryunabhängige Shortcut Index von Peer 1

Abb.5 verdeutlicht diesen Vorgang. Der Peer1 interessiert sich für die Frage Education/UML und flutet diese Abfrage zu den anderen Peers (angenommen seine Shortcut Liste ist noch leer). Peer2 leitet diese Abfrage an Peer3 und Peer 5 weiter, da diese Peers auf gleiche Query schon einmal mit Erfolg geantwortet haben und daher in seinem Shortcut Index stehen. Peer1 bekommt die Statements von Peer3 und Peer5 und speichert diese als Content Provider Peers. Dann schaut er im Message Pfad, welcher Peer sie empfohlen hat und speichert Peer2 als ein Recommender Peer (von Typ R). Die Ergebnis Shortcut Liste von Peer1 sieht man in der Tabelle 2. Die QueryHits für den Recommender Peer sind die Summe aller Statements, die seine empfohlenen Peers geliefert haben.

Passives Lernen durch kontrolliertes Hören – bei jedem eingegangenen Query, wenn dessen Thema das Interesse des lokalen Peers trifft, wird der Abfragende Peer als Recommender zur Liste hinzugefügt. Das Query trifft das Interesse des lokalen Peers, wenn die Ähnlichkeit sim (q,r) zwischen den veröffentlichten Ressoucen r und dem Query q größer ist als eine Ähnlichkeitsschwelle t. Ein hohes t reduziert die Anzahl der Peers, die auf diese Weise entdeckt werden, die gefundenen sind dafür aber Themenspezifischer. PID des Quell-Peers wird dem Message Pfad entzogen. QueryHits wird auf eins gesetzt, da die Information über die Anzahl der Ergebnis Statements für diese Abfrage nicht bekannt ist.

2.3.3 Index Replacement Policy und Indexgröße

Es ist empfehlenswert die Größe des Shortcut Index einzuschränken. Mit der Zeit wächst Shortcut Index. Mit Hilfe einer Replacement Policy ersetzt die Shortcut Management Komponente alte Shortcuts durch Neue. Bei INGA wird eine Nutzwertbasierte Policy angewendet – der Shortcut mit dem niedrigsten Nutzwert wird durch den Neuen ersetzt. Der Nutzwert ist ein statistischer Wert und hängt von zwei anderen Werten ab: *successful usages*- wie oft ein Shortcut erfolgreich genutzt worden ist und *total usages* – wie oft er insgesamt genutzt worden ist. Utility gibt das Maß an, wie hilfreich ein Shortcut bei der Abfrage war.

$$utility(\phi) = \frac{|\, succesfulusages \,|}{|\, totalusages \,|}$$

2.4 Queryunabhängige Shortcuts

Die Auswahl des Bootstrapping Knotens bietet eine höhere Trefferwahrscheinlichkeit im Vergleich zu default Routing Strategie wie z.B. Flooding (Überflutung). Bootstrapping Knoten sind die Peers mit vielen Shortcuts zu vielen remote Peers.
Der Bootstrapping Index wird aktualisiert während der Peer online ist. Jedes Query beinhaltet bootstrapping Information vom Quell-Peer. Wenn die Bootstrapping Fähigkeit von dem neuen Peer höher ist, als schon vorhandene Shortcuts, wird der Alte durch den Neuen ersetzt.
Wenn das lokale Wissen über Content Provider oder Recommender nicht vorhanden ist, sendet ein Peer seine Anfrage zu den „besten“ bootstrapping Knoten und fängt an, das Wissen über Content Provider zu sammeln.
Die Bootstrapping Fähigkeit eines Peers wird berechnet durch die Multiplikation von den Shortcuts, die er erzeugt hat mit Anzahl der Peers, die er kennt:

P.Bo = | *Shortcuts*| x |*Peers*|

P.Bo drückt dabei die Fähigkeit des Peers aus, das breite und diverse Wissen über andere Peers zu sammeln. Tabelle 3 zeigt die Bootstrapping Fähigkeit des Peer1 in dem Queryunabhängigen Shortcut Index von Peer2

