Università degli studi
di Modena e Reggio Emilia

Facoltà di Ingegneria
Corso di Laurea Specialistica
in Ingegneria Informatica

Approssimazione semantica per routing di interrogazioni in un PDMS

Simona Sassatelli

Tesi di Laurea

Relatore:
Dott. Federica Mandreoli

Correlatore:
Ing. Riccardo Martoglia

Anno Accademico 2004/2005

Ringraziamenti

Desidero ringraziare la Dott. Federica Mandreoli e l'Ing. Riccardo Martoglia per l'infinita disponibilità, la costante assistenza, la preziosa collaborazione e la fiducia che mi hanno offerto in questi mesi di lavoro.
Ringrazio i miei genitori, per avermi sempre sostenuto ed appoggiato.
Grazie infine ai miei amici e compagni di studi, che tanto mi hanno dato in questi anni.

Parole chiave

PDMS
Query routing
Schema matching
Approssimazione semantica
Routing index

Indice

Introduzione **1**

I Stato dell'arte **5**

1 Introduzione al problema **7**

2 Sistemi Peer-to-Peer (P2P) **11**
2.1 Classificazione 12
2.2 Applicazioni 14

3 Peer Data Management System (PDMS) **17**
3.1 Caratteristiche 18
3.2 Panoramica sui sistemi esistenti 19
3.2.1 Piazza 19
3.2.2 Hyperion 21
3.2.3 Edutella 24
3.2.4 Stanford Peers 27
3.3 Soluzioni progettuali 29
3.3.1 Problematiche 29
3.3.2 Rappesentazione dei dati 30
3.3.3 Connessione dei dati 31
3.3.4 Risoluzione delle interrogazioni 40
3.3.5 Ricerca dei dati 49

II Tecniche di approssimazione semantica per query routing in un PDMS **53**

4 Migrazione di XML S^3MART verso un ambiente P2P **55**
4.1 Il sistema XML S^3MART 56
4.1.1 Schema matching 57
4.1.2 Automatic query rewriting 66

4.2 Architettura P2P di riferimento 68
4.3 Modifiche al sistema XML S^3MART 71

5 Routing di interrogazioni in un PDMS 79
5.1 Routing by mapping 83
5.1.1 Modello matematico di riferimento 84
5.1.2 Indici di routing semantici 86
5.1.3 Creazione degli indici di routing semantici 90
5.2 Gestione dell'ambiente P2P 105
5.2.1 Protocollo di interazione tra i peer 105
5.2.2 Algoritmo del modulo P2P 109
5.3 Evoluzioni future 118
5.3.1 Utilizzo degli indici di routing semantici 119
5.3.2 Informazioni quantitative sui dati 121

6 Prove sperimentali e risultati ottenuti 127
6.1 Ambiente di simulazione 129
6.2 Scenario per le prove sperimentali 130
6.3 Risultati sperimentali 133
6.3.1 Confrontabilità dei mapping 133
6.3.2 Efficacia degli indici di routing semantici 135

Conclusioni e sviluppi futuri 141

Elenco delle figure

3.1 Piazza: architettura di esempio per un PDMS. 20
3.2 Hyperion: architettura del sistema. 22
3.3 Edutella: architettura di JXTA. 25
3.4 Edutella: schema del processo di risoluzione delle query. 26
3.5 SON: esempio. 27
3.6 SON: processo di creazione e utilizzo. 28
3.7 Hyperion: esemio di due schemi. 34
3.8 Hyperion: esempio di mapping. 35
3.9 Xyleme: esempio di due DTD concrete e della corrispondente DTD astratta. 36
3.10 XML S^3MART: schema matching. 37
3.11 Piazza: rappresentazione grafica per query e mapping. 42
3.12 Xyleme: struttura di una query. 45
3.13 Xyleme: esempio di riscrittura. 46
3.14 Edutella: procedimento di risoluzione delle query. 48

4.1 XML S^3MART: struttura concettuale ad albero di un documento. 58
4.2 XML S^3MART: algoritmo per la matching computation. 61
4.3 XML S^3MART: esempio di grafo etichettato diretto. 62
4.4 XML S^3MART: costruzione del Pairwise Connectivity Graph. . . 63
4.5 XML S^3MART: costruzione dell'Induced Propagation Graph. . . 65
4.6 Schema del progetto WISDOM. 69

5.1 Propagazione delle interrogazioni in un sistema P2P. 80
5.2 Meccanismo di query routing. 81
5.3 SRI: esempio. 87
5.4 SRI: esempio. 89
5.5 SRI: operazione di aggregazione. 92
5.6 SRI: operazione di composizione. 93
5.7 SRI: implementazione dell'operazione di aggregazione. 96
5.8 SRI: implementazione dell'operazione di composizione. 100
5.9 Protocollo di interazione tra i peer. 106
5.10 Schema delle strutture dati mantenute da ogni peer. 110

5.11 Esempio di nuova connessione (I) 114
5.12 Esempio di nuova connessione (II) 115
5.13 Esempio di nuova connessione (III) 116
5.14 Esempio di nuova connessione (IV) 117
5.15 Esempio di nuova connessione (V) 118
5.16 Esempio di nuova connessione (VI) 119
5.17 Esempio di nuova connessione (VII) 120
5.18 QRI: esempio. 121
5.19 QRI: creazione di una nuova connessione. 122
5.20 QRI: aggiornamento dei vicini. 123
5.21 QRI: operazione di aggregazione. 124

6.1 Schema del processo simulativo in una rete fittizia. 129
6.2 Esempi di schemi di argomento sportivo. 131
6.3 Esempi di schemi di argomento musicale. 131
6.4 Esempi di schemi di argomento musicale. 132
6.5 Esempi di schemi di argomento bibliografico. 132
6.6 Scenario per le prove sperimentali. 133
6.7 Scenario per la prima tipologia di prove sperimentali. 134
6.8 Scenario iniziale per la seconda tipologia di esperimenti. 136
6.9 Scenario modificato per la seconda tipologia di esperimenti. . . . 138

Introduzione

La grande quantità di informazioni e servizi che sono oggi disponibili sul Web rende estremamente complesso, per i suoi utenti, riuscire ad eseguire in tale ambito ricerche efficaci ed efficienti. Uno dei fattori che maggiormente ostacolano la possibilità di ottenere risposte rilevanti è costituito dal fatto che i dati, in questo contesto, spesso mancano di una rappresentazione semantica in grado di farne comprendere il significato alle macchine che devono processarli. Gli strumenti e i linguaggi classicamente impiegati nel campo dei database, che consentono di rappresentare i dati in modo da esplicitarne la semantica e formulare su di essi interrogazioni significative ed espressive, non sono infatti direttamente applicabili alla realtà del World Wide Web (WWW), poiché si configura come un ambiente estremamente vasto, eterogeneo e dinamico, all'interno del quale l'insieme dei documenti disponibili è in continua evoluzione.

Le caratteristiche di flessibilità, evoluzione rapida e decentralizzazione sono ancora più evidenti all'interno dei sistemi *Peer-to-Peer (P2P)*, per i quali si è recentemente assistito ad una notevole diffusione nell'ambito della rete Internet. Questi sistemi sono caratterizzati dal fatto di essere costituiti da un vasto numero di unità di elaborazione alla pari (i *peer*), indipendenti e capaci di organizzarsi autonomamente per condividere grandi quantità di risorse. Le reti e le applicazioni P2P sono quindi in grado di consentire ai partecipanti l'accesso a molti contenuti, rimanendo tuttavia piuttosto carenti per quanto riguarda le capacità semantiche di gestione e recupero dei contenuti stessi.

In questo contesto si inseriscono i *Peer Data Management System (PDMS)*, che si configurano come una recente proposta per tentare di attuare una sintesi tra le caratteristiche di flessibilità tipiche dei sistemi P2P e l'espressività semantica propria delle applicazioni relative alle basi di dati. Questi sistemi si prefiggono di implementare un'architettura decentralizzata e facilmente estensibile per la gestione dei dati, all'interno della quale ad ogni utente sia concesso,

in ogni istante, di agire con la massima libertà sui propri contenuti e allo stesso tempo di accedere a contenuti che risiedono su altri peer. L'obiettivo è infatti che i partecipanti ad un PDMS siano entità autonome ed indipendenti, che decidono liberamente di condividere i propri dati e si trovano di conseguenza a dover collaborare nel risolvere le richieste a loro sottoposte dagli utenti. Tuttavia per realizzare questo fine ultimo c'è ancora molto da indagare a livello di ricerca.

Nell'ambito di questa Tesi di Laurea sono stati affrontati diversi aspetti riguardanti il processo di routing delle interrogazioni in un PDMS e, partendo da un'approfondita analisi delle numerose problematiche ad esso relative, sono state proposte tecniche innovative che consentono di operare sulla base di informazioni semantiche. In particolare, sono stati ideati algoritmi e strutture dati che permettono di raccogliere queste informazioni e mantenerle aggiornate, in modo che esse possano essere adeguatamente impiegate nelle operazioni di ricerca dei contenuti.

La prima problematica affrontata in questo contesto riguarda la gestione dell'eterogeneità che caratterizza i partecipanti ad un PDMS. I singoli peer, essendo entità indipendenti, possono infatti adottare modalità differenti per rappresentare i propri dati ed occorre quindi che tra di essi vengano stabiliti degli opportuni collegamenti semantici (i *mapping*) in grado di esprimerne le corrispondenze.

L'approccio proposto per creare questi mapping si basa sul concetto di "approssimazione semantica" ed è stato realizzato a partire da *XML* S^3*MART*, un sistema sviluppato presso l'Università di Modena e Reggio Emilia per consentire la riscrittura di interrogazioni su insiemi di documenti eterogenei. Non essendo originariamente applicabile in un contesto P2P, XML S^3MART è stato sottoposto durante la realizzazione della Tesi ad un adeguato processo di reingegnerizzazione, nel corso del quale sono state eseguite le modifiche alla sua struttura e al suo funzionamento necessarie a renderlo idoneo al nuovo ambiente. Grazie ai cambiamenti realizzati, il sistema può dunque essere impiegato da ogni entità di un PDMS per stabilire come i suoi concetti vengono approssimati dalle entità ad essa connesse (i suoi "vicini") tramite l'attribuzione di un punteggio numerico che ne esprime la vicinanza semantica.

Dallo studio delle problematiche relative al routing delle query in un PDMS è inoltre emerso che in tale ambito non sempre risulta conveniente propagare un'interrogazione verso altri nodi: coinvolgere entità che non trattano informazioni collegate a quelle richieste risulta infatti inefficiente sia perché in tal modo

si rischia di sovraccaricare il richiedente con dati poco significativi, sia perché tutto ciò comporta un'inutile e gravosa crescita del traffico di rete.

Per risolvere il problema sono state ideate e realizzate delle apposite strutture dati, dette *indici di routing semantici*, all'interno delle quali ogni peer può memorizzare un punteggio numerico che esprime un'informazione riassuntiva di come i suoi contenuti vengono approssimati semanticamente all'interno delle sottoreti che hanno origine dai nodi vicini. Grazie a queste strutture dati, ogni entità di un PDMS può quindi selezionare, per ogni interrogazione pervenuta, quali tra i suoi vicini sono potenzialmente in grado di risolverla meglio e scegliere di propagarla solo a loro, realizzando in definitiva un meccanismo che può essere definito di *"routing by mapping"*.

Nel progettare gli indici di routing semantici è stato necessario condurre uno studio sulle modalità da adottare per ottenere le informazioni riassuntive che devono essere mantenute al loro interno ed è a tal fine stato introdotto un modello matematico che si rifà alla logica fuzzy. É stato inoltre definito un protocollo che consente ai vari peer di cooperare nei processi di creazione e aggiornamento di tali indici, le cui funzionalità sono state implementate in uno specifico modulo software da utilizzare congiuntamente alla nuova versione di XML S^3MART.

Infine, il sistema realizzato è stato inserito all'interno di un ambiente di simulazione, appositamente implementato per rappresentare le caratteristiche di un PDMS, tramite il quale è stato possibile verificarne il comportamento in ambito distribuito e valutarne l'efficacia.

La presente Tesi è strutturalmente organizzata in sei capitoli, suddivisi in due parti, la **Prima** delle quali riporta gli argomenti studiati per sviluppare il lavoro di ricerca e implementazione presentato nella **Seconda Parte**.

Nel **Capitolo 1** vengono illustrate le problematiche relative alla gestione dei dati nell'ambito dei sistemi Peer-to-Peer, che sono a loro volta presentati nel **Capitolo 2**. Il **Capitolo 3** descrive invece i PDMS, che rappresentano l'ambiente di riferimento per le tematiche affrontate nella Tesi.

Il **Capitolo 4** inizia la seconda parte della trattazione e descrive il processo di migrazione verso l'ambiente P2P a cui, durante la realizzazione della tesi, è stato sottoposto il sistema XML S^3MART. In questo capitolo vengono per prima cosa presentate la versione originaria di XML S^3MART e l'ambiente P2P di destinazione, in modo da poter adeguatamente introdurre di seguito le modifiche che a tale scopo è stato necessario eseguire.

Nel **Capitolo 5** viene affrontato nello specifico il problema del routing di interrogazioni in un PDMS e sono presentati gli indici di routing semantici che in tale contesto si vogliono impiegare. Nel descrivere queste strutture viene anche introdotto il modello matematico fuzzy a cui si fa riferimento per interpretare le informazioni riassuntive che vengono mantenute al loro interno. Nel seguito dello stesso capitolo viene illustrato il protocollo progettato per consentire ai peer di cooperare nei processi di creazione e aggiornamento degli indici di routing semantici e l'algoritmo del modulo che implementa questo protocollo. Il capitolo si conclude con alcune considerazioni relative alle tematiche presentate e potenziali evoluzioni future.

Il **Capitolo 6** presenta infine le prove sperimentali eseguite sul sistema realizzato e descrive l'ambiente di simulazione che è stato a questo scopo implementato, lo scenario impiegato per eseguire le prove e i risultati ottenuti.

Parte I

Stato dell'arte

Capitolo 1

Introduzione al problema

La grande quantità di informazioni e servizi che sono oggi disponibili sul Web rende estremamente complesso riuscire ad eseguire in tale ambito ricerche efficaci ed efficienti. Uno dei fattori che maggiormente ostacolano la possibilità di ottenere risposte interessanti è costituito dal fatto che i dati spesso mancano di una rappresentazione semantica che sia in grado di farne comprendere il significato alle macchine che devono processarli.

Il mondo delle **basi di dati** in realtà offre strumenti che consentono di rappresentare i dati in modo da esplicitare la loro semantica e linguaggi che permettono la formulazione di interrogazioni significative ed espressive, ma tali tecniche presentano il grande svantaggio di necessitare di una onerosa attività iniziale di progettazione per definire uno schema comune che consenta la condivisione delle informazioni e che risulta difficilmente estendibile ad eventuali nuovi concetti.

Questi strumenti non sono quindi compatibili con quella che è la realtà del **World Wide Web (WWW)**, dal momento che si tratta di un ambiente non solo vastissimo ma anche assolutamente aperto, eterogeneo e dinamico, nel quale la collezione dei documenti disponibili è in continua evoluzione.

Il problema della mancanza di una caratterizzazione semantica (che sia comprensibile da una macchina) dei dati del Web è stato parzialmente fronteggiato con l'introduzione del linguaggio **XML** [7] nel quale il significato delle informazioni rappresentate viene manifestato attraverso l'impiego di specifici tag; tuttavia anche in questo caso emergono problematiche legate al fatto che risulta necessario disporre di un ambiente di riferimento comune in cui i diversi concetti devono essere definiti.

Più recentemente si è cercato di affrontare la questione nell'ambito del cosid-

detto **Sematic Web** [4], che si pone appunto lo scopo di rappresentare i dati in un formato che includa informazioni semantiche, in modo che esse possano essere sfruttate per eseguire interrogazioni complesse. L'idea alla base del Semantic Web consiste nel considerare il Web come una base di conoscenza, in cui è possibile rappresentare i concetti, il loro significato (collegandoli ad una particolare ontologia) e le loro relazioni impiegando appositi linguaggi come ad esempio RDF [10], RDF Schema [9] e OWL [8]. Tuttavia il problema di riuscire creare un punto di incontro tra le tante diverse ontologie esistenti, specialmente su larga scala, permane.

Una soluzione parziale del problema è fornita dai **sistemi per l'integrazione dei dati** (come ad esempio [22], [40] e [3]): si tratta di sistemi costituiti da un grande numero di sorgenti di dati indipendenti ed eterogenee nei quali le interrogazioni possono essere risolte grazie al fatto che l'amministratore del sistema definisce uno schema globale comune rappresentativo del dominio di interesse e specifica le corrispondenze semantiche esistenti tra tale schema e le sorgenti dei dati. In questo modo le interrogazioni, che vengono in un primo momento formulate sullo schema globale, possono essere riscritte secondo le relazioni locali delle sorgenti dei dati, le quali si preoccupano infine di fornire le risposte. Si tratta dunque di sistemi potenzialmente ampi e flessibili in cui vengono rese disponibili informazioni semantiche, ma purtroppo l'impiego di uno schema mediato risulta essere un punto debole: esso deve essere infatti progettato adeguatamente da un'entità centrale, che si deve anche occupare di effettuare tutte le necessarie successive modifiche, limitando in questo modo le possibilità di cambiamento ed evoluzione delle sorgenti locali.

Le limitazioni di questo approccio risultano ancora più evidenti se si considera il crescente interesse che si è diffuso negli ultimi anni nell'ambito della rete Internet per una particolare categoria di sistemi distribuiti detti **Peer-to-Peer (P2P)** [57]. Gli esempi più noti sono sistemi impiegati per la condivisione di file musicali e digitali, come Napster [47], Gnutella [24] e Freenet [12], o per la computazione distribuita come seti@home [53] [55] e genome@home [23] [39], o ancora per la comunicazione in tempo reale come ICQ [32] . In tutti questi sistemi i partecipanti sono uguali ed autonomi (possono cioè eseguire le stesse funzioni e unirsi o lasciare la rete in ogni istante) e possono condividere in maniera diretta risorse di diversa natura (informazioni, capacità computazionale o di memorizzazione), senza necessitare di alcun tipo di controllo centralizzato. Essi

sono dunque dotati della capacità di funzionare e organizzarsi senza la necessità di alcun amministratore globale, in presenza di una popolazione di nodi estremamente variabile e soggetta a potenziali guasti: sono quindi intrinsecamente scalabili, robusti e facilitano la condivisione delle risorse in forma anonima. Tuttavia questi sistemi sono carenti per quanto riguarda le capacità di gestione dei dati che sono tipiche dei database, come ad esempio la capacità di amministrare e trasformare dati complessi, supportare le relazioni tra di essi ed esprimerne la semantica, con conseguenze negative per ciò riguarda le attività di collocamento e recupero dei dati. Gli attuali sistemi P2P nella maggior parte dei casi sono in grado di gestire solamente richieste semplici e non particolarmente precise, in cui l'oggetto cercato può essere trovato solamente se si è a conoscenza del suo identificativo e che non prevedono dunque la possibilità di rappresentare informazioni semantiche.

Affiora quindi la necessità di disporre di una nuova classe di sistemi per la condivisione dei dati che permettano di esprimere indicazioni semantiche e di eseguire interrogazioni complesse, ma nei quali le operazioni per la gestione e la condivisione possano avvenire in modo completamente decentralizzato e distribuito. Ad ognuno dei partecipanti al sistema dovrebbe infatti essere consentito di modificare i propri dati o inserirne di nuovi ed eventualmente collegarli a quelli già presenti in modo del tutto autonomo.

Una soluzione proposta è costituita dai cosiddetti **Peer Data Management System (PDMS)** che, implementati su un'architettura di tipo Peer-to-Peer, si propongono come sistemi che tentano di attuare una sintesi tra le caratteristiche di estensibilità e flessibilità tipiche del WWW e dei sistemi P2P e l'espressività semantica propria delle applicazioni appartenenti al mondo delle basi di dati e del Semantic Web.

Capitolo 2

Sistemi Peer-to-Peer (P2P)

Con il termine Peer-to-Peer (P2P) ci si riferisce ad una particolare categoria di sistemi distribuiti la cui caratteristica principale è di essere composti da unità di elaborazione alla pari. I partecipanti a tali sistemi sono infatti nodi indipendenti connessi in rete (detti *peer*) con identiche capacità e responsabilità, che sono in grado di organizzarsi autonomamente e senza necessitare di alcun tipo di autorità centrale per condividere risorse di varia natura (ad esempio contenuti, cicli di CPU, memoria o banda).

Si tratta quindi di architetture estremamente decentralizzate e flessibili, i cui componenti non hanno ruoli predefiniti e sono invece liberi di agire e intraprendere le loro attività secondo le singole necessità. Dal momento che non esiste poi alcun tipo di gestore centrale, la visione che ogni peer ha del sistema non è completa, ma piuttosto limitata alla serie di nodi a cui risulta collegato e che rappresentano i suoi *vicini*.

L'aspetto più attraente dei sistemi P2P è la loro capacità di consentire ai partecipanti l'accesso ad una grande quantità di risorse, minimizzando allo stesso tempo i costi necessari per gestirle grazie alla possibilità di **suddividerli e bilanciare il carico** tra i vari peer.

Essi si presentano poi come sistemi fortemente **flessibili ed estensibili**, in grado di supportare popolazioni estremamente variabili di nodi, che possono decidere autonomamente di connettersi o abbandonare il sistema in ogni istante.

Per questo motivo sono anche intrinsecamente **robusti e tolleranti ai guasti**, poiché possono adattarsi agli eventuali fallimenti continuando a garantire

prestazioni accettabili e un certo grado di connessione tra i nodi, favorendo in tal modo la disponibilità e l'integrità delle risorse.

A causa della loro natura fortemente distribuita sono anche facilmente **scalabili** e in grado di garantire un certo **anonimato ai partecipanti** al fine di consentire una condivisione delle risorse che sia il più possibile libera e non controllata.

Nonostante le enormi potenzialità che offrono, le architetture P2P sono però caratterizzate da problematiche progettuali e implementative di non semplice soluzione.

Inevitabili sono infatti le difficoltà legate all'ambito della **sicurezza**, soprattutto a causa della loro natura aperta ed anonima in cui non risulta banale verificare l'autenticità delle risorse e le intenzioni dei partecipanti.

La maggior parte dei sistemi P2P esistenti è inoltre carente per quanto riguarda le possibilità di **gestione e amministrazione delle risorse** e offre in quest'ambito solo servizi molto elementari.

La questione che probabilmente riveste maggiore importanza nel decretare il successo di un'applicazione P2P è poi la possibilità di disporre di tecniche efficienti per la **ricerca e il recupero delle risorse** e anche questo aspetto non è certo facilitato dalla loro architettura fortemente decentralizzata e distribuita.

2.1 Classificazione

Nel costituire un sistema P2P i vari nodi si organizzano in una rete che viene creata al di sopra della sottostante rete fisica e che viene definita *overlay network*. Sebbene le architetture P2P nella loro forma più pura siano concepite come completamente decentralizzate, in realtà i sistemi esistenti sono caratterizzati da gradi di centralizzazione differenti per la loro overlay network ed è quindi possibile distinguere tra:

- *Architetture puramente decentralizzate.* Si tratta di quei sistemi in cui, come accade in Gnutella [24] o Freenet [12], non è presente alcun tipo di gestore centrale, tutti i peer possono quindi intraprendere esattamente le stesse funzionalità assumendo diversi ruoli a seconda delle loro necessità.

- *Architetture ibride.* In questi sistemi, di cui Napster [47] è un esempio, viene impiegato un server centrale che funge da indice e a cui tutti i peer si rivolgono per localizzare le risorse, solo la successiva operazione di recupero dei contenuti è effettuata in modo distribuito e diretto tra i singoli peer.

- *Architetture parzialmente centralizzate.* Sono quei sistemi, come ad esempio KaZaA [36], che presentano caratteristiche appartenenti ad entrambe le tipologie precedenti. Essi prevedono infatti la presenza di alcuni nodi che sono più importanti degli altri (i cosiddetti supernodi o superpeer) e che agiscono come indici centrali locali per i file contenuti in un certo sottoinsieme dei peer. I singoli peer presentano quindi le interrogazioni solo al loro supernodo da cui ricevono i risultati, analogamente a quanto avviene nelle architetture ibride. Tuttavia i vari superpeer sono considerati sullo stesso livello e collegati tra loro come accade nelle architetture pure.

Un altro criterio impiegato per classificare i sistemi P2P è la struttura della loro overlay network e si può quindi distinguere tra:

- *Architetture non strutturate.* La rete viene creata in modo non deterministico e i vari file vengono posizionati all'interno del sistema in maniera del tutto indipendente dalla sua topologia. In questi sistemi è dunque necessario implementare metodi appositi che consentano di localizzare i contenuti. Nella maggior parte dei casi vengono impiegati approcci semplici, che prevedono di esplorare la rete inondandola di query fino a quando non viene raggiunto il nodo desiderato. Sebbene tutto ciò sia limitante per la scalabilità, questo tipo di architettura è la più adatta nel caso di sistemi che devono gestire popolazioni di nodi estremamente variabili, come accade ad esempio in Napster [47], Publius [59], Gnutella [24], KaZaA [36] ed Edutella [49]. Inoltre il meccanismo di localizzazione dei contenuti implementato può essere tale da consentire ricerche per parole chiave.

- *Architetture strutturate.* In questo caso la rete viene creata seguendo regole adeguatamente specificate e i file sono posizionati in locazioni definite in modo preciso. Per mantenere le corrispondenze tra ogni risorsa (ad esempio l'identificativo di un file) e la sua locazione (ad esempio l'indirizzo di un peer) si ricorre all'impiego di tabelle di routing distribuite, che sono

utilizzate per dirigere le interrogazioni verso i nodi opportuni. La ricerca diventa in questo modo particolarmente efficiente e scalabile, ma possibile solo per le cosiddette query di matching esatto (ovvero quelle per cui è noto l'esatto identificatore della risorsa cercata). Questa tipologia di architettura è poco applicabile ai sistemi caratterizzati da popolazioni di nodi molto variabili poiché la struttura cha la caratterizza è complessa da mantenere. Esempi tipici di questa categoria sono quei sistemi che ricorrono all'impiego delle tabelle di hashing distribuite come Chord [54], CAN [52], PAST [16] e Tapestry [62].

2.2 Applicazioni

Il recente successo delle architetture P2P ha causato la loro diffusione in diversi ambiti, determinandone l'impiego in una grande varietà di applicazioni differenti.

Esistono infatti sistemi P2P impiegati come infrastrutture per facilitare la **comunicazione** e la collaborazione diretta e in tempo reale tra i vari peer (ad esempio ICQ [32] e Jabber [33]), sistemi con lo scopo di realizzare un'efficiente **computazione distribuita** sfruttando la potenza computazionale dei partecipanti (ad esempio seti@home [53] [55] e genome@home [23] [39]) e ancora sistemi che forniscono supporto per diversi **servizi Internet**.

L'applicazione principale per il mondo P2P è però la **distribuzione di contenuti**: si tratta cioè di quei sistemi che nascono con lo scopo di realizzare infrastrutture per permettere la condivisione tra i partecipanti di file e dati di varia natura. Esistono numerosi esempi anche molto famosi di questi sistemi, basti pensare a Napster [47], Gnutella [24], KaZaA [36], Freenet [12], MojoNation [46], Oceanstore [38]. Alcuni sono molto semplici e offrono solamente i servizi basilari per consentire la condivisione dei file, altri sono invece più sofisticati e costruiscono ambienti più complessi, all'interno dei quali è possibile pubblicare, organizzare, aggiornare e ricercare i contenuti in modo più sicuro ed efficiente. Tuttavia la maggior parte dei sistemi esistenti in quest'ambito è abbastanza carente per quanto concerne le capacità di gestione e amministrazione dei dati e non consente per esempio di esprimerne la semantica o supportare le relazioni tra di essi, con conseguenze negative per ciò che riguarda le attività di localizzazione e recupero [46].

Si pone in questo contesto una recente area di ricerca con l'obiettivo di realizzare sistemi per la gestione dei dati in ambito P2P [47] che superino tali limitazioni combinando aspetti provenienti dal mondo dei sistemi P2P, dei database e del Semantic Web. L'idea alla base di molti di questi sistemi è di realizzare reti P2P in cui i partecipanti impieghino metadati per descrivere i loro contenuti in modo da fornirne una più ricca ed espressiva rappresentazione e facilitarne quindi il recupero. In quest'ottica si collocano i cosiddetti **Peer Data Management System (PDMS)** che vengono presentati nel capitolo seguente.

Capitolo 3

Peer Data Management System (PDMS)

Proponendosi come punto di incontro tra il mondo dei database e quello delle reti P2P, i Peer Data Management System (PDMS) sono sistemi che intendono fornire un'architettura decentralizzata e facilmente estensibile per la gestione dei dati, all'interno della quale ad ogni utente sia concesso in ogni istante di agire con la massima libertà sui propri contenuti e allo stesso tempo di accedere a contenuti che risiedono su altri peer. I partecipanti ad un sistema di questo tipo sono infatti entità connesse in rete autonome, indipendenti e verosimilmente eterogenee che decidono di condividere il proprio contenuto informativo con gli altri componenti del sistema stesso.

Si tratta quindi di sistemi che si occupano di integrare semanticamente sorgenti informative eterogenee ma, a differenza di quanto accade per i classici sistemi impiegati per questi scopi, sono basati su un'architettura P2P e sono quindi completamente decentralizzati. In un PDMS non si ricorre infatti all'impiego di un singolo schema logico globale dei dati, ogni peer può invece continuare ad operare secondo il proprio schema individuale e l'integrazione è resa possibile grazie al fatto che tra i singoli peer vengono stabilite delle relazioni semantiche locali, solitamente chiamate *semantic mapping*. Queste relazioni permettono ad ogni nodo di esprimere le corrispondenze che esistono tra il suo contenuto informativo e quello di alcuni degli altri nodi presenti e rappresentano la sua conoscenza del sistema.

In un sistema di questo tipo non esistono quindi ruoli predefiniti e, con riferimento alla struttura dei sistemi per l'integrazione, si può affermare che ogni

peer può svolgere la funzione di sorgente informativa, mediatore o addirittura entrambi. Tutto ciò giova alla flessibilità del sistema, infatti, non esistendo alcuna autorità centrale, i singoli peer hanno la massima libertà di azione per quanto riguarda le possibilità di estensione e modifica dell'architettura globale e delle sorgenti locali.

L'obiettivo finale di un PDMS è però quello di sfruttare le relazioni semantiche locali per risolvere le interrogazioni degli utenti in modo da presentare loro tutte le informazioni utili contenute all'interno dei diversi nodi. Ogni interrogazione viene così risolta navigando i mapping semantici, che consentono di propagarla ai diversi peer in modo che ognuno di essi possa contribuire con i propri dati alla risposta completa, che deve essere infine presentata al richiedente.

3.1 Caratteristiche

Essendo costruiti su un'architettura P2P, i PDMS presentano molte delle caratteristiche e potenzialità che sono proprie di tali sistemi e che sono state illustrate in precedenza. Essi sono infatti caratterizzati da un'**architettura decentralizzata e distribuita**, in cui i partecipanti sono **nodi di uguale importanza** e senza ruoli predefiniti. Questi nodi sono anche **autonomi ed indipendenti** e risulta quindi necessario per un PDMS non solo essere in grado di gestire la loro **dinamicità** ma anche la loro **eterogeneità**, soprattutto perché, rispetto ai tradizionali sistemi P2P per la condivisione di contenuti, i PDMS si propongono di fornire **tecniche più ricche per la gestione dei dati**, che permettano di rappresentarne la semantica ed eseguire su di essi interrogazioni complesse.

Questi sistemi possono essere quindi considerati una sorta di naturale **evoluzione dei classici sistemi per l'integrazione** dei dati, in cui il singolo schema logico comune viene sostituito da una serie di mapping locali tra i nodi. Non impiegando uno schema logico globale, permettono di evitare le onerose attività iniziali di progettazione necessarie alla sua creazione e semplificano notevolmente tutte le operazioni di aggiornamento dovute all'evoluzione del sistema. Ogni peer può inoltre effettuare interrogazioni sull'intero sistema semplicemente utilizzando il proprio schema personale, senza essere costretto ad adottarne uno nuovo globale.

I PDMS si presentano poi come sistemi particolarmente interessanti per la realtà del **Semantic Web**, che si propone di condividere a livello idealmente

universale dati semanticamente arricchiti: la quantità di ontologie e schemi che vengono in questo contesto coinvolti è talmente grande da rendere assolutamente inadeguata l'adozione di strutture centralizzate che devono essere definite a priori.

3.2 Panoramica sui sistemi esistenti

Attualmente esistono diversi progetti di ricerca che si concentrano sullo sviluppo di sistemi per la gestione dei dati in ambito P2P e nel seguito viene presentata una breve descrizione di alcuni di questi progetti.

3.2.1 Piazza

Piazza [28] [30] [29] [56] è un progetto condotto dall'Università di Washington con lo scopo di realizzare un sistema distribuito facilmente estensibile per la condivisione decentralizzata e scalabile dei dati di un vasto numero di sorgenti eterogenee.

In questo progetto gli autori affrontano molti degli aspetti logici, algoritmici e implementativi che sono propri del contesto della gestione dei dati in un ambiente P2P, soffermandosi in particolare sulle problematiche legate alla necessità di un meccanismo decentralizzato per la mediazione tra i peer. In questa prospettiva propongono un formalismo molto flessibile detto *PPL (Piazza Peer Language)* da impiegare per la rappresentazione dei mapping semantici e su cui costruiscono un algoritmo per la risoluzione delle query. Si dedicano inoltre allo sviluppo di strumenti automatici da utilizzare nella generazione dei mapping e alla ricerca di ottimizzazioni da applicare al processo di riformulazione delle query. Sottolineano infine come un PDMS rappresenti un'infrastruttura ideale per costruire applicazioni per il Semantic Web e affrontano uno studio delle problematiche collegate, segnalando la rilevanza di tale ambito per gli sviluppi futuri di Piazza.

L'architettura di Piazza è costituita da una serie di nodi peer connessi in rete, ognuno dei quali rappresenta una sorgente di informazioni che è interessato a condividere con gli altri. Ogni nodo ha un proprio schema (*peer schema)* che

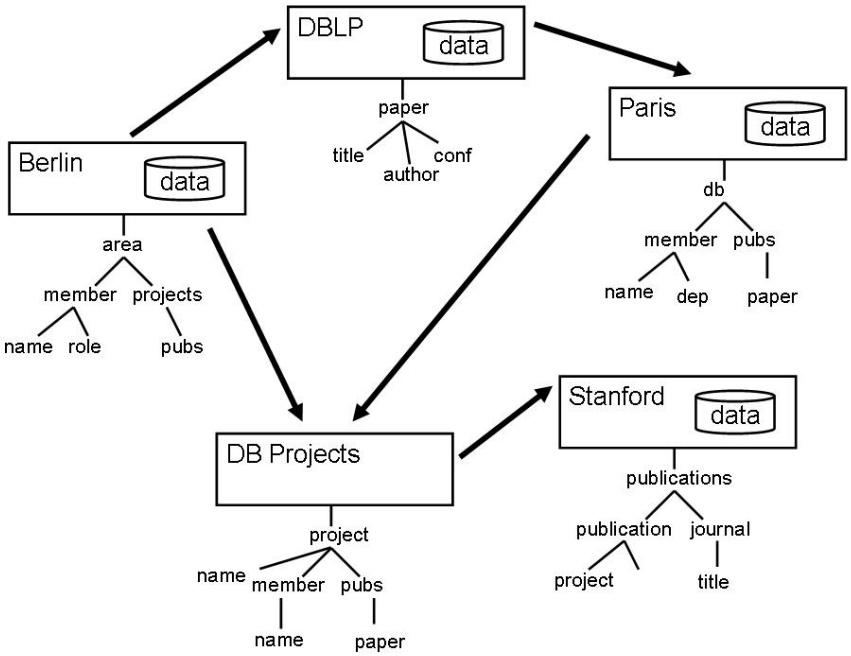

Figura 3.1: Piazza: architettura di esempio per un PDMS.

ne rappresenta il dominio di interesse ovvero la sua " visione del mondo" , verosimilmente differente da quella degli altri e contenente le relazioni (*peer relation*) che connettono tra loro gli elementi dello schema. Ogni peer può inoltre (non obbligatoriamente) fornire al sistema dei dati effettivi, memorizzati in forma di *stored relation* che li collegano al peer schema corrispondente, e mantiene alcuni *mapping semantici* locali che specificano le corrispondenze tra il suo schema e quello dei nodi vicini.

Un esempio è mostrato in Figura 3.1, nella quale ogni rettangolo rappresenta un peer che può possedere o meno dati effettivi. Sotto ad ogni peer è raffigurato un frammento del suo schema XML e le frecce rappresentano i mapping verso gli schemi degli altri peer.

Il meccanismo di risoluzione delle query prevede la presenza su ogni nodo di due moduli principali:

- *Query reformulation engine*, che, presa in ingresso una query posta ad un determinato peer, impiega i mapping semantici disponibili per ottenere in uscita una serie di query equivalenti da sottoporre ad altri nodi;

- *Query evaluation engine*, che si occupa invece dell'effettiva valutazione e risoluzione delle query.

Ogni interrogazione che viene posta ad un nodo del sistema viene infatti riformulata sulla base dei suoi mapping semantici per essere trasferita dal nodo iniziale ai suoi vicini ed essere presso di loro valutata. Tale processo si ripete ricorsivamente e in questo modo è possibile ottenere una risposta contenente tutti i dati del sistema provenienti dai peer collegati da un percorso semantico che viene infine ridiretta al nodo richiedente.

La riformulazione delle interrogazioni viene eseguita sul nodo a cui vengono poste sfruttando l'esistenza di un catalogo di sistema globale che fornisce l'accesso a tutti i mapping relativi ad ogni particolare peer relation. Questo catalogo deve essere reso disponibile a tutti i peer del sistema in modo che ognuno di essi abbia la possibilità di consultarlo secondo le proprie necessità ed è quindi attualmente realizzato in forma centralizzata (per migliorare le prestazioni del sistema i mapping sono inoltre memorizzati in cache presso i singoli peer). In questo senso si può affermare che la versione corrente di Piazza non si presenta come un sistema puramente P2P, ma piuttosto come un'architettura ibrida (si faccia riferimento alla sezione 2.1) dal momento che le operazioni di riformulazione vengono eseguite in modo centralizzato e solo le successive operazioni di effettiva valutazione delle query già riformulate sono distribuite. Come gli stessi autori sottolineano, sarebbe in realtà possibile aumentare il grado si decentralizzazione del sistema (e contemporaneamente quindi anche la sua scalabilità e robustezza) implementando una versione distribuita di tale catalogo, per esempio sfruttando alcune delle tecniche impiegate nelle architetture Peer-to-Peer strutturate (si veda la sezione 2.1) come le distributed hash table.

3.2.2 Hyperion

Si tratta di un progetto [1] condotto dalle Università di Toronto e Ottawa con lo scopo di realizzare un'architettura per un PDMS basata su una rete di nodi peer in grado di coordinarsi a runtime nell'esecuzione delle classiche operazioni tipiche dei DBMS.