PID	Shortcuts	Peers	P.Bo
1	3	3	9

Tabelle 3: Queryunabhängige Shortcut Index des Peer2

2.5 Peer Selektions-Algorithmus

Der Benutzer stellt eine Frage an das Peer-to-Peer Netzwerk. Das Query wird evaluiert, entweder lokal in Abhängigkeit von lokalen Datenbanken, die Antwort ist dann präsent oder durch das Netzwerk weitergeleitet.
Wenn ein Peer ein Query von einem remote Peer empfängt, wird als erstes der update Algorithmus gestartet. Es wird die Information über Quell-Peer von dem Query Message extrahiert und der bootstrapping Index sowie Recommender Index aktualisiert. Im zweiten Schritt versucht der Peer diese Query zu beantworten. Wenn die maximale Anzahl der Sprünge (von einem Peer zu den anderen) noch nicht erreicht ist, wird der Weiterleitungs-Algorithmus ausgeführt. Dadurch werden Top-k Peers (hier k=2) ausgewählt, die für diese Query am kompetentesten erscheinen.
Der Quell-Peer bekommt eine Antwort- die Information von dem Message Pfad wird extrahiert, um die lokalen Content Provider und Recommender Shortcuts zu aktualisieren.

2.5.1 Shortcut Index Aktualisierungs-Algorithmus

Nach dem der Peer ein Query Message bekommt, wird erst:

Zeile 2-5: bootstrapping Information über das Quell-Query extrahiert und der Bootstrappingindex nach der oben genannten Strategie aktualisiert. Wenn der P.Bo des aktuellen Peers höher ist als die Bootstrapping Fähigkeit von denen, die schon im bootstrapping Index gespeichert sind, wird der Peer mit dem niedrigsten P.Bo durch den Neuen ersetzt.

Zeile 6-13: Der Recommender Index wird aktualisiert, wenn das Query das eigene Interesse des lokalen Peers trifft. Das geschieht durch den Vergleich des Querys mit den lokal gespeicherten Content Provider Shortcuts (Zeile 8).

Zeile 15: Die Zahl der Sprünge wird erhöht. Diese Zahl zeigt, wie viele Peers dieses Query schon bekommen haben. Die maximale Anzahl der Sprünge ist festgelegt.

Zeile 16: Die eigene PID wird zum Message Pfad zugefügt.

Require: QueryMessage qm, LocalShortCutIndex 0
1: Peer p=getQueryingPeer(qm)
2: if getBootstrappingInfo(qm)>getBootstrappingInfo(local peer)

```
    then
 3:    /* Add querying peer to bootstrapping index if it outperforms*/
       /* local  value*/
 4:    addBootstrappingPeer(o,p,getBootstrappingInfo(qm))
 5: end if
 6: Query q=getQuery(qm)
 7: for all c   ContentProviderShortcuts do
 8:    if Sim(q,c)>  t          then
 9:    /* Add new recommender shortcut with one query hit if incoming
       query matches ones interests*/
10:    addRecommenderPeer(o,q,p,l)
11:    break
12:  end if
13: end for
14: /* Update Query*/
15:incTTL(qm)
16: addToMessagePath(qm,CurrentPeerID).
```

2.5.2 Weiterleitungs-Algorithmus

Aufgabe dieses Algorithmus ist die k Peer zu bestimmen zu denen die gegebene Abfrage weitergeleitet werden soll. Die Queryabhängige Selektion wird vor der Queryunabhängigen aufgeführt, in der unten aufgezählten Reihenfolge, bis k Peers ausgewählt sind, an die das Query gesendet werden soll.

Queryabhängige Weiterleitungs-Strategie

1. Exakte Treffer - Grenzwert für das Ähnlichkeitsmaß zwischen Query und Queryabhängigen Shortcuts ist gleich 1.

2. Wenn durch den 1. Schritt weniger als k Peers ausgesucht worden sind, wird der Shortcut mit dem höchsten Ähnlichkeitsrang (Grenzwert 0.15) gesucht.