Alla base del progetto sta infatti l'idea di concepire un PDMS come un convenzionale DBMS al quale viene aggiunto un nuovo strato software (un *P2P layer*) che ha il compito di implementare le funzionalità che sono richieste dai

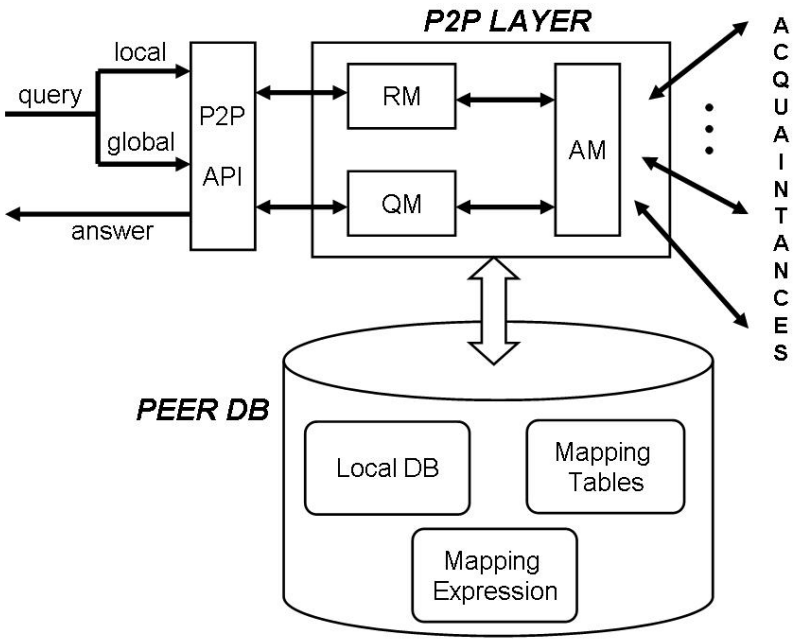

Figura 3.2: Hyperion: architettura del sistema.

peer per condividere e coordinare i dati senza compromettere in alcun modo la loro autonomia. Questo strato si occupa di stabilire e gestire, a runtime e in maniera semi-automatica, le relazioni tra i peer, costruendo in tal modo una rete P2P logica.

Gli sviluppatori del progetto si concentrano principalmente sulle problematiche che riguardano la definizione e la gestione dei matadati logici che permettono la condivisione dei dati e il coordinamento tra i peer.

La condivisione dei dati è attuata mediante l'impiego di *mapping table e mapping expression*, che permettono di definire le corrispondenze che esistono tra valori memorizzati in peer differenti. Uno studio approfondito sulle mapping table è affrontato dagli autori in [37], dove presentano un algoritmo per ricavare nuove tabelle dalle vecchie esistenti.

Per quanto riguarda il coordinamento, viene affrontato il problema della consistenza dei dati contenuti nei diversi peer e sono definite tecniche che permettono di specificare le politiche che regolano la propagazione dei dati quando il sistema si evolve.

Dal punto di vista architetturale il sistema risulta composto da una serie di

peer che sono database relazionali locali e che stabiliscono tra di loro dei legami in modo da definire una rete P2P. Come è illustrato in Figura 3.2, ognuno dei nodi risulta costituito da tre elementi principali: un'interfaccia (*P2P API*), uno strato P2P (*P2P layer*) e un *DBMS*.

All'interno del DBMS sono contenuti i dati locali del nodo e la serie di mapping table e mapping expression che devono essere impiegate per lo scambio dei dati con gli altri peer.

La *P2P API* è l'interfaccia utente che consente all'utilizzatore del singolo nodo di accedere all'intero sistema e di specificare se le interrogazioni devono essere risolte localmente oppure coinvolgendo anche gli altri peer.

il P2P layer risulta a sua volta composto di tre moduli:

- *Acquaintance manager (AM).* Implementa un metodo semi-automatico, basato sia su informazioni immesse dagli utenti sia su strumenti automatici, per creare i legami che i peer hanno tra di loro e che sono espressi attraverso mapping table e mapping expression. Si occupa inoltre di gestire le evoluzioni delle mapping table e delle mapping expression dovute alla dinamicità del sistema.

- *Query Manager (QM).* Gestisce l'esecuzione delle query che sono poste al sistema dai suoi utilizzatori. Si possono distinguere due tipi di query:

  1. le *query locali*, che devono essere risolte considerando solo i dati del peer locale;
  2. e le *query globali*, durante l'esecuzione delle quali è invece necessario ricorrere all'utilizzo della rete P2P per conciliare i dati locali con i dati che risiedono su altri peer. In questo secondo caso il QM si deve occupare di riscrivere la query, che viene posta ad un determinato peer e risulta quindi formulata secondo il suo schema, sugli schemi degli altri peer e per fare ciò utilizza i servizi forniti dall'AM.

- *Rule Manager (RM).* Si occupa di garantire che vengano rispettate le politiche di consistenza definite tra i peer per gestire le possibili evoluzioni e modifiche del sistema. Tali politiche vengono espresse dagli utenti attraverso la P2P API.

Le interrogazioni vengono quindi sottoposte al sistema dagli utenti attraverso la P2P API di un qualsiasi nodo per essere successivamente gestite dal suo Query

Manager. Nel caso di query globali, questo modulo non si occupa solamente di fornire in risposta i dati locali rilevanti ma impiega i mapping memorizzati nel DBMS locale per riscrivere l'interrogazione sui nodi collegati. Le query riscritte possono quindi essere inviate a questi nodi per essere valutate anche presso di loro.

Attraverso la P2P API un utente può anche specificare determinate politiche di consistenza che i peer devono rispettare a fronte delle possibili evoluzioni del sistema e dei suoi dati. Tali politiche sono gestite dal Rule Manager che si occupa di applicarle ai mapping memorizzati localmente e di comunicarle agli altri peer attraverso l'Acquaintance Manager.

3.2.3 Edutella

Si tratta di un progetto di ricerca [49] condotto dalle Università di Hannover e Karlsruhe con lo scopo di sviluppare un'infrastruttura in grado di agevolare l'interoperabilità e la condivisione delle risorse nelle reti P2P. L'idea di base è di combinare reti P2P e metadati che descrivono le risorse amministrate dai singoli peer, in modo da consentire ad applicazioni eterogenee di interagire e cooperare, secondo i principi che sono propri del Semantic Web.

Gli autori si concentrano principalmente sulle problematiche che riguardano i linguaggi e i meccanismi di comunicazione tra i vari peer e le affrontano proponendo un'infrastruttura basata su un'architettura di superpeer (si veda la sezione 2.1) costruita su RDF, il linguaggio standard per i metadati del W3C.

Il primo scenario applicativo pensato per Edutella è una rete P2P per lo scambio di risorse didattiche ed educative, ma l'intento degli autori è di estenderlo anche ad altri contesti.

Per quanto riguarda la sua effettiva implementazione, Edutella è costruita su JXTA [25] [34], un progetto open source della Sun Microsystems. JXTA è essenzialmente un ambiente java per creare applicazioni P2P costituito da una serie di protocolli basati su XML che forniscono varie funzionalità utili in quest'ambito, come ad esempio servizi per l'accesso remoto, la scoperta dei peer, il loro raggruppamento e monitoraggio. I differenti strati dell'architettura JXTA (Core, Service, Application) sono mostrati in Figura 3.3.

Edutella si basa su JXTA, ma ne arricchisce l'architettura con i seguenti componenti:

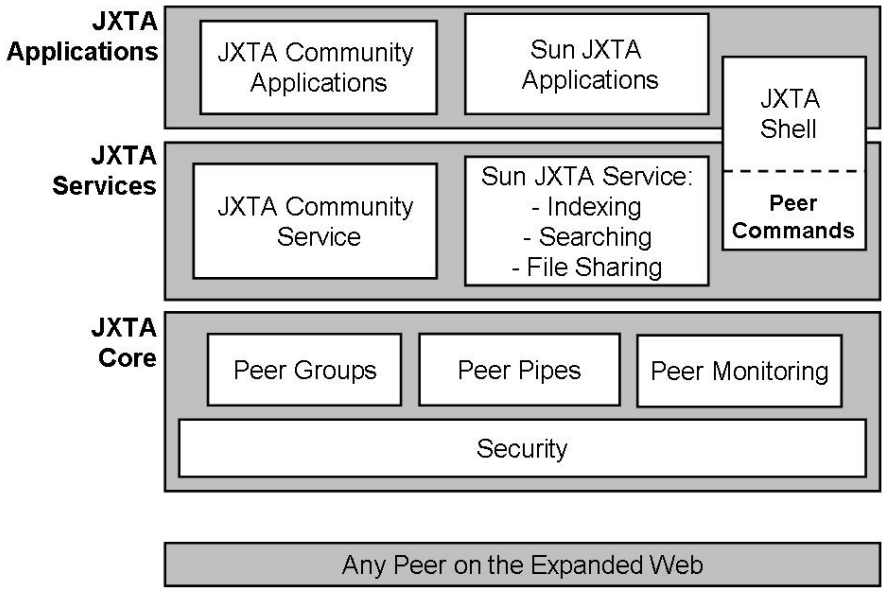

Figura 3.3: Edutella: architettura di JXTA.

- *Edutella Services.* Completa lo strato JXTA Service (costruito sul JXTA Core) e si preoccupa di definire i formati e i protocolli da impiegare nello scambio dei dati.

- *Edutella Peers.* Si pone nello strato di Application e utilizza le funzionalità fornite sia dagli Edutella Services che dallo strato JXTA Services per implementate tutti i servizi necessari al funzionamento del sistema, tra i quali i fondamentali sono:

  1. *Annotation service.* Si occupa del processo di annotazione di tutto il materiale memorizzato nel sistema e fornisce a tale scopo un meccanismo di visualizzazione dei vari documenti che permette di estrarne i metadati.

  2. *Mapping service.* Si occupa di attuare le traduzioni tra vocabolari differenti di metadati necessarie per consentire l'interoperabilità tra i vari peer.

  3. *Mediation service.* Si occupa di fornire viste coerenti per dati che

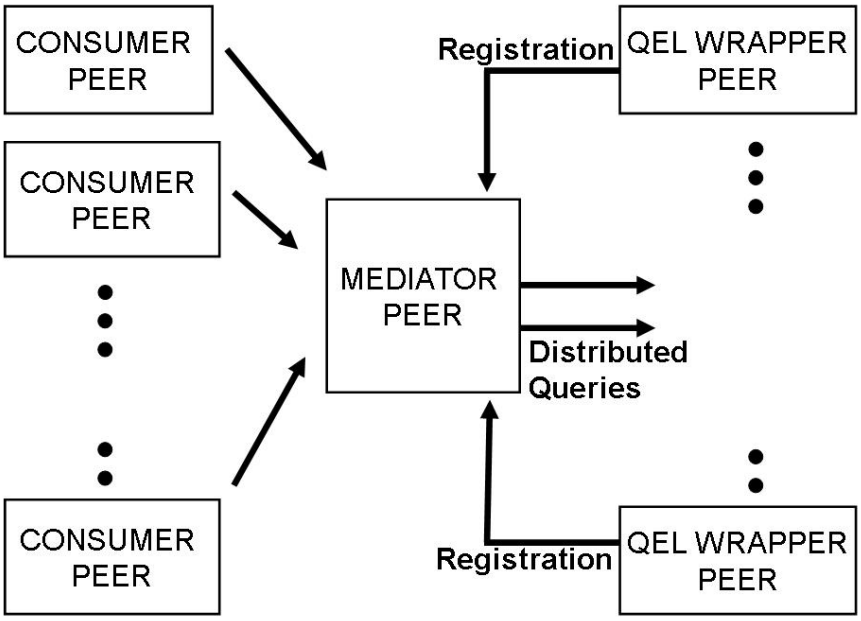

Figura 3.4: Edutella: schema del processo di risoluzione delle query.

appartengono a sorgenti differenti risolvendo i problemi legati all'eventuale presenza di informazioni ridondanti e in conflitto.

4. *Query service.* È il servizio fondamentale del sistema che definisce e standardizza i metodi per porre le interrogazioni e risolverle.

Il sistema Edutella ha un'architettura parzialmente centralizzata (si veda la sezione 2.1) che prevede la presenza di nodi superpeer che si comportano da sistemi a mediatore per il sottoinsieme di peer a loro collegati. Questi superpeer si occupano di integrare i metadati e fondere i dati provenienti dalle varie sorgenti risolvendo gli eventuali conflitti, ma agiscono anche da centri di registrazione e distribuzione delle query.

I vari peer devono infatti registrare presso di loro i tipi di interrogazioni che sono in grado di gestire e una certa quantità di informazioni riguardanti gli schemi di metadati supportati: in questo modo ogni superpeer diventa capace di selezionare i peer rilevanti a cui inviare le effettive interrogazioni pervenute. Ogni peer porge infatti le sue interrogazioni al proprio superpeer, che formula la risposta sia utilizzando le proprie informazioni sia interrogando altri superpeer

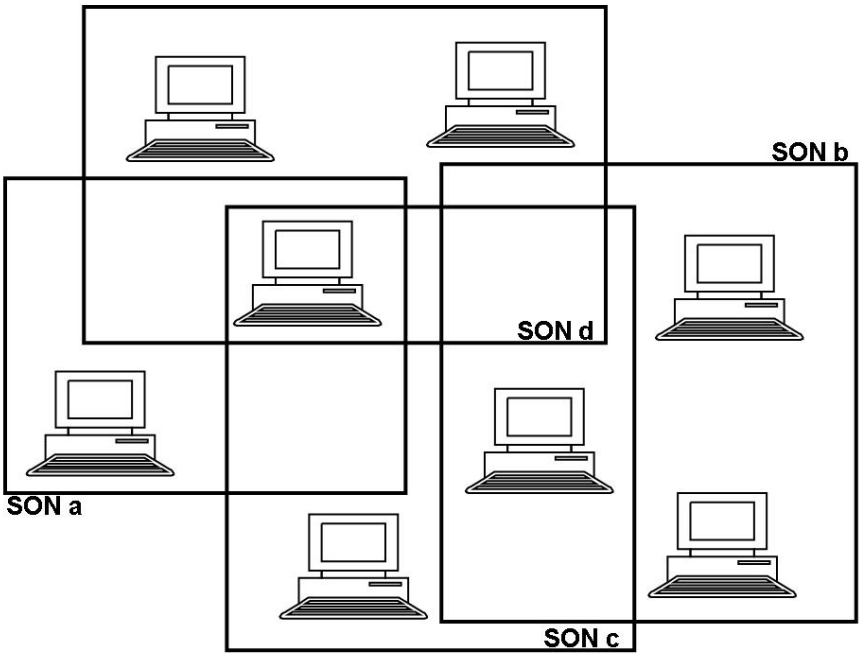

Figura 3.5: SON: esempio.

e si occupa inoltre di fondere i risultati per recapitare una singola risposta al richiedente. Lo schema di tale procedimento è illustrato nella seguente Figura 3.4.

3.2.4 Stanford Peers

Presso l'Università di Stanford è presente un gruppo di ricerca molto attivo nell'area delle architetture P2P che si concentra in modo particolare su tutte le problematiche legate alla localizzazione e organizzazione delle risorse nei sistemi per la condivisione dei dati [27]. In quest'ambito è stato sviluppato un progetto [15] che propone un'organizzazione flessibile e innovativa per le reti di questi sistemi, in cui le connessioni tra i nodi sono influenzate dal loro contenuto. Questo approccio prevede infatti che i peer non siano collegati tra loro in modo arbitrario a formare una singola overlay network come accade nei classici sistemi P2P, ma che i collegamenti vengano stabiliti tra nodi che contengono dati semanticamente simili per costruire varie Semantic Overlay Network (SON). All'interno delle singole SON viene poi definita un'organizzazione gerarchica sulla base di come

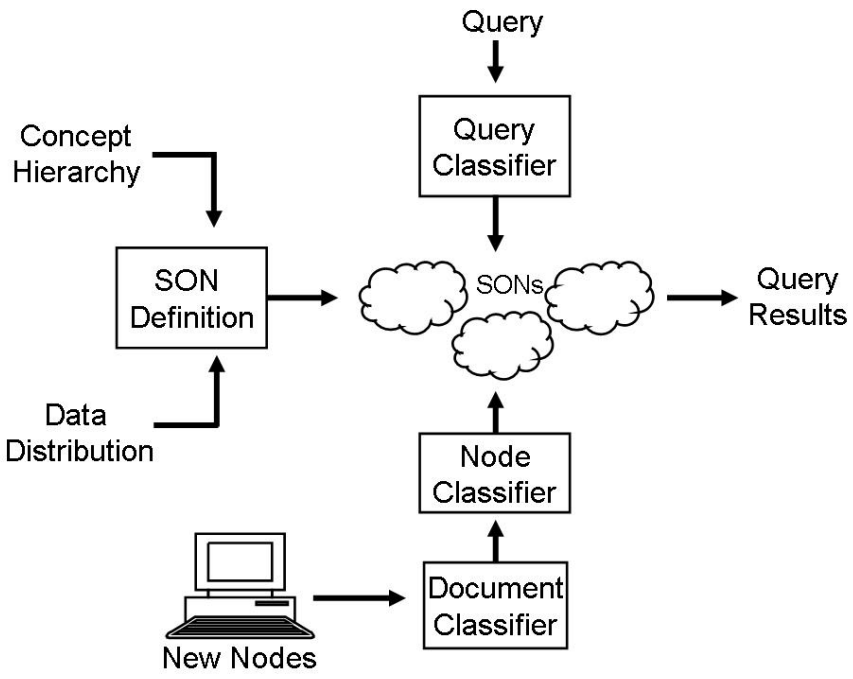

Figura 3.6: SON: processo di creazione e utilizzo.

si specializzano i vari argomenti. Il risultato è quindi una rete di superpeer (si veda la sezione 2.1) in cui è possibile indirizzare le query solamente verso le SON che trattano argomenti a loro correlati, aumentando in tal modo la probabilità di ottenere i dati desiderati velocemente e riducendo allo stesso tempo il carico sui nodi che hanno contenuti non correlati. Un esempio di struttura di questi sistemi è rappresentata in figura 3.5, nel quale si può anche osservare che ogni singolo nodo può appartenere a più di una SON.

Come è illustrato nella Figura 3.6 il processo di costruzione delle SON si articola in tre fasi:

- Per prima cosa sulla base dei dati effettivamente distribuiti nel sistema viene scelta la gerarchia da impiegare per classificare i *concetti* che definiscono le varie SON.

- Ogni nodo che si connette al sistema esegue quindi un *document classifier*, che associa un concetto della gerarchia ad ognuno dei suoi documenti.

- Infine il nodo esegue un *node classifier* che, sulla base dei risultati prodotti dal document classifier, lo assegna alle specifiche SON.

In Figura 3.6 è anche schematizzato il processo di risoluzione delle query che prevede di sottoporle ad un opportuno classificatore in modo da individuare le SON da contattare. A questo punto i nodi appropriati possono essere selezionati dai superpeer delle singole SON (alternativamente si può pensare di propagare la query a tutti i nodi della SON).

In [15] gli autori affrontano in modo specifico le problematiche che riguardano la scelta delle gerarchie di classificazione e dei classificatori da impiegare per le query e i documenti, proponendo per il momento tecniche manuali e parzialmente automatiche. Inoltre valutano diverse alternative per il meccanismo di assegnamento dei nodi alle SON e propongono un approccio detto Layered SON, secondo il quale un nodo viene assegnato ad una SON solo se possiede almeno una certa percentuale di documenti della corrispondente categoria.

3.3 Soluzioni progettuali

3.3.1 Problematiche

La realizzazione di un sistema per la gestione dei dati in ambito P2P comporta numerose difficoltà che derivano principalmente dalle carenze che tali architetture mostrano in materia di capacità di amministrazione dei contenuti e della loro semantica. Si possono individuare quattro macro-problematiche progettuali fondamentali:

1. **Rappresentazione dei dati.** Le sorgenti informative che partecipano al sistema sono entità autonome e indipendenti, è perciò verosimile supporre che esse adottino modalità diverse per rappresentare i propri dati ed occorre quindi definire dei linguaggi comuni.

2. **Connessione dei dati.** A causa dell'eterogeneità delle sorgenti informative che compongono il sistema occorre un meccanismo di integrazione decentralizzato che permetta ai peer di condividere i dati e cooperare. Nella maggior parte dei casi si ricorre all'impiego di collegamenti semantici locali, detti mapping, che vengono stabiliti tra i vari peer. Importanti decisioni relative a tali collegamenti riguardano il loro processo di creazione, i formalismi/linguaggi che vengono impiegati per rappresentarli e la scelta dei criteri che permettono di stabilire quali peer collegare tra loro.

3. **Risoluzione delle interrogazioni.** Il sistema deve risolvere le interrogazioni fornendo tutte le informazioni utili contenute all'interno dei diversi nodi ed occorre quindi che le query inizialmente poste ad un peer vengano opportunamente riformulate per essere propagate anche agli altri. Al termine del processo occorre poi fondere i risultati in un'unica risposta da presentare al richiedente. Tutte queste operazioni risultano particolarmente critiche per le prestazioni del sistema e la sua efficienza poiché la maggior parte delle informazioni ad esse necessarie devono essere desunte a tempo di esecuzione.

4. **Ricerca dei dati.** Siccome un'architettura P2P non prevede l'impiego di alcun controllore globale occorre disporre di un meccanismo decentralizzato che permetta la localizzazione e il recupero dei dati. Si tratta di una problematica strettamente collegata alla risoluzione delle interrogazioni ma riguarda trasversalmente molti altri aspetti come la collocazione dei dati e la gestione dell'evoluzione del sistema.

I vari sistemi esistenti propongono soluzioni progettuali differenti a questi quesiti, alcune delle quali sono completamente innovative, mentre altre derivano da ambiti differenti e possono però essere adattate al contesto P2P.

3.3.2 Rappesentazione dei dati

Per quanto riguarda la rappresentazione dei dati, alcuni sistemi si limitano al caso relazionale, supportando solo sorgenti che sono classici database e quindi descritte da uno schema espresso secondo questo modello. Un esempio è costituito dal progetto **Hyperion** [1], che concepisce se stesso come una rete di DBMS, ma presenta la particolarità di consentire ai peer di coordinare i propri dati secondo regole appositamente create dagli utenti per gestire l'evoluzione del sistema.

Sempre nel contesto relazionale si pone la proposta del **Local Relational Model** (LRM) [5] [47], in cui gli autori si concentrano in modo specifico sul problema della rappresentazione dei contenuti e definiscono un modello per i dati relazionali che si adatta al contesto P2P.

Anche la prima versione del progetto **Piazza** [28] era pensata per essere applicata al solo contesto relazionale, ma i suoi autori hanno in seguito deciso di abbandonarlo in favore dell'ambiente XML, indotti dal fatto che la maggior parte delle attuali applicazioni per la gestione delle informazioni impiega questo linguaggio per esportare i dati. Il linguaggio XML costituisce infatti probabilmente il modo più utile e semplice per rappresentare i dati in una forma che ne includa una certa descrizione semantica e ne faciliti la condivisione. Un'altra osservazione degli autori di Piazza a favore di questa scelta è l'impiego dell'XML nel formato RDF dei dati per il Semantic Web, un ambito di cui sottolineano l'importanza per gli sviluppi futuri del sistema. All'interno di Piazza viene già fornito un meccanismo che consente la presenza di sorgenti RDF, anche se bisogna puntualizzare che non è in grado di cogliere la ricchezza semantica di queste fonti.

Un sistema maggiormente orientato alla realtà del Semantic Web è **Edutella** [49] che adotta un modello dei dati basato su RDF, lo standard per i metadati del World Wide Web definito dal W3C. La rete Edutella connette peer estremamente differenti tra loro e supporta sorgenti informative di diversa natura, ma per ognuna di esse esprime in RDF i metadati che ne descrivono il contenuto. Ogni nodo di questo sistema è quindi considerato un deposito di metadati RDF (ovvero una base di conoscenza) che consiste di una serie di espressioni RDF e descrive tali metadati in accordo a schemi definiti in RDF Schema.

3.3.3 Connessione dei dati

Il meccanismo che i vari sistemi impiegano per connettere i dati dipende fortemente dal modello adottato per rappresentarli.

In **Piazza** [28] i vari peer sono collegati tra loro attraverso mapping semantici locali che vengono stabiliti tra i loro schemi per esprimerne le corrispondenze e che a seconda del numero di peer coinvolti vengono definiti:

- *pairwise mapping*, quando sono stabiliti tra due soli peer,
- *join mapping*, quando sfruttando degli operatori di join lo schema di un peer viene messo in relazione con gli schemi di un numero maggiore di peer.

Siccome il modello dei dati di Piazza è basato su XML, il linguaggio impiegato per esprimere i mapping *(Piazza Peer Language, PPL)* è una versione leggermente semplificata di XQuery, che opera attraverso una o più *mapping definition* che sono direzionali da uno schema *source* a uno schema *target*: il procedimento di definizione consiste nel considerare una parte dello schema target ed annotarlo con espressioni XQuery che definiscono quali dati dello schema source corrispondono a quella parte dello schema target. Si osservino ad esempio i due schemi di peer raffigurati in Listing 3.1 e Listing 3.2, nei quali le rientranze rappresentano la struttura annidata e i suffissi * indicano che per l'elemento in questione sono possibili zero o più istanze.

Listing 3.1: Piazza: esempio di schema S1.xml

```
people
    faculty*
        name
        advisee*
    student
```

Listing 3.2: Piazza: esempio di schema S2.xml

```
people
    faculty*
    student
        name
        advisor*
```

In Listing 3.3 viene presentato un esempio di mapping dallo schema source S1.xml allo schema target S2.xml, nel quale le variabili vengono inserite tra le parentesi graffe {} e le espressioni XQuery impiegate per l'annotazione sono introdotte tra i simboli {: :}.

Per quanto riguarda la costruzione dei mapping, in Piazza è al momento necessario l'intervento umano, ma sono comunque forniti strumenti automatici per assistere l'uomo al fine di agevolarlo. Il processo di creazione si articola in due fasi:

Listing 3.3: Piazza: mapping da S1 a S2.

```
<S2>
    <people> {: $people = document(``S1.xml'')/people :}
        <faculty>
        {: $name IN $people/faculty/name/text() :}
        { $name }
        </faculty>
        <student>
        {: $student IN $people/student/text() :}
            <name> { $student } </name>
            <advisor>{: $faculty IN $people/faculty,
                        $name IN $faculty/name/text(),
                        $advisee IN $faculty/advisee/text()
                        WHERE $advisee = $student :}
                        { $name }
            </advisor>
        </student>
    </people>
</S2>
```

1. nella prima fase detta di "schema matching" vengono rintracciate in maniera completamente automatizzata le corrispondenze esistenti tra gli elementi degli schemi considerati. Questa fase si divide a sua volta di due sottoparti:

   (a) per prima cosa le corrispondenze vengono ricercate impiegando un algoritmo che combina diverse tecniche di machine learning;

   (b) in seguito, poiché l'attività di schema matching è spesso caratterizzata da ripetizioni, la ricerca prosegue sfruttando le esperienze passate. In quest'ambito è tuttora allo studio uno strumento di schema matching detto corpus-based [42] che si basa sull'utilizzo di una raccolta di vecchi schemi, mapping, dati e metadati su cui possono essere calcolate delle statistiche da impiegare nelle nuove creazioni e che si evolve nel tempo con un processo di feedback.

2. nella seconda fase si ricorre invece all'intervento umano per raffinare le cor-

ALPHA-AIR DATABASE

AA_Passenger	
pid	name
1	Renee
2	Verena
4	Huyu

AA_Tickets		
pid	fno	meal
1	AA210	Meat
2	AA378	Veg

AA_Flight				
fno	date	dest	sold	cap
AA210	01/05/03	L.A.	120	256
AA341	01/15/03	N.Y.	160	160
AA378	01/21/03	S.F.	90	124

BETA-AIR DATABASE

BA_Flight				
fno	date	to	sold	aid
BA1023	01/07/03	LAX	67	B-3
BA1078	01/15/03	JFK	118	B-2
BA1109	01/21/03	ORD	164	B-1

BA_Fleet		
aid	type	capacity
B-1	Boeing 747	340
B-2	Boeing 737	130
B-3	Boeing 737	107

BA_Passenger	
pid	name
1	John
2	Renee

BA_Reserve	
pid	fno
1	BA1023
2	BA1078
3	BA1109

Figura 3.7: Hyperion: esemio di due schemi.

rispondenze precedentemente evidenziate ed ottenere dei mapping precisi.

Al contrario di quanto accade in Piazza, dove i mapping vengono stabiliti tra gli schemi dei peer, nel progetto **Hyperion** [1] i collegamenti semantici sono stabiliti a livello dei singoli oggetti e gli strumenti impiegati per esprimerli sono le *mapping expression* e le *mapping table*. Le mapping expression altro non sono che generalizzazioni di espressioni GLAV, mentre le mapping table sono tabelle all'interno delle quali vengono memorizzate le corrispondenze che esistono tra valori di attributi concettualmente simili che risiedono su peer differenti.

Un esempio di questi costrutti è mostrato in figura 3.8 relativamente al caso di due ipotetici peer che rappresentano due compagnie aeree (figura 3.7). La mapping expression indica che ogni passeggero della compagnia Beta-Air è considerato anche un passeggero della compagnia Alpha-Air. La mapping table (a) associa invece i nomi delle città del primo database con i codici degli aeroporti del secondo, mentre la mapping table (b) associa i numeri dei voli dei due database quando essi hanno la stessa città di destinazione. Infine il mapping (c)

MAPPING EXPRESSION:

AA_Passenger(p,n) ≥ BA_Passenger(p,n)

MAPPING TABLE:

dest	to
N.Y.	JFK
N.Y.	LGA
L.A.	LAX
L.A.	ONT

(a)

fno_{AA}	fno_{BA}
AA210	BA1023
AA341	BA1078
AA341	BA1080

(b)

$date_{AA}$	$date_{BA}$
x	*x*

(c)

Figura 3.8: Hyperion: esempio di mapping.

usa una variabile x per rappresentare la funzione identità, ovvero ogni valore di data del primo database viene fatto corrispondere a se stesso nel secondo.

Sia per quanto riguarda le mapping expression che le mapping table le tecniche impiegate per la loro creazione sono manuali o parzialmente automatiche. In particolare, in [37] gli autori affrontano uno studio molto approfondito sulle mapping table, discutendo diverse possibili semantiche e presentando un linguaggio che permette di esprimerle. Dimostrano poi come esse debbano essere considerate vincoli che regolano lo scambio dei dati tra i peer su cui è possibile ragionare per gestirle e generarne di nuove. Relativamente a quest'utimo aspetto propongono infine un algoritmo efficiente che, partendo da una serie di tabelle già create, è in grado di generarne automaticamente di nuove derivandole dalle vecchie.

Un approccio complesso e interessante per definire e scoprire i mapping è adottato da **Xyleme** [13], un progetto che si pone l'obiettivo di costruire un deposito per memorizzare documenti XML provenienti da diverse sorgenti del Web. Xyleme definisce i mapping tra i singoli path degli schemi e li impiega per realizzare delle *viste* che gli consentono di gestire in un un'unica struttura i dati provenienti da diverse sorgenti. In Listing 3.4 è riportato un esempio dei mapping che vengono impiegati per costruire una vista sul dominio *culture* che

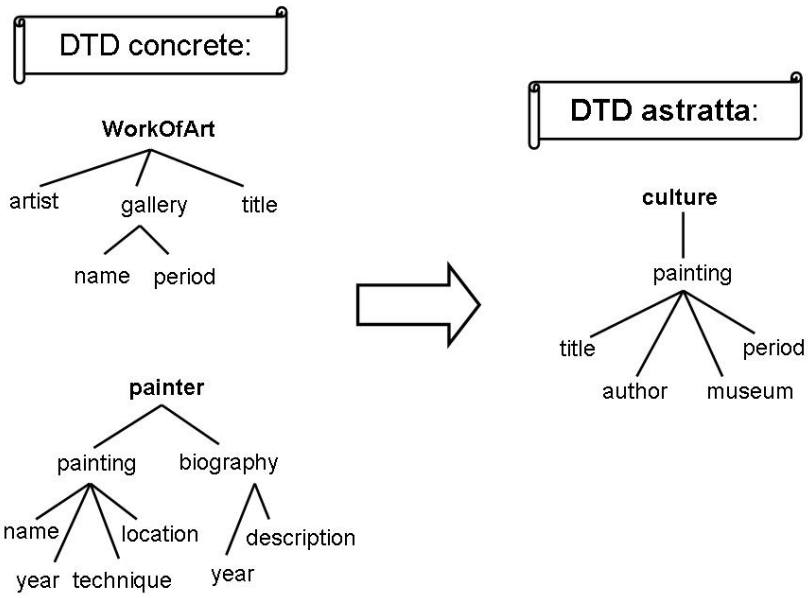

Figura 3.9: Xyleme: esempio di due DTD concrete e della corrispondente DTD astratta.

è rappresentato in figura 3.9.

Listing 3.4: Xyleme: esempio di mapping.

```
culture/painting --> WorkOfArt
culture/painting --> painter/painting

culture/painting/title --> WorkOfArt/title
culture/painting/title --> painter/painting/name

culture/painting/author --> WorkOfArt/artist/name
culture/painting/author --> painter

culture/painting/museum --> WorkOfArt/gallery
culture/painting/museum --> painter/painting/location
```

Analogamente a quanto accade nel contesto relazionale, in Xyleme ogni vista ha un *dominio*, uno *schema* e una *definizione*. Nel sistema si indica come *concreto*

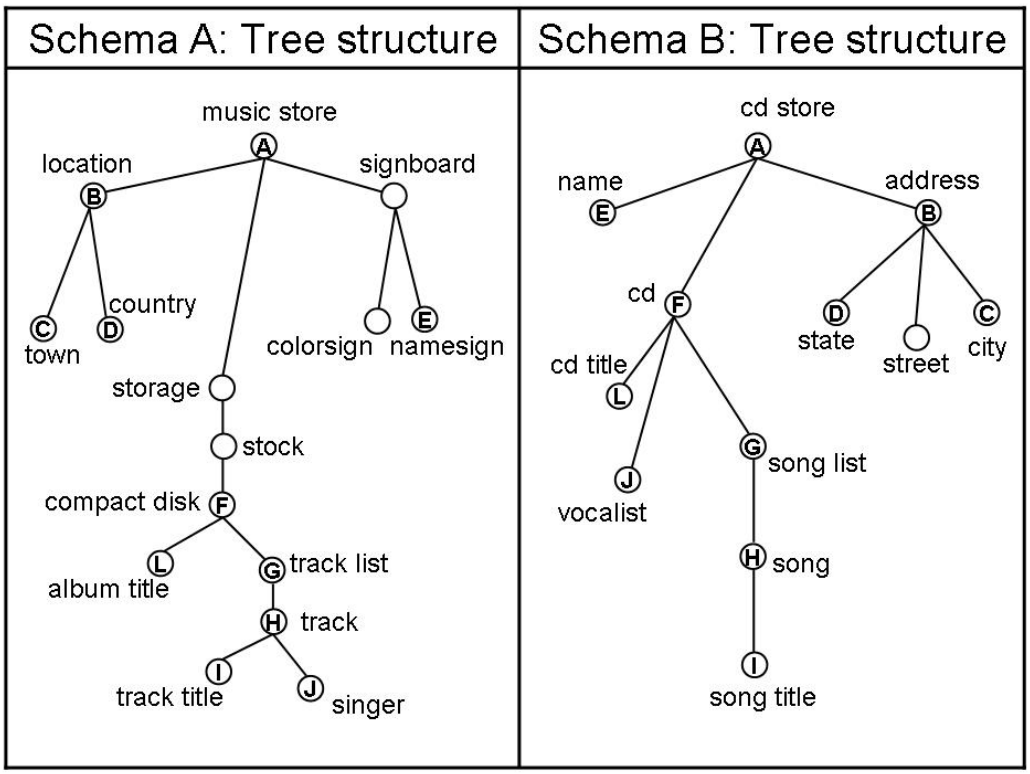

Figura 3.10: XML S^3MART: schema matching.

tutto ciò che ha a che fare con il dominio, mentre tutto quello che riguarda la vista stessa è detto *astratto*.

Il dominio di una vista è un insieme di cluster contenenti documenti XML semanticamente collegati e la cui struttura è descritta dalle loro DTD concrete (rappresentabili come alberi). Lo schema di una vista è invece una DTD astratta, ovvero un albero di concetti che descrive documenti astratti. La definizione di una vista è infine espressa tramite una serie di coppie (p, p') in cui p è un path della DTD astratta mentre p' è un path di una DTD concreta. Queste coppie sono dette mapping e stabiliscono quindi le corrispondenze tra i path di una DTD astratta con i path di una DTD concreta.

All'interno del progetto viene inoltre definito un metodo per costruire questi mapping in maniera semiautomatica considerando il loro contesto e gli elementi che li compongono.

Una definizione dei mapping simile a quella di Xyleme è proposta all'interno

del sistema **XML S^3MART (XML Semantic Structural Schema Matcher for Automatic query RewriTing)** [43]. Si tratta di un sistema sviluppato con l'obiettivo di consentire ricerche efficaci ed efficienti (espresse tramite interrogazioni XQuery) su insiemi di documenti XML che, seppur dotati di una struttura differente, sono in qualche modo in relazione tra loro per i contenuti che trattano. In XML S^3MART i mapping sono direzionali e vengono stabiliti in maniera automatica tra i singoli elementi degli schemi mediante un processo di schema matching che, per ogni coppia di schemi considerati, estrae le similarità sia strutturali che semantiche esistenti tra gli elementi che vi compaiono ed identifica il match migliore. L'estrazione della similarità è resa possibile grazie all'annotazione dei termini mediante l'impiego di un thesaurus comune e permette di associare ad ogni mapping un valore numerico compreso tra zero e uno che esprime la vicinanza semantica e strutturale tra i concetti. Per avere un esempio si considerino i due schemi di figura 3.10 che, pur adottando strutture e termini differenti, descrivono entrambi i contenuti relativi ad un negozio di prodotti musicali e agli album venduti; il processo di schema matching è in grado di evidenziare le corrispondenze tra gli elementi indicati con le stesse lettere.

All'interno del sistema **Edutella** [49] [41] ogni superpeer raggruppa i nodi relativi ad uno specifico dominio e si occupa di gestire l'eterogeneità dei loro schemi costruendo delle regole di trasformazione dette corrispondenze che permettono di identificare le correlazioni esistenti tra gli elementi che compaiono in tali schemi. Queste corrispondenze sono strutturate come mapping che vengono stabiliti tra ogni singolo schema di un nodo e lo schema unico definito dall'amministratore del superpeer. Si consideri l'esempio riportato in Listing 3.5 di uno schema di un superpeer attinente a contenuti di e-learning.

Listing 3.5: Edutella: schema di un superpeer.

```
lectures(lecture: identifier, lecture:language,
            lecture:subject, lecture:educationalcontext)
```

Per prima cosa vengono definite le corrispondenze esistenti tra gli attributi di questo schema e gli attributi degli schemi dei nodi a lui collegati, come è mostrato in Listing 3.6 e Listing 3.7 per i peer di esempio P1 e P2, che usano rispettivamente schemi di metadati di tipo Dublin Core (DC) [2] e Learning Object Metadata (LOM) [26].

Listing 3.6: Edutella: corrispondenze per il nodo P1.

```
1.  lectures:Identifier = dc:identifier
    lectures:language = dc:lang
    lectures:subject = dc:subject
```

Listing 3.7: Edutella: corrispondenze per il nodo P2.

```
2.  lectures:Identifier = lom:general.identifier
    lectures:language = lom:general.language
    lectures:context = lom:educational.context
```

Sulla base di tali corrispondenze vengono quindi create delle viste sugli specifici schemi dei peer, come è illustrato in Listing 3.8.

Listing 3.8: Edutella: viste per P1 e P2

```
1. lecturesViewDC(lectures:Identifier,
                  lectures:language,
                  lectures:subject)
   <-DC(dc:Identifier, dc:lang, dc:subject)

2. lecturesViewLOM(lectures:Identifier,
                  lectures:language,
                  lectures:context)
   <-LOM(lom:general.identifier,
         lom:general.language,
         lom:educational.context)
```

Successivamente vengono stabilite le correlazioni tra gli attributi dello schema del superpeer e quelli delle viste prima definite (Listing 3.9).

Infine, combinando tutte le informazioni precedenti è possibile ottenere le due corrispondenze finali mostrate in Listing 3.10.

Listing 3.9: Edutella: correlazioni tra gli attributi del superpeer e quelli delle viste.

```
1.  lectures(lectures:identifier,lectures:language,
            lectures:subject,-)
    <- lecturesViewDC(lectures:Identifier,lectures:Language,
            lectures:subject)

2.  lectures(lecture:identifier,lecture:language,-,
            lecture:context)
    <-lecturesViewLOM(lectures:Identifier,lectures:Language,
            lectures:context)
```

Listing 3.10: Edutella: corrispondenze finali.

```
P1.Correspondence1:
    lectures(lectures:identifier,lectures:language,-,
            lectures:educationalcontext)
    <-v(lectures:Identifier,lectures:language,
            lectures:context)
    <-LOM(lom:general.identifier,lom:general.language,
            lom:educational.context)

P2.Correspondence2:
    lectures(lectures:identifier,lectures:language,
            lectures:subject,-)
    <-v(lectures:Identifier,lectures:language,
            lectures:subject)
     <-DC(dc:identifier,dc:subject,dc:lang)
```

3.3.4 Risoluzione delle interrogazioni

La risoluzione delle interrogazioni è una problematica complessa, che risulta pesantemente influenzata dalle soluzioni adottate per la rappresentazione dei dati e per la loro connessione.

Per la risoluzione delle interrogazioni in **Piazza** [28] viene impiegato un algoritmo che prevede di navigare i mapping semantici disponibili per riformulare

ogni query, inizialmente posta su un determinato nodo, sugli altri peer che attraverso di essi è possibile raggiungere. L'obiettivo è di costruire una risposta finale che contenga tutti i dati rilevanti nel sistema, anche se appartenenti a nodi differenti e distanti.

L'algoritmo prende in ingresso una query Q posta sullo schema di un determinato nodo P e per prima cosa, se P contiene dei dati, formula la risposta a Q basata sui dati di P. Successivamente vengono ad uno ad uno considerati i vicini di P, ovvero tutti quei nodi che sono a lui collegati attraverso mapping semantici, e tali mapping sono impiegati per riformulare la query Q sul loro schema. Ad ogni passo si ottiene quindi una query Q' riformulata rispetto ad un nodo P' vicino di P, che viene a sua volta valutata sui dati di P' (se esso ne contiene) per inserire le nuove informazioni ottenute in un insieme comune delle risposte (dal quale sono eliminate le eventuali ridondanze). Il processo viene poi ripetuto per ogni nodo P', seguendo tutti i mapping semantici esistenti verso i suoi vicini e si procede ricorsivamente fino a quando non rimangono più percorsi utili da esplorare.

Siccome i mapping in Piazza sono direzionali da un nodo source S a un nodo target T, il processo di riformulazione può essere di due tipi a seconda della direzione dei mapping disponibili. Nel caso in cui la query iniziale sia posta su T, e debba quindi essere riformulata su S, è necessaria un'operazione di *unfolding*: per accedere ai dati di S si compone la query Q con le query del mapping che definiscono T in termini di S. Questa operazione è analoga a quella che viene eseguita per le *viste* nei tradizionali database relazionali e al processo di *query reformulation GAV-based* dei sistemi per l'integrazione. Nell'altro caso la query originale è posta su S e bisogna riformularla su T compiendo un'operazione di *rewriting*. Questa volta sia T che Q sono definiti come query su S e quindi per riformulare Q bisogna utilizzare il mapping in direzione inversa. Il caso è analogo al problema relazionale della *riscrittura delle query utilizzando le viste*, ma in Piazza viene esteso ai dati XML che, a differenza dei dati relazionali, sono semistrutturati.

Come esempio si faccia riferimento agli schemi introdotti nella sezione precedente (3.3.3 Connessione dei dati) e si supponga di voler riformulare sullo schema S2.xml la query rappresentata in Listing 3.11 e inizialmente posta sul S1.xml per recuperare tutti gli advisee di Ulmann.

Il processo di riscrittura opera confrontando la struttura della query con i

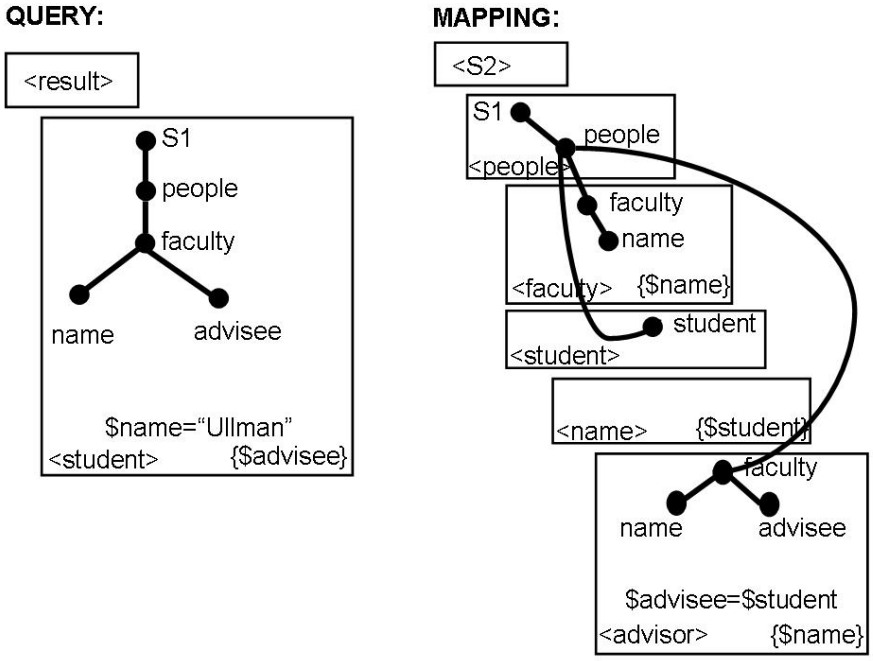

Figura 3.11: Piazza: rappresentazione grafica per query e mapping.

Listing 3.11: Piazza: esempio di query.

```
<result> {
    FOR $faculty IN document(``S1.xml'')/people/faculty,
        $name IN $faculty/name/text(),
        $advisee IN $faculty/advisee/text()
    WHERE $name = ``Ullman''
    RETURN
        <student> {$advisee} </student>
}
</result>
```

path presenti nella mapping definition per determinare se è possibile trovare una corrispondenza; in caso di esito positivo la query viene riorganizzata in modo analogo a quanto fa il mapping sul suo schema in ingresso.

Nella rappresentazione grafica della figura 3.11 la corrispondenza è evidenziata dalla linea più spessa; la query dell'esempio può quindi essere riscritta su S2 nella forma riportata in Listing (3.12).

Listing 3.12: Piazza: esempio di query riformulata.

```
<result> {
    FOR $student IN /S2/people/student,
        $advisor IN $student/advisor/text(),
        $name IN $student/name/text()
    WHERE $advisor = ``Ullman''
    RETURN
        <student> { $name } </student>
}
</result>
```

Gli autori di Piazza si soffermano infine sull'importanza di disporre un algoritmo per la risoluzione delle query che sia efficiente (poiché l'operazione di risoluzione delle query deve avvenire a runtime) e propongono quindi in [56] alcune possibili ottimizzazioni che consentono di evitare di seguire tutti i percorsi semantici disponibili:

- *Eliminazione delle riformulazioni superflue.* Il fatto che spesso sono presenti percorsi multipli di mapping tra le stesse coppie di peer porta a riformulazioni delle query superflue che possono essere eliminate prima di essere valutate.

- *Minimizzazione delle riformulazioni ridondanti.* Spesso le numerose elaborazioni che vengono eseguite sulle interrogazioni le allargano pesantemente includendo al loro interno espressioni ridondanti ed è quindi possibile minimizzarle prima di valutarle.

- *Impiego di strategie di ricerca.* Vengono studiati nel dettaglio gli effetti legati alla strategia di ricerca adottata.

- *Calcolo di percorsi semantici.* Viene dimostrato che pre-calcolare alcuni percorsi semantici nella rete può comportare diversi benefici e ne vengono studiati i costi.

Analogamente a quanto accade in Piazza, nel progetto **Hyperion** [1] le interrogazioni vengono risolte riformulandole sugli schemi dei nodi messi in relazione

tra loro tramite i collegamenti semantici disponibili, ovvero le mapping table e le mapping expression precedentemente introdotte. Per avere un esempio si faccia riferimento al caso presentato nella sezione precedente (3.3.3) e si consideri di voler riformulare sul database Beta-Air la query q1 inizialmente posta sul database Alpha-Air. Impiegando le mapping expression e le mapping table precedentemente introdotte, q1 (Listing 3.13) viene tradotta in q2 (Listing 3.14).

Listing 3.13: Hyperion: esempio di query q1.

```
SELECT  date
FROM    AA_Flight
WHERE   dest = ``LA''
```

Listing 3.14: Hyperion: esempio di query riformulata q2.

```
SELECT  date
FROM    BA_Flight
WHERE   to = ``LAX''
OR      to = ``ONT''
```

In **Xyleme** [13] il linguaggio impiegato per le interrogazioni è un'estensione di OQL che è in grado di sopportare le path expression. Si consideri come esempio la query in Listing 3.15 e la sua struttura ad albero (figura 3.12).

Listing 3.15: Xyleme: esempio di query.

```
SELECT  p/title
FROM    doc IN culture
        P IN doc/painting
WHERE   p/author CONTAINS ``Van Gogh''
AND     p/museum CONTAINS ``Orsay''
```

La risoluzione delle interrogazioni viene eseguita in due fasi.

1. Per prima cosa viene analizzata la struttura ad albero della query e viene generata un'espressione algebrica che permette di selezionare gli elementi

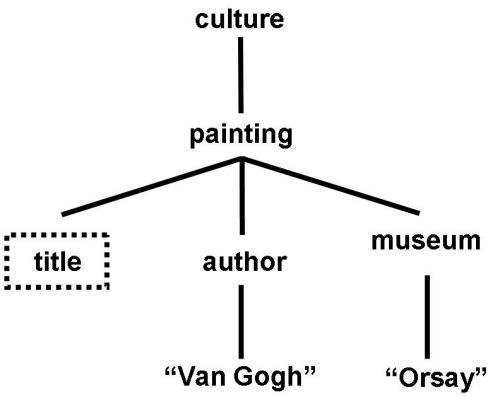

Figura 3.12: Xyleme: struttura di una query.

che compaiono in tale schema. Questa espressione algebrica viene impiegata da un apposito operatore detto *Patter Scan* per selezionare le viste i cui schemi corrispondono alla query.

2. Nella fase precedente sono stati selezionati solamente degli schemi astratti e in questa fase si impiega quindi un operatore detto *Abstract to Concrete (A2C)* che impiega i mapping per tradurre gli schemi astratti prima selezionati in schemi concreti. In particolare, per migliorare le prestazioni dell'algoritmo, non vengono considerati tutti i path concreti disponibili, ma solo quelli che appartengono alla stessa DTD concreta e che preservano le relazioni di discendenza della query. Ogni schema concreto generato viene poi assegnato ad un operatore detto *FTIScan* che recupera dai singoli documenti gli elementi selezionati.

In figura 3.13 viene mostrato un esempio di traduzione A2C.

Per eseguire la riscrittura delle interrogazioni su documenti XML eterogenei il sistema **XML** S^3**MART** [43] adotta un approccio basato sul concetto di *similarità* che applica alla struttura dei documenti considerati. Infatti, a causa della natura intrinseca dei dati XML, anche la componente strutturale di un documento è coinvolta nella formulazione di una query e viene quindi impiegata nel processo di ricerca, con la conseguenza che un tradizionale query engine restituisce solo documenti per cui esiste una perfetta corrispondenza tra il loro schema e quello della query. L'idea alla base di XML S^3MART è invece

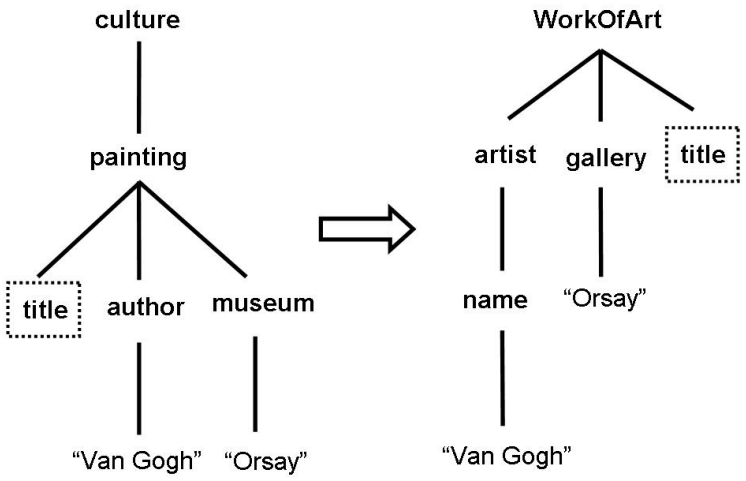

Figura 3.13: Xyleme: esempio di riscrittura.

che anche documenti strutturalmente differenti possono essere utili per risolvere un'interrogazione se i loro schemi mostrano delle similarità, che possono essere sia strutturali (hanno una struttura simile nell'albero XML) che semantiche (usano termini che hanno significati simili). Ad esempio il documento rappresentato in Listing 3.17 può essere verosimilmente utile per risolvere la query in Listing 3.16, anche se la sua struttura e i nomi impiegati per gli elementi non corrispondono.

Listing 3.16: XML S^3MART: esempio di query.

```
FOR     $x IN /musicStore
WHERE   $x/storage/cd/singer = ``Elisa''
RETURN  $x/name
```

Nella fase di rewriting, le informazioni estratte durante la fase di schema matching precedentemente illustrata (si veda la sezione 3.3.3) vengono sfruttate per riscrivere le query in maniera automatica tramite un procedimento di unfolding che le rende compatibili con le strutture dei documenti disponibili e quindi risolvibili da un normale query engine XML. Sulla base delle similarità estratte nella fase di schema matching, ad ogni operazione di riscrittura viene inoltre assegnato un punteggio da impiegare per valutare l'accuratezza delle risposte.

Listing 3.17: XML S^3MART: esempio di documento.

```
<cdStore>
        <name>Music World Shop</name>
        <cd>
            <vocalist>Elisa</vocalist>
            <cdTitle>Then comes the sun</cdTitle>
        </cd>
...
</cdStore>
```

Con riferimento all'esempio di schema matching del paragrafo precedente (3.3.3) la query in Listing 3.16 viene riscritta sullo schema B nel modo specificato in Listing 3.18.

Listing 3.18: XML S^{3}MART: esempio di query riscritta.

```
FOR     $x IN /cdStore
WHERE   $x/cd/vocalist = ``Elisa''
RETURN  $x/name
```

All'interno del progetto **Edutella** [49] la possibilità di supportare diverse tipologie di sorgenti informative è garantita grazie all'impiego di un modello comune per i dati *(Edutella Common Data Model - ECDM)* e un linguaggio comune per le interrogazioni *(Edutella Query Exchange Language - RDF QEL)* basati su RDF che forniscono la sintassi e la semantica necessarie alla definizione di un'interfaccia standard per le query. Per effettuare le necessarie traduzioni tra modello dei dati locale dei singoli peer e modello comune ECDM, Edutella impiega dei particolari moduli software detti *wrapper*, che si connettono poi alla rete impiegando le primitive JXTA per trasmettere su di essa le query espresse in RDF QEL.

Ogni superpeer mantiene poi localmente i mapping espressi come *corrispondenze* tra il suo schema e quello degli altri nodi a lui collegati ed è quindi in grado quando riceve una query di identificarne le eventuali conformità con gli schemi gestiti. Si faccia riferimento all'esempio introdotto nella sezione prece-

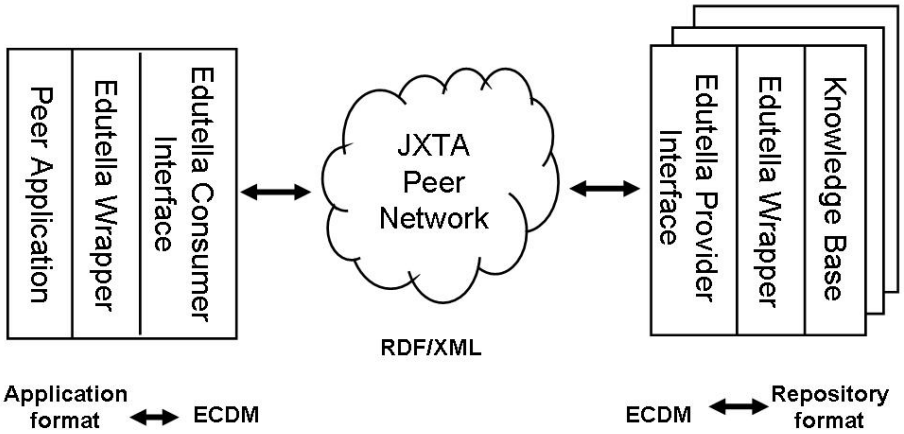

Figura 3.14: Edutella: procedimento di risoluzione delle query.

dente (3.3.3) e si consideri il caso in cui il superpeer riceva una query il cui schema è espresso in Listing 3.19.

Listing 3.19: Edutella: esempio di query.

```
lecture (lecture:identifier,lecture:language,lecture:subject,
         lecture:educationalcontext)
```

Il superpeer identifica quindi le corrispondenze P1:Correspondence1 e P2:Correspondence2 (Listing 3.10) come rilevanti per la risoluzione della query e interroga perciò i peer P1 e P2, occupandosi infine di raccogliere e conciliare i risultati da loro forniti.

Un aspetto interessante del sistema Edutella è poi la sua abilità di gestire le diverse capacità che i vari peer possono avere in materia di gestione delle interrogazioni, prevedendo la definizione di diversi livelli del linguaggio RDF-QEL che consentono ad ogni peer di specificare quale tipo di interrogazioni è in grado di supportare (query congiuntive, algebra relazionale, chiusura transitiva, ecc...). Tutti i livelli di linguaggio sono comunque basati sullo stesso modello

dei dati ECDM e per ognuno di essi è possibile utilizzare una rappresentazione delle query in RDF.

3.3.5 Ricerca dei dati

La possibilità di disporre di tecniche efficienti ed efficaci per ricercare e recuperare i dati all'interno di un sistema P2P è una problematica di fondamentale importanza poiché influisce pesantemente sulla sua usabilità. In tale ambito sono state implementate varie metodologie, con diversi pregi e difetti, e la scelta di quale di esse adottare dipende dunque fortemente dalle necessità dell'applicazione in questione.

Nei sistemi che presentano **architetture strutturate** (si veda la sezione 2.1) è possibile usufruire di meccanismi di ricerca efficienti in grado di garantire che i contenuti vengano sempre localizzati, se presenti, seguendo un percorso di lunghezza limitata. Esempi di questi sistemi sono **Chord** [54], **Past** [16], **Tapestry** [62] e **CAN** [52] che per localizzare le risorse ricorrono all'impiego di tabelle di routing memorizzate in forma distribuita tra i vari nodi sotto forma di *distributed hash table (DHT)*. Tali tabelle mantengono le corrispondenze tra l'identificativo di ogni risorsa e la sua locazione, stabilite applicando funzioni hash all'identificativo e impiegando il risultato così prodotto per risalire al nodo ospitante.

Un esempio di applicazione che impiega queste tecniche è **PIER (Peer-to-Peer Information Exchange and Retrieval)** [31] un motore di esecuzione di query costruito su CAN con l'obiettivo di arricchire il mondo scalabile e flessibile delle reti P2P con le capacità di interrogazione dei dati tipiche dei DBMS relazionali. Per realizzare tale scopo PIER rilassa i vincoli di consistenza che caratterizzano i classici DBMS e si concentra piuttosto sugli aspetti di disponibilità e scalabilità, proponendo di accedere ai dati tramite wrapper in grado di estrarne le informazioni.

Le tecniche delle architetture strutturate sono quindi molto efficienti, ma necessitano di uno stretto controllo sulla topologia della rete e sul posizionamento delle risorse e presentano inoltre lo svantaggio di supportare solamente ricerche per identificativo, presentandosi come non adatte per le applicazioni che intendono fornire servizi di ricerca più complessi.

Le **architetture non strutturate** (si veda la sezione 2.1) offrono invece la possibilità di sviluppare meccanismi di ricerca basati sul contenuto delle risorse (come le ricerche per parole chiave) e risultano inoltre più adeguate delle precedenti per i sistemi caratterizzati da popolazioni di nodi estremamente variabili, nei quali non è semplice mantenere una struttura.

Le architetture più essenziali, come ad esempio Gnutella [24], impiegano però metodi molto elementari di ricerca, che operano esplorando la rete inondandola di query fino a quando non viene raggiunto il nodo desiderato. Si tratta di approcci semplici e robusti, ma molto costosi in termini di traffico di rete generato e quindi poco efficienti.

Un'altra soluzione poco efficiente è quella adottata dalle architetture ibride (presentate nella sezione 2.1), che ricorrono all'impiego di un indice centralizzato [47] che permette di localizzare tutti i dati del sistema e che risulta quindi poco robusto e difficile da mantenere aggiornato.

Soluzioni molto più interessanti prevedono invece l'impiego di indici distribuiti sui singoli peer detti **Indici di Routing (RI)** [60] [35] [21] [14]. L'idea che sta alla base di questi costrutti è di utilizzarli per indicare non tanto la locazione esatta di un certo documento, ma piuttosto la direzione migliore da seguire per recuperalo. Le loro dimensioni sono quindi contenute, poiché risultano proporzionali al numero dei vicini di ogni nodo piuttosto che al numero totale di documenti del sistema. Ogni peer mantiene infatti nel suo IR un breve sommario per ognuno dei nodi confinanti relativamente ai dati che quest'ultimo o i suoi vicini sono in grado di fornire. Le indicazioni dei sommari possono essere ricavate dalle query processate in precedenza o dallo scambio diretto di informazioni tra i peer, ma la cosa importante è che tramite gli IR ogni nodo può stimare i contenuti raggiungibili nel propagare una query su ognuno dei suoi collegamenti e dunque selezionare quelli più promettenti in accordo con la loro rilevanza rispetto alla query.

Crespo e Garcia Molina in [14] presentano tre tipi di indici di routing:

- **Compound RI.** Memorizzano per ogni collegamento in uscita il numero di risorse che sono globalmente raggiungibili seguendo quel percorso e le raggruppano inoltre per argomenti. I limiti principali di questa tipologia sono dovuti al fatto che possono incorrere in percorsi ciclici e che considerano tutti i peer del sistema allo stesso livello senza valutare la loro distanza dal nodo corrente.

- **Hop-count RI.** Memorizzano per ogni collegamento in uscita il numero di risorse che possono essere raggiunte suddividendole sulla base della loro distanza dal peer corrente. La distanza viene espressa in numero di hop e vengono considerati solo i peer raggiungibili attraverso un numero fissato di hop detto *orizzonte*, in modo da non incorrere in percorsi ciclici.

- **Exponential RI.** Cercano di adottare gli aspetti vantaggiosi di entrambe le tipologie precedenti ed ereditano dagli Hop-count RI il principio di attribuire rilevanza maggiore ai nodi più vicini e dai Compound RI l'idea di non avere un orizzonte, in modo da poter fornire informazioni sull'intera rete. Per quanto riguarda la loro struttura sono del tutto simili ai Compound-IR, perché memorizzano per ogni collegamento in uscita il numero di istanze globalmente raggiungibili seguendo quel percorso e le raggruppano inoltre per argomento. La differenza sta nel fatto che durante il processo di costruzione i valori da memorizzare vengono corretti con un opportuno fattore che tiene in considerazione la distanza dal peer corrente.

In [44] viene poi presentato l'algoritmo **Adaptive Probabilistic Search (APS)** che può essere in un certo senso considerato una variante degli Exponential RI. In questo lavoro ogni nodo mantiene una lista di tutti gli oggetti per cui in passato ha inviato richieste ad ognuno dei suoi vicini. Il valore di ogni elemento della lista rappresenta la probabilità di quel vicino di rispondere con successo alla stessa richiesta.

Le fasi di ricerca e aggiornamento sono fuse in un unico algoritmo che opera in tre passi:

1. Il peer iniziale propaga la query a un numero fissato k di suoi vicini (mentre gli altri nodi che la ricevono devono scegliere solamente un vicino a cui inviarla).

2. Quando un peer riceve una query, la invia al vicino che ha la probabilità più alta di rispondere e adotta un approccio che può essere ottimistico o pessimistico. Nel caso ottimistico si assume che il peer scelto sarà in grado di fornire un risultato e quindi la sua probabilità viene aumentata; nel caso pessimistico si assume al contrario che il risultato non arriverà e quindi la probabilità viene diminuita.

3. La query può terminare il suo percorso in due casi: in caso di successo se viene trovata la risposta e in caso di insuccesso se il suo TTL si è azzerato oppure se ha raggiunto un nodo che aveva già visitato. A seconda che sia stato adottato l'approccio ottimistico o pessimistico, bisogna segnalare rispettivamente il caso di fallimento o di successo.

Gli indici di routing fino ad ora presentati sono costruiti su un modello di rappresentazione della conoscenza basato su keyword, poiché hanno la forma di una lista di parole chiave che indicano i contenuti indirizzati. Un approccio più articolato è invece affrontato all'interno del sistema **Edutella** [49] [41] [48], i cui indici sono in grado di fornire informazioni più accurate. Questi indici vengono mantenuti dai soli superpeer della rete e possono essere di due tipi:

- *indici di routing SP/P*, che descrivono i peer direttamente connessi ad un superpeer,
- *indici di routing SP/SP*, che si riferiscono invece agli altri superpeer.

La struttura delle due tipologie è comunque omogenea e costituita da una serie di strutture dati che memorizzano diversi attributi sui dati indirizzati. L'adozione di un modello di questo tipo permette dunque di migliorare notevolmente l'esecuzione delle query, perché consente di specificare proprietà utili alla loro risoluzione e di operare eventualmente a diversi livelli d granularità.

Parte II

Tecniche di approssimazione semantica per query routing in un PDMS

Capitolo 4

Migrazione di XML S^3MART verso un ambiente P2P

Nell'ambito della presente Tesi di Laurea vengono affrontate alcune delle problematiche relative al processo di routing delle interrogazioni in un PDMS.

In particolare, l'architettura di riferimento è rappresentata da un sistema Peer-to-Peer parzialmente centralizzato (si veda la sezione 2.1), costituito da una rete di superpeer che devono poter collaborare tra loro nella risoluzione delle interrogazioni poste dagli utenti. Perché tale collaborazione sia possibile occorre quindi che ogni superpeer possa stabilire, con gli altri superpeer a cui è connesso, degli opportuni collegamenti semantici (detti *mapping*, si veda la sezione 3.3.3) che esprimono le corrispondenze esistenti tra i rispettivi contenuti. Questi mapping devono infatti essere impiegati da ogni superpeer per "riscrivere" le interrogazioni a lui pervenute sulla base dei contenuti degli altri superpeer, in modo che anch'essi possano processarle e quindi risolverle.

L'approccio proposto in questa Tesi per realizzare tali mapping si basa su un processo di "approssimazione semantica", che viene applicato agli elementi degli schemi che descrivono il contenuto informativo dei superpeer. L'idea di base è che ogni superpeer stabilisca un mapping verso ognuno dei suoi vicini collegando ogni elemento del suo schema con quello dello schema di tale vicino che ne rappresenta la migliore approssimazione semantica. Ad ogni singola coppia di elementi si vuole poi associare un punteggio numerico che esprima la qualità di tale approssimazione semantica, in modo che ogni superpeer possa valutare come i suoi elementi possono essere approssimati muovendosi verso i suoi vicini.

Per implementare queste funzionalità si è scelto di impiegare XML S^3MART

[43], un sistema nato con lo scopo di consentire la riscrittura di interrogazioni su insiemi di documenti dotati di schemi differenti. XML S^3MART è adatto ad essere utilizzato nel processo di creazione dei mapping tra i superpeer, perché opera individuando le similarità sia semantiche che strutturali esistenti tra elementi che compaiono in schemi differenti e le esprime attraverso dei punteggi numerici.

XML S^3MART non è stato tuttavia progettato per operare in un contesto Peer-to-Peer e per impiegarlo in un PDMS è stato quindi necessario apportare delle opportune modifiche al suo funzionamento, in modo da renderlo idoneo al nuovo ambiente di esecuzione.

Nel seguito di questo capitolo viene per prima cosa presentato il sistema XML S^3MART, in modo da illustrarne in maniera dettagliata la struttura e il funzionamento. In seguito viene presentata l'architettura di riferimento del PDMS in cui XML S^3MART deve essere inserito e vengono infine descritte le modifiche su di esso eseguite per renderlo idoneo a tale architettura.

4.1 Il sistema XML S^3MART

XML S^3MART (XML Semantic Structural Schema Matching for Automatic query RewriTing) [43] è un sistema sviluppato presso l'Università di Modena e Reggio Emilia con l'obiettivo di consentire l'esecuzione di ricerche efficaci ed efficienti all'interno di collezioni di documenti eterogenei espressi in formato XML.

L'utilità di uno strumento di questo tipo risulta evidente se si considera che grazie ai notevoli progressi in materia di risorse computazionali e al considerevole calo dei costi di digitalizzazione, negli ultimi anni si è assistito ad una rapida proliferazione di sistemi in grado di memorizzare elettronicamente, consentire l'accesso e la diffusione attraverso il Web di grandi quantità di documenti digitali e dati multimediali. Queste collezioni spesso raggruppano però documenti che provengono da sorgenti differenti e sono quindi eterogenei per quanto riguarda la struttura adottata per la loro rappresentazione, pur essendo correlati tra loro per i contenuti che trattano.

Inoltre, come è stato detto nel capitolo 3.3.2, il linguaggio XML [7] è diventato una sorta di "standard de facto" per lo scambio e la rappresentazione di dati eterogenei nella rete Internet e sono quindi molte le collezioni di documenti

accessibili via Web i cui dati sono espressi in XML e associati a Schemi XML [58]. I motori di ricerca che operano in questo ambito sono poi in grado di risolvere interrogazioni espresse in linguaggi, come ad esempio XQuery, la cui espressività permette agli utenti di porre richieste strutturali, ovvero quesiti nella cui formulazione viene coinvolta anche la struttura dei documenti.

Proprio per questo motivo, l'esecuzione di ricerche efficaci ed efficienti in questo contesto è un problema di non semplice soluzione, dal momento che documenti XML riguardanti lo stesso argomento e descriventi la stessa realtà (e quindi ugualmente utili per soddisfare le richieste degli utenti) ma provenienti da fonti diverse, possono avere strutture ed impiegare termini molto differenti tra loro.

Con lo scopo di favorire lo scambio di informazioni tra sistemi diversi, XML S^3MART propone quindi un approccio per la risoluzione di interrogazioni approssimate su insiemi di documenti XML eterogenei che, invece di operare direttamente sui dati, cerca di interpretare la componente strutturale delle interrogazioni elaborando gli schemi dei documenti disponibili. In particolare, tali schemi (che si suppongono espressi in XML Schema) sono sottoposti ad un processo che opera in due fasi:

1. per prima cosa, durante la fase di **schema matching**, vengono estratte le similarità sia semantiche che strutturali esistenti tra gli elementi che compaiono negli schemi coinvolti, in modo che sia possibile evidenziarne le corrispondenze;

2. tali similarità vengono poi impiegate nella seguente fase di **query rewriting** per produrre riformulazioni delle interrogazioni che siano compatibili con le strutture dei documenti disponibili e quindi processabili da un normale esecutore di query XML.

4.1.1 Schema matching

La fase di **schema matching** opera sull'insieme di schemi XML che caratterizzano le parti strutturali dei documenti nella collezione e, processandoli a coppie, identifica per ognuna di esse il miglior mapping possibile, ovvero associa ad ogni elemento che compare in uno dei due schemi l'elemento dell'altro schema che

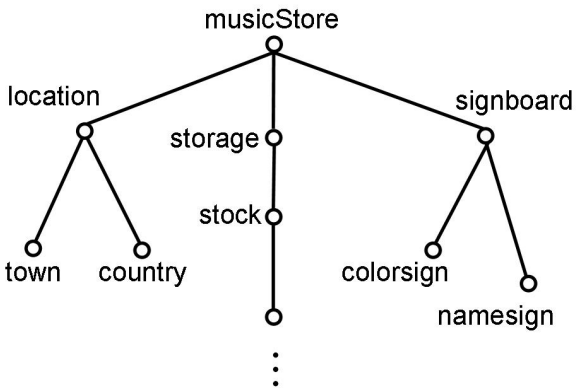

Figura 4.1: XML S^3MART: struttura concettuale ad albero di un documento.

meglio gli corrisponde. Questa fase si compone a sua volta di tre sottoprocessi, i primi due dei quali (detti **structural schema expansion** e **terminology disambiguation**), sono necessari per massimizzare l'efficacia della terza fase, nella quale viene effettivamente eseguita la **matching computation**.

Structural schema expansion

XML Schema [58] è il linguaggio raccomandato dal W3C per la definizione della struttura e dei tipi di dati dei documenti XML. Lo scopo di un XML Schema è quindi la definizione di una classe di documenti XML accomunati dal fatto che essi descrivono la loro componente strutturale impiegando gli elementi (e le relazioni tra di essi) introdotti in tale schema.

XML Schema è però un linguaggio molto ricco e per eseguire un processo di schema matching efficace è invece necessario operare su schemi che rappresentino la loro sottostante struttura concettuale in modo semplice e immediato. La fase di **structural schema expansion** si occupa quindi di modificare ogni XML Schema, risolvendo e riscrivendo i suoi costrutti in modo da rendere più esplicite le relazioni strutturali tra i suoi elementi e rappresentare così in maniera migliore la classe di documenti XML che definisce.

Si considerino per esempio i frammenti relativi ad un XML Schema e alla sua corrispondente versione espansa presentati in Listing 4.1 e 4.2 per una classe di documenti riguardanti dei negozi di musica, la cui struttura concettuale ad

Listing 4.1: Frammento di XML Schema originale.

```
<xsd:schema xmlns:xsd =``http://www.w3.org/2001/xmlSchema''>
    <xsd:element name=``musicStore'' type=``musicStoreType''/>
    <xsd:complexType name =``musicStoreType''>
    <xsd:all>
        <xsd:element name =``location'' type=``locationType''>
        ...
    </xsd:all>
    </xsd:complexType>
    <xsd:all>
        <xsd:element name =``town'' type =``xsd:string''/>
        <xsd:element name=``country''type=``xsd:string''/>
        ...
    </xsd:all>
```

Listing 4.2: Frammento di XML Schema espanso.

```
<xsd:schema xmlns:xsd =``http://www.w3.org/2001/xmlSchema''>
    <xsd:element name =``musicStore''>
        <xsd:element name =``location''>
            <xsd:element name =``town'' type =``xsd:string''/>
            <xsd:element name=``country''type=``xsd:string''/>
            ...
        </xsd:element>
```

albero è rappresentata in figura 4.1. Come si può osservare, la versione espansa dello schema fornisce una rappresentazione molto più esplicita e diretta della struttura concettuale, poichè rispetto alla versione originale sono stati eliminati tutti i costrutti contenenti espressioni regolari, keywords e definizioni di tipi. Lo schema espanso ha così una forma molto semplice nella quale ogni elemento ed ogni attributo hanno un nome e nel caso siano elementi intermedi hanno figli identificabili in maniera immediata. Gli elementi foglie e gli attributi hanno invece un valore testuale il cui tipo primitivo è specificato mediante l'attributo "type".

Terminology disambiguation

Dopo aver reso esplicite le relazioni strutturali di uno schema mediante il processo di structural schema expansion, viene applicato un altro stadio di elaborazione che consente di raffinare ulteriormente la quantità di informazioni da fornire alla fase di matching computation. Tale stadio è detto di **terminology disambiguation** e si concentra sulla componente semantica dei termini che compaiono nello schema stesso. Ognuno di questi termini viene infatti in questa fase disambiguato e ne viene cioè esplicitato il significato, in modo che esso possa essere in seguito impiegato per individuare la sua similarità semantica con i termini che compaiono negli altri schemi.

Per eseguire questa operazione viene impiegato WordNet (WN), uno dei più conosciuti database lessicali per la lingua inglese in cui i vari concetti sono organizzati in insiemi di sinonimi, i cosiddetti *synset*. In WordNet un synset raggruppa quindi tutti i termini della lingua inglese ai quali è associato lo stesso significato ed è inoltre caratterizzato da una descrizione testuale detta *gloss*. Siccome poi le parole della lingua sono per loro natura polisemiche (ovvero possiedono più di un significato) all'interno di WordNet ognuna di esse può appartenere a diversi synset.

Lo scopo del processo di terminology disambiguation è di associare ad ognuno dei termini che compaiono nello schema XML il corretto synset di WordNet. Nella prima versione di XML S^3MART tale associazione veniva stabilita mediante un procedimento semi-automatico in cui, tramite un'apposita interfaccia grafica, l'operatore umano poteva "annotare" ogni termine dello schema con il miglior candidato tra i termini disponibili in WordNet e selezionare poi il suo synset. Recentemente è stato poi introdotto un nuovo modulo detto **Strider (STRucture-based Information Disambiguation ExpeRt)** che, esplorando il contesto dei termini da disambiguare e sfruttando le informazioni fornite da WordNet, consente di eseguire tutto il processo in maniera completamente automatica. All'utente è comunque concesso di intervenire manualmente per raffinare ed eventualmente correggere le associazioni stabilite in maniera automatica.

Matching computation

Questa fase si occupa di eseguire l'effettiva operazione di matching tra gli schemi espansi ed annotati che sono stati prodotti dalle fasi precedenti. In particolare,

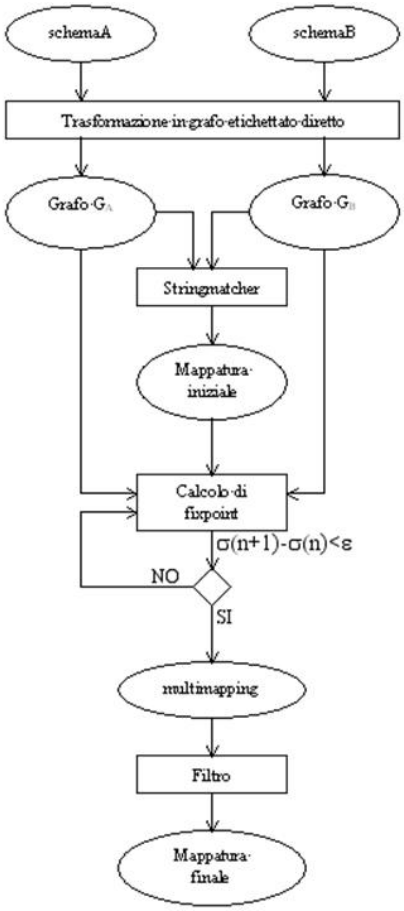

Figura 4.2: XML S^3MART: algoritmo per la matching computation.

gli schemi della collezione vengono ora processati a coppie e per ogni coppia viene individuato il miglior mapping tra gli elementi che vi compaiono, considerando sia le informazioni strutturali che semantiche ricavate in precedenza. É importante considerare entrambe queste tipologie di informazioni perché, se da un lato il significato dei termini dello schema fornisce già di per sé un'indicazione semantica del contenuto dei documenti che fanno riferimento a tale schema, dall'altro la componente strutturale consente di individuare per ogni termine il suo contesto, che è rappresentato dalla posizione del nodo corrispondente nell'albero XML.