Queryunabhängige Selektion

3. Peers mit höchstem bootstrapping Rang.

4. Default Suchstrategie z.B flooding-basierte Strategie - die Nachbarn bis zu k-Peers hinzugefügt.

Require: QueryMessage *qm*, int *k*
Ensure: getTTL(*qm*) < *maxTTL*
1: **Query** *q* = getQuery(*qm*)
2: /* Start Forwarding */
3: **Queue** *selectedPeers*:=0 ;
4: /*————————— 1 ———————————————————————*/
5: /* Add all matching peers above t */
6: **Queue** *selectedShortcuts*:=0;
7: **for all** qds 2 QueryDependentShortcuts **do**

8: **if** Sim(*qds*,*q*) > t **then**
9: *selectedShortcuts*.append(*qds*)
10: **end if**
11: **end for**
12: *selectedPeers* = *selectedShortcuts*.getDistinctPeers()
13: /* ———————— 2 ———————————————————— */
14: **Queue** *topPeers*:=0; 15: **int** *peersToAppend*= *k-selectedPeers*.size()
16: **if** (*selectedPeers*.size()< *k*) **then**
17: **Queue** *selectedShortcuts*:= 0; 18: /* Add matching peers above t */
19: **for all** qds 2 QueryDependentShortcuts **do**
20: **if** Sim(*qds*,*q*) > t **then**
21: *selectedShortcuts*.append(*qds*)
22: **end if**
23: **end for**
24: rankBySimilarity(*selectedShortcuts*
25: *topPeers*=*selectedShortcuts*.getDistinctPeers(*peersToAppend*)
26: *selectedPeers*.append(*topPeers*)
27: **end if**
28: /* ———————— 3 ———————————————— */
29: **if** (*selectedPeers*.size()< *k*) **then**
30: *peersToAppend*= *k-selectedPeers*.size()
31: *topPeers*=getTopBootstrappingPeers(*peersToAppend*)
32: *selectedPeers*.append(*topPeers*)
33: **end if**
34: /* ———————— 4 ———————————————————— */
35: **if** (*selectedPeers*.size()< *k*) **then**
36: *peersToAppend*= *k-selectedPeers*.size()
37: *topPeers*= getDefaultNetworkPeers(*peersToAppend*)
38: *selectedPeers*.append(*topPeers*)
39: **end if**
40: *selectedPeers*=randomContribution(0.15,*selectedPeers*)
41: **Return** *selectedPeers*.

Der Weiterleitungs-Algorithmus wird aufgerufen, wenn Query die maximale Anzahl an Sprüngen nicht erreicht hat.

3. Bibster - semantik-basierte Peer-to-Peer System

Ein weiteres Beispiel für ein semantisches Peer-to-Peer System ist Bibster – ein Assistent zum Suchen und Managen der bibliographischen Metadaten (z.B. von BibTex-File)
Bibster erlaubt dem Benutzer:
- Import eigener bibliographische Metadaten – Ohne großen Verwaltungsaufwand können die Nutzer sich individuell gepflegte bibliographische Daten gegenseitig zur Verfügung stellen.
- Formulieren der Anfrage – als Beschreibung wie z.B. Autor, Publikations-Typ etc. oder Festlegen des Themas in einer Themenhierarchie.
Das Routing folgt dem Prinzip semantischer Rangordnung und das Duplizieren von Antworten wird durch das Messen der semantischen Ähnlichkeit unterschiedlicher Antworten vermieden