Le operazioni che vengono eseguite per il calcolo di matching sono in parte derivate da quelle proposte nell'algoritmo di Similarity Flooding [44], i cui passi sono rappresentati in figura 4.2 e possono essere raggruppati in quattro fasi

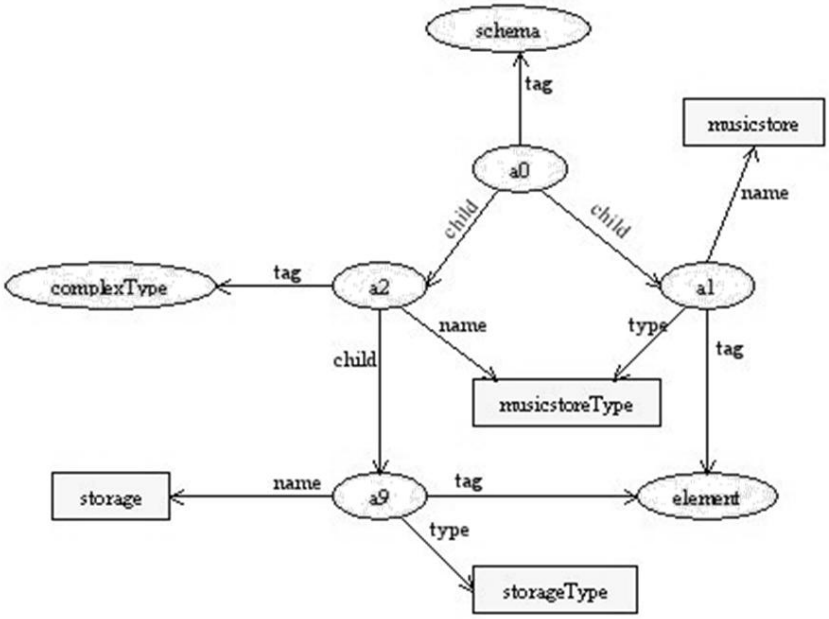

Figura 4.3: XML S^3MART: esempio di grafo etichettato diretto.

principali.

1. I due schemi XML considerati vengono per prima cosa convertiti in due **grafi etichettati diretti** seguendo le specifiche RDF [10]. Ogni arco di questi grafi è rappresentato da una tripla (s, p, o), in cui s ed o sono i nodi soggetto e oggetto, mentre p è l'etichetta dell'arco stesso. I nodi possono essere *entità* (rappresentati graficamente come ellissi) o *letterali* (rappresentati graficamente come rettangoli) mentre le etichette sugli archi che rappresentano i *predicati* possono essere di quattro tipi: *child*, *name*, *type* e *tag*. Ogni entità del grafo rappresenta un elemento o un attributo dello schema XML ed è identificato dal suo full path, ogni letterale rappresenta invece un particolare nome o un tipo primitivo che compaiono nello schema XML e che diversi elementi possono condividere. La struttura gerarchica dello schema XML viene rappresentata grazie agli archi con etichetta *child*. Un esempio di un frammento del grafo etichettato diretto ottenuto dallo schema di Listing 4.2 è rappresentato in figura 4.3.

   Dai due grafi etichettati diretti ottenuti per una coppia di schemi viene poi costruito un unico **grafo di connettività a coppie (Pairwise Connec-**

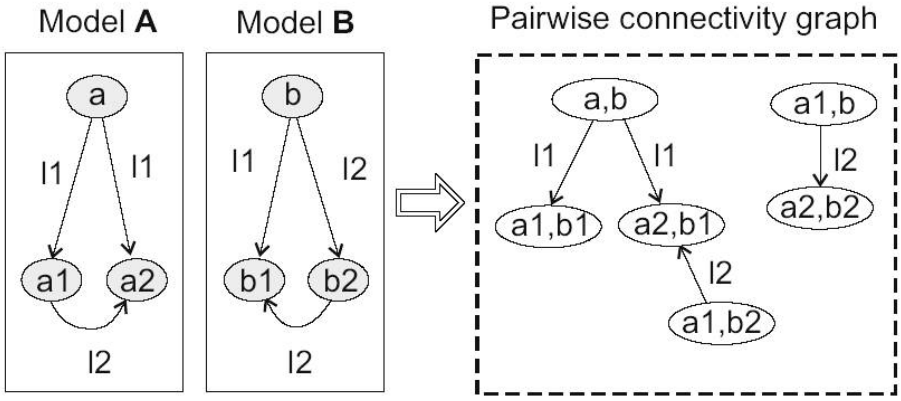

Figura 4.4: XML S^3MART: costruzione del Pairwise Connectivity Graph.

tivity Graph (PCG)) accoppiando i nodi che nei due grafi di partenza sono collegati dagli stessi archi. I nodi del grafo risultante sono quindi costituiti da coppie di elementi, uno per ogni grafo di partenza, e sono collegati tra loro solo se lo stesso arco collega tra loro anche i nodi originari nei grafi di partenza. Formalmente si ha che se A e B sono due grafi etichettati diretti, il loro grafo di connettività a coppie è definito come:

$$((x, y), p, (x', y')) \in PCG(A, B) \Leftrightarrow (x, p, x') \in A \ \wedge \ (y, p, y') \in B$$

Il procedimento è schematizzato in figura 4.4.

2. Successivamente per ogni coppia di elementi, corrispondente ad un nodo del grafo di connettività a coppie, viene calcolato un punteggio di similarità iniziale compreso tra 0 ed 1 che esprime la loro vicinanza semantica.

   Nel caso in cui i due elementi siano un'entità ed un letterale, oppure due entità diverse tra loro, tale punteggio viene impostato al valore minimo, mentre viene scelto il valore massimo se le due entità sono uguali. Se i due elementi sono letterali, a differenza di quanto avviene nell'algoritmo di Similarity Flooding in cui si procede semplicemente confrontando i loro prefissi e suffissi comuni, in XML S^3MART viene adottato un approccio più complesso che sfrutta la semantica dei termini degli schemi XML ricavata durante la fase di terminology disambiguation. Per calcolare tale punteggio si adotta infatti un approccio linguistico che quantifica la distanza tra i significati dei due elementi confrontando le gerarchie di ipernimi

di WordNet dei loro synset. Il punteggio per ogni coppia di termini (a, b) è così ottenuto calcolando il rapporto tra le profondità dei loro synset in tale gerarchia e la lunghezza del percorso che li collega, secondo la seguente formula:

$$\sigma^0(a, b) = \frac{2 \cdot depth(\ LCA(a, b)\ )}{depth(a) + depth(b)}$$

nella quale l'operatore *depth* indica la profondità del synset nella gerarchia di ipernimi di WordNet e *LCA (Lowest Common Ancestor)* identifica l'antenato comune ai due synset che si trova al livello più basso di tale gerarchia.

3. In seguito i valori di similarità iniziali, che riflettono la vicinanza semantica delle singole coppie di elementi, vengono raffinati applicando un algoritmo iterativo di fixpoint che inserisce nel calcolo le informazioni strutturali sugli schemi.

   Per applicare questo algoritmo viene per prima cosa costruita una struttura ausiliaria detta **induced propagation graph**. Come si può osservare in figura 4.5, questa struttura è ricavata dal grafo di propagazione a coppie semplicemente aggiungendovi, per ogni arco, un ulteriore arco diretto in verso opposto. Ad ogni arco viene poi associato un peso compreso tra 0 e 1 chiamato *coefficiente di propagazione* che indica quanto la similarità di una coppia di elementi influenza la similarità delle coppie vicine e viceversa e che, come è indicato in [44], può essere calcolato applicando diverse formule, a seconda della modalità di propagazione che si intende rappresentare. Ad esempio, l'approccio di figura 4.5 suppone che ogni tipologia di arco presente nel grafo, e identificata da una stessa etichetta e da uno stesso nodo sorgente, fornisca un contributo pari a 1.0: ogni singolo arco ha perciò un coefficiente di propagazione pari al quoziente tra 1.0 e il numero di archi di quel tipo uscenti dalla coppia.

   L'algoritmo di fixpoint procede quindi impiegando questa struttura per calcolare in modo iterativo i punteggi di similarità, basandosi sul principio che due nodi appartenenti a due schemi XML distinti sono tanto più simili

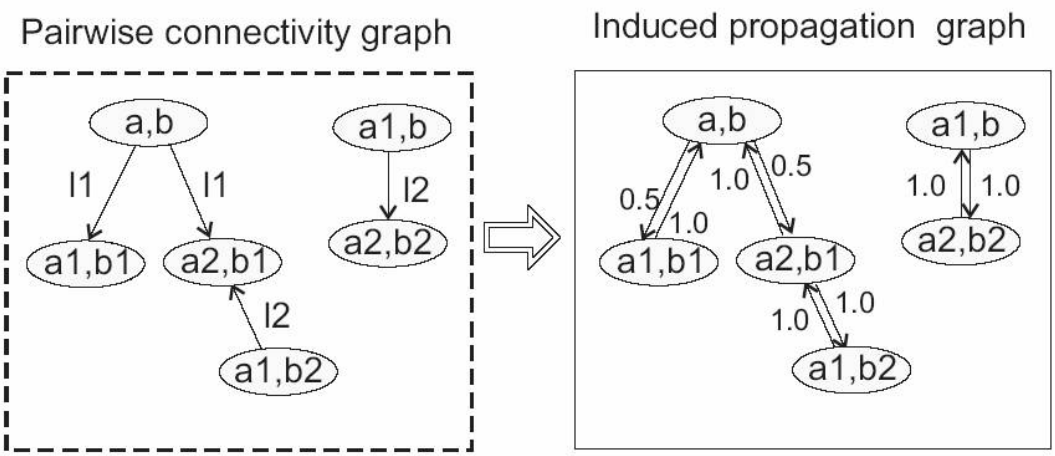

Figura 4.5: XML S^3MART: costruzione dell'Induced Propagation Graph.

quanto più lo sono i loro nodi adiacenti: la similarità di ogni coppia si propaga alle coppie vicine con un peso pari a quello espresso dal coefficiente di propagazione. Ad ogni iterazione il punteggio di similarità di ogni coppia viene così incrementato sommandogli i valori di similarità delle coppie vicine moltiplicati per i coefficienti associati agli archi che a loro li collegano. Tutti i punteggi calcolati vengono poi normalizzati dividendoli per il massimo valore di similarità trovato per l'iterazione corrente, in modo da ottenere dei risultati compresi tra 0 ed 1 che vengono poi utilizzati come valori di partenza per eseguire il calcolo all'iterazione successiva.

Formalmente, il punteggio di similarità tra i termini x ed y all'iterazione (i+1)-esima (indicato con $\sigma^{i+1}(x,y)$), viene calcolato a partire dal valore calcolato all'iterazione precedente $\sigma^i(x,y)$ secondo la formula (in cui per consentire una maggiore comprensibilità viene omessa l'operazione di normalizzazione):

$$\sigma^{i+1}(x,y) = \sigma^i(x,y) \; + \sum_{(a_u,p,x)\in A,(b_u,p,y)\in B} \sigma^i(a_u,b_u)\cdot w((a_u,b_u),(x,y)) \; + \sum_{(x,p,a_v)\in A,(y,p,b_v)\in B} \sigma^i(a_v,b_v)\cdot w((a_v,b_v),(x,y))$$

in cui con $w((a_u,b_u),(x,y))$ si indica la funzione che restituisce il coefficiente di propagazione per l'arco che collega la coppia (a_u,b_u) alla coppia

(x, y). Il calcolo viene eseguito in maniera iterativa fino a quando non si raggiunge la convergenza (tutti i punteggi si stabilizzano, ovvero la lunghezza del vettore Euclideo residuo $\Delta(\sigma^n, \sigma^{n-1})$ rimane sotto ad un certo fissato ϵ) oppure un numero massimo di iterazioni.

Come è indicato in [44], anche per la formula di fixpoint è possibile scegliere tra diverse alternative che permettono di tenere in maggiore o minore considerazione i valori di similarità calcolati inizialmente.

4. Poiché le fasi precedenti dell'algoritmo producono un insieme molto grande di possibili mapping, occorre infine applicare un apposito filtro per selezionare solo i risultati significativi. In XML S^3MART è stato scelto un filtro di "matrimonio stabile" in grado di garantire che, per ogni coppia di elementi (x, y), non esiste nessuna altra coppia (x', y') tale che x è più simile a y' di quanto non lo sia y e y' è più simile a x di quanto non lo si x'.

4.1.2 Automatic query rewriting

La fase di **automatic query rewriting** impiega i mapping ricavati tramite le precedenti operazioni di schema matching per riscrivere una data query, formulata rispetto ad uno schema source, sugli schemi target disponibili. Ad ogni riscrittura viene inoltre assegnato un punteggio, che esprime una misura della sua fedeltà rispetto alla query originaria.

Le fasi secondo cui si articola il processo di riscrittura sono le seguenti:

1. per prima cosa la query viene processata sostituendo tutti i path che compaiono nelle sue clausole FOR e WHERE con i loro corrispondenti full path;

2. tutti questi full path vengono poi automaticamente riscritti sostituendoli con i corrispondenti mapping forniti dalla fase di schema matching;

3. viene costruita una variabile da inserire nella clausola FOR che permette di collegare tutti i path riscritti e il cui valore è uguale al più lungo prefisso comune a tutti i path coinvolti;

4. ad ogni query riscritta viene infine assegnato un punteggio, che è calcolato come la media dei punteggi assegnati durante la matching computation ad ogni path riscritto.

Al momento XML S^3MART è in grado di supportare la riscrittura di query congiuntive che impiegano variabili, predicati e wildcards. Di seguito si riportano alcuni esempi nei quali le query originarie sono poste rispetto allo schema source A e vengono riscritte sullo schema target B di figura 3.10. Con la Query 1 (Listing 4.3) viene mostrato un esempio di riscrittura (Listing 4.4) che coinvolge path con wildcards.

Listing 4.3: XML S^3MART: Query 1

```
FOR     $x IN /musicStore
WHERE   $x/storage/*/compactDisk//singer = ``Elisa''
AND     $x/track/songTitle = ``Gift''
RETURN  $x/signboard/namesign
```

Listing 4.4: XML S^3MART: Query 1 riscritta

```
FOR     $x IN /cdStore
WHERE   $x/cd/vocalist = ``Elisa''
AND     $x/cd/tracklist/passage/title = ``Gift''
RETURN  $x/name
```

La Query 2 (Listing 4.5) mostra invece il comportamento della riscrittura nella gestione delle variabili. Come si può osservare in Listing 4.6, il valore della variabile non viene semplicemente tradotto e riportato direttamente nella riscrittura, ma viene invece ricostruito solo alla fine del processo, dopo che tutti i path sono stati sostituiti con i corrispondenti full path.

La Query 3 (Listing 4.7) mostra infine la gestione dei predicati che coinvolgono valori testuali. Come si può osservare in Listing 4.8, ogni volta che un mapping per un elemento contenente un valore coinvolge un elemento intermedio, i predicati espressi su questo elemento vengono riscritti in una clausola OR su tutti gli elementi che sono discendenti dell'elemento target corrispondente e che contengono un valore compatibile.

Listing 4.5: XML S^3MART: Query 2

```
FOR $x IN /musicStore/storage/stock/compactDisk/songlist/track
WHERE   $x/singer = ``Elisa''
AND     $x/songTitle = ``Gift''
RETURN  $x
```

Listing 4.6: XML S^3MART: Query 2 riscritta

```
FOR     $x IN /cdStore/cd
WHERE   $x/cd/vocalist = ``Elisa''
AND     $x/tracklist/passage/title = ``Gift''
RETURN  $x/tracklist/passage
```

Listing 4.7: XML S^3MART: Query 3

```
FOR     $x IN /musicStore
WHERE   $x/storage/stock/compactDisk = ``Gift''
AND     $x/location = ``Modena''
RETURN   $x
```

Listing 4.8: XML S^3MART: Query 3 riscritta

```
FOR     $x IN /cdStore
WHERE       ($x/cd/vocalist = ``Gift''
OR      $x/cd/title = ``Gift''
OR      $x/cdtracklist/passage/title = ``Gift'')
AND     ($x/address/city = ``Modena''
OR      $x/address/street = ``Modena''
OR      $x/address/state = ``Modena'')
RETURN  $x
```

4.2 Architettura P2P di riferimento

La presente Tesi di Laurea è stata sviluppata nell'ambito del progetto nazionale **WISDOM (Web Intelligent Search based on DOMain ontologies)**, un progetto di ricerca cofinanziato dal MIUR a cui collaborano, oltre all'Università

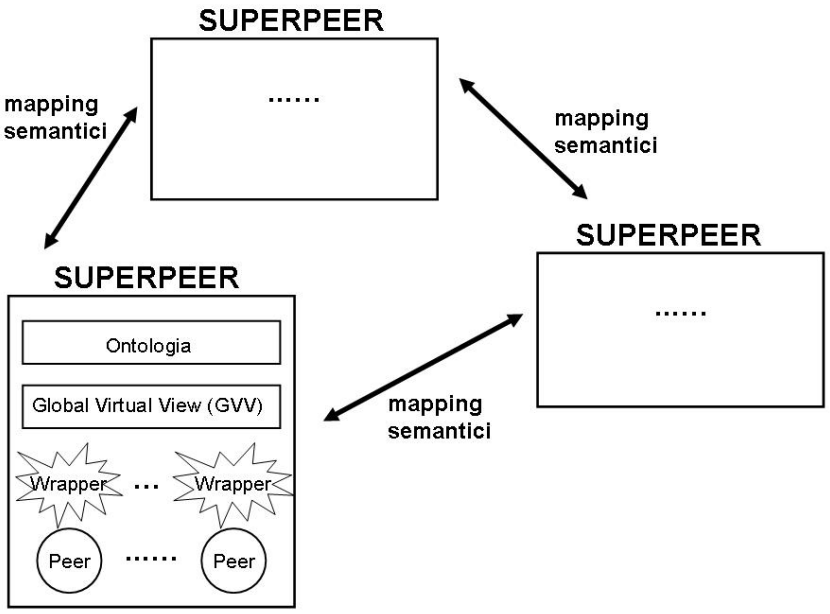

Figura 4.6: Schema del progetto WISDOM.

di Modena e Reggio Emilia, l'Università di Trento, l'Università di Bologna e l'Università di Roma. Il progetto si occupa della definizione di tecniche e strumenti per la ricerca, la localizzazione e la fruizione personalizzata di risorse informative accedibili via Web. Lo scopo è di costruire un'infrastruttura software in grado di consentire alle entità che vi partecipano di trarre vantaggio dalla grande quantità di informazioni che tramite il Web sono rese disponibili.

Le soluzioni proposte sono basate sull'adozione di ontologie di dominio e di un'architettura distribuita e decentralizzata che si ispira ai sistemi Peer-to-Peer e in particolare ai Peer Data Management System (PDMS) (si veda il capitolo 3). La proposta architetturale per WISDOM fa infatti riferimento ai sistemi parzialmente centralizzati (si veda il capitolo 2.1) poiché è strutturata come una rete di superpeer in cui il processo di integrazione semantica dei contenuti viene eseguito su due livelli:

- un primo livello di *integrazione forte*, che prevede l'integrazione semantica di un esiguo insieme di sorgenti (che rappresentano i singoli peer) in un unico nodo informativo (un superpeer);

- e un secondo livello di *integrazione debole*, basato sulla definizione di mapping semantici (si veda il capitolo 3) tra i concetti che sono propri dei vari superpeer.

La struttura di base è quindi quella classica di un sistema parzialmente centralizzato in cui ogni singolo peer risulta collegato ad un suo superpeer di riferimento, al quale invia le proprie interrogazioni e dal quale riceve i risultati. I vari superpeer sono poi collegati tra loro attraverso una specifica overlay network (si veda il capitolo 2), tramite la quale possono collaborare nel processare le interrogazioni a loro poste dai singoli peer. In questo modo le interrogazioni presentate da un nodo del sistema possono essere risolte facendo riferimento sia alle informazioni disponibili localmente presso il suo supernodo di riferimento, sia a quelle offerte da altri supernodi.

Ogni superpeer si occupa dunque in primo luogo di integrare i dati del sottoinsieme di peer a lui collegati ed è a tal fine costituito da un'architettura a strati *wrapper-mediatore* (figura 4.6). Lo strato di *wrapper* si occupa di mascherare le modalità di accesso fisico ai dati che sono proprie di ogni singolo peer: un wrapper è infatti un modulo software, specifico per una certa sorgente informativa, che opera sui suoi dati con lo scopo di consentirne un accesso trasparente rispetto alle modalità di rappresentazione adottate. Lo strato di *mediatore* si occupa invece di conciliare i dati forniti dai wrapper, e relativi ai nodi collegati ad un superpeer, in un'unica *GVV (Global Virtual View)*: lo scopo di questa GVV è di offrire una vista integrata, coerente e consistente delle informazioni presenti nelle sorgenti.

Ad ogni superpeer è poi associata una certa ontologia, che descrive l'offerta informativa del superpeer stesso in termini di entità concettuali e delle relazioni esistenti tra di esse. Attraverso tale ontologia vengono inoltre stabilite le relazioni con gli altri superpeer mediante la creazione di opportuni mapping semantici. Questi mapping sono direzionali, vengono cioè definiti da un superpeer source verso un superpeer target, ed esprimono le relazioni esistenti tra le loro diverse ontologie. Lo scopo è di consentire la riscrittura di query originariamente poste su un nodo source, e quindi espresse sulla sua ontologia, in query espresse sull'ontologia del nodo target e quindi da lui processabili.

La proposta di questa Tesi, che si inserisce nell'ambito del lavoro svolto dall'Università di Modena e Reggio Emilia, è di progettare e implementare le operazioni necessarie ai superpeer per collaborare realizzando un modulo che

comprende ed amplia le funzionalità offerte dal sistema XML S^3MART [43], che è stato presentato in precedenza. In particolare, tale modulo potrà essere utilizzato dai vari superpeer per creare i mapping semantici da impiegare nel processo di routing e risoluzione delle interrogazioni.

4.3 Modifiche al sistema XML S^3MART

Il primo problema incontrato nell'inserire XML S^3MART in un ambiente Peer-to-Peer è costituito dal fatto che nella sua versione originale il sistema processa gli schemi XML a coppie e, poiché il suo obiettivo principale è di identificare il miglior mapping possibile tra gli elementi che compaiono nei due schemi, si disinteressa del fatto che i punteggi di mapping ottenuti risultano relativi solamente alla coppia di schemi corrente. Questo significa che tali punteggi possono essere impiegati per confrontare la vicinanza dei termini solamente nell'ambito di una stessa coppia di schemi e non hanno invece significato se si riferiscono a coppie di schemi diverse.

Nel nuovo ambiente Peer-to-Peer introdotto risulta però di fondamentale importanza che i punteggi ottenuti nel calcolare il mapping dallo schema di un certo peer source verso gli schemi dei suoi vicini siano comparabili tra di loro, perché si vuole che il peer stesso possa impiegarli per decidere quale dei suoi vicini è in grado di approssimare meglio i suoi concetti.

In particolare, la motivazione della non comparabilità dei punteggi ottenuti con la versione originaria di XML S^3MART è da ricercare nell'operazione di normalizzazione che viene eseguita ad ogni passo di iterazione dell'algoritmo di similarity flooding. Come è stato spiegato nel capitolo 4.1, l'operazione di schema matching viene infatti eseguita considerando gli schemi a coppie ed applicando a tutte le coppie un algoritmo iterativo che ad ogni passo calcola i punteggi di mapping sulla base dei risultati ottenuti al passo precedente. Al termine di ogni iterazione tutti i punteggi correnti vengono inoltre normalizzati dividendoli per il loro valore massimo, in modo da ottenere dei valori appartenenti all'intervallo $[0, 1]$. L'algoritmo viene dunque iterato fino a quando non si raggiunge la convergenza (ovvero i valori calcolati in due iterazioni successive differiscono per una quantità minore di un certo fissato ϵ arbitrariamente piccolo), oppure un numero massimo di iterazioni. Lo pseudocodice dell'algoritmo è presentato in Listing 4.9.

Listing 4.9: Versione originaria dell'algoritmo di schema matching.

```
//Agoritmo di Schema Matching:
FOR (ogni schema source)
{
    FOR (ogni schema target)
    {
        applica algoritmo iterativo;
    }
}

//Algorimo Iterativo:
FOR (numero di iterazioni max)
{
    applica un passo algoritmo;
    normalizza i risultati del passo corrente;
    IF (convergenza raggiunta)
        break;
}
```

I valori finali per termini appartenenti a coppie di schemi diversi non sono quindi comparabili, perché i massimi impiegati nelle normalizzazioni a cui vengono sottoposti sono locali ad ogni coppia di schemi considerata e quindi diversi tra loro. Essi possono inoltre essere sottoposti ad un diverso numero di operazioni di normalizzazione, perché le varie coppie di schemi possono raggiungere la convergenza in istanti diversi.

Per ottenere dei punteggi confrontabili si è dunque pensato di modificare l'algoritmo di schema matching in modo che esso consideri contemporaneamente tutti gli schemi disponibili e possa normalizzare di conseguenza ad ogni iterazione tutti i punteggi impiegando gli stessi valori. In particolare, i punteggi che devono essere confrontabili sono solo quelli calcolati per uno stesso schema source, ovvero tutti quelli calcolati da un certo fissato peer verso tutti i suoi vicini. La nuova versione dell'algoritmo fissa quindi per prima cosa uno schema source e applica ogni passo dell'algoritmo iterativo a tutti gli altri schemi disponibili, preoccupandosi inoltre di volta in volta di individuare tra tutti i punteggi calcolati per quella iterazione il valore massimo locale alla coppia di schemi con-

siderata. I risultati ricavati ad ogni passo per tutti gli schemi target vengono poi normalizzati dividendoli per il massimo globale di quella iterazione, ovvero il valore massimo tra tutti i massimi locali ottenuti per i singoli schemi target alla stessa iterazione. In questo modo si ottiene che i valori impiegati ad ogni passo per la normalizzazione (i massimi globali) sono gli stessi per tutte le coppie caratterizzate dallo stesso schema source. Lo pseudocodice della nuova versione dell'algoritmo viene riportata in Listing 4.10.

Listing 4.10: Nuova versione dell'algoritmo di schema matching.

```
//Nuova versione Algoritmo di Schema Matching:
FOR (ogni schema source)
{
    FOR (numero di iterazioni max)
  {
    maxGlobale = 0;
    FOR (ogni schema target)
    {
        applica un passo algoritmo iterativo;
        maxLocale =
        max(punteggi di questa iterazione per coppia corrente);
        IF(maxLocale > maxGlobale)
            maxGlobale = maxLocale;
    }
    normalizza il passo corrente impiegando maxGlobale;
    IF (convergenza raggiunta per tutte le coppie di schemi)
        break;
  }
}
```

Come si può osservare, rispetto alla versione precedente è stato invertito l'ordine di esecuzione dei due cicli che consentono di iterare sul numero di passi dell'algoritmo e sugli schemi target. Inoltre in questa nuova versione la confrontabilità dei risultati è garantita anche grazie al fatto che tutte le coppie di schemi vengono sottoposte allo stesso numero di iterazioni dell'algoritmo, poiché il calcolo si arresta solo quando la convergenza è stata raggiunta per tutte le coppie considerate. Il fatto che per alcune coppie di schemi l'algoritmo continui

ad essere applicato anche dopo il raggiungimento della convergenza non crea problemi, dal momento che i risultati ottenuti nelle iterazioni successive a tale istante possono essere considerati uguali a meno di un ϵ.

I punteggi per lo stesso schema source ottenuti in questo modo risultano quindi confrontabili tra loro, tuttavia nel contesto di un ambiente Peer-to-Peer, che per sua natura è libero, dinamico e caratterizzato da un elevato numero di entità, non è possibile pensare di applicare un procedimento che prevede di dover processare tutti gli schemi dei partecipanti contemporaneamente. Si tratta infatti di un meccanismo troppo centralizzato, poiché comporta che tutti i punteggi vengano ri-calcolati ogni volta che un nuovo nodo entra a far parte del sistema o decide di apportare delle modifiche al proprio schema.

La soluzione proposta per risolvere questo nuovo problema si basa sull'osservazione che per ogni schema source, lo schema a lui più simile tra tutti quelli disponibili è sicuramente lo schema stesso. Allo stesso modo, per ogni concetto di un qualsiasi schema source, il concetto a lui più vicino è sicuramente rappresentato dallo stesso concetto di quello stesso schema. Per ogni concetto di ogni schema source, il mapping con punteggio più alto sarà dunque quello che, eseguendo l'operazione di schema matching dello schema con se stesso, associa il concetto con se stesso. L'operazione di schema matching di uno schema con se stesso viene chiamata *self schema matching* e il mapping che ne costituisce il risultato e che associa ogni concetto con se stesso è detto *self mapping*.

Processando tutti gli schemi insieme, se si inserisce anche lo stesso schema source tra gli schemi target, si ottiene dunque che ad ogni iterazione il massimo corrente da impiegare per la normalizzazione è rappresentato sicuramente dal punteggio di un self mapping (e precisamente dal self mapping che per quell'iterazione ha ottenuto il punteggio maggiore). Dal momento che questi valori dipendono dal solo schema source è dunque possibile per ogni nodo calcolarli a priori, prima di connettersi al sistema, in modo da averli a disposizione quando in seguito dovrà creare i mapping con i vicini.

L'algoritmo è stato quindi ulteriormente modificato separandolo in due parti, la prima delle quali si occupa di eseguire l'operazione di *self schema matching*, che è necessaria per la corretta esecuzione della seconda fase, in cui viene eseguito il reale processo di *schema matching*.

1. Fase di *Self Schema Matching*.
   La prima parte della nuova versione dell'algoritmo deve essere eseguita da

ogni nodo sul suo solo schema prima di stabilire un qualunque collegamento con altri peer e si occupa di calcolare il suo self schema matching. Questa parte opera in maniera del tutto analoga a quanto fa l'algoritmo di schema matching della versione originaria di XML S^3MART se ad esso viene dato in ingresso uno stesso schema come source e come target, con la fondamentale differenza che ad ogni passo di iterazione viene memorizzato il valore di massimo locale impiegato nell'eseguire la normalizzazione. Nel self mapping risultante ogni concetto dello schema viene dunque ovviamente associato con se stesso, ma eseguendo questa attività ogni peer acquisisce due informazioni di fondamentale importanza:

- i valori che ha impiegato ad ogni passo di esecuzione dell'algoritmo di similarity flooding per eseguire l'operazione di normalizzazione,
- e i punteggi finali di self mapping.

La prima informazione è importante perché permette al peer di calcolare tutti i suoi futuri mapping (ovvero quelli di cui sarà lo schema source) impiegando gli stessi valori nell'eseguire le operazioni di normalizzazione ed ottenere quindi dei risultati comparabili.

L'utilità della seconda informazione verrà dettagliata meglio in seguito, per ora si segnala solo che, dal momento che lo schema che meglio approssima lo schema del peer source è lo schema del peer source stesso, i punteggi che si ottengono eseguendo questo calcolo rappresentano il più alto punteggio di approssimazione possibile per ogni concetto.

2. Fase di *Schema Matching.*
   La seconda parte dell'algoritmo si occupa invece di eseguire il processo di schema matching tra lo schema source e gli schemi dei vicini. Come nella versione originaria del sistema, anche qui gli schemi vengono considerati a coppie e per ognuna di esse viene applicato il solito algoritmo iterativo che ad ogni passo calcola i punteggi di mapping sulla base dei risultati ottenuti al passo precedente. In questo caso però, al termine di ogni iterazione, la normalizzazione viene eseguita impiegando lo stesso valore che per quella stessa iterazione è stato impiegato dal peer source nell'eseguire il suo self schema matching e che durante tale fase è stato opportunamente memorizzato.

In questo modo, ogni volta che un nuovo peer decide di connettersi al sistema, i nodi già presenti possono impiegare l'algoritmo della fase di schema matching per creare i loro mapping verso il nuovo arrivato senza dover ri-calcolare anche gli altri loro mapping già esistenti verso altri nodi. La confrontabilità dei punteggi ottenuti per i nuovi mapping con quelli già presenti è poi garantita dal fatto che per calcolarli vengono impiegati gli stessi valori di normalizzazione. La versione ulteriormente modificata dell'algoritmo viene riportata in Listing 4.11.

Listing 4.11: Versione modificata del nuovo algoritmo di schema matching.

```
//Nuova versione Algoritmo di Schema Matching:
FOR (ogni schema source)
{
  vettoreNormalizzazione = self schema matching;
  FOR (ogni altro schema)
  {
    schema matching(vettoreNormalizzazione);
  }
}

//self schema matching:
FOR (numero di iterazioni max)
{
  applica un passo algoritmo iterativo;
  normalizza i risultati del passo corrente;
  vettoreNormalizzazione[passo corrente] =
            valore usato per normalizzare in questo passo;
}

//schema matching:
FOR (numero iterazioni max)
{
  applica un passo algoritmo iterativo;
  normalizza i risultati del passo corrente impiegando
        il valore in vettoreNormalizzazione[passo corrente];
}
```

Come si può osservare, analogamente a quanto avveniva nella versione prece-

dente, anche qui l'algoritmo viene iterato per tutte le coppie per lo stesso numero di passi, in modo da garantire che i punteggi finali siano pienamente comparabili. In questo caso però, poiché non si dispone a priori di tutte le coppie di schemi a cui deve essere applicato, non è possibile imporne la terminazione quando si è verificata la convergenza per ognuna di esse e si è quindi scelto di eseguirlo per un numero fissato di passi, visto che, come è stato spiegato in precedenza, eseguire un numero di iterazioni maggiori di quelle richieste per la convergenza non crea problemi.

Un'ulteriore considerazione sulla nuova versione del sistema riguarda il significato concettuale che viene assegnato ai punteggi di mapping. Siccome le due versioni hanno scopi e ambiti applicativi differenti, è infatti differente anche l'interpretazione dei risultati che forniscono.

Nella versione originaria di XML S^3MART lo scopo principale del sistema è di stabilire le associazioni migliori tra i termini che compaiono nei due schemi di volta in volta processati. Per questo motivo, l'operazione di normalizzazione che viene eseguita ad ogni passo dell'algoritmo iterativo prevede la divisione di tutti i punteggi relativi ad ogni iterazione per il loro valore massimo: procedendo in questo modo viene infatti attribuito ad ogni coppia un punteggio che esprime quanto i due concetti della coppia sono vicini rispetto ai concetti della coppia che, avendo ottenuto il punteggio maggiore, è formata dai concetti in assoluto più vicini tra loro rispetto a tutti gli altri.

Nella nuova versione del sistema ogni punteggio del mapping finale deve invece esprimere quanto il concetto dello schema target è vicino al corrispondente concetto dello schema source, in modo da fornire una descrizione numerica di quanto bene esso lo approssima. Per questo motivo i punteggi ricavati dall'esecuzione delle operazioni di schema matching non sono di per sé abbastanza significativi e devono invece essere interpretati sulla base di quella che è la migliore approssimazione possibile per ogni concetto dello schema source, che deve essere a sua volta essere impostata al valore 1. Come è stato spiegato in precedenza, la migliore approssimazione possibile per ogni concetto è costituita dal concetto stesso e i punteggi ottenuti dallo schema matching devono quindi essere interpretati come una frazione di quelli di self mapping che sono stati memorizzati da ogni peer source durante l'esecuzione della fase di self schema matching.

L'algoritmo finale deve quindi essere ulteriormente modificato in modo da inserire la parte che prevede la divisione dei risultati ottenuti dallo schema mat-

ching per i punteggi di self mapping dei concetti corrispondenti. Si noti che sottoponendo alla stessa operazione gli stessi punteggi di self mapping, essi vengono impostati al valore 1 e tale risultato è corretto poiché l'unità rappresenta la migliore approssimazione possibile, ovvero la certezza. Lo pseudocodice risultante è riportato in Listing 4.12.

Listing 4.12: Versione definitiva dell'algoritmo di schema matching.

```
//Nuova versione Algoritmo di Schema Matching:
FOR (ogni schema source)
{
  vettoreNormalizzazione = self schema matching;
  FOR (ogni altro schema)
  {
    schema matching(vettoreNormalizzazione);
    dividi punteggi schema matching
              per i corrispondenti punteggi di self mapping
  }
}

//self schema matching:
FOR (numero di iterazioni max)
{
  applica un passo algoritmo iterativo;
  normalizza i risultati del passo corrente;
  vettoreNormalizzazione[passo corrente] =
        valore usato per normalizzare in questo passo;
}

//schema matching:
FOR (numero iterazioni max)
{
  applica un passo algoritmo iterativo;
  normalizza i risultati del passo corrente impiegando
       il valore in vettoreNormalizzazione[passo corrente];
}
```

Capitolo 5

Routing di interrogazioni in un PDMS

Come è stato illustrato nel capitolo 4.1, il sistema XML S^3MART nella sua versione originaria è stato progettato per consentire la definizione di mapping e la risoluzione di interrogazioni nell'ambito di collezioni di documenti XML eterogenei, come i web repository o le digital library.

L'obiettivo della presente Tesi di Laurea, è invece di introdurlo in un ambiente Peer-to-Peer, in modo che i vari peer partecipanti al sistema possano impiegare le sue funzionalità per stabilire tra i loro schemi i mapping semantici (si veda il capitolo 3) necessari per poter collaborare nella risoluzione delle interrogazioni.

Un sistema Peer-to-Peer (si veda il capitolo 2) è infatti costituito da un numero potenzialmente elevato di unità di elaborazione connesse in rete autonome, indipendenti e verosimilmente eterogenee che decidono di condividere il loro contenuto informativo. Lo scopo è di permettere che le interrogazioni poste dagli utenti di ogni peer vengano risolte recuperando le informazioni utili possedute non solo da tale peer, ma anche da tutti gli altri partecipanti al sistema. Quando un utente presenta un'interrogazione ad un certo nodo, questo non si limita quindi a risolverla impiegando i dati disponibili localmente, ma la propaga agli altri nodi a cui è connesso in modo che anch'essi possano provvedere a loro volta a risolverla e propagarla. Il procedimento è schematizzato nell'esempio della figura 5.1, in cui è raffigurata una piccola porzione di una rete Peer-to-Peer formata dai dieci nodi identificati dalle lettere dall'alfabeto e connessi tra di loro nel modo rappresentato dalle linee. Come si può osservare dalla figura, la query Q che viene posta da un utente al nodo A, viene da lui propagata anche ai peer

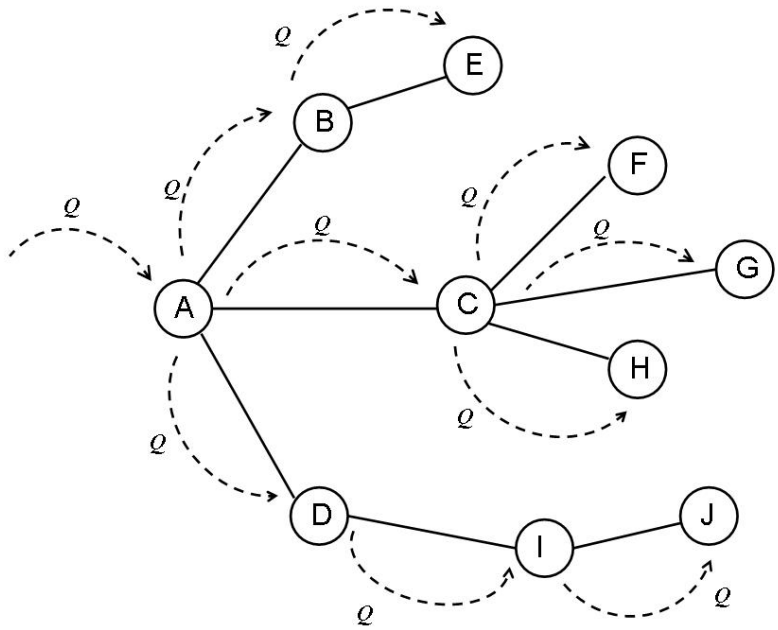

Figura 5.1: Propagazione delle interrogazioni in un sistema P2P.