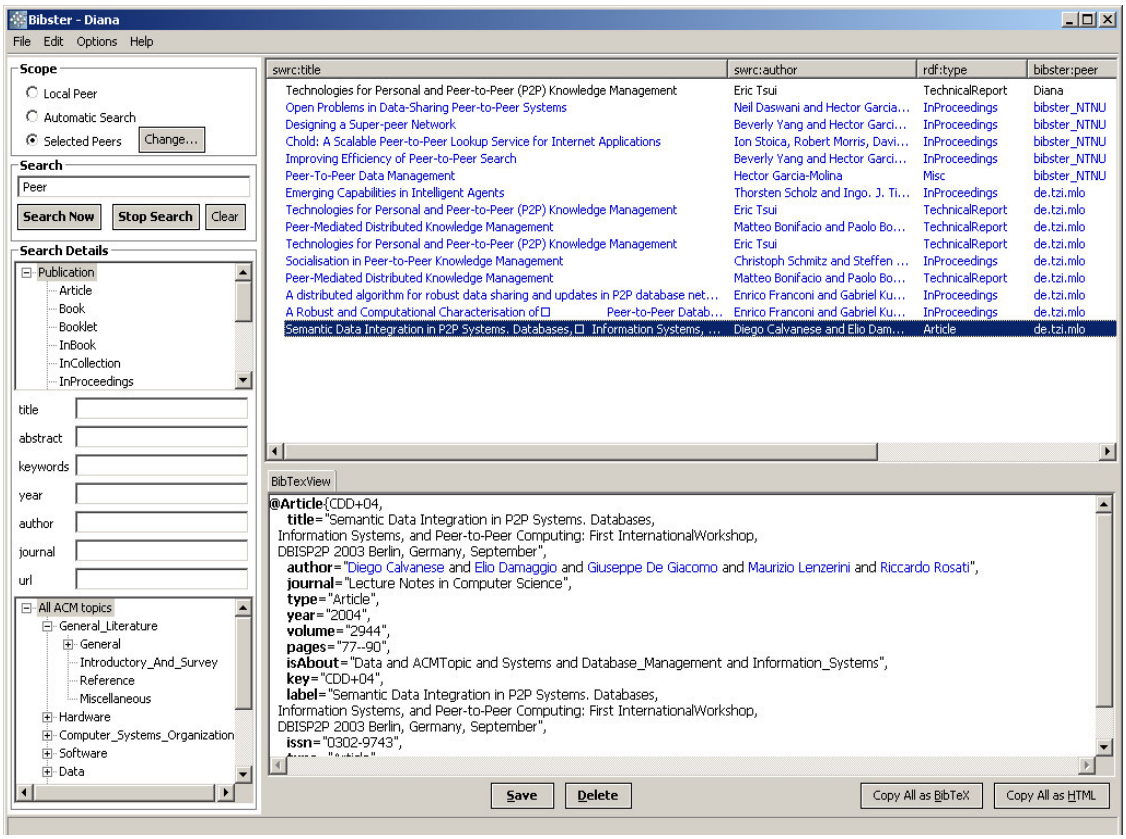

Abb.6 Bibster

4. Zusammenfassung

Im Hauptinteresse der Arbeit sind die Verbesserungsmöglichkeiten der Standart Peer-to-Peer Systeme. Die Überbrückung von P2P und Grid Architektur, die in Pyragrid realisiert wurde ist ein Versuch, bei dem ein P2P wie ein Data Grid mit breiterer Grid Architektur handelt und die kollaborative Natur des Grids die Funktionalität des P2P ergänzt.
Durch das semantische Routing wird im Vergleich zu Standard P2P Netzwerken die Anzahl der Messages halbiert, dadurch Performance des Systems verbessert und die Kosten gesenkt.

Literaturverzeichnis

1. Löser A., Tempich C., Staab S., Nejdl W.:Learning and Recommending Shortcuts in Semantic Peer-to-Peer Networks.
2. Viglas S. D.: Pyragrid: Bringing Peer-to-Peer and Grid Architectures Together. Sixth Thematic Workshop of the EU Netzwork of Excellence DELOS on Digital Library
3. Haase P., Broekstra J., Ehrig M., Menken M., Mika P., Plechawski M., Pyszlak P., Schnizler B., Siebes R., Staab S., Tempich C.: Bibster – A Semantics-Based Bibliogaphic Peer-to-Peer System