B, C e D, che a loro volta fanno lo stesso per i loro vicini.

Come è stato illustrato nel capitolo 4, a causa dell'eterogeneità delle sorgenti coinvolte, perché siano in grado di collaborare è però necessario che tra di esse vengano stabiliti dei mapping semantici, che permettono ad ogni nodo di esprimere le corrispondenze esistenti tra i suoi contenuti e quelli dei peer vicini.

In particolare, nell'ambito di questa Tesi ci si pone in un ambiente Peer-to-Peer in cui si suppone che tutti i partecipanti siano caratterizzati da uno schema espresso in XML Schema. Si intende quindi impiegare la versione modificata del sistema XML S^3MART, realizzata durante la stesura della Tesi e presentata nel capitolo 4.3, in modo che ogni peer possa eseguire il processo di schema matching tra il suo schema e quello di ognuno dei suoi vicini, allo scopo di individuare il miglior mapping possibile per ognuno di essi ed impiegarlo per propagare verso di lui le interrogazioni.

In un contesto Peer-to-Peer bisogna poi osservare che non sempre risulta conveniente propagare un'interrogazione verso altri peer: coinvolgere peer che non trattano informazioni collegate a quelle richieste risulta infatti inefficiente sia perché in tal modo si rischia di sovraccaricare il richiedente con dati poco signi-

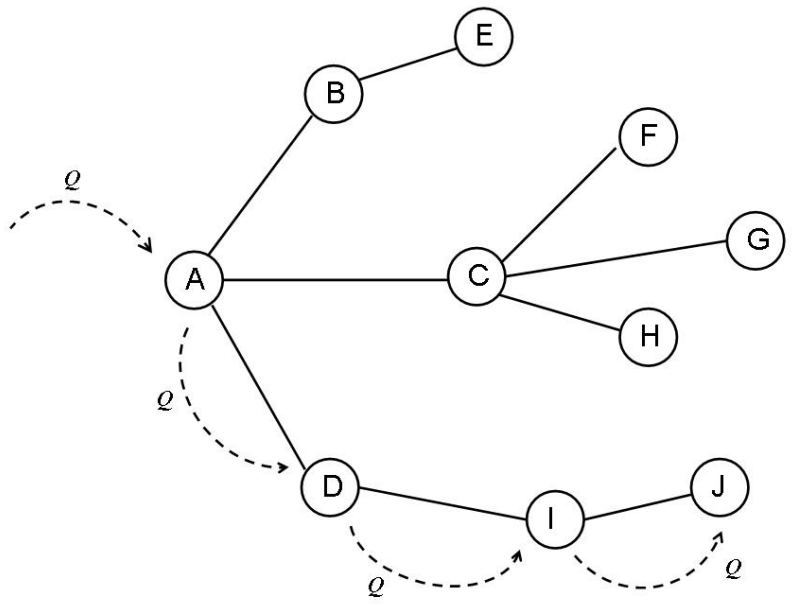

Figura 5.2: Meccanismo di query routing.

ficativi, sia perché tutto ciò comporta un'inutile e gravosa crescita del traffico di rete. Per evitare questa inefficienza occorrerebbero degli appositi meccanismi in grado di permettere ad ogni peer di selezionare, per ogni interrogazione pervenuta, quali tra i suoi vicini sono potenzialmente in grado di risolverla meglio. Sfruttando le informazioni fornite da questi meccanismi, ogni peer potrebbe così inviare ogni interrogazione solamente al sottoinsieme di peer suoi vicini ritenuti rilevanti per risolverla. Facendo riferimento all'esempio precedente, se il peer A fosse a conoscenza del fatto che, tra i suoi vicini, il solo peer a contenere informazioni rilevanti per risolvere la query Q è D, potrebbe propagare tale query solo verso di lui e, come può essere osservato nella figura 5.2, ne conseguirebbe una notevole riduzione del traffico di rete.

L'idea proposta in questo contesto è di re-interpretare i punteggi di mapping calcolati dal sistema XML S^3MART, in modo che essi possano essere impiegati da ogni peer per decidere quali suoi vicini sono in grado di approssimare meglio i concetti della query ed hanno quindi una probabilità maggiore di risolverla fornendo dati interessanti per l'utente.

Nella scelta dei nodi a cui propagare una query ogni peer dovrebbe inoltre

poter considerare come i vicini di questi nodi sono in grado a loro volta di approssimare i concetti coinvolti, poiché la query dovrà essere in seguito propagata anche a loro. Idealmente sarebbe quindi utile poter eseguire le operazioni di schema matching anche tra gli schemi di questi nodi e lo schema del nodo originario, in modo da identificare anche per loro i mapping con i relativi punteggi e poterli quindi impiegare nel processo di instradamento delle query. Tuttavia una soluzione di questo tipo non è chiaramente adottabile in un ambiente Peer-to-Peer, poiché procedendo in questo modo ogni peer dovrebbe inviare il suo schema ad ogni altro peer del sistema e calcolare inoltre i mapping verso ognuno di loro, con conseguenze disastrose in materia di traffico di rete prodotto e risorse computazionali impiegate.

La soluzione adottata prevede invece la costruzione per ogni nodo di un'apposita struttura dati locale, detta *indice di routing semantico*, in cui i punteggi di mapping del nodo stesso verso ognuno dei suoi vicini vengono combinati in modo opportuno con i punteggi che ognuno di questi vicini ha a sua volta verso i peer a cui lui è collegato. Questi ultimi punteggi provengono dagli indici di routing semantici dei vicini stessi e sono stati quindi ottenuti in modo del tutto analogo ai precedenti, realizzando così una sorta di meccanismo iterativo che consente la propagazione delle informazioni relative ai punteggi di mapping verso tutti i nodi del sistema. Grazie a queste strutture dati, ogni nodo ha quindi a disposizione un'idea generica di come ogni suo concetto può essere approssimato dalla porzione di rete cha ha origine a partire da un certo suo vicino e può quindi impiegare queste informazioni nella scelta dei nodi a cui propagare le interrogazioni in una sorta di meccanismo di "routing by mapping".

Nelle parti seguenti del capitolo vengono innanzitutto esposte nel dettaglio le problematiche che è necessario affrontare per implementare il procedimento di "routing by mapping". Sono poi introdotti gli indici di routing semantici e il modello matematico adottato per interpretarli, descrivendo inoltre le operazioni che devono essere eseguite per crearli e mantenerli aggiornati. Infine, viene presentato il nuovo modulo software che è stato realizzato per consentire la costruzione e l'utilizzo delle strutture di indice all'interno di un ambiente Peer-to-Peer. Tale modulo deve essere affiancato dalla versione modificata del sistema XML S^3MART presentata nel capitolo 4.3 e basa il suo funzionamento su un protocollo di interazione tra i peer che nel corso del capitolo viene opportunamente definito ed illustrato.

5.1 Routing by mapping

I punteggi di mapping, che ogni peer calcola eseguendo l'operazione di schema matching tra il suo schema (lo schema source) e ognuno dei suoi vicini (lo schema target), rappresentano un'indicazione numerica di come ogni concetto del suo schema viene semanticamente approssimato dal corrispondente concetto dello schema del peer vicino. L'idea alla base del meccanismo di *routing by mapping* è che ogni peer impieghi tali punteggi per propagare ogni query pervenuta solamente ai vicini che presentano un buon grado di approssimazione per i concetti richiesti dalla query stessa, poiché essi hanno verosimilmente una maggiore probabilità di fornire dati utili per l'utente.

Tuttavia bisogna osservare che, nella versione originaria del sistema XML S^3MART, tale punteggio viene calcolato operando solamente sugli schemi dei due peer in questione (il source e il target) e non prende in considerazione le possibilità di approssimazione dello schema target verso gli schemi degli altri peer che a lui sono collegati. Queste informazioni sono invece di fondamentale importanza nel nuovo ambiente Peer-to-Peer introdotto, poiché in tale ambito ogni nodo che riceve una query da un altro nodo deve preoccuparsi a sua volta di propagarla verso i suoi vicini.

In altre parole è come se, applicandoli ad un ambiente Peer-to-Peer, i punteggi della versione originaria di XML S^3MART fossero ricavati facendo un'assunzione di "mondo perfetto" e cioè supponendo che lo schema target sia in grado di approssimare a sua volta ogni suo concetto con fedeltà massima e pari a 1.0 sugli schemi dei suoi vicini. Procedere in questo modo risulta quindi corretto per il contesto di applicazione originario del sistema, ma richiede un'assunzione troppo forte in ambito Peer-to-Peer e i punteggi in questo modo calcolati non sono perciò significativi per implementare un corretto meccanismo di routing by mapping.

Occorre dunque modificare opportunamente il procedimento di calcolo dei punteggi di mapping, in modo che attraverso di essi ogni peer possa venire a conoscenza delle possibilità di approssimazione per un determinato concetto che, rivolgendosi ad un certo vicino, si hanno verso tutti i peer appartenenti alla porzione di rete cha da tale vicino ha origine. Questo tipo di conoscenza permette infatti ad ogni nodo di avere un'idea più veritiera della probabilità che ha di ottenere risultati utili instradando la query in una certa direzione.

Idealmente sarebbe quindi utile che ogni peer potesse eseguire il processo di schema matching anche verso i nodi componenti le sottoreti che hanno origine dai suoi vicini, in modo da disporre anche dei punteggi di mapping che mettono in relazione i suoi concetti con quelli di questi nodi. Tuttavia un procedimento di questo tipo non si presta ad essere adottato in un ambiente Peer-to-Peer, sia per il traffico di rete che deriverebbe dal meccanismo di scambio degli schemi, sia perché comporterebbe un elevato impegno computazionale da parte dei singoli peer per calcolare tutti i mapping necessari.

La soluzione proposta prevede invece che ogni peer si costruisca delle particolari strutture dati locali dette *indici di routing semantici*, compatte e semplici da aggiornare, per memorizzarvi le informazioni necessarie ad avere un'idea generica di come i vari concetti possono essere approssimati lungo le porzioni di rete che si dipartono dai suoi vicini. Al contrario di quanto accadrebbe se ogni nodo calcolasse tutti i mapping possibili, queste strutture non consentono dunque di prevedere con precisione il comportamento che si ottiene nel propagare una query ad un determinato vicino, ma, rispetto al precedente approccio, presentano il grande vantaggio di richiedere per la loro gestione l'impiego di molte meno risorse computazionali e di rete.

5.1.1 Modello matematico di riferimento

Come è stato illustrato nel capitolo 4.1.1, l'attività di schema matching del sistema XML S^3MART ha come risultato la produzione di una serie di mapping direzionali che mettono in relazione i concetti di uno schema source con quelli di uno schema target e assegnano ad ogni coppia un punteggio compreso tra 0 ed 1 che esprime la loro vicinanza semantica e strutturale. Tale punteggio può essere considerato una misura dell'errore che si commette nell'approssimare il concetto dello schema source con il corrispondente concetto dello schema target e stabilisce il grado di appartenenza di ogni coppia di concetti alla relazione di mapping definita tra i loro due schemi.

Questa situazione può essere modellata ricorrendo alla teoria della logica fuzzy [63], nell'ambito della quale è possibile interpretare ogni mapping come una relazione fuzzy definita sui due schemi e il punteggio di ogni coppia di termini come il grado di membership di tale coppia nella relazione.

Definizione 5.1 ***Relazione fuzzy.***
Siano $X = \{X_1, ..., X_p\}$ ed $Y = \{Y_1, ..., Y_q\}$ due insiemi non vuoti. Una relazione fuzzy R tra X ed Y è definita come un sottoinsieme fuzzy del prodotto cartesiano $X \times Y$.

Essa è quindi caratterizzata da una funzione μ_R, definita a sua volta da $X \times Y$ in $[0, 1]$, che ad ogni coppia (X_i, Y_j) associa un valore compreso tra 0 e 1 descrivente il valore di verità dell'espressione: "X_i è in relazione R con Y_j":

$$\mu_R : X \times Y \mapsto [0, 1]$$

Definizione 5.2 ***Grado di membership.***
Sia R una relazione fuzzy tra X ed Y. Allora $\mu_R(X_i, Y_j)$ è un valore appartenente all'intervallo $[0, 1]$ che viene interpretato come il grado di membership della coppia ordinata (X_i, Y_j) in R.

Nel contesto di un PDMS, i mapping vengono stabiliti tra coppie di peer adiacenti. In particolare un **mapping** $M(S_1, S_2)$, definito da uno schema source S_1 a uno schema target S_2 (non necessariamente distinti), associa ogni elemento di S_1 con l'elemento di S_2 che meglio gli corrisponde, sulla base di un punteggio appartenente all'intervallo $[0, 1]$ che esprime il grado di approssimazione semantica incontrato nel muoversi da S_1 a S_2.

Questa definizione può essere generalizzata al caso di un insieme di schemi di destinazione $\mathcal{S}$. In questo secondo caso il mapping $M(S_1, \mathcal{S})$, definito da S_1 a $\mathcal{S}$, associa ogni elemento di S_1 con un insieme di elementi appartenenti agli schemi contenuti in $\mathcal{S}$.

Formalmente, un **mapping generalizzato** può quindi essere interpretato come una relazione fuzzy:

$$M(S_1, \mathcal{S}) \subseteq S_1 \times_{S \in \mathcal{S}} S \cup \{null\}$$

dove ogni istanza $(C_1, \bar{C}_2)$ ha un grado di membership indicato con $\mu(C_1, \bar{C}_2) \in [0, 1]$.

Questa relazione fuzzy soddisfa le seguenti proprietà:

- è una **Funzione-0**,
  cioè per ogni $C_1 \in S_1$ esiste esattamente un $\bar{C}_2$ nel codominio tale che $\mu(C_1, \bar{C}_2) \geq 0$.

- è **Riflessiva**,
  ovvero, ponendo $S_1 = S_2$, per ogni $C_1 \in S_1$ si ha che $\mu(C_1, C_1) = 1$.

Si noti quindi che ogni concetto mappato sulla tupla nulla è un concetto non mappato e il suo grado di membership è di conseguenza irrilevante. Inoltre, il self mapping $M(S, S)$, stabilito tra uno schema e se stesso, corrisponde alla relazione identità.

5.1.2 Indici di routing semantici

Ispirandosi a quanto proposto in [14], si vuole che ogni peer possieda un suo *indice di routing semantico (SRI)* all'interno del quale possa memorizzare, per ogni elemento del suo schema e per ogni suo peer vicino, il grado di approssimazione che si ha per quel concetto verso la sottorete che ha origine da tale vicino. L'indice di routing semantico di un certo peer consiste quindi in una lista di vettori, ognuno dei quali è associato ad uno dei suoi vicini. I vettori che compongono l'indice di uno stesso peer hanno tutti una dimensione pari al numero di concetti dello schema di tale peer e ogni loro posizione è inoltre associata in maniera univoca ad ognuno di tali concetti. All'interno di ogni posizione del vettore relativo ad un certo vicino viene poi memorizzato il punteggio di mapping che esprime come il termine corrispondente a quella posizione può essere approssimato lungo i percorsi che hanno origine da quel vicino.

Formalmente, nel caso generico di un peer P collegato ad un certo numero n di vicini, numerati da 1 a n, si ha quindi che il suo indice di routing semantico è costituito da una matrice SRI_P, di $n+1$ righe (i vettori relativi ai vicini) e m colonne, una per ogni elemento dello schema di P (indicato con $S(P)$).

Per avere un esempio si consideri la piccola rete di peer rappresentata in figura 5.3 e costituita dai soli nodi A, B e C. L'indice di routing semantico di A è costituito da due vettori (che rappresentano le righe della matrice SRI_A), associati rispettivamente ai suoi vicini B e C, mentre sia B che C possiedono entrambi un solo vettore relativo al loro unico vicino A. Supponendo che lo schema

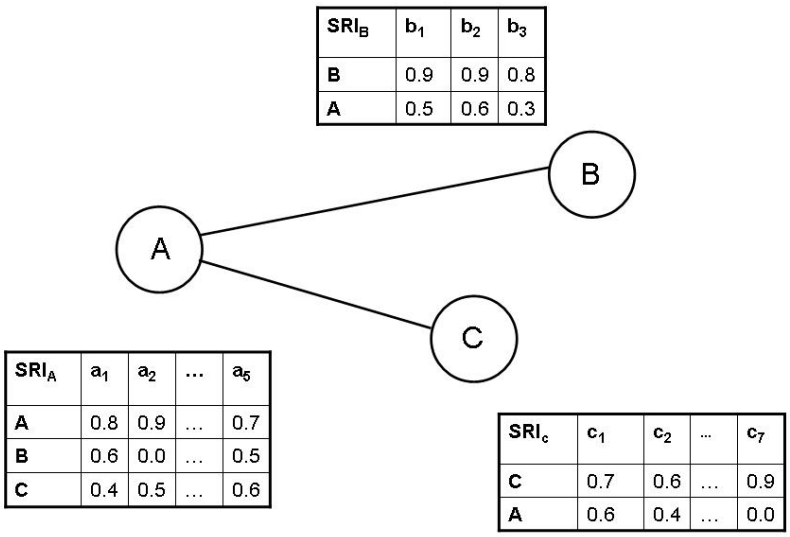

Figura 5.3: SRI: esempio.

del peer A contenga cinque elementi, i vettori del suo indice di routing semantico sono costituiti entrambi da cinque posizioni (le colonne della matrice SRI_A), ognuna relativa ad un preciso elemento: i cinque elementi dello schema A vengono cioè ordinati ed associati alle rispettive posizioni dei due vettori. Allo stesso modo si suppone che gli schemi dei peer B e C possiedano rispettivamente tre e sette concetti e i loro vettori contengono quindi altrettante posizioni associate in maniera univoca a tali concetti. In ognuno degli indici di routing semantici dei tre nodi è poi presente un'ulteriore vettore, inserito in prima posizione, che è associato al nodo stesso ed è adibito a contenere in ogni sua posizione il punteggio di self mapping (si veda il paragrafo 4.3) del concetto corrispondente.

I valori numerici che ogni peer inserisce all'interno di ognuno dei suoi vettori devono quindi rappresentare adeguatamente le possibilità di approssimazione che si hanno, per i concetti del suo schema, all'interno delle sottoreti che hanno origine da ogni corrispondente vicino. Dal punto di vista del modello matematico introdotto, ogni riga i di un SRI_A di un nodo A è dunque un fuzzy set, in cui $SRI_A[i][j]$ è il grado di membership dell'elemento $A_j \in S(A)$ e rappresenta il livello di approssimazione semantica per quell'elemento nella sottorete che ha

origine dall'i-esimo vicino. Un caso particolare è rappresentato dalla prima riga, poiché essa si riferisce al peer locale A ed ogni elemento $SRI_A[0][j]$ è quindi il grado di membership $\mu(A_j, A_j)$ in $M(S(A), S(A))$. Come è stato spiegato nel paragrafo 5.1.1, ognuno di questi valori assume il valore 1, tuttavia per le motivazioni spiegate nel capitolo 4.3, risulta utile memorizzare in questa riga i valori di self mapping precedenti all'operazione di normalizzazione che li divide per se stessi, in modo che ogni peer possa disporne per calcolare in modo corretto tutti i suoi mapping.

Nel caso semplificato della figura 5.3, i peer B e C non hanno altri vicini oltre ad A e da loro non ha quindi origine alcuna sottorete. I punteggi nell'indice di A devono perciò esprimere solo le possibilità di approssimazione dei suoi concetti verso i concetti degli schemi dei peer B e C. Tali valori corrispondono dunque ai punteggi di mapping che possono essere calcolati applicando il processo di schema matching alle coppie di schemi A - B e A - C. In particolare, per riempire il vettore relativo allo schema B, il peer A deve innanzitutto recuperare lo schema del peer B, in modo da poter poi applicare l'algoritmo di schema matching tra il suo schema (il source) e lo schema di B (il target). I punteggi ottenuti per il mapping risultante devono poi essere ordinati secondo l'ordine assegnato ai concetti di A, per essere infine inseriti nelle posizioni del vettore corrispondenti a tali concetti. Si osservi inoltre che, dal momento che lo schema di A possiede un numero di concetti maggiore rispetto allo schema di B, obbligatoriamente alcune posizioni del vettore contengono il valore 0.0, che sta a significare che quel concetto di A non è mappabile sullo schema di B e non esiste cioè alcun concetto a lui corrispondente in tale schema. La creazione dell'altro vettore di A deve essere eseguita in modo del tutto analogo, ma impiegando lo schema di C.

La situazione è più complessa per il calcolo degli indici di routing dei peer B e C. Il peer B ha un solo vicino rappresentato dal peer A e si potrebbe quindi pensare che, in modo del tutto analogo al caso precedente, esso debba inserire all'interno del suo vettore i valori risultanti dall'algoritmo di schema matching eseguito tra il suo schema e quello dello schema A. Tuttavia tale informazione non è completa, perché non esprime il fatto che il peer A è a sua volta connesso a C e quindi il peer B, attraverso il suo collegamento verso A, può raggiungere tutta la sottorete formata dai peer A e C. Perché B abbia un'idea completa di come i suoi concetti possono essere approssimati da tutta questa sottorete,

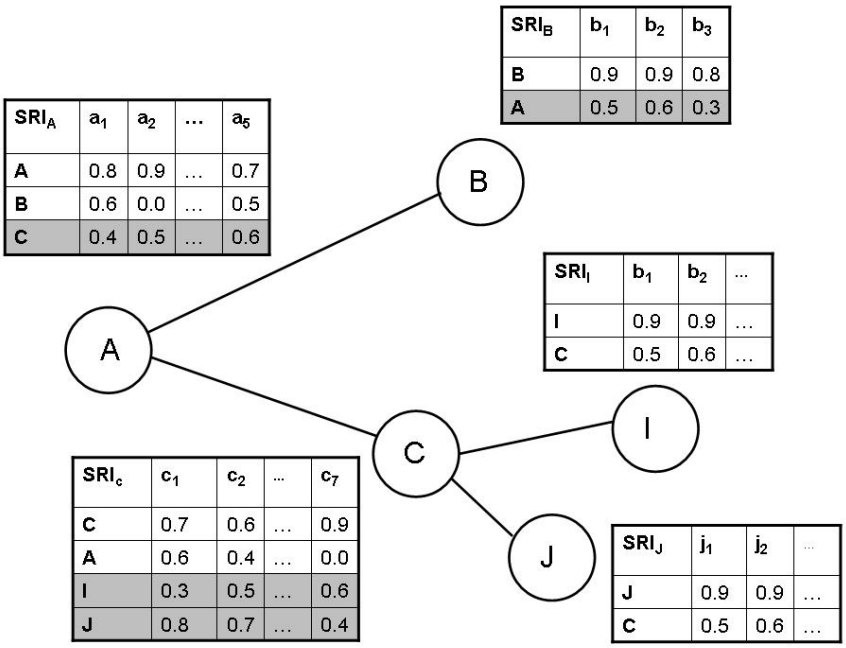

Figura 5.4: SRI: esempio.

deve quindi essere a conoscenza sia dei suoi punteggi di mapping verso il solo peer A, sia dei punteggi di mapping che il nodo A ha a sua volta verso C. La soluzione adottata prevede che B mantenga un'informazione riassuntiva di questi due punteggi, combinandoli in modo adeguato tramite l'impiego di un'opportuna funzione matematica di *composizione*, i cui risultati vengono inseriti nel vettore dell'indice di routing semantico di B. L'operazione matematica di composizione tra due vettori di punteggi di mapping ha il significato di esprimere che attraverso il peer che rappresenta il target del primo vettore è possibile raggiungere anche il peer che rappresenta il target del secondo vettore.

In modo del tutto analogo anche C mantiene all'interno del suo vettore un'informazione riassuntiva delle possibilità di approssimazione dei suoi concetti verso la sottorete formata dai peer A e B. Tale informazione è ottenuta in questo caso mediante la *composizione* dei punteggi di mapping che esso avrebbe verso il solo schema di A, con quelli che A ha verso il solo schema di B.

Un caso ancora più complesso è rappresentato in figura 5.4 in cui è raffigurata la stessa rete dell'esempio precedente, in cui però sono stati aggiunti due nuovi peer I e J, collegati entrambi al solo nodo C.

Il nuovo indice di routing semantico di C è quindi ora costituito da due ulteriori vettori, ognuno dei quali è relativo ad uno dei due nuovi peer e, poiché attraverso di essi C non può accedere ad alcuna sottorete, contiene i punteggi di mapping calcolati solo eseguendo l'algoritmo di schema matching tra le coppie di schemi C - I e C - J.

Anche l'indice di routing semantico di A è però diverso rispetto al caso precedente, perché esso deve essere ora a conoscenza del fatto che, attraverso C, può raggiungere anche i nodi I e J. In modo analogo a quanto avveniva in precedenza, si vuole dunque che esso possa inserire all'interno del suo vettore un'informazione riassuntiva delle possibilità di approssimazione dei suoi concetti all'interno della sottorete che ha origine da C. Per costruire tale informazione, A deve quindi disporre di due dati, in modo da poterli combinare impiegando la stessa funzione matematica di composizione sopra introdotta:

- il vettore dei punteggi di mapping che A ha verso il solo peer C, così come sono calcolati semplicemente applicando l'algoritmo di schema matching;
- un vettore di valori che esprima un'informazione riassuntiva di come i concetti di C possono essere approssimati dalla sottorete che ha origine da C e che comprende i due nodi I e J.

Quest'ultimo tipo di informazione deve descrivere il fatto che attraverso C è possibile raggiungere sia il peer I che il peer J e deve quindi essere implementata da una funzione matematica che calcoli l'*aggregazione* (detta anche *unione*) dei punteggi di mapping che C ha verso il solo peer I, con quelli che C ha verso il solo peer J. C può quindi fornire ad A questa informazione eseguendo l'aggregazione dei suoi vettori verso I e verso J, in modo che A possa poi comporla con il suo vettore verso C, per ottenere infine l'informazione riassuntiva desiderata.

5.1.3 Creazione degli indici di routing semantici

I processi di creazione e aggiornamento degli indici di routing semantici, che si ispirano in parte a quanto descritto in [14], devono essere tali da garantire che all'interno di ognuno di essi vengano inserite tutte le informazioni che sono state specificate nel paragrafo precedente.

Ovviamente tali indici sono poi utili se contengono delle informazioni corrette sullo stato corrente del sistema. Come è stato adeguatamente spiegato nel

capitolo 2, la topologia della rete di un sistema Peer-to-Peer non è però statica e può invece cambiare in qualsiasi istante in conseguenza della creazione di un nuovo collegamento tra due peer. La struttura degli indici di routing semantici deve quindi adattarsi a questa realtà ed essere tale da evolversi rispecchiando ciò che accade nel sistema. Per questo motivo, gli indici devono essere creati e aggiornati in modo incrementale ogni volta che si verifica un cambiamento nella rete.

Come si può facilmente intuire, l'indice di routing semantico di un peer appena creato e non ancora connesso a nessun altro nodo è costituito da un solo vettore (ovvero una matrice di una sola riga) contenente i punteggi di self mapping per ogni elemento del suo schema. Tale indice viene poi in seguito aggiornato ogni volta che il peer si connette ad un altro nodo, in modo da includere le informazioni relative alla nuova porzione di rete conosciuta.

In particolare, quando un peer (indicato ad esempio con A) si connette ad un altro nodo (indicato ad esempio con B), per prima cosa entrambi devono scambiarsi l'un l'altro le informazioni sul livello di approssimazione a cui si può accedere tramite le loro rispettive sottoreti. A questo scopo, ogni peer deve *aggregare* le righe del suo indice di routing semantico e inviarle, insieme al suo schema, all'altro nodo. Quando uno dei due peer, ad esempio A, riceve questo tipo di informazione da B, può calcolare il mapping $M(S(A), S(B))$ verso B ed estendere il suo indice di routing semantico aggiungendovi una nuova riga con l'identificativo di B. I valori da inserire in questa nuova riga sono ottenuti *componendo* i punteggi di approssimazione semantica dati dal mapping $M(S(A), S(B))$ appena calcolato, con l'informazione aggregata fornita da B. Un'operazione completamente analoga viene eseguita anche da B quando riceve le informazioni inviate da A.

Dopo aver esteso i propri indici di routing semantici, entrambi i peer devono poi informare i loro rispettivi vicini a riguardo del cambiamento avvenuto nella rete, in modo che anch'essi possano aggiornare di conseguenza i loro indici di routing. A tal fine ognuno dei due peer deve preoccuparsi di inviare ad ogni suo vicino un aggregato del suo indice di routing, dal quale è stata esclusa la riga relativa a quel vicino. Quando un vicino riceve questa informazione aggregata, può quindi aggiornare il proprio indice sostituendo la riga relativa al mittente dell'informazione con i nuovi valori.

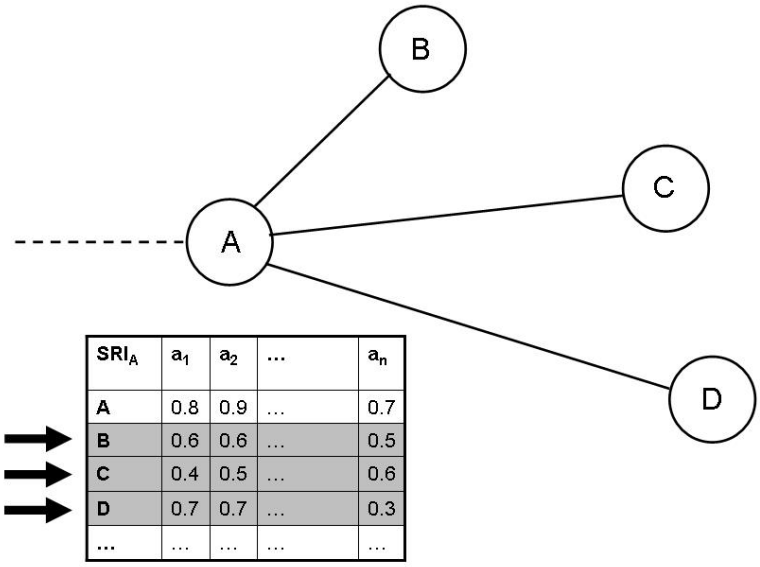

Figura 5.5: SRI: operazione di aggregazione.

Le operazioni di aggregazione e composizione

Descrivendo la struttura e il processo di creazione degli indici di routing semantici sono state introdotte due diverse operazioni, che possono essere eseguite sui vettori che contengono i punteggi di mapping per costruirne delle informazioni riassuntive: l'aggregazione e la composizione.

L'operazione di *aggregazione* viene impiegata nei casi in cui si vuole ottenere un'informazione riassuntiva per un certo numero di mapping che hanno in comune uno stesso schema source. Si consideri per esempio lo scenario rappresentato in figura 5.5 e si supponga che il peer A desideri costruire un'informazione riassuntiva di ciò che accade ai punteggi di mapping nella sottorete che da lui ha origine e che comprende i peer B, C e D. Il peer A deve a tal scopo selezionare dal suo indice di routing semantico i vettori relativi ai peer B, C e D per eseguire poi su di essi l'operazione di aggregazione (si noti che tali vettori sono tutti caratterizzati dallo stesso schema source che è quello di A stesso). Il significato dell'operazione di aggregazione è dunque di esprimere, in forma compatta tramite un solo vettore di punteggi, che un certo peer source è collegato a tutti

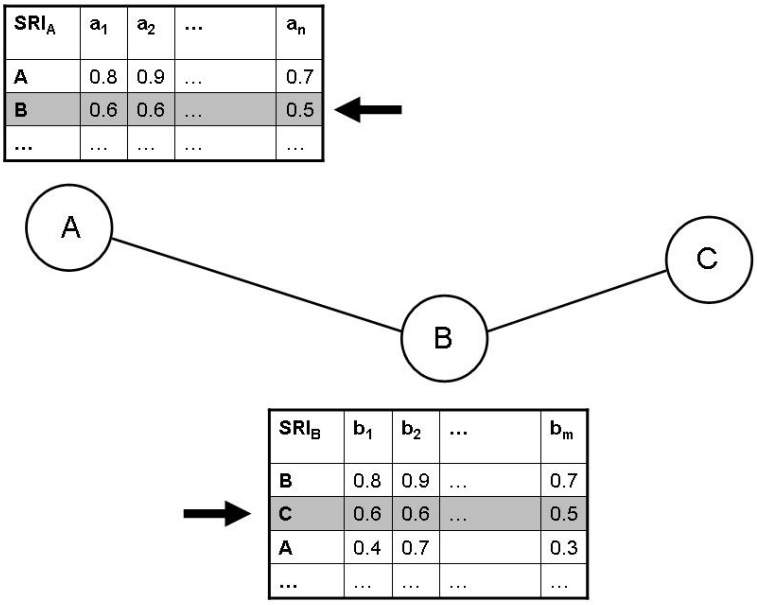

Figura 5.6: SRI: operazione di composizione.

i peer che rappresentano i target dei mapping che si vanno ad aggregare e che attraverso tale source è quindi possibile raggiungere tutti quei target.

La *composizione* viene invece impiegata quando si vuole creare un'informazione riassuntiva di due mapping caratterizzati dal fatto che lo schema target del primo mapping corrisponde allo schema source del secondo. Si consideri ad esempio lo scenario rappresentato in figura 5.6 e si supponga che il peer A desideri costruire un'informazione riassuntiva di ciò che accade ai punteggi di mapping nella sottorete che da lui ha origine e che comprende i peer B e C. Per ottenere tale informazione A deve quindi disporre di due informazioni:

- il vettore dei punteggi di mapping che A ha verso il solo peer B e che possono essere calcolati semplicemente applicando l'algoritmo di schema matching alla coppia di schemi A - B,
- il vettore dei punteggi di mapping che il peer B ha a sua volta verso il peer C, che A deve recuperare dall'indice di routing semantico di B, nel cui vettore relativo a C sono memorizzati i risultati dell'esecuzione dell'algoritmo di schema matching alla coppia di schemi B - C.

Una volta recuperati questi due vettori, A può quindi eseguire su di essi l'operazione di composizione (si noti che il primo mapping ha come schema target lo schema di B, che è anche lo schema source del secondo mapping) e memorizzare infine il risultato nel vettore relativo a B del suo indice di routing semantico. Il significato dell'operazione di composizione è quindi di esprimere, in forma compatta tramite un solo vettore di punteggi, che il peer target di un certo mapping è collegato ad un altro peer che attraverso tale peer target è possibile raggiungere.

Nei paragrafi seguenti vengono affrontati e approfonditi gli aspetti che riguardano l'effettiva implementazione matematica delle due funzioni presentate.

Implementazione dell'operazione di aggregazione

Come è stato illustrato nel paragrafo precedente, l'operazione di **aggregazione** deve essere eseguita tra due o più vettori di punteggi che hanno lo schema source in comune e sono quindi caratterizzati da uguali dimensioni corrispondenti al numero di concetti che compaiono in tale schema source. Ognuna delle posizioni dei due vettori è inoltre associata ad un concetto dello schema source e contiene il valore che esprime il punteggio di mapping per tale concetto verso lo schema target. Il risultato dell'operazione di aggregazione tra due vettori è quindi un vettore che ha le stesse dimensioni dei due vettori da aggregare e le cui posizioni sono associate agli stessi concetti associati alle posizioni dei vettori da aggregare. In ogni posizione esso deve poi contenere un valore numerico che esprime un'informazione riassuntiva di come il concetto corrispondente è approssimato dall'aggregazione (ovvero l'unione) dei due schemi target dei vettori di partenza. Tale valore numerico deve quindi essere calcolato a partire dai punteggi che sono riportati nei due vettori di partenza per quello stesso concetto, applicando una funzione matematica significativa per rappresentare l'operazione di aggregazione. Il procedimento è schematizzato in figura 5.7.

Formalmente si ha quindi che, dato l'indice di routing semantico del peer A SRI_A e una funzione di aggregazione g, la procedura di aggregazione opera sui fuzzy set corrispondenti alle righe di SRI_A e produce come risultato un fuzzy set strutturato come una tupla:

$$g(SRI_A) = [\mu_1, \ldots, \mu_m]$$

i cui valori sono i gradi di membership, ovvero i livelli di approssimazione, degli

m concetti che compongono lo schema di A, $S(A)$, verso la sottorete che dal peer A ha origine.

In particolare, ogni μ_j è poi ottenuto applicando la funzione di aggregazione g ai gradi di membership del concetto A_j contenuti nella colonna j-esima dell'indice SRI_A:

$$\mu_j = g(SRI_A[1][j], \ldots, SRI_A[n][j])$$

Si noti che l'operazione di aggregazione viene eseguita su tutti gli elementi della colonna j-esima dell'indice SRI_A, escludendo la posizione 0, poiché essa contiene il valore di approssimazione del concetto verso lo stesso peer A (il valore di self mapping). Questa procedura permette quindi ad A di ottenere l'aggregato delle informazioni contenute nel suo indice di routing semantico e relative a tutti i suoi vicini che deve inviare ai nodi con cui intende stabilire nuove connessioni.

Nel caso in cui A debba invece costruire un'informazione aggregata per aggiornare un nodo a cui è già connesso, la riga i-esima di SRI_A corrispondente al vicino a cui deve essere inviato l'aggiornamento deve ovviamente essere esclusa dall'aggregato e quindi si ottiene:

$$\mu_j = g(SRI_A[1][j], \ldots, SRI_A[i-1][j], SRI_A[i+1][j], \ldots, SRI_A[n][j]).$$

Si osservi che il tutto può essere interpretato anche in termini delle relazioni fuzzy che sono rappresentate dai fuzzy set che vengono aggregati. Si consideri a tal fine il caso generico di un peer A collegato ad un certo numero n di peer vicini $P_1, P_2, \ldots, P_n$. I fuzzy set che corrispondono alle righe di SRI_A altro non sono che i gradi di membership delle relazioni fuzzy di mapping:

$$M(S(A), S(A)), M(S(A), S(P_1 \downarrow)), \ldots, M(S(A), S(P_n \downarrow))$$

dove con $P_i \downarrow$ si indica il mapping dell'i-esimo vicino P_i verso tutta la sottorete che da P_i ha origine.

Il mapping aggregato $M(S(A), S(A \downarrow))$ è quindi il mapping dato dall'unione di tutti i mapping dei vicini di A verso le loro sottoreti:

$$\{A_j \mapsto \bigcup_{P \in A\downarrow} M(A, P)(A_j) \mid A_j \in S(A)\}$$

che associa ad ogni elemento A_j di $S(A)$ un grado di membership determinato applicando un'opportuna funzione matematica di aggregazione g alle funzioni di membership dei mapping da aggregare.

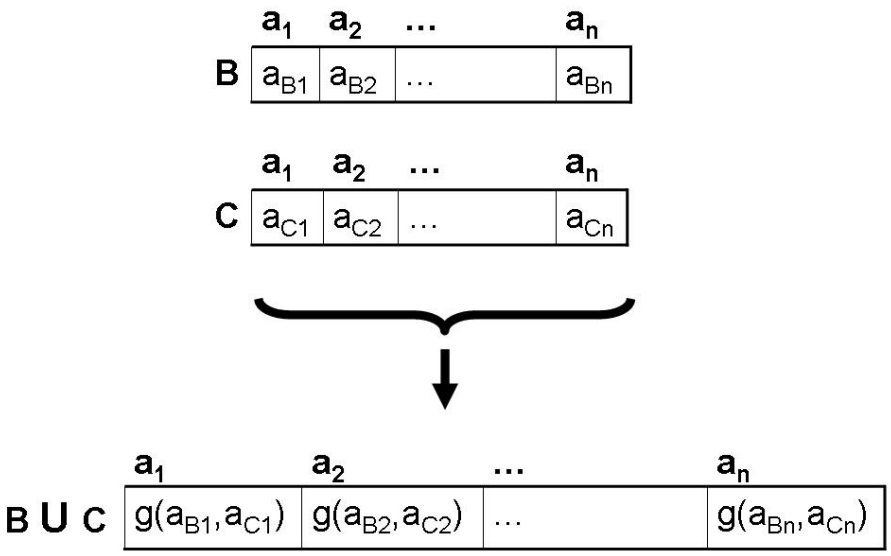

Figura 5.7: SRI: implementazione dell'operazione di aggregazione.

Per quanto riguarda la specifica funzione matematica di aggregazione g da impiegare nel calcolo dei punteggi risultanti sono possibili diverse alternative.

Perché tale funzione rappresenti in maniera adeguata il significato dell'aggregazione, per essa devono comunque essere soddisfatte le proprietà elencate di seguito:

1. g è una funzione *limitata*, ovvero:
   $g(0, \ldots, 0) = 0$ e $g(1, \ldots, 1) = 1$

2. g è una funzione *monotona*, ovvero:
   per ogni coppia di tuple $(a_1, \ldots, a_n)$ e $(b_1, \ldots, b_n)$ tali che $a_i, b_i \in [0, 1]$ per ogni $i \in [1, n]$, se $a_i \leq b_i$ per ogni $i \in [1, n]$ allora $g(a_1, \ldots, a_n) \leq g(b_1, \ldots, b_n)$.

3. g è una funzione *continua*.

Inoltre, essendo una funzione di aggregazione che viene applicata a fuzzy

set, per essa devono valere due ulteriori proprietà che caratterizzano il significato della nozione di aggregazione in tale ambito:

4. g è una funzione *simmetrica* per tutti i suoi argomenti,ovvero:
   $g(a_1, \ldots, a_n) = g(a_{p(1)}, \ldots, a_{p(n)})$ per ogni permutazione p su $[1, n]$

5. g è una funzione di *idempotenza*, ovvero:
   $g(a, \ldots, a) = a$ per ogni $a \in [0, 1]$

Nell'ambito della logica fuzzy il significato di aggregazione viene infatti reso attraverso l'operazione di unione, che secondo le regole semantiche classiche fuzzy è definita come segue:

Definizione 5.3 ***Unione.***
Siano R ed S due relazioni fuzzy. Il risultato dell'unione di R ed S è dato da:

$$(R \vee S)(X_i, Y_j) = \max \{\mu_R(X_i, Y_j),\ \mu_S(X_i, Y_j)\}$$

Tuttavia, la scelta della funzione di *max( )* non è l'unica possibile e in letteratura sono state fatte molte altre proposte di regole che assegnano un grado di membership all'unione fuzzy sulla base dei gradi degli operandi. In particolare, in [19] viene presentata una classe di funzioni che rispettano le proprietà sopra elencate e si prestano quindi ad essere impiegate per implementare l'operazione di aggregazione: le *co-norme triangolari* [18].

Definizione 5.4 ***Co-norma triangolare*** *Una co-norma triangolare è una funzione di aggregazione binaria s che soddisfa le seguenti proprietà.*

- ***$\vee$-Conservazione:***
  $s(1, 1) = 1$
  $s(x, 0) = s(0, x) = x$

- ***Monotonia:***
  $s(x_1, x_2) \leq s(x'_1, x'_2)$ *se* $x_1 \leq x'_1$ *e* $x_2 \leq x'_2$

- ***Commutatività:***
  $s(x_1, x_2) = s(x_2, x_1)$

- ***Associatività:***
  $s(s(x_1, x_2), x_3) = s(x_1, s(x_2, x_3))$

Di seguito vengono riportati alcuni esempi di specifiche funzioni che sono co-norme triangolari [45].

Maximum: $s(x_1, x_2) = \max\{x_1, x_2\}$

Drastic sum: $t(x_1, x_2) = \begin{cases} \max\{x_1, x_2\} & \text{, se } \min\{x_1, x_2\} = 0 \\ 1 & \text{, altrimenti.} \end{cases}$

Bounded sum: $s(x_1, x_2) = \min\{1, x_1 + x_2\}$

Einstein sum: $s(x_1, x_2) = (x_1 + x_2)/(1 + x_1 \cdot x_2)$

Algebraic sum: $s(x_1, x_2) = x_1 + x_2 - x_1 \cdot x_2$

Hamacher sum $s(x_1, x_2) = (x_1 + x_2 - 2 \cdot x_1 \cdot x_2)/(1 - x_1 \cdot x_2)$

Le funzioni di aggregazione fino ad ora presentate sono tutte binarie, ma nel caso sia necessario gestire operazioni di unione che coinvolgono più di due elementi, è possibile impiegarle per costruire appropriate funzioni di aggregazione con cardinalità m. Una funzione di aggregazione con cardinalità m è infatti una funzione definita da $[0, 1]^m$ in $[0, 1]$ e si può quindi pensare di crearla iterando una funzione binaria associativa. Ad esempio, nel caso in cui la congiunzione binaria sia definita dalla funzione di aggregazione binaria s, la congiunzione con molteplicità 3 può essere definita da $s(s(x_1, x_2), x_3)$. In particolare, se la funzione di aggregazione con molteplicità m si ottiene iterando una co-norma triangolare, per essa continuano a valere le proprietà presentate in precedenza.

Implementazione dell'operazione di composizione

Il caso dell'operazione di **composizione** è più complesso rispetto al precedente perché non si tratta di un'operazione che può essere eseguita direttamente tra fuzzy set come avviene per l'aggregazione, ma di un'operazione che coinvolge due relazioni fuzzy definite su domini differenti. Mentre l'aggregazione può quindi essere applicata direttamente alle colonne dell'indice di routing semantico che sono immediatamente disponibili, la composizione deve essere invece applicata alle relazioni che sono rappresentate da tali colonne. In particolare, facendo riferimento alla figura 5.6, le due relazioni fuzzy da comporre sono rappresentate da due mapping che hanno la forma:

- $M_1 = M(S(A), S(B))$, che rappresenta il mapping stabilito dal peer A verso il peer B;
- $M_2 = M(S(B), S(B \downarrow))$, che rappresenta il mapping del peer B aggregato verso tutta la sottorete che da lui ha origine (indicata col simbolo $B \downarrow$).

Il risultato dell'operazione di composizione è quindi una relazione fuzzy

$$M(S(A), S(B \downarrow)) \subseteq S(A) \times S(B \downarrow)$$

che associa ad ogni coppia di elementi $(C_1, \bar{C}_2)$ un grado di membership appartenente all'intervallo $[0, 1]$ sulla base di quanto stabilito da una funzione di membership:

$$\mu_{M_1 \circ M_2}(C_1, \bar{C}_2) : \ S(A) \times S(B \downarrow) \mapsto [0, 1]$$

che opera applicando una opportuna funzione matematica f alle funzioni di membership μ_{M_1} e μ_{M_2} delle relazioni da comporre, secondo lo schema seguente:

$$\mu_{M_1 \circ M_2}(C_1, \bar{C}_2) = f(\mu_{M_1}(C_1, x), \mu_{M_2}(x, \bar{C}_2))$$

dove x è un elemento dello schema $S(B)$.

L'operazione di composizione viene quindi eseguita tra due vettori caratterizzati dal fatto che lo schema target del primo vettore è anche lo schema source del secondo. Il vettore risultato della composizione di due vettori è un vettore che, per numero di posizioni e concetti ad esse associati, è uguale al primo vettore da comporre. All'interno di ogni posizione esso deve poi contenere un valore numerico che è determinato dalla funzione di membership $\mu_{M_1 \circ M_2}$ ed esprime come il

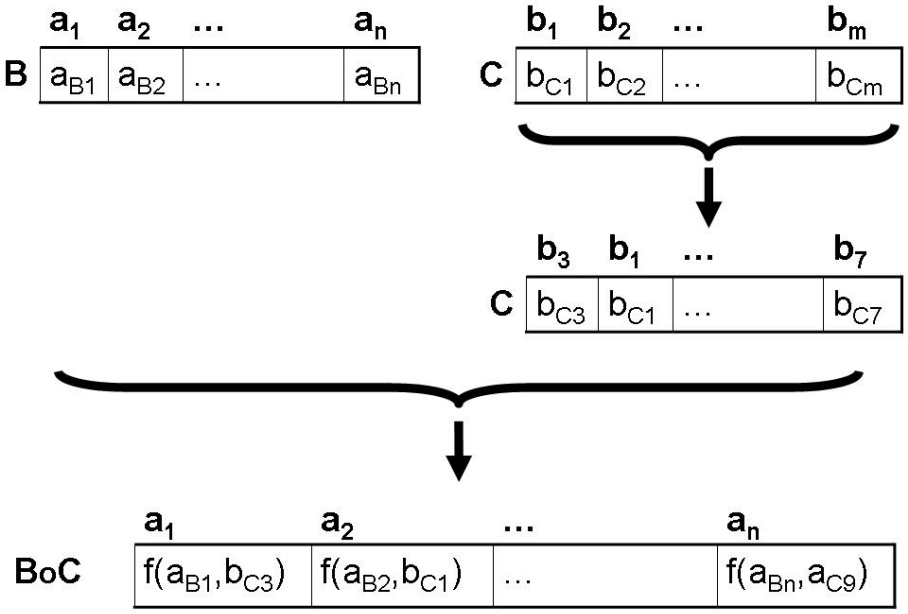

Figura 5.8: SRI: implementazione dell'operazione di composizione.

concetto corrispondente è approssimato lungo il percorso composto dallo schema target del primo vettore e lo schema target del secondo vettore . Tale valore numerico per ogni posizione deve quindi essere calcolato a partire dai punteggi che si trovano:

- nella stessa posizione del primo vettore,
- nella posizione del secondo vettore associata al concetto su cui è mappato, tramite il mapping del primo vettore, il concetto di cui si sta calcolando il valore.

Per poter eseguire l'operazione tra i valori giusti occorre quindi che il secondo vettore venga riordinato in modo da renderlo allineato all'ordine con cui i suoi concetti source compaiono come target del primo vettore. In questo modo i valori risultanti possono poi essere calcolati applicando alla stessa posizione di entrambi i vettori una opportuna funzione matematica f che sia significativa per rappresentare la composizione. Il procedimento è schematizzato in figura 5.8.

Per quanto riguarda la specifica funzione matematica f da impiegare nel calcolo dei punteggi risultanti sono possibili anche in questo caso diverse alternative.

perché tale funzione rappresenti in maniera adeguata il significato della composizione, per essa devono comunque essere soddisfatte le proprietà elencate di seguito:

1. f è una funzione *limitata*, ovvero:
   $f(0, \ldots, 0) = 0$ e $f(1, \ldots, 1) = 1$

2. f è una funzione *monotona*, ovvero:
   per ogni coppia di tuple $(a_1, \ldots, a_n)$ e $(b_1, \ldots, b_n)$ tali che $a_i, b_i \in [0, 1]$ per ogni $i \in [1, n]$, se $a_i \leq b_i$ per ogni $i \in [1, n]$ allora $f(a_1, \ldots, a_n) \leq f(b_1, \ldots, b_n)$.

3. f è una funzione *continua*.

   Inoltre, essendo una funzione di composizione che viene applicata a relazioni fuzzy, per essa deve valere un'ulteriore proprietà che caratterizza il significato della nozione di composizione in tale ambito:

4. f è una funzione di *max-min composition*, ovvero:
   data una coppia di tuple $(a_1, \ldots, a_n)$ e $(b_1, \ldots, b_n)$, tali che $a_i, b_j \in [0, 1]$ per ogni $i, j \in [1, n]$, si ha che $f(a_i, b_j) \leq max(a_i, b_j)$

Nell'ambito della logica fuzzy, l'operazione di composizione tra relazioni viene infatti definita nel modo seguente:

Definizione 5.5 ***Composizione max-min.***
Si considerino i tre fuzzy set:

$$X = \{X_i\}_{i=1,\ldots,p} \ , \ Y = \{Y_i\}_{i=1,\ldots,q}, \ Z = \{Z_i\}_{i=1,\ldots,r}$$

e le due relazioni fuzzy R ed S, definite rispettivamente sugli insiemi $X \times Y$ e $Y \times Z$:

$$R \subseteq X \times Y$$
$$S \subseteq Y \times Z$$

La relazione fuzzy composta $R \circ S$ è data dalla funzione di membership $\mu_{R \circ S}$, definita su $X \times Z$ come:

$$\mu_{R \circ S}(X_i, Z_k) = max_j(min(\mu_R(X_i, Y_j), \mu_S(Y_j, Z_k)))$$

Si osservi che l'ultima proprietà presentata è di importanza fondamentale nel contesto di un PDMS, perché consente di esprimere il fatto che, tanto più ci si allontana attraverso la rete dal peer che possiede dei determinati concetti, tanto più il livello di approssimazione per tali concetti aumenta (e quindi il punteggio di approssimazione diminuisce).

Nella scelta della funzione f è tuttavia possibile considerare un'ulteriore componente e cioè il fatto che le operazioni di aggregazione e composizione hanno un significato simile a due operatori dell'algebra relazionale classicamente impiegati nell'ambito dei database: la composizione corrisponde infatti all'operatore di *join*, mentre l'aggregazione è assimilabile all'operatore di *unione*. In particolare, dal momento che tali operazioni devono essere applicate in un contesto fuzzy, si può dedurre che è possibile implementarle in modo analogo a quanto viene fatto per i corrispondenti operatori nell'ambito dei cosiddetti *database fuzzy* [51, 6, 61].

I database fuzzy costituiscono un'interessante area di ricerca che combina la teoria della logica fuzzy con la tecnologia delle basi di dati, allo scopo di ampliare l'applicabilità dei database tradizionali, introducendo tecniche che siano in grado di trattare con informazioni incerte e imprecise. La necessità di disporre di questo tipo di strumenti è molto forte negli ultimi anni a causa della disponibilità sempre maggiore di archivi di dati complessi, come database di dati semistrutturati o multimediali, il cui contenuto semantico non può essere definito in modo preciso [20, 11]. L'idea alla base dei database fuzzy è quindi di estendere i database tradizionali tramite l'introduzione di opportuni modelli per i dati e appropriati linguaggi di interrogazione che si rifanno alla logica fuzzy per catturare l'incertezza delle informazioni semantiche di questi dati complessi.

Come è illustrato in [50], i database fuzzy sono quindi costituiti da *relazioni fuzzy*, ognuna delle quali è a sua volta composta da una insieme di *tuple* a cui è assegnato un grado di membership appartenente all'intervallo $[0, 1]$. Tale grado di membership viene associato ad ogni tupla sulla base di quanto stabilito da una *funzione di membership* μ, che denota quanto la tupla in questione è rappresentativa per la relazione specificata. In [50] viene anche presentata una definizione formale della semantica dei vari operatori che possono essere impiegati in questo ambito. In particolare, per quanto riguarda gli operatori di join ed unione, viene specificato che essi operano sulle tuple a cui vengono applicati in

maniera del tutto analoga ai corrispondenti operatori dell'algebra relazionale; alla tupla risultante deve poi essere assegnato un grado di membership che può essere calcolato nel modo seguente:

- per l'operatore di *JOIN* il punteggio della tupla risultante è la congiunzione (ovvero l'intersezione) dei punteggi delle varie tuple,
- per l'operatore di *UNIONE* il punteggio della tupla risultante è invece calcolato come la disgiunzione (ovvero l'unione) dei punteggi delle tuple componenti.

L'interpretazione dell'operazione di aggregazione coincide quindi completamente con quella presentata in precedenza, mentre per l'operazione di composizione viene introdotta l'operazione di congiunzione fuzzy che, secondo le regole semantiche classiche, è definita nel modo seguente:

Definizione 5.6 ***Congiunzione.***
Siano R ed S due relazioni fuzzy definite tra gli insiemi X ed Y. Il risultato della congiunzione di R ed S è dato da:

$$(R \wedge S)(X_i, Y_j) = \min \{\mu_R(X_i, Y_j),\ \mu_S(X_i, Y_j)\}$$

Anche in questo caso, la scelta della funzione di *min()* non è l'unica possibile e in letteratura è possibile ritrovare molte altre proposte. In particolare, in [19] viene presentata una classe di funzioni che rispettano le proprietà sopra elencate e si prestano quindi ad essere impiegate per implementare l'operazione di composizione: le *norme triangolari* [17].

Definizione 5.7 ***Norma triangolare*** *Una norma triangolare è una funzione di aggregazione binaria t che soddisfa le seguenti proprietà.*

- ***$\wedge$-Conservazione:***
  $t(0,0) = 0$
  $t(x,1) = t(1,x) = x$
- ***Monotonia:***
  $t(x_1, x_2) \leq t(x'_1, x'_2)$ *se* $x_1 \leq x'_1$ *e* $x_2 \leq x'_2$

- ***Commutatività:***
  $t(x_1, x_2) = t(x_2, x_1)$
- ***Associatività:***
  $t(t(x_1, x_2), x_3) = t(x_1, t(x_2, x_3))$

Di seguito vengono riportati alcuni esempi di funzioni specifiche che rientrano nella classe delle norme triangolari [45].

Minimum: $t(x_1, x_2) = \min\{x_1, x_2\}$

Drastic product: $t(x_1, x_2) = \begin{cases} \min\{x_1, x_2\} & \text{, se } \max\{x_1, x_2\} = 1 \\ 0 & \text{, altrimenti.} \end{cases}$

Bounded difference: $t(x_1, x_2) = \max\{0, x_1 + x_2 - 1\}$

Einstein product: $t(x_1, x_2) = (x_1 \cdot x_2)/(2 - (x_1 + x_2 - x_1 \cdot x_2))$

Algebraic product: $t(x_1, x_2) = x_1 \cdot x_2$

Hamacher product: $t(x_1, x_2) = (x_1 \cdot x_2)/(x_1 + x_2 - x_1 \cdot x_2)$

Si propone quindi che per calcolare i punteggi dei vettori risultanti dalle operazioni di aggregazione e composizione vengano utilizzate delle funzioni prese tra quelle proposte in questo paragrafo. In particolare, nel sistema realizzato sono rese disponibili diverse alternative per le funzioni da impiegare e il sistema può quindi essere opportunamente configurato in modo da utilizzare le funzioni effettivamente scelte.

5.2 Gestione dell'ambiente P2P

Il sistema XML S^3MART nella sua versione originaria viene applicato ad un ambiente di esecuzione che può essere definito centralizzato, poiché tutti i documenti su cui deve operare gli sono resi disponibili in un archivio locale. Esso agisce dunque sull'intero insieme di documenti a sua disposizione processandoli a coppie per eseguire le opportune operazioni di schema matching.

Come è stato descritto nel capitolo 2, un ambiente Peer-to-Peer si distanzia molto da questa situazione poiché si configura come un contesto completamente decentralizzato e distribuito, composto da un numero potenzialmente elevato di unità distinte ed autonome, ognuna delle quali è caratterizzata da un proprio schema per la rappresentazione dei dati verosimilmente diverso da quelli delle altre. Essendo completamente indipendenti, queste entità hanno poi un comportamento totalmente autonomo e possono quindi scegliere liberamente gli istanti in cui intendono connettersi al sistema e le altre entità a cui desiderano collegarsi.

La nuova versione del sistema, presentata nel capitolo 4.3, non è adatta per essere direttamente applicata al nuovo ambiente di esecuzione ed è stato quindi necessario completarla realizzando un nuovo modulo (chiamato "modulo P2P") che si occupa del controllo delle varie problematiche legate all'introduzione dell'ambiente di esecuzione distribuito e gestisce il comportamento che le varie entità hanno in un contesto Peer-to-Peer.

5.2.1 Protocollo di interazione tra i peer

Prima di poter sviluppare il modulo che si occupa della gestione dell'ambiente Peer-to-Peer è stato necessario progettare un protocollo che definisce come i peer devono interagire all'interno del sistema. Secondo questo protocollo i peer comunicano tra loro per mezzo di quattro tipologie di messaggi:

1. **RICHIESTA di CONNESSIONE**, si tratta del messaggio che viene inviato dal peer che ne è il mittente ad un certo destinatario per comunicargli la sua volontà di creare una nuova connessione verso di lui;

2. **RISPOSTA di CONNESSIONE**, è il messaggio con cui un peer che ha ricevuto una RICHIESTA di CONNESSIONE può rispondere al suo mittente, in modo da completare il processo di creazione della nuova connessione;

Figura 5.9: Protocollo di interazione tra i peer.

3. **RICHIESTA di AGGIORNAMENTO**, è il messaggio che viene inviato dal peer che riceve una RISPOSTA di CONNESSIONE al suo mittente per chiedergli di essere informato sulle nuove porzioni di rete che tramite la connessione appena stabilita è in grado ora di raggiungere;

4. **RISPOSTA di AGGIORNAMENTO**, è un messaggio che rappresenta la risposta ad un messaggio della tipologia precedente o della stessa tipologia e viene impiegato per propagare nella rete gli aggiornamenti conseguenti alla creazione di una nuova connessione.

La ricezione di ognuno di questi messaggi scatena nel peer destinatario una serie di comportamenti e l'invio di un messaggio di risposta secondo quanto specificato dallo schema di figura 5.9.

Secondo quanto stabilito dal protocollo, ogni nodo, prima di potersi connettere alla rete P2P, viene dunque per prima cosa inizializzato tramite l'esecuzione sul suo schema dell'attività di Self Schema Matching. Come è stato illustrato

nel capitolo 4.3, grazie all'esecuzione di questa operazione il nodo può infatti memorizzare, all'interno delle proprie strutture dati, i valori di normalizzazione impiegati in tale calcolo e i valori di self mapping che ne rappresentano il risultato, in modo da poterli poi impiegare nell'esecuzione di tutte le successive operazioni di Schema Matching.

Dopo lo svolgimento di queste operazioni preliminari, il peer è dunque libero di scegliere uno o più nodi già presenti nella rete P2P, per manifestargli la sua volontà di connettersi a loro tramite l'invio di un messaggio di **RICHIESTA di CONNESSIONE**. Nel contattare tali peer, il nuovo arrivato si preoccupa anche di informarli sulle sue modalità di rappresentazione dei dati, inserendo all'interno del messaggio il suo stesso XML Schema.

Quando un peer connesso alla rete P2P riceve una richiesta di connessione, si preoccupa innanzitutto di aggiornare il suo indice di routing semantico inserendo un nuovo vettore relativo al mittente del messaggio. In seguito estrae dallo stesso messaggio lo XML Schema che il nuovo arrivato gli ha inviato ed esegue l'operazione di Schema Matching tra il suo schema source e lo schema target ricevuto. Il nuovo vettore inserito nell'indice di routing semantico viene così riempito impiegando i punteggi di mapping calcolati tramite questa operazione. Il peer si preoccupa inoltre di replicare al messaggio di richiesta di connessione con un suo messaggio di **RISPOSTA di CONNESSIONE**, in cui inserisce il proprio XML Schema.

Il nuovo nodo riceve dunque la risposta del peer che ha contattato ed aggiorna per prima cosa il suo indice di routing semantico inserendovi un vettore relativo al mittente di tale messaggio. Esegue poi a sua volta l'attività di Schema Matching tra il suo schema source e lo schema target recuperato dal messaggio di risposta, in modo da inserire nel nuovo vettore i punteggi di mapping in questo modo calcolati.

A questo punto bisogna osservare che, nel caso più generico possibile, il peer che ha iniziato il processo di creazione della nuova connessione, potrebbe essere in realtà un nodo già appartenente al sistema, che ha deciso di ampliare il numero dei suoi vicini inserendo tra di essi anche il nodo che ha scelto di contattare. In questo caso, è presumibile aspettarsi che tale peer "iniziatore" possieda già delle connessioni instaurate con altri nodi del sistema, i cui punteggi di mapping sono memorizzati nel suo indice di routing semantico. Esso deve quindi informare di questo fatto il nodo contattato in modo da portarlo a conoscenza delle

informazioni riguardanti la porzione di rete che da lui ha origine.

Dopo aver aggiornato il suo indice di routing semantico in conseguenza del messaggio di risposta di connessione ricevuto, il nodo iniziatore si preoccupa quindi di controllare se nel suo stesso indice di routing semantico sono presenti altri vettori oltre a quello appena creato e a quello contenente i punteggi di self mapping. In caso positivo si occupa quindi di aggregare tali vettori, costruendone un'informazione riassuntiva tramite l'esecuzione su di essi dell'operazione di aggregazione. Esso invia poi al nodo contattato un messaggio di **RICHIESTA di AGGIORNAMENTO**, in cui inserisce il vettore ottenuto come risultato dell'operazione di aggregazione.

Quando un peer riceve un messaggio di richiesta di aggiornamento si preoccupa in primo luogo di verificare se al suo interno è presente un vettore aggregato che contiene i punteggi relativi alla porzione di rete che ha origine dal mittente del messaggio. In caso positivo esso si occupa quindi di combinare questa informazione con l'informazione relativa ai punteggi di mapping che ha precedentemente calcolato verso lo schema del mittente e che ha memorizzato nell'appropriato vettore del suo indice di routing semantico. Esegue dunque a tal scopo l'operazione di composizione tra tale vettore del suo indice di routing semantico e il vettore ricevuto, per memorizzare infine il risultato nel primo dei due, in modo da aggiornare il suo indice con le informazioni riguardanti la nuova porzione di rete raggiungibile.

Il peer deve poi a questo punto rispondere alla richiesta di aggiornamento informando il mittente dei suoi eventuali collegamenti instaurati in tempi precedenti verso altri nodi del sistema. A tal scopo, e in maniera del tutto analoga a quanto aveva fatto il mittente a suo tempo, controlla quindi se nel suo indice di routing semantico sono presenti altri vettori oltre a quello relativo al mittente della richiesta e a quello contenente i punteggi di self mapping. In caso positivo procede quindi ad aggregare tali vettori eseguendo su di essi l'operazione di aggregazione ed inserisce il vettore che ne rappresenta il risultato all'interno di un messaggio di **RISPOSTA di AGGIORNAMENTO** che invia al nodo che era stato il mittente della rispettiva richiesta.

Le attività di questo peer non sono però ancora terminate, poiché esso deve anche preoccuparsi di propagare a tutti i suoi vicini, mittente della richiesta di aggiornamento escluso, le informazioni riguardanti la nuova porzione di rete ora raggiungibile (tramite appunto la connessione con il mittente della richiesta di

aggiornamento). Per ognuno di tali vicini esso si preoccupa quindi di comporre tutti i vettori del suo indice, escluso quello per quel vicino e quello di self mapping, e inviare a quel vicino il vettore risultante in un messaggio di **RISPOSTA di AGGIORNAMENTO**.

Ogni peer che riceve un messaggio di risposta di aggiornamento deve quindi per prima cosa preoccuparsi di aggiornare il vettore del suo indice relativo al mittente del messaggio con il risultato della composizione di tale vettore con il vettore contenuto nel messaggio stesso. In secondo luogo deve poi preoccuparsi di propagare l'aggiornamento ricevuto ad ognuno dei suoi vicini, mittente della risposta di aggiornamento escluso. A tale scopo, per ognuno di tali vicini, aggrega tutti i vettori del suo indice tranne quello relativo al vicino stesso e quello di self mapping, ed invia il risultato al vicino stesso all'interno di un messaggio di **RISPOSTA di AGGIORNAMENTO**.

5.2.2 Algoritmo del modulo P2P

In Listing 5.1 viene riportato lo pseudocodice dell'algoritmo che modella il funzionamento del nuovo modulo P2P realizzato e relativo alla gestione dell'ambiente di esecuzione Peer-to-Peer. In particolare, questo modulo rispecchia fedelmente quanto stabilito dal protocollo presentato nel paragrafo 5.2.1 e richiama al suo interno le funzionalità fornite dal modulo relativo alla versione modificata di XML S^3MART presentata nel capitolo 4.3.

Ogni nodo che compone il PDMS inizia l'esecuzione di questo algoritmo nel momento in cui entra a far parte della sua rete Peer-to-Peer e la continua per tutto il suo ciclo di vita.

In particolare, ogni peer che esegue l'algoritmo è identificato da un id univoco ed è caratterizzato da un suo XML Schema che è indicato nello pseudocodice col nome di `''mioSchema''`. Tra le sue strutture dati (che sono schematizzate in figura 5.10) esso mantiene poi un suo indice di routing semantico che ha la forma di un array ed è chiamato appunto `''indiceRouting''`. Si noti che in tale array nella posizione 0 viene inserito, durante la fase di inizializzazione, il vettore contenente i punteggi di self mapping, mentre i vettori relativi ai vari peer vicini vengono inseriti al momento opportuno nelle posizioni a loro associate durante la fase di aggiornamento (con la notazione `''indiceRouting[peer]''` si intende quindi il vettore dell'indice di routing semantico relativo al peer vicino "peer"). Ogni

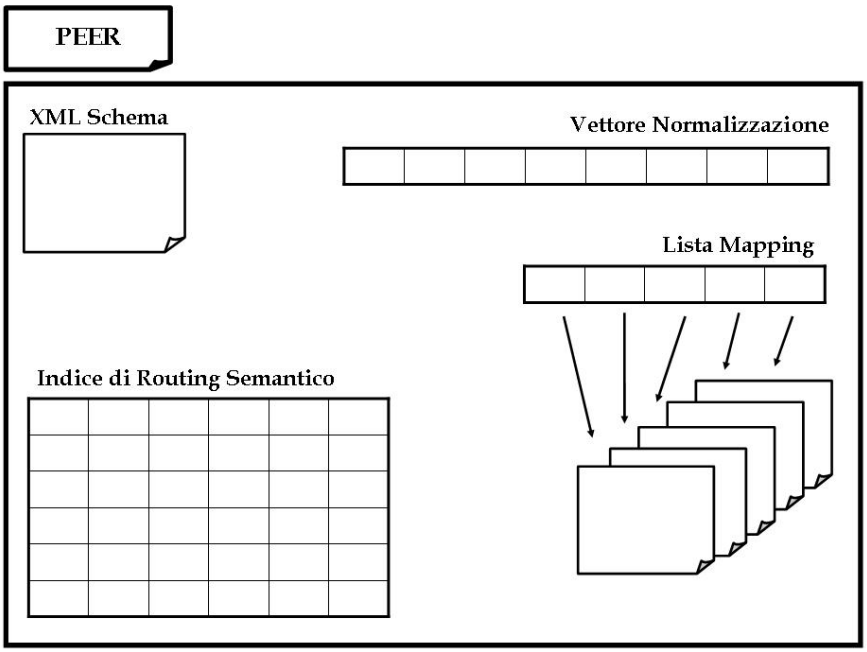

Figura 5.10: Schema delle strutture dati mantenute da ogni peer.

peer possiede inoltre un ulteriore vettore, detto ``vettoreNormalizzazione'', che impiega per memorizzare, durante la fase di inizializzazione, i valori da impiegare per eseguire tutte le future operazioni di normalizzazione. Infine, ogni peer mantiene anche una lista detta "Mapping[ ]" in cui inserisce i mapping risultanti dall'esecuzione delle operazioni di schema matching con ognuno dei suoi vicini, in modo da poter recuperare in ogni istante come gli elementi del suo schema sono associati con gli elementi degli schemi di ognuno dei vicini.

Listing 5.1: Algoritmo del modulo P2P.

```
//Fase di Inizializzazione
selfSchemaMatching(mioSchema);
vettoreNormalizzazione = valori usati nel calcolo
                                    selfSchemaMatching;
indiceRouting[0] = punteggi di self mapping;
Mapping[ ] = null;
```

```
//Fase di Creazione degli Indici
scelta dei vicini;
FOR (ogni vicino scelto)
    {
    send(vicino scelto,RICHIESTA_CONNESSIONE,mioSchema);
    }


//Fase di Aggiornamento degli Indici
WHILE(true)
    {
    aspetta di ricevere un messaggio;
    mess = messaggio ricevuto;

    IF(mess == RICHIESTA_CONNESSIONE)
        {
        indiceRouting[Mittente] =
                schemaMatching(mioSchema,schemaMittente);
        Mapping[Mittente] = risultato di schemaMatching;

        send(mittente,RISPOSTA_CONNESSIONE,mioSchema);
        }

    IF(mess == RISPOSTA_CONNESSIONE)
        {
        indiceRouting[Mittente] =
                schemaMatching(mioSchema,schemaMittente);
        Mapping[Mittente] = risultato di schemaMatching;

        aggregato = unioneEccetto(mittente);
        send(mittente,RICHIESTA_AGGIORNAMENTO,aggregato);
        }

  IF(mess == RICHIESTA_AGGIORNAMENTO)
    {
    indiceRouting[mittente] =
            composizione(indiceRouting[mittente],aggregato);
```

```
    aggregato = unioneEccetto(mittente);
    send(mittente,RISPOSTA_AGGIORNAMENTO, aggregato);

    FOR(ogni vicino)
        {
        IF(vicino != mittente)
            {
            aggregato = unioneEccetto(vicino);
            send(vicino,RISPOSTA_AGGIORNAMENTO, aggregato);
            }
        }
    }

  IF(mess == RISPOSTA_AGGIORNAMENTO)
    {
    indiceRouting[mittente] =
            composizione(indiceRouting[mittente], aggregato);

    FOR(ogni vicino)
        {
            IF(vicino != mittente)
                {
                aggregato = unioneEccetto(vicino);
                send(vicino,RISPOSTA_AGGIORNAMENTO, aggregato);
                }
            }
        }
}
```

L'algoritmo è suddiviso in tre parti principali:

1. nella *Fase di Inizializzazione* vengono eseguite le operazioni preliminari necessarie per la connessione del peer al sistema e per la sua futura esecuzione di tutte le altre attività;

2. nella *Fase di Creazione degli Indici* il peer sceglie in maniera completa-

mente autonoma uno o più nodi a cui intende connettersi e li contatta inviandogli un messaggio di "richiesta di connessione";

3. nella *Fase di Aggiornamento degli Indici* il peer, mentre esegue le sue normali attività, rimane pronto a ricevere i messaggi che gli altri peer possono inviargli per informarlo sulle modifiche da eseguire sul suo indice di routing semantico e a cui lui stesso replica con le proprie informazioni.

Le funzioni introdotte per descrivere il funzionamento dell'algoritmo sono:

- `selfSchemaMatching(schema)`, che si occupa di calcolare le operazioni di Self Schema Matching per lo schema "schema";
- `schemaMatching(schema1, schema2)`, che si occupa di calcolare le operazioni di Schema Matching tra lo schema source "schema1" e lo schema target "schema2";
- `unioneEccetto(peer)`, che calcola l'aggregato di tutti i vettori dell'indice di routing semantico eccetto quelli corrispondenti ai punteggi di self mapping (in posizione 0) e al peer vicino "peer";
- `composizione(vettore1, vettore2)`, che calcola l'operazione di composizione tra il "vettore1" e il "vettore2".
- `send(peerDest, tipo messaggio, contenuto)`, che invia un messaggio di tipo "tipo messaggio" e contenente le informazioni specificate in "contenuto" al peer indicato come "peerDest". Per quanto riguarda il campo "tipo messaggio" sono possibili le quattro alternative introdotte nel paragrafo precedente (5.2.1):

  1. RICHIESTA_CONNESSIONE, per iniziare le operazioni di creazione di una nuova connessione;
  2. RISPOSTA_CONNESSIONE, inviato in risposta al precedente per completare il processo di creazione di una nuova connessione;
  3. RICHIESTA_AGGIORNAMENTO, per iniziare le operazioni di aggiornamento necessarie se una nuova connessione ha reso raggiungibili nuove porzioni di rete,

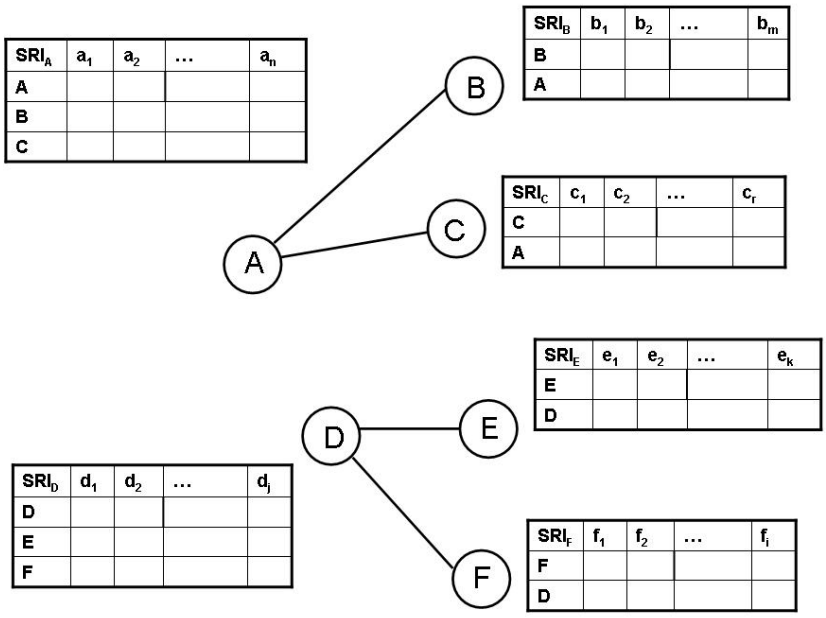

Figura 5.11: Esempio di nuova connessione (I): situazione iniziale.

4. RISPOSTA_AGGIORNAMENTO, inviato in riposta al precedente o ad un messaggio della stessa tipologia per propagare gli aggiornamenti disponibili.

Esempio di creazione di una nuova connessione

Per chiarire meglio il funzionamento dell'algoritmo P2P presentato nel capitolo precedente (5.2.2) vengono ora approfondite tramite un esempio le sue due fasi di creazione ed aggiornamento degli indici di routing semantici (la fase di inizializzazione si riduce a quanto è stato spiegata nel paragrafo 4.3). Si consideri a tal scopo la porzione di rete rappresentata in figura 5.11 e si supponga che inizialmente non esista alcuna connessione tra il peer A e il peer D.

Il peer A decide poi di volersi collegare al peer D e gli invia di conseguenza un messaggio di **RICHIESTA di CONNESSIONE** contenente il suo XML Schema.

Quando D riceve il messaggio di richiesta di connessione inviatogli da A (figura 5.12), aggiorna il suo indice di routing semantico aggiungendovi un nuovo

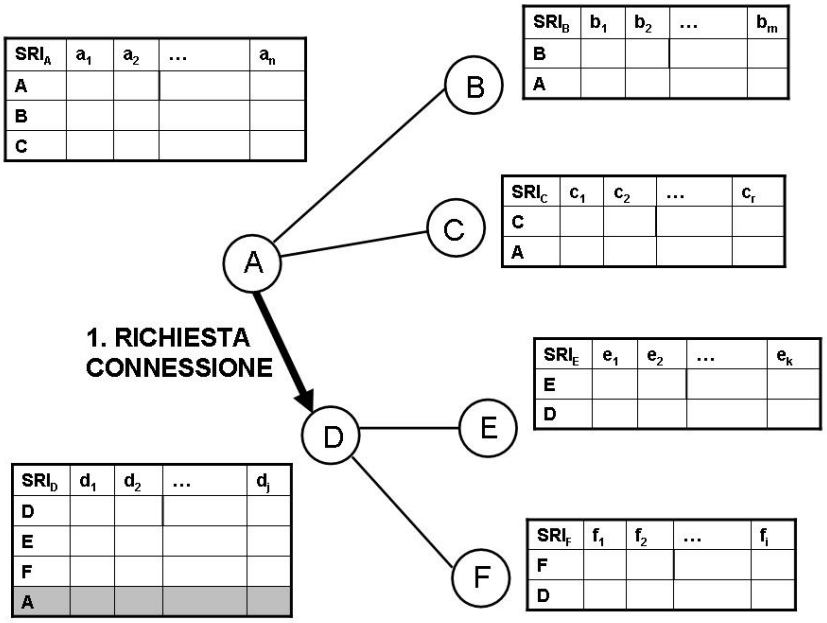

Figura 5.12: Esempio (II): il peer A si connette al peer D.

vettore per A e calcola i punteggi da inserire all'interno di questo vettore applicando l'algoritmo di Schema Matching tra il suo schema source e lo schema target del nodo A. A questo punto il peer D invia ad A un messaggio di **RISPOSTA di CONNESSIONE** contenente il suo XML Schema.

Quando A riceve il messaggio di risposta di connessione inviatogli da D (figura 5.13), aggiorna a sua volta il suo indice di routing semantico aggiungendovi un nuovo vettore per D e calcola i punteggi da inserire all'interno di questo vettore applicando l'algoritmo di Schema Matching tra il suo schema source e lo schema target del nodo D. A questo punto, A deve però preoccuparsi di informare D del fatto che lui era a sua volta collegato già da tempi precedenti ai nodi B e C, poiché D deve essere messo al corrente del fatto che, tramite la nuova connessione appena stabilita con A, ora può raggiungere anche i nodi B e C. A tale scopo, A aggrega quindi i due vettori relativi a B e C del suo indice di approssimazione e inserisce il vettore risultante all'interno del messaggio di **RICHIESTA di AGGIORNAMENTO** che invia a D.

Quando D riceve il messaggio di richiesta di aggiornamento inviatogli da A (figura 5.14), per prima cosa aggiorna il vettore relativo a D del suo indice

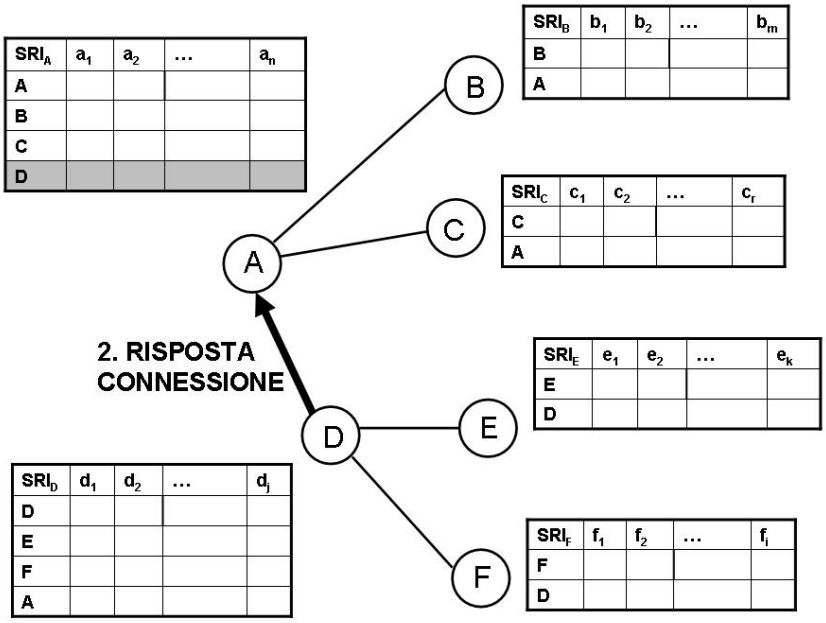

Figura 5.13: Esempio (III): D risponde alla richiesta di connessione di A.

di routing semantico componendolo con il vettore ricevuto nel messaggio. In secondo luogo, D si preoccupa a sua volta di informare A del fatto che lui era già collegato, tramite connessioni stabilite in precedenza, con i nodi E ed F. A tal scopo, D aggrega quindi i vettori relativi ad E ed F del suo indice di routing semantico ed invia il vettore risultante ad A all'interno di un messaggio di **RISPOSTA di AGGIORNAMENTO** (figura 5.15). I compiti di D non sono però ancora terminati, poiché esso deve preoccuparsi di informare anche E ed F del nuovo collegamento stabilito verso A, poiché anch'essi possono ora raggiungere tutta la porzione di rete che si trova oltre A. A questo scopo quindi, D invia il vettore appena calcolato e relativo ad A del suo indice di routing semantico ad entrambi i nodi E ed F all'interno di due messaggi di **RISPOSTA di AGGIORNAMENTO** a loro diretti (figura 5.16).

Quando E ed F ricevono il messaggio di risposta di aggiornamento da parte di D (figura 5.16) essi aggiornano il loro indice di routing semantico componendo il vettore relativo a D con il vettore appena ricevuto. Poiché poi tali nodi non hanno altri vicini oltre a D, non devono di preoccuparsi di propagare ulteriormente l'aggiornamento.

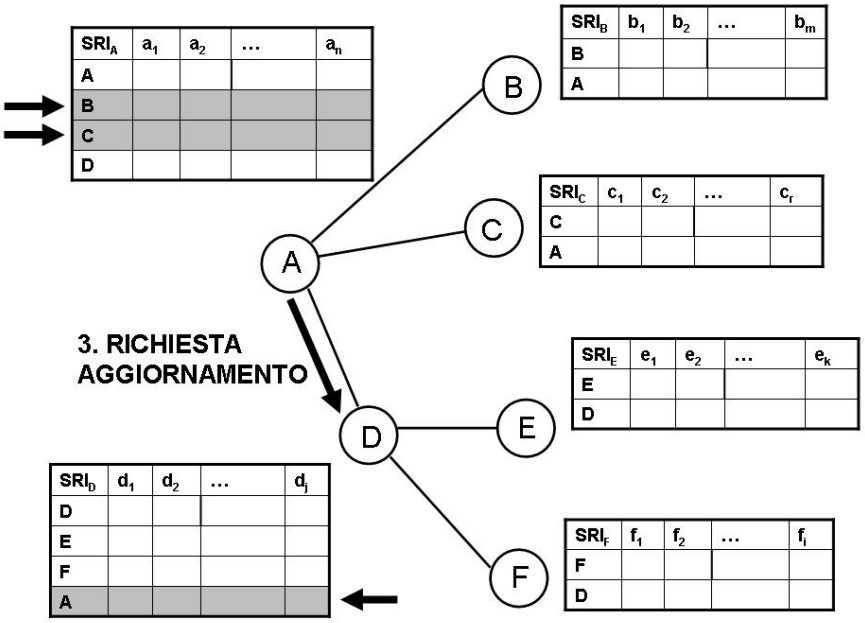

Figura 5.14: Esempio (IV): A chiede a D di essere aggiornato.

Quando A riceve il messaggio di risposta di aggiornamento da parte di D (figura 5.15), anch'esso aggiorna il suo indice di routing semantico, componendo il vettore relativo a D con il vettore ricevuto. Tuttavia, visto che A è a sua volta collegato ai nodi B e C, deve preoccuparsi di informare anche loro dell'aggiornamento ricevuto. A tal scopo A inserisce quindi il vettore appena calcolato e relativo a D del suo indice di routing semantico all'interno di due messaggi di risposta di aggiornamento che invia ai nodi B e C (figura 5.17).

Quando B e C ricevono il messaggio di risposta di aggiornamento da parte di A, in modo del tutto analogo al comportamento adottato da E ed F, aggiornano il loro indice di routing semantico componendo il vettore relativo ad A con il vettore appena ricevuto, e poiché non hanno altri vicini oltre ad A, non devono preoccuparsi di propagare ulteriormente gli aggiornamenti (figura 5.17).

Tutti i nodi della rete hanno ora le informazioni complete e corrette sulla nuova connessione che è stata creata.

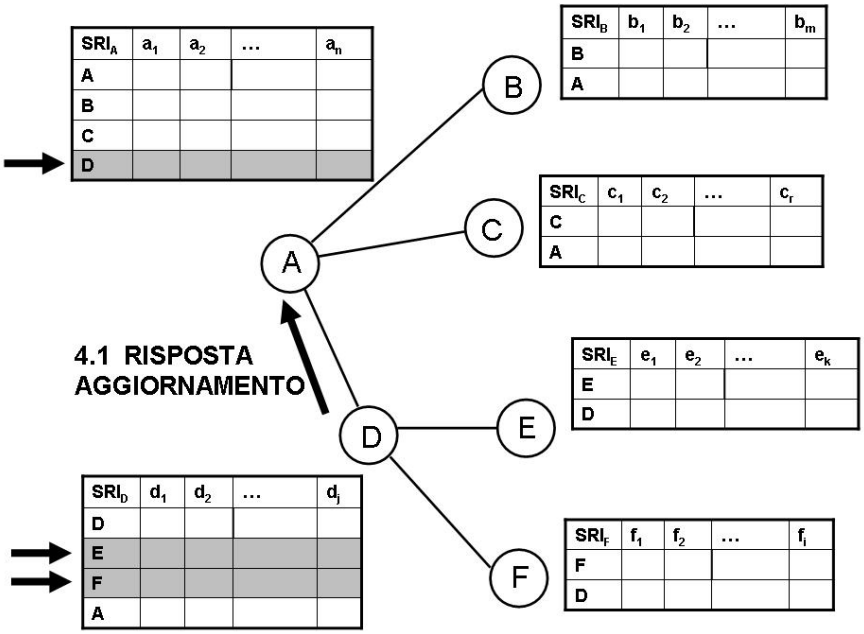

Figura 5.15: Esempio (V): D risponde alla richiesta di aggiornamento di A.

5.3 Evoluzioni future

In questa sezione vengono presentate alcune ulteriori considerazioni che riguardano il lavoro svolto ed introducono dei suggerimenti per la sua eventuale futura evoluzione.

In primo luogo viene presentata una proposta per quanto riguarda l'effettivo impiego degli indici di routing semantici all'interno di un PDMS e viene quindi illustrato come le informazioni al loro interno mantenute possono essere interpretate per realizzare il meccanismo di routing delle interrogazioni.

In seguito viene invece sottolineato che il meccanismo di routing proposto potrebbe essere migliorato se fosse possibile disporre di altre informazioni da considerare congiuntamente a quelle degli indici di routing semantici. I punteggi di approssimazione mantenuti all'interno degli indici si configurano infatti come una sorta di indicazione puramente qualitativa sui dati recuperabili. In questa parte ci si sofferma invece in particolare sugli aspetti quantitativi relativi a tali dati

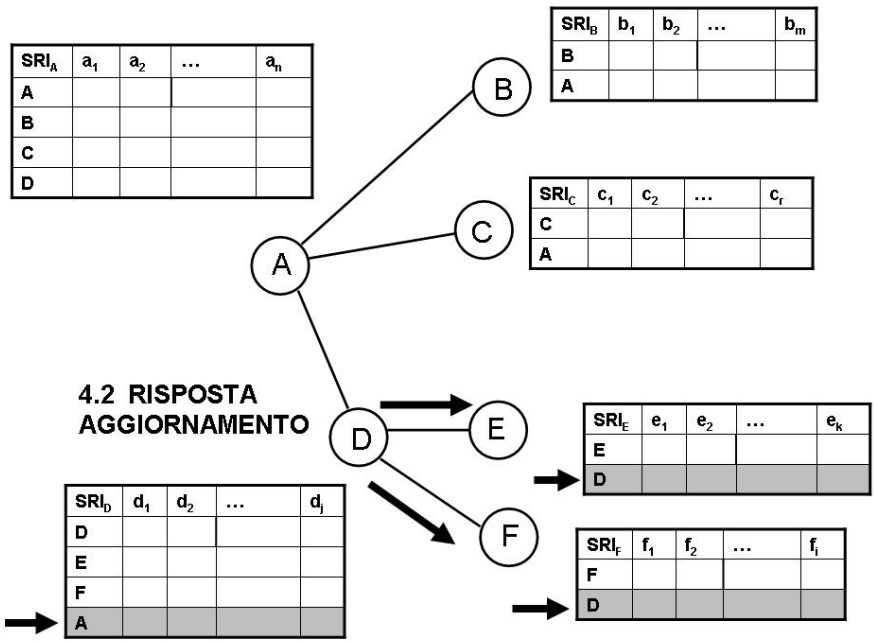

Figura 5.16: Esempio (VI): D propaga ad E ed F gli aggiornamenti ricevuti da A.

5.3.1 Utilizzo degli indici di routing semantici

Gli indici di routing semantici si configurano in definitiva come strutture dati che consentono ad ogni membro di un PDMS di comprendere come gli elementi del suo schema possono essere approssimati lungo i rami della rete che da lui si dipartono. La proposta che viene fatta all'interno di questo paragrafo è che tali indici vengano impiegati per instradare tra i vari peer componenti il PDMS le query sottoposte dagli utenti e realizzare in questo modo il meccanismo di "routing by mapping" presentato nel capitolo 5.1.

In particolare, si propone che ogni nodo del sistema impieghi i punteggi disponibili all'interno del suo indice di routing semantico per stabilire quali tra i suoi vicini sono rilevanti per le query a lui pervenute, ovvero hanno una maggiore probabilità di risolverle fornendo dati interessanti per gli utenti. Lo scopo finale è quindi che ogni peer possa propagare le interrogazioni solamente verso i peer che sulla base del suo indice di routing sono considerati rilevanti.

Il meccanismo proposto in questo ambito prevede dunque che ogni peer, ogni volta che riceve una nuova query, tenti in primo luogo di risolverla sulla base

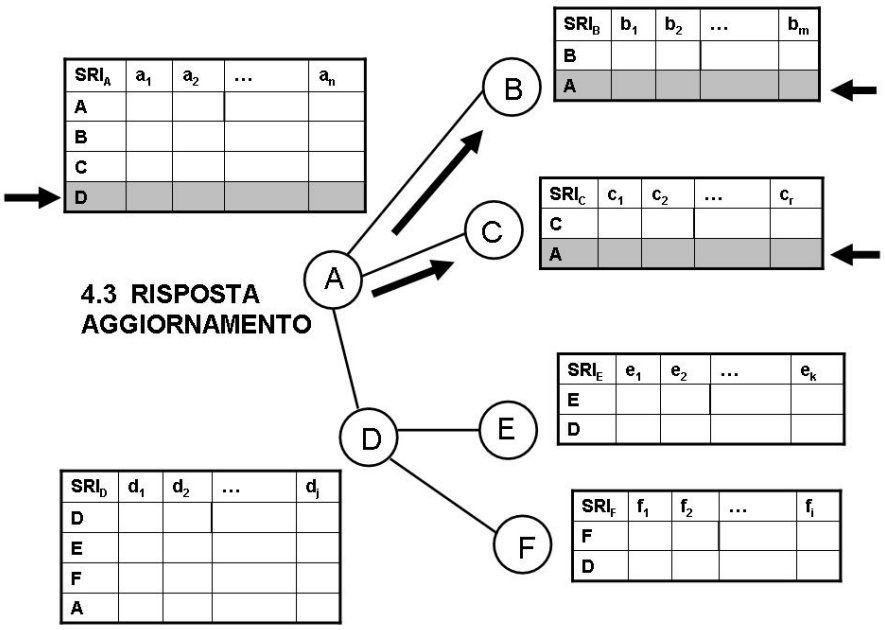

Figura 5.17: Esempio(VII): A propaga a B e C gli aggiornamenti ricevuti da D.

dei dati disponibili localmente. In seguito, il peer deve impiegare le informazioni mantenute all'interno del suo indice di routing per stabilire un ranking tra i suoi vicini, che sia significativo per esprimerne la rilevanza nei confronti della query. In questo modo la query può essere propagata solamente ai vicini che si trovano nella posizioni più alte di tale ranking.

Per un peer, stabilire la rilevanza dei suoi vicini nei confronti di una query sulla base del suo indice di routing, non è tuttavia un procedimento immediato e dipende dal tipo di query che si sta considerando.

Il caso più semplice è rappresentato da una query che coinvolge un solo concetto. Come è stato spiegato nel capitolo 5.1.2, ogni peer mantiene all'interno del suo indice di routing il livello di approssimazione per ogni concetto del suo schema verso ognuno dei suoi vicini. In questo caso la rilevanza di ogni vicino può quindi essere ricavata in maniera diretta dall'indice, poiché corrisponde con il livello di approssimazione per il concetto richiesto.

Nel caso di query più complesse che coinvolgono più concetti è invece necessario accedere, per ogni vicino, ai valori di approssimazione relativi a tutti i concetti richiesti e combinarli in modo opportuno sulla base delle operazioni

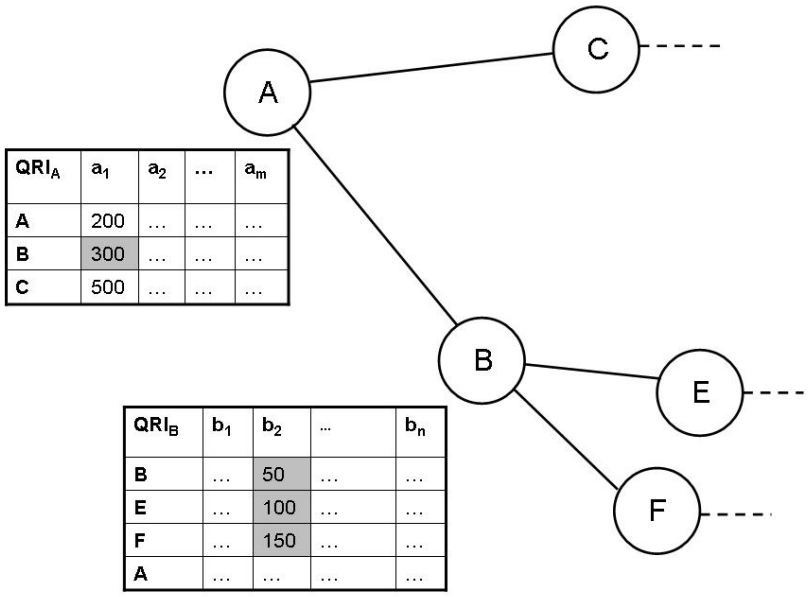

Figura 5.18: QRI: esempio.

richieste dalla query. Nello specifico, si propone che la combinazione dei punteggi relativi ai singoli concetti venga eseguita utilizzando le tecniche impiegate classicamente per i valori fuzzy in ambito relazionale.

Se ad esempio la query ha la forma della congiunzione di due concetti, per stabilire la rilevanza dei vari vicini occorre quindi recuperare i punteggi dei due concetti e combinarli eseguendo la congiunzione dei loro livelli di approssimazione fuzzy: secondo la classica semantica fuzzy il risultato corrisponde al minimo dei due valori.

Un discorso analogo può essere fatto nel caso di query disgiuntive o che coinvolgono altri operatori dell'algebra relazionale. Una descrizione accurata di come tali operatori possono essere tradotti in ambito fuzzy è presentata in [50].

5.3.2 Informazioni quantitative sui dati

Come è stato più volte ribadito, la dinamica della rete di un sistema Peer-to-Peer coinvolge molte diverse componenti e nell'implementare un generico meccanismo di routing delle interrogazioni per tale ambito è quindi possibile pensare di te-

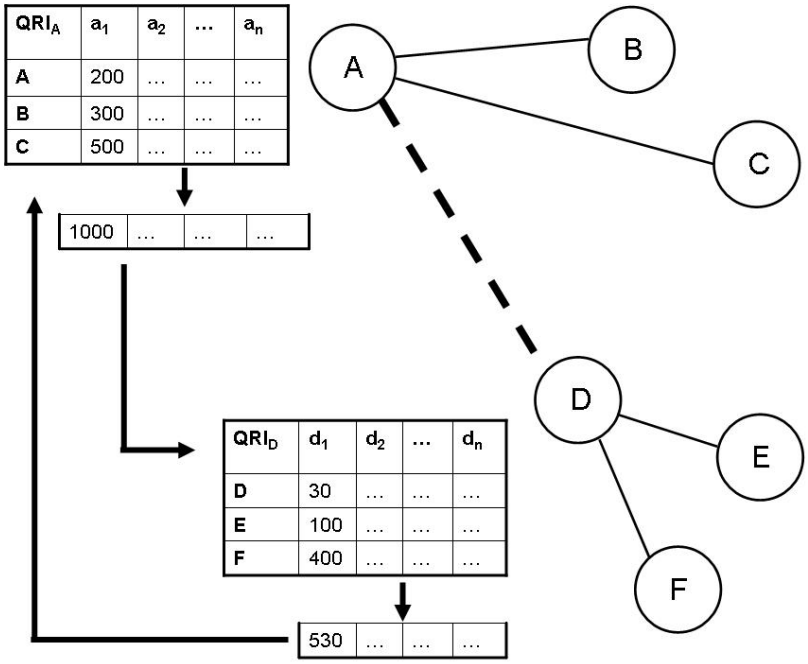

Figura 5.19: QRI: creazione di una nuova connessione.

nere in considerazione anche informazioni ulteriori rispetto ai soli punteggi di approssimazione semantica.

In particolare, un tipo di informazione che sarebbe sicuramente utile poter considerare è rappresentato dalla quantità di documenti che sono effettivamente presenti su ogni peer, poiché lo scopo finale dell'utente nel sottoporre le interrogazioni al sistema è proprio di recuperare i dati contenuti in questi documenti. I vari punteggi di mapping impiegati per instradare le interrogazioni dovrebbero quindi essere interpretati anche in riferimento alla quantità di dati che con tale livello di approssimazione è possibile recuperare. In questo modo ogni nodo potrebbe selezionare il vicino a cui propagare una query basandosi non solamente sul punteggio con cui sono rappresentati i suoi concetti nella sottorete che da tale vicino ha origine, ma anche sulla quantità di documenti contenenti quei concetti che in quella sottorete sono effettivamente presenti.

Questo tipo di informazione è di importanza fondamentale perché permette di dare un'interpretazione migliore ai punteggi di approssimazione semantica memorizzati all'interno degli indici di routing. Come è stato spiegato nel capitolo 5.1.3, i punteggi di approssimazione per i concetti di un certo peer decrescono

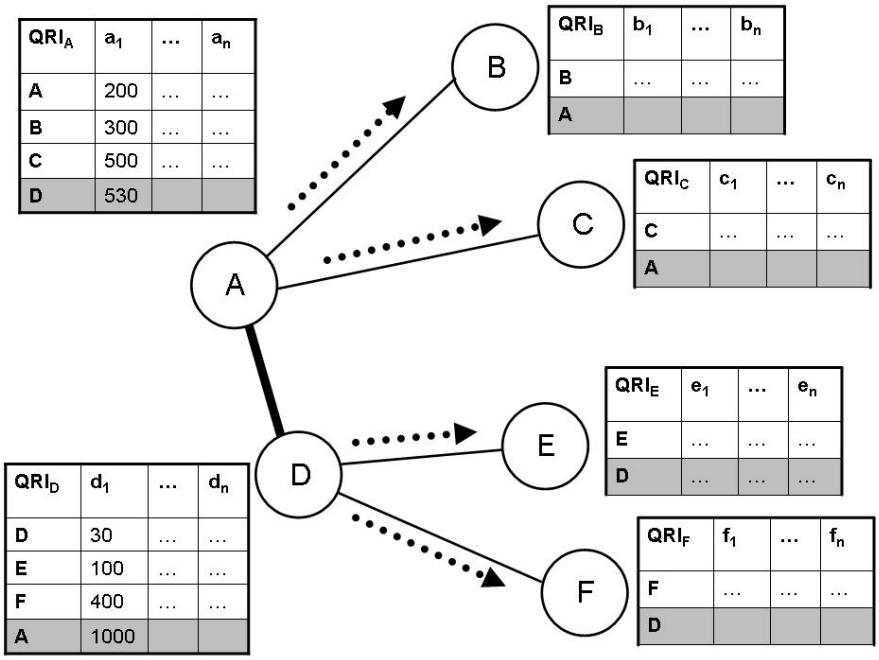

Figura 5.20: QRI: aggiornamento dei vicini.

infatti allontanandosi da tale peer lungo i vari rami della rete che da lui si dipartono, a causa delle varie operazioni di combinazione a cui sono sottoposti. Questo comportamento è corretto poiché rispecchia il fatto che, quanto più ci si allontana da un peer, tanto più aumenta il livello di incertezza che caratterizza la conoscenza del suo schema e di conseguenza il livello di approssimazione per i suoi concetti. Tuttavia bisogna osservare che, quanto più ci si allontana da un peer, tanto più aumenta anche il numero di altri peer con cui si entra in contatto e i cui documenti risultano di conseguenza accedibili. Propagando la query all'interno del sistema si ottiene quindi che il numero di documenti restituibili all'utente aumenta, pur diminuendo la precisione che i concetti di questi documenti hanno rispetto ai concetti richiesti.

Se si suppone quindi che ogni nodo sia a conoscenza del numero di documenti disponibili localmente, è possibile pensare di trattare questa informazione in maniera analoga a quanto viene fatto per i punteggi di approssimazione semantica. Ispirandosi a quanto è presentato in [14] ogni nodo può infatti mantenere un'appropriata struttura di indice organizzata come una tabella (detta indice di routing quantitativo, QRI), le cui righe sono associate ai suoi vicini (la prima

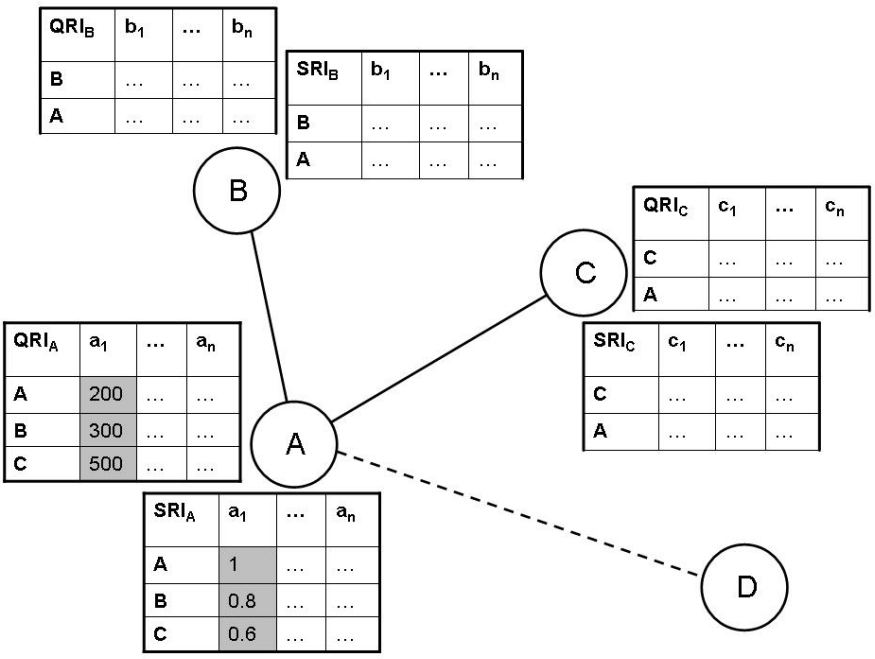

Figura 5.21: QRI: operazione di aggregazione.

riga è associata al nodo stesso) e le cui colonne corrispondono ai concetti del suo schema. I valori memorizzati nella tabella di ogni peer sono quindi, per ogni concetto del suo schema, le informazioni quantitative relative a se stesso e alle sottoreti che dai suoi vicini hanno origine. Si consideri ad esempio la tabella rappresentata in figura 5.18 per il nodo A, il cui schema si suppone essere costituito dagli m elementi $a_1, a_2 \ldots, a_m$. Il valore 300, posto in corrispondenza della colonna relativa ad a_1 e della riga riferita a B, sta ad indicare che nella sottorete che ha origine da B (e che include lo stesso B) è possibile recuperare 300 documenti contenenti l'elemento corrispondente ad a_1.

In particolare, l'informazione memorizzata per ogni vicino deve rappresentare il numero totale di documenti a lui locali e da lui accedibili nella sua sottorete e può quindi essere ricavata da tale vicino semplicemente aggregando le informazioni che lui ha a sua volta a disposizione all'interno della sua tabella. Ad esempio in figura 5.18 le informazioni nella tabella di A relative a B rappresentano l'aggregato delle informazioni della tabella di B stesso che B può ricavare semplicemente eseguendone la somma per colonne, in modo da poter poi inviare il risultato ad A. Più nel dettaglio, supponendo che l'elemento a_1 dello schema

di A corrisponda all'elemento b_2 dello schema di B, B può informare A del fatto che i documenti recuperabili nella sua sottorete sono i 300 dati dalla somma dei suoi 50 con i 100 di E e i 150 di F.

In maniera analoga a quanto è stato spiegato per gli indici di routing semantici, anche questi indici "quantitativi" devono poi essere mantenuti aggiornati al verificarsi di evoluzioni nella topologia della rete e devono quindi essere gestiti in modo simile a quanto presentato nel paragrafo 5.2.2. In particolare, ogni volta che due peer si connettono, entrambi devono innanzitutto estendere le loro tabelle, inserendovi una nuova riga relativa al nuovo vicino e contenente le informazioni quantitative accedibili mediante la sua sottorete. Ogni peer si preoccupa di costruire questa informazione semplicemente sommando per colonne i valori specificati nella sua tabella originaria e di inviare poi il risultato all'altro peer. Si noti che in questo caso l'operazione di composizione non va a inficiare i valori risultanti, poiché ogni nodo inserisce già nel proprio aggregato anche le informazioni relative a se stesso. L'operazione di composizione, che viene eseguita quando ognuno dei due peer riceve l'aggregato dell'altro, si riduce al semplice riordino dei valori ricevuti sulla base delle corrispondenze stabilite tra gli elementi a cui quei valori si riferiscono e gli elementi della tabella locale. Quando ognuno dei due peer riceve l'aggregato dell'altro, deve quindi preoccuparsi solamente di riordinare i valori ricevuti in modo concorde alle sue colonne sulla base del mapping stabilito con l'altro peer e può in seguito inserire direttamente i valori nella sua tabella. Il procedimento è schematizzato in figura 5.19.

Come accade per gli indici di routing semantici, dopo aver esteso le loro tabelle, i due peer devono aggiornare i loro vicini per informarli sulla nuova connessione stabilita. A tal fine, ognuno di loro costruisce per ogni suo vicino un aggregato, da cui esclude la riga della tabella relativa al vicino stesso, e si preoccupa di inviargliela. Ogni vicino che riceve questa informazione aggiorna poi la sua tabella semplicemente sostituendo la riga relativa al mittente con l'informazione ricevuta, dopo averla opportunamente riordinata in modo concorde alle sue colonne. Il procedimento è schematizzato in figura 5.20.

Supponendo quindi di disporre di queste informazioni quantitative, è possibile pensare di impiegarle durante il processo di aggregazione degli indici di routing semantici (si veda il paragrafo 5.1.3), allo scopo di ottenere dei risultati più fedeli nell'esprimere le effettive possibilità di recupero di dati rilevanti. In particolare, dal momento che mediante queste tabelle ogni peer conosce il numero di docu-

menti recuperabili da se stesso e dai vicini i cui punteggi di approssimazione semantica deve aggregare, può interpretare tali quantità in percentuale ed applicarle ai punteggi corrispondenti prima di aggregarli. Ad esempio in figura 5.21, il peer A dispone della tabella relativa alle informazioni quantitative (indicata con QRI_A) e del suo indice di routing semantico (indicato con SRI_A). Accedendo a queste strutture A è quindi a conoscenza del fatto che per il concetto a_1 è in grado si recuperare su di se, e quindi con approssimazione 1, 200 documenti, mentre muovendosi verso B gli è possibile recuperarne 300 ma con approssimazione 0.8 e infine verso C 500 con approssimazione 0.6. Questo significa che i dati recuperabili attraverso A, sono per il 20% disponibili con approssimazione 1 (ovvero informazioni certe), per il 30% con approssimazione 0.8 e per il 50% con approssimazione 0.6. L'informazione aggregata risultante per D viene quindi calcolata da A applicando l'operazione di aggregazione ai 3 valori di approssimazione, dopo averli opportunamente corretti pesandoli con le percentuali a loro relative. Il risultato in questo modo ottenuto rispecchia in modo più fedele le possibilità che il peer A ha di risolvere con successo (fornendo cioè dati rilevanti per l'utente) una query contenente il concetto a_1.

Valutando i punteggi in questo modo calcolati per ognuno dei suoi vicini, ogni peer può quindi fare una scelta più accurata delle direzioni in cui è conveniente propagare le query pervenute, con la conseguenza che il meccanismo di routing risultante tende a fornire risultati migliori.

Capitolo 6

Prove sperimentali e risultati ottenuti

In questo capitolo vengono presentate le prove sperimentali eseguite sul sistema implementato durante la stesura della Tesi e i risultati che da queste prove sono stati ottenuti.

Come è stato spiegato nel capitolo 4, il sistema è stato realizzato per essere impiegato all'interno di un PDMS. I PDMS sono tuttavia una tecnologia nata di recente e il cui studio è iniziato solo da breve tempo. Il maggior impedimento per testare il sistema è quindi rappresentato proprio dalla mancanza di PDMS esistenti da poter impiegare a questo scopo.

Le prove sperimentali non sono quindi state eseguite all'interno di un PDMS reale, ma utilizzando uno strumento di simulazione in grado di riprodurre le principali condizioni che caratterizzano tale ambiente. In particolare, lo strumento di simulazione è stato introdotto per imitare il comportamento che le varie entità possono avere all'interno di un PDMS relativamente alle operazioni di creazione delle connessioni. In questo modo è stato possibile verificare il funzionamento del sistema senza dovere impiegare una reale rete Peer-to-Peer e disinteressandosi di conseguenza di tutte le problematiche di rete che i peer di un sistema reale devono affrontare per potere interagire.

L'attività sperimentale è stata eseguita in due fasi, con lo scopo di valutare componenti differenti del sistema:

1. la prima parte di esperimenti è stata dedicata alla valutazione delle modifiche eseguite sul sistema XML S^3MART e presentate nel capitolo 4.3;

2. nella seconda parte di esperimenti è stato invece implementato il protocollo di comunicazione tra i peer proposto nel capitolo 5.2.1, in modo da verificare come il sistema progettato può essere effettivamente applicato ad un ambiente distribuito. Lo scopo di questa parte è inoltre di ottenere una valutazione indicativa di come gli indici di routing semantici vengono modificati in conseguenza dei cambiamenti che possono verificarsi nella struttura della rete dell'ambiente di esecuzione.

Si sottolinea che i risultati sperimentali ottenuti non possono essere interpretati come informazioni accurate e devono invece essere considerati semplici indicazioni orientative, poiché mancano allo stato attuale le componenti necessarie ad eseguire una sperimentazione approfondita. Per svolgere una valutazione accurata sarebbe infatti doveroso un preliminare studio a riguardo delle modalità da impiegare per tale valutazione. In particolare, sarebbe necessario implementare complessi meccanismi di valutazione propri dei sistemi di information retrieval e introdurre parametri specifici per misurare le grandezze tipicamente impiegate in tale ambito come la precision e la recall.

Tutto ciò non rientra però tra gli obiettivi della presente Tesi, nell'ambito della quale, per quanto riguarda le prove sperimentali, ci si è concentrati soprattutto sulla non banale attività di produzione del codice necessario per implementare l'ambiente di simulazione. Viene quindi presentato solo un insieme volutamente ridotto di prove, che hanno lo scopo principale di introdurre un ambiente di esecuzione distribuito per il sistema e fornire un'indicazione della sua efficacia in tale ambiente. I risultati ottenuti rappresentano comunque un interessante suggerimento per l'eventuale lavoro futuro che potrà essere eseguito in quest'ambito.

Sono state inoltre trascurate le questioni relative all'efficienza perché richiederebbero un altrettanto approfondito studio di tutte le problematiche di rete coinvolte.

Nel seguito del capitolo vengono introdotti l'ambiente di simulazione e lo scenario impiegati per eseguire le prove e sono infine presentati i risultati raccolti.

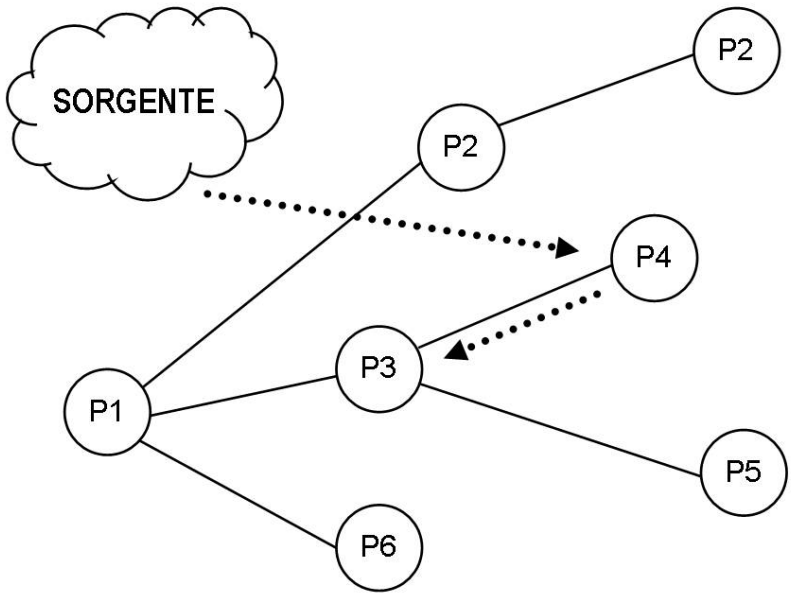

Figura 6.1: Schema del processo simulativo in una rete fittizia.

6.1 Ambiente di simulazione

Le prove sperimentali sono state eseguite impiegando l'ambiente di simulazione **SimJava** nella versione 2.0. SimJava è uno strumento di simulazione discreto ed orientato agli eventi, strutturato come un insieme di API che forniscono le componenti necessarie per definire ed eseguire la simulazione di programmi scritti in linguaggio Java. All'interno di SimJava ogni sistema è considerato come un insieme di entità che interagiscono e comunicano tra loro scambiandosi degli eventi; il tempo di simulazione avanza quindi sulla base di questi eventi.

Prima di realizzare il codice per implementare il simulatore è stato necessario eseguire un'attività di analisi e progettazione che ha permesso di ricavare un modello per il sistema da studiare. In particolare, poiché il sistema da simulare si configura come un PDMS, le sue entità componenti sono state identificate nei singoli peer che ne fanno parte. Gli eventi che tali entità si scambiano corrispondono quindi ai messaggi che i peer devono inviarsi l'un l'altro per potere interagire.

Per modellare poi il fatto che in un sistema Peer-to-Peer i singoli nodi possono

decidere in modo libero ed indipendente quando entrare a far parte del sistema e a quali altri nodi connettersi, è stata introdotta un'ulteriore entità, che si configura come una sorgente, con il compito di attivare le altre entità (i peer) in modo casuale. Ogni entità attivata dalla sorgente sceglie poi a sua volta in modo casuale un certo numero di altre entità a cui collegarsi. Il procedimento è schematizzato in figura 6.1, nella quale le frecce rappresentano le scelte fatte dalle entità di un generico scenario di esempio in un ipotetico istante.

Ogni entità del simulatore corrisponde quindi ad un nodo del sistema reale che, una volta attivato, richiama le funzionalità della versione modificata del sistema XML S^3MART (presentata nel capitolo 4) e interagisce con le altre entità secondo quanto stabilito dal meccanismo di interazione illustrato nel capitolo 5.2.1.

Dal modello si è anche ricavato che le grandezze che si vogliono considerare come risultato della simulazione sono rappresentate dai valori contenuti all'interno degli indici di routing semantici che, in conseguenza dell'esecuzione della simulazione, vengono creati ed aggiornati dalle singole entità. Si vuole infatti osservare come i punteggi di approssimazione memorizzati all'interno di questi indici vengono modificati quando si verificano dei cambiamenti nella struttura della rete dovuti alla creazione di nuove connessioni tra i peer.

6.2 Scenario per le prove sperimentali

Per eseguire gli esperimenti è stato considerato uno scenario composto da un numero di peer relativamente esiguo, ma sufficiente ai fini dello studio qualitativo da intraprendere. Ognuno di questi peer è stato inoltre dotato di un suo schema espresso nel linguaggio XML Schema e diverso da quello degli altri.

Sono stati a tal fine realizzati un certo numero di schemi raggruppabili a seconda del loro argomento in tre diverse categorie:

- la prima categoria di schemi descrive contenuti di argomento sportivo, come le caratteristiche delle squadre o degli atleti che praticano diversi sport o le informazioni relative alle gare sportive sostenute tra diverse squadre (si veda la figura 6.2);

- la seconda categoria contiene schemi che descrivono la realtà di alcuni

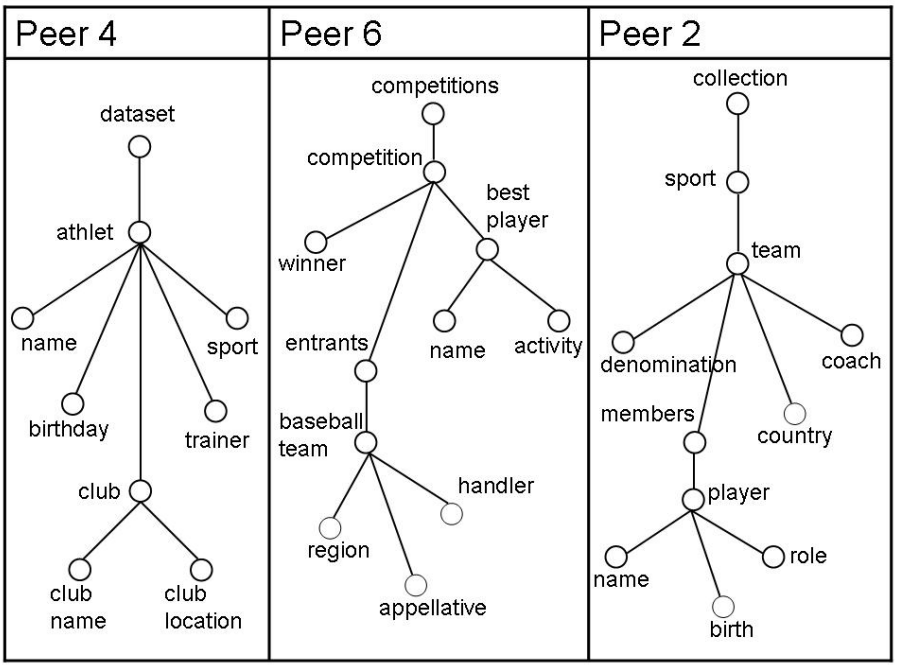

Figura 6.2: Esempi di schemi di argomento sportivo.

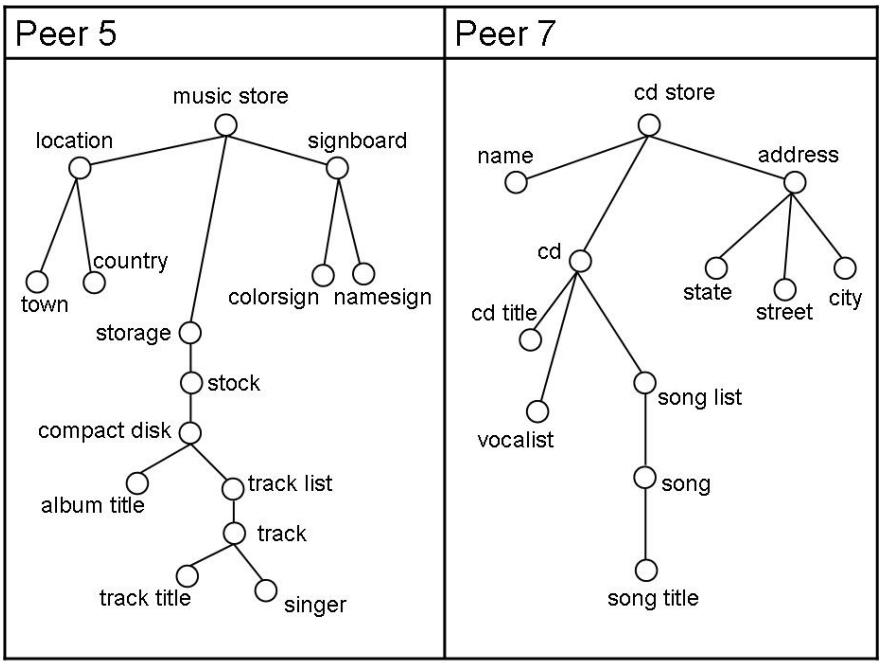

Figura 6.3: Esempi di schemi di argomento musicale.

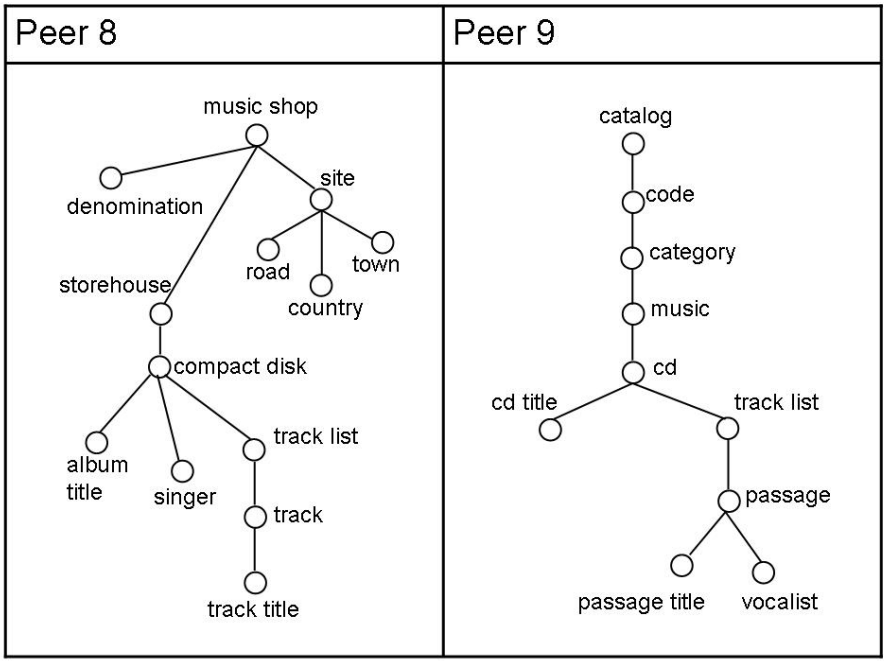

Figura 6.4: Esempi di schemi di argomento musicale.

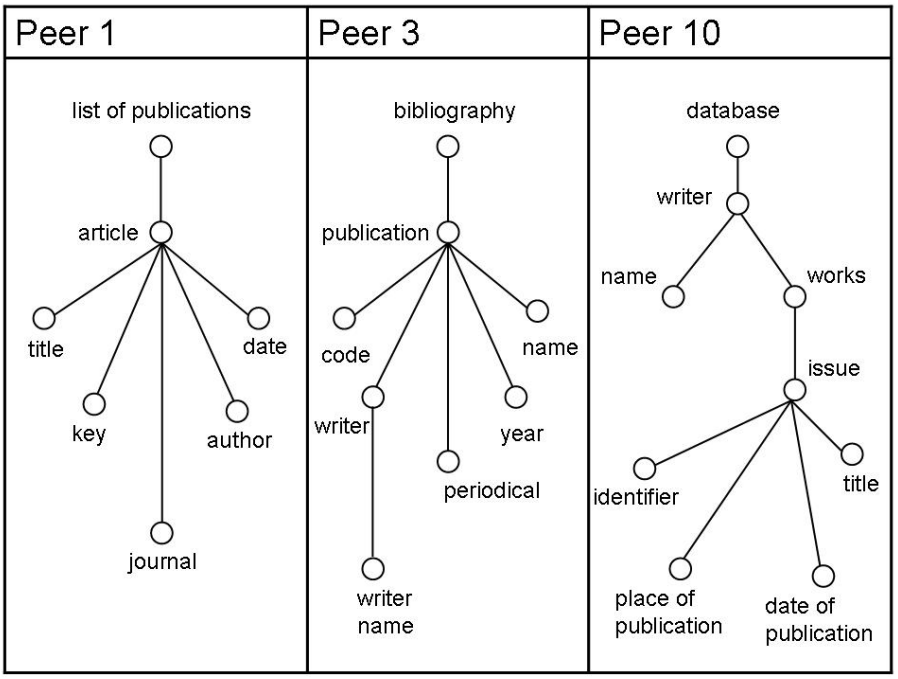

Figura 6.5: Esempi di schemi di argomento bibliografico.

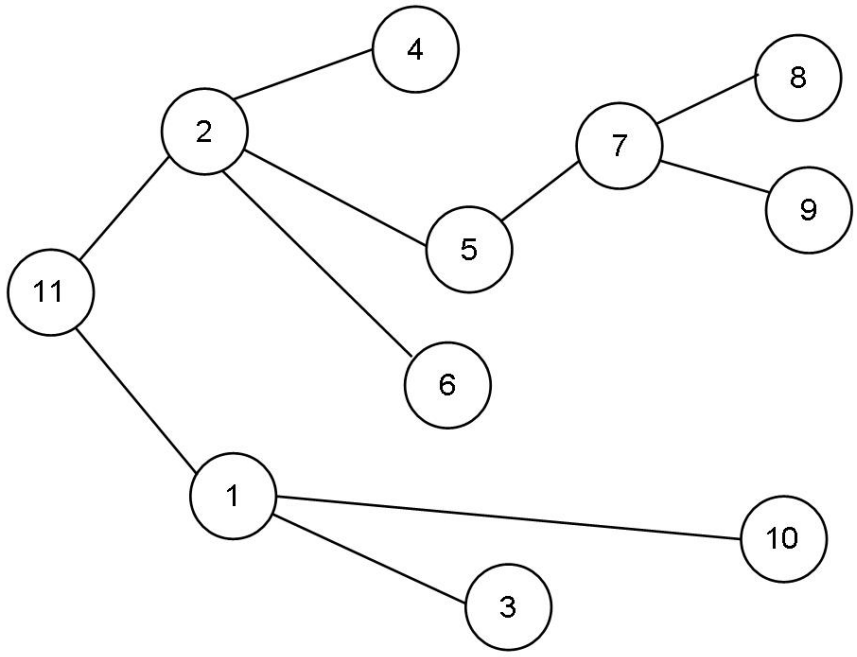

Figura 6.6: Scenario per le prove sperimentali.

negozi di cd musicali, di cui alcuni esempi sono rappresentati nelle figure 6.3 e 6.4;

- alla terza categoria appartengono invece schemi che contengono informazioni bibliografiche relative ad articoli e pubblicazioni scientifiche, come quelli di figura 6.5.

In figura 6.6 viene rappresentata una porzione della rete impiegata per la simulazione in cui è indicato come i vari schemi sono stati assegnati ai peer di tale porzione.

6.3 Risultati sperimentali

6.3.1 Confrontabilità dei mapping

La prima parte delle prove sperimentali è stata eseguita con lo scopo di verificare se la nuova versione realizzata per il sistema XML S^3MART e presentata nel

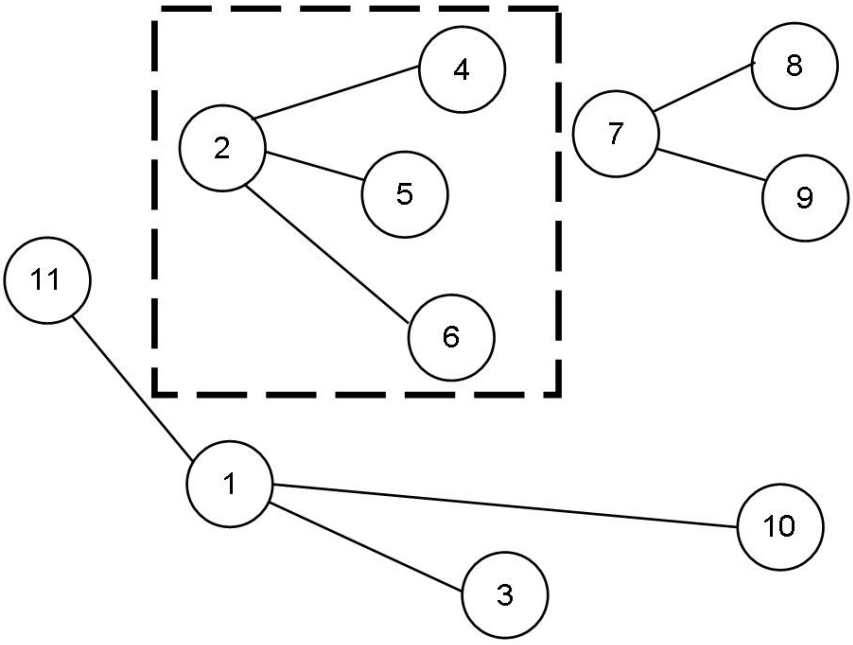

Figura 6.7: Scenario per la prima tipologia di prove sperimentali.

capitolo 4 si comporta effettivamente come richiesto nel calcolare i punteggi di mapping.

In particolare, lo scopo era di verificare se i valori risultanti dall'applicazione dell'algoritmo di schema matching tra uno schema source e un certo insieme di schemi target sono effettivamente confrontabili tra loro per ogni termine dello schema di partenza. Come è stato spiegato nel capitolo 4, se tali punteggi sono confrontabili significa infatti che possono essere impiegati dal peer che corrisponde allo schema source per comprendere quale peer, tra quelli descritti dagli schemi target, approssima in maniera più fedele i suoi concetti.

Sono state quindi eseguite numerose prove applicando l'algoritmo di schema matching agli schemi presentati nel paragrafo 6.2 e confrontando i risultati ottenuti con quanto può essere stabilito secondo il senso comune da qualsiasi essere umano. I diversi casi osservati hanno permesso di concludere che generalmente il comportamento del sistema corrisponde a quanto viene correttamente previsto.

Di seguito viene riportato un esempio di questa tipologia di esperimenti. Nell'interpretare l'esempio si tenga presente che i punteggi di approssimazione tra i concetti non hanno di per sé significato se vengono interpretati come valori

Peer 2	**Peer 4**		**Peer 5**		**Peer1**	
collection	dataset	0.38124	musicstore	0.23863	list	0.23210
sport	sport	0.19652	storage	0.01931		
team	club	0.12020	track	0.03564	article	0.05854
denomination	clubname	0.38228	tractitle	0.33562	key	0.07239
country	clublocation	0.22385	location	0.21234		
coach	trainer	0.38582	signboard	0.07654	journal	0.06064
members			stock	0.05332		
player	athlete	0.17219	singer	0.08345	author	0.09620
name	name	0.40997	namesing	0.40433	title	0.36028
birth	birthday	0.09855	colorsign	0.01542	date	0.08965
role			albumtitle	0.01511		

Tabella 6.1: Valori dell'indice di routing semantico del peer 2.

assoluti e devono invece essere considerati in relazione ai punteggi che si hanno per lo stesso concetto source verso altri concetti target. L'esempio è relativo alla porzione di rete che comprende i peer 2, 4, 5 e 1, collegati tra loro come in figura 6.7. I risultati per i termini dello schema di partenza del peer 2 verso i termini degli schemi degli altri tre peer sono riportati nella tabella 6.1. Come si può osservare, l'algoritmo associa in maniera corretta ogni termine dello schema 2 al corrispondente sinonimo dello schema 4. Negli schemi 5 e 1 non sono invece presenti dei termini che corrispondono a quelli dello schema 2 e l'algoritmo crea comunque le associazioni che ritiene migliori poiché il suo compito è di creare le coppie in ogni caso. L'aspetto significativo è però che ad ognuna di queste coppie, che per il senso comune sono considerate "sbagliate", viene attribuito un punteggio minore rispetto alla corrispondente coppia "corretta" con lo schema 4. Per tutti i termini dello schema del peer 2, i punteggi maggiori si ottengono infatti per il corrispondente termine dello schema del peer 4 e il peer 2 può quindi dedurre che il peer 4 è quello tra i suoi vicini che approssima meglio tutti i suoi concetti.

6.3.2 Efficacia degli indici di routing semantici

Nella seconda parte delle prove sperimentali è stato implementato il protocollo di interazione tra i peer presentato nel capitolo 5.2.1, in modo da introdurre il sistema nell'ambiente Peer-to-Peer simulato con SimJava.

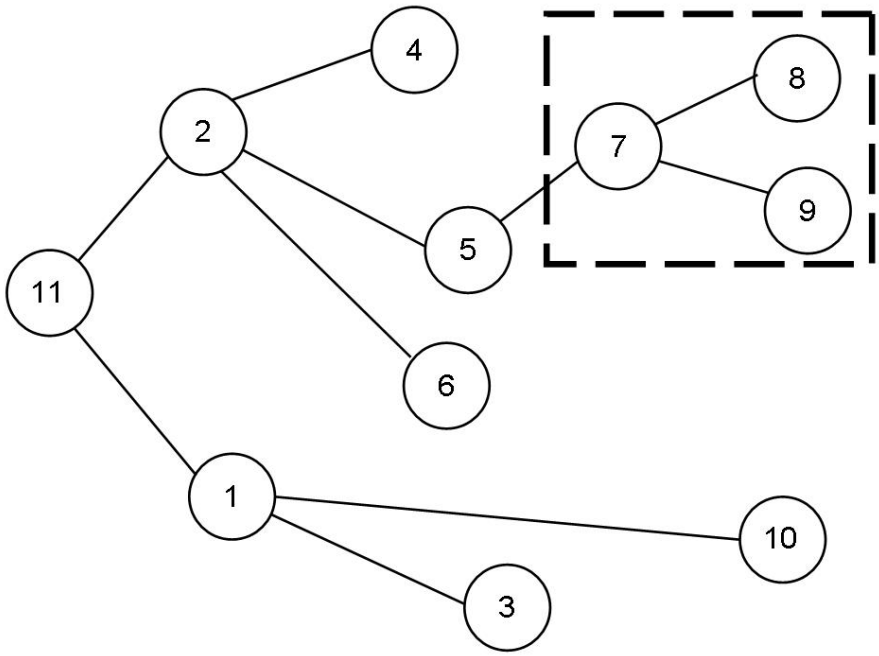

Figura 6.8: Scenario iniziale per la seconda tipologia di esperimenti.

Sono state fatte diverse prove, eseguendo più volte il programma di simulazione, in modo da ottenere diverse configurazioni finali per la rete di peer e potere di conseguenza osservare come gli indici di routing semantici di questi peer vengono modificati quando cambiano le sottoreti a cui tali peer sono collegati.

É stato inoltre considerato l'aspetto relativo alla scelta delle specifiche funzioni matematiche da impiegare per implementare le operazioni di composizione e aggregazione che vengono applicate alle righe degli indici di routing semantici. Come è stato illustrato nel capitolo 5.1.3, per la scelta di queste operazioni sono infatti possibili diverse alternative, opportunamente selezionabili al momento dell'esecuzione del sistema. Sono stati quindi raccolti i risultati relativi a prove eseguite con diverse combinazioni disponibili.

Nell'insieme delle prove eseguite si è potuto osservare che generalmente, in modo concorde con quanto desiderato, l'indice di routing semantico di un peer collegato ad una sottorete che approssima bene i suoi concetti, vede diminuire i suoi valori se tale sottorete viene sostituita da un'altra sottorete che tratta concetti più lontani.

Si consideri ad esempio la porzione di rete in figura 6.8, in cui il peer 5 ha

Peer 5	Peer 7	a	b	c	d
town	city	0.0	0.0	0.0	0.0
tracklist	songlist	0.268992	0.025158	0.042001	0.268991
track	song	0.132590	0.008048	0.012210	0.132590
singer	vocalist	0.375080	0.064790	0.104290	0.375080
namesign	name	0.376637	0.013937	0.023522	0.383815
compact disk	cd	0.211248	0.011255	0.019025	0.211248
storage		0.0	0.0	0.0	0.0
country	state	0.0	0.0	0.0	0.0
album title	cd title	0.384806	0.068632	0.108081	0.384806
stock		0.0	0.0	0.0	0.0
location	address	0.097514	0.0	0.0	0.119395
colorsign		0.0	0.0	0.0	0.0

Tabella 6.2: Valori dell'indice di routing semantico del peer 5.

uno schema di argomento musicale ed è collegato alla sottorete che ha origine dal peer 7 e che comprende tre peer dello stesso argomento.

I valori della riga dell'indice di routing semantico del peer 5 relativa al peer 7 sono rappresentati in tabella 6.8, nella quale sono riportate quattro diverse alternative risultanti dall'applicazione delle funzioni matematiche corrispondenti a:

- a: massimo per l'aggregazione e minimo per la composizione;
- b: massimo per l'aggregazione e prodotto per la composizione;
- c: somma algebrica per l'aggregazione e minimo per la composizione;
- d: somma algebrica per l'aggregazione e prodotto per la composizione.

Come si può osservare i risultati sono molto diversi nei quattro casi considerati. Tuttavia i valori assoluti non hanno di per sé significato e per essere interpretati correttamente devono invece essere considerati in relazione a quelli calcolati in seguito modificando la rete.

Se la sottorete che ha origine dal peer 7 viene sostituita con la sottorete che ha origine dal peer 1 e che comprende tre peer che trattano contenuti scientifici (si veda la figura 6.9), i punteggi dell'indice di routing semantico di 5 per la riga corrispondente al vicino 1 diventano quelli riportati in tabella 6.9.

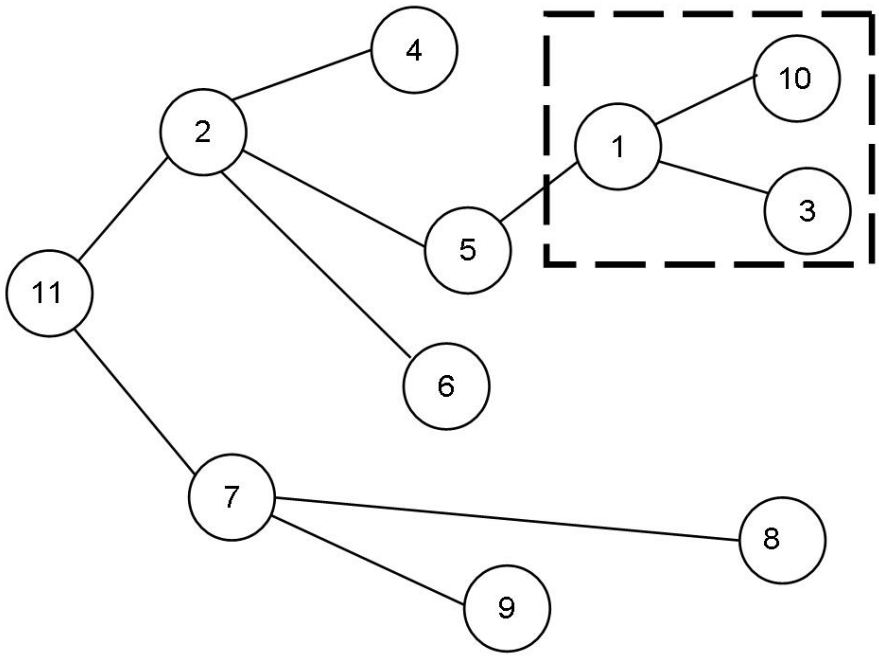

Figura 6.9: Scenario modificato per la seconda tipologia di esperimenti.

Peer 5	**Peer 1**	**a**	**b**	**c**	**d**
town		0.0	0.0	0.0	0.0
tracklist	key	0.190745	0.007994	0.014141	0.198278
track	article	0.053089	0.0	0.0	0.053088
singer	author	0.215323	0.040951	0.064299	0.215322
namesign		0.0	0.0	0.0	0.0
compact disk		0.0	0.0	0.0	0.0
storage		0.0	0.0	0.0	0.0
country		0.0	0.0	0.0	0.0
album title	date	0.012744	0.002380	0.002747	0.012744
stock		0.0	0.0	0.0	0.0
location		0.0	0.0	0.0	0.0
colorsign		0.0	0.0	0.0	0.0

Tabella 6.3: Valori dell'indice di routing semantico del peer 5.

Come si può osservare, anche in questo caso i punteggi dipendono fortemente dalle funzioni matematiche scelte, ma l'aspetto importante è che i valori che si ottengono sono in ogni caso minori rispetto a quelli corrispondenti ottenuti in precedenza. Questo significa che i valori dell'indice di routing del peer 5 rispecchiano in maniera corretta il fatto che i suoi concetti vengono approssimati meglio dalla sottorete che ha origine da 7 rispetto a quella che ha origine da 1.

Per quanto riguarda la scelta delle specifiche funzioni matematiche da impiegare si ha che, per gli esempi considerati, di cui quello riportato è una rappresentazione significativa, la combinazione che sembra mostrare il comportamento migliore è la d (somma algebrica per l'aggregazione e prodotto per la composizione). Questa conclusione deriva dall'osservazione dei risultati ottenuti per diversi peer, dopo averli inseriti in una situazione analoga a quella presentata. La combinazione d sembra essere preferibile rispetto alle altre perché, nella maggior parte dei casi, è quella che produce risultati numerici più distanziati tra loro quando la sottorete di un peer, a lui semanticamente vicina, viene sostituita da una nuova sottorete che tratta contenuti diversi.

Conclusioni e sviluppi futuri

Nell'ambito di questa Tesi di Laurea è stato affrontato lo studio delle problematiche relative al processo di routing delle interrogazioni in un PDMS.

In particolare, si è scelto di impiegare il sistema XML S^3MART per eseguire il processo di creazione dei mapping tra i vari peer, che consente di risolvere l'eterogeneità semantica esistente tra i loro dati. Non essendo originariamente applicabile in un contesto P2P, XML S^3MART è stato sottoposto ad un adeguato processo di reingegnerizzazione, nel corso del quale sono state eseguite le modifiche necessarie ad inserirlo in un PDMS. La nuova versione realizzata per il sistema si presta quindi ad essere impiegata da ogni entità componente un PDMS per calcolare i mapping verso le entità vicine ed è in grado di associare a questi mapping dei punteggi numerici, appartenenti all'intervallo [0,1] e confrontabili tra loro, che esprimono il livello di approssimazione semantica tra i concetti coinvolti.

Si è inoltre pensato di poter sfruttare le informazioni espresse dai punteggi di approssimazione semantica per implementare un meccanismo di routing capace di indirizzare le query verso i nodi che meglio approssimano i concetti da esse richiesti e che di conseguenza hanno una maggiore probabilità di risolverle con successo. A tal fine sono state progettate delle particolari strutture dati, dette *indici di routing semantici*, all'interno delle quali ogni peer mantiene, per ogni suo vicino, un punteggio descrivente un'informazione riassuntiva di come i suoi concetti vengono approssimati nella sottorete che ha origine dal vicino stesso. Il processo di creazione delle informazioni riassuntive da inserire e mantenere all'interno degli indici di routing semantici è stato modellato sulla base di una teoria matematica che si ispira alla logica fuzzy.

É stato inoltre progettato un protocollo che consente ai vari peer di collaborare nelle operazioni di creazione e aggiornamento degli indici di routing ed è stato definito e implementato l'algoritmo che definisce il comportamento dei

peer descritto da questo protocollo.

Infine, è stato creato un appropriato ambiente di simulazione all'interno del quale è stato possibile eseguire il sistema realizzato ed effettuare le prove sperimentali volte a testarne l'efficacia.

Il lavoro prodotto si inserisce nell'ambito di una problematica ben più vasta di quanto è possibile trattare in una Tesi di Laurea e si presta di conseguenza a molti e interessanti futuri sviluppi sotto diversi aspetti.

Alcune potenziali evoluzioni sono già state parzialmente introdotte nel capitolo 5.3, nel quale è stato presentato come gli indici di routing semantici potrebbero essere effettivamente impiegati all'interno di un PDMS per implementare il meccanismo di "routing by mapping". L'idea è che ogni peer impieghi i valori disponibili all'interno del suo indice per stabilire un ranking tra i vicini, che rispecchia la loro correlazione con i concetti di ogni query pervenuta. In questo modo, ogni query potrebbe essere propagata ai soli vicini che sono rilevanti per essa e che si trovano di conseguenza nelle posizioni più alte del ranking.

Sicuramente sarebbe poi utile modificare il processo di routing sfruttando anche ulteriori informazioni rispetto ai soli valori di approssimazione semantica. In particolare, come è esposto nel capitolo 5.3.2, un tipo di informazione che sarebbe sicuramente utile poter considerare è rappresentato dal numero di documenti che sono presenti su ogni peer, poiché in tal caso sarebbe possibile interpretare i punteggi di mapping semantici anche in riferimento alla quantità di dati che con tale punteggio sono effettivamente recuperabili. Queste informazioni quantitative potrebbero essere a loro volta mantenute all'interno di apposite strutture dati, la cui creazione ed evoluzione potrebbe essere gestita in maniera del tutto analoga a quanto avviene per gli indici di routing semantici.

Un'ulteriore tipologia di informazione di natura quantitativa potrebbe essere relativa non solo agli elementi richiesti dalle query, ma anche ai valori che tali elementi devono avere per soddisfare le query stesse. Queste informazioni quantitative sulla estensione dei concetti di ogni peer potrebbero essere mantenute all'interno di strutture dette *content summary*, in grado di esprimere una sorta di conoscenza statistica sui dati dei peer.

Per quanto riguarda i punteggi di approssimazione semantica sarebbe poi possibile pensare di mantenerne di più accurati, calcolandoli separatamente per i peer che si trovano a diverse distanze (espresse in numero di hop) dal peer sour-

ce. In questo modo sarebbe possibile valutare meglio la qualità di informazioni raggiungibili allontanandosi lungo la rete e i costi da sostenere per recuperarle.

Molto lavoro è inoltre ancora possibile per quanto riguarda la simulazione. Per prima cosa si potrebbe testare il sistema con reti dalla topologia più complessa e costituite da un numero maggiore di peer, in modo da verificare il suo comportamento anche in queste condizioni. In secondo luogo si potrebbe introdurre la dimensione di analisi legata all'efficienza, inserendo di conseguenza le componenti legate alle problematiche di rete coinvolte, quali il traffico generato e i ritardi sperimentati. Infine sarebbe interessante riuscire a simulare il comportamento in presenza di query reali, che coinvolgono più concetti ed operatori relazionali.

Per quanto riguarda invece l'effettivo impiego di quanto realizzato all'interno del progetto WISDOM, sarà necessario introdurre l'ambiente ontologico ed abbandonare di conseguenza il linguaggio XML. Tutto ciò comporterà delle modifiche sia alla fase di schema matching sia al successivo processo di riscrittura, a cui il sistema XML S^3MART dovrà di conseguenza essere adattato.

Bibliografia

[1] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The Hyperion project: from data integration to data coordination. *SIGMOD Record, 32(3)*, 2003.

[2] D. Beckett, E. Miller, and D. Brickley. Expressing simple Dublin Core in RDF/XML. Technical report, Dublin Core Metadata Initiative, 2002.

[3] D. Beneventano and S. Bergamaschi. The MOMIS Methodology for Integrating Heterogeneous Data Sources. In *18th World Computer Congress of International Federation for Information Processing (IFIP)*, 2004.

[4] T. Berners-Lee and O. Lassila J. Hendler. *The Semantic Web*. Scientific American, 2001.

[5] P. Bernstein, F.Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A Vision. In *Proceedings of the 5th International Workshop on the Web and Databases (WebDB)*, 2002.

[6] P. Bosc, M. Galibourg, and G. Hamon. Fuzzy Querying with SQL: Extensions and Implementation Aspects. *Fuzzy Sets and Systems*, 1988.

[7] T. Bray, J. Paoli, C. M. Sperger-McQueen, E. Maller, and F. Yergeau. *Extensible Markup Language (XML) 1.0*, 2004. W3C Recommendation.

[8] D. Brickley and R. V. Guha. *OWL Web Ontology Language Overview*, 2004. W3C Recommendation.

[9] D. Brickley and R. V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*, 2004. W3C Recommendation.

[10] J. Caroll and G. Klyne. *Resource Description Framework (RDF): Concept and Abstract Syntax*, 2004. W3C Recommendation.

[11] P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. Fuzzy Query Languages for Multimedia Data. In *Design and Management of Multimedia Information Systems: Opportunuties and Challenges*. IDEA Group Publishing, 2001.

[12] I. Clarke, O. Sandberg, and B. Wiley. Freenet: a distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability (DIAU)*, 2000.

[13] S. Cluet, P. Veltri, and D. Vodislav. Views in a large scale XML repository. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, 2001.

[14] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[15] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks. Technical report, Stanford University, 2003.

[16] P. Druschel and A. Rowstron. Past: a large scale, persistent peer-to-peer storage utility. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS)*, 2001.

[17] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, 1980.

[18] D. Dubois and H. Prade. A Review of Fuzzy Set Aggregation Connectives. *Information Sciences, 36*, 1990.

[19] R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems (PODS)*, 1996.

[20] R. Fagin. Fuzzy queries in Multimedia Database Systems. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems (PODS)*, 1998.

[21] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Processing Queries in a Large Peer-to-Peer System. In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAISE)*, 2003.

[22] H. Garcia-Molina, Y. Papakonstantinou, A. Rajaraman D. Quass, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. *Journal of Intelligent Information Systems*, 1997.

[23] The genome@home project web site. http://genomeathome.stanford.edu.

[24] The Gnutella web site: http://gnutella.wego.com.

[25] L. Gong. Project JXTA: A technology overview. Technical report, SUN Microsystems, 2001.

[26] Learning Object Metadata Working Group. Draft standard for learning object metadata. Technical report, IEEE Learning Technology Standards Committee(LTSC), 2002.

[27] Stanford Peers Group. Peer-to-Peer Research at Stanford. *SIGMOD Record*, 2003.

[28] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. *IEEE Transactions on Knowledge and Data Engineering (IKDE), 16(7)*, 2004.

[29] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the 12th International Conference on World Wide Web (WWW*, 2003.

[30] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engeneering (ICDE)*, 2003.

[31] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the internet with PIER. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, 2003.

[32] The ICQ web site. http://web.icq.com.

[33] The Jabber web site. http://www.jabber.org.

[34] The project JXTA web site. http://www.jxta.org.

[35] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *Proceedings of the 2002 ACM International Conference on Information and Knowledge Management (CIKM)*, 2002.

[36] The Kazaa web site. http://www.kazaa.com.

[37] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In *SIGMOD Conference*, 2003.

[38] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, S. Gummadi, H. Weatherspoon, W. Weimer, C.Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. In *Proceedings of the 9th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[39] S. Larson, C. Snow, and V. Pande. *Modern Methods in Computational Biology* . Horizon Press, 2003.

[40] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB)*, 1996.

[41] A. LoSer, W. Nejdl, M. Wolpers, and W. Siberski. Information Integration in Schema-Based Peer-to-Peer Networks. In *Proceedings of the 15th Conference On Advanced Information Systems Engineering (CAISE)*, 2003.

[42] J. Madhavan, P. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based Schema Matching. In *Proceedings of IJCAI Workshop on Information Integration on the Web (IIWeb)*, 2003.

[43] F. Mandreoli, R. Martoglia, and P. Tiberio. Approximate Query Answering for a Heterogeneous XML Document Base. In *Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE)*, 2004.

[44] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: a Versatile Graph Matching Algorithm and its Application to Schema Matching. In *Proceedings of the 18th International Conference on Data Engeneering (ICDE)*, 2002.

[45] M. Mizumoto. Pictorial Representations of Fuzzy Connectives: Cases of T-norms, T-conorms and Averaging Operators. *Fuzzy Sets and Systems, 31*, 1989.

[46] The MojoNation web site. http://www.mojonation.net.

[47] The Napster web site: http://napster.com.

[48] W. Nejdl, M.Wolpers, W. Siberski, C. Schmitz, M. T. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based Routing and Clustering Strategies

for RDFbased Peer-to-Peer Networks. In *Proceedings of the 12th ACM International World Wide Web Conference (WWW)*, 2003.

[49] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. Edutella: a p2p networking infrastructure based on RDF. In *Proceedings of the 12th International Conference on World Wide Web (WWW)*, 2003.

[50] W. Penzo. Rewriting Rules to Permeate Complex Similarity and Fuzzy Queries within a Relational Database System. *IEEE Transactions on Knowledge and Data Engineering (IKDE), 17(2)*, 2005.

[51] H. Prade and C. Testemale. Generalizing Database Relational Algebra for the Treatment of Incomplete and Uncertain Information and Vague Queries. *Information Sciences, 34*, 1984.

[52] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *Proceedings of the 16th ACM Conference on Application, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, 2001.

[53] The seti@home project web site. http://setiathome.ssl.berkeley.edu.

[54] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 16th ACM Conference on Application, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM)*, 2001.

[55] W. Sullivan, D. Werthime, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson. A new major seti project based on project serendip data and 100,000 personal computers. In *Proceedings of the 5th International Conference on Bioastronomy*, 1997.

[56] I. Tatarinov and A. Halevy. Efficient Query Reformulation in Peer Data Management Systems. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2004.

[57] S. A.. Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Survey*, 2004.

[58] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelshon. *XML Schema*, 2001. W3C Recommendation.

[59] M. Waldman, R. AD, and C. LF. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[60] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[61] Q. Yang, W. Zhang, C. Liu, J. Wu, C. Yu, H. Nakajima, and N. D. Rishe. Efficient Processing of Nested Fuzzy SQL Queries in a Fuzzy Database. *IEEE Transactions on Knowledge and Data Engineering (IKDE), 17(2)*, 2005.

[62] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, University of California, 2001.

[63] H. J. Zimmermann. *Fuzzy Set Theory*. Kluwer Academic Publishers, 1996.