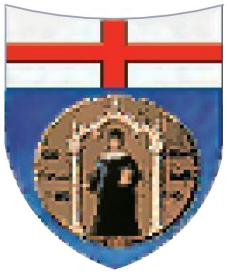
Università degli Studi di Genova
Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea in Informatica



Anno Accademico 2003/2004

Tesi di Laurea

SINAPSY P2P: un approccio basato su ontologie per il recupero di dati in un sistema P2P

Candidato
Chiara Casanova

Relatore
Prof. Giovanna Guerrini
Dott. Viviana Mascardi

Correlatore
Prof. Barbara Catania

Indice

Introduzione

Il World Wide Web (WWW) ha cambiato drasticamente la disponibilità delle informazioni elettroniche accessibili in Internet, ma il suo successo e la sua crescita esponenziale hanno reso difficile il ritrovamento, l'accesso, la presentazione ed il mantenimento delle informazioni utilizzate da una grande quantità di utenti. Questo ha condotto alla necessità di rivisitare la nozione di accesso alle informazioni in tale contesto poiché le tecniche usate nelle basi di dati distribuite, sviluppate per gestire i dati distribuiti su un numero limitato di server, hanno raggiunto i loro limiti nel contesto di Internet non supportando la scalabilità. È quindi stata sviluppata una nuova classe di sistemi distribuiti, i sistemi Peer to Peer (P2P) [1], che consentono di supportare migliaia di nodi fornendo soluzioni interessanti dal punto di vista della scalabilità.

Durante gli ultimi anni il paradigma di rete P2P è diventato molto popolare sia fra i privati sia nel settore aziendale, in contrasto con il ben noto modello di rete Client-Server che è predominante sia in Internet sia nelle reti LAN (Local Area Network). Come suggerisce il suo nome esso definisce una partecipazione a pari livello dei nodi nella rete che possono ricoprire indifferentemente i ruoli di client e server e la cui conoscenza della rete è limitata ad un sottoinsieme di nodi (vicini).

L'essenza del P2P è quella che i nodi nella rete sfruttano direttamente le risorse presenti su altri nodi della rete senza l'intervento di un server centrale. L'enorme successo di sistemi quali Napster [40], Gnutella [22], SETI@home [55] e KazaA [31] e le iniziative di industrie altamente visibili come JXTA [29] di Sun ed il P2P Working Group che include HP, IBM ed Intel, hanno mostrato che il P2P è uno dei paradigmi più potenti quando si tratta di condividere risorse su Internet senza una base di dati centrale, senza un'amministrazione centralizzata e fra utenti che necessitano di lavorare in modo robusto e scalabile.

Una delle funzionalità principali dei sistemi P2P è la scoperta di risorse nella rete. Tuttavia, malgrado l'*Information Retrieval (IR)* sia stato descritto come una delle maggiori applicazioni dei protocolli P2P, l'attività di ricerca

in questo campo è ancora in fase embrionale.

Esistono innumerevoli sorgenti di informazione potenziali su Internet ognuna con le proprie caratteristiche, politiche, architetture hardware e software. Si ritiene che le soluzioni P2P possano agevolare l'utilizzo di Internet tramite la costruzione di gruppi di peer attivi e collaborativi che si scambiano informazioni riguardanti le caratteristiche della rete sottostante e delle politiche di recupero.

In quasi tutti i tipi di architetture P2P la risoluzione delle interrogazioni è basata su algoritmi di *flooding* che consentono la propagazione da nodo a nodo sinché il numero di "rimbalzi" tra i nodi è considerato sufficiente. Il nodo ricevente risolve l'interrogazione localmente e la propaga all'insieme dei suoi vicini. Tuttavia il principio della propagazione ha qualche inconveniente [14]. Infatti, in reti non strutturate la definizione di vicino di un nodo non possiede nessuna semantica particolare e la propagazione delle interrogazioni genera un considerevole numero di messaggi attraverso la rete. I sistemi ed i protocolli P2P che trasmettono tutte le interrogazioni a tutti i peer non sono scalabili. È pertanto richiesto un instradamento dell'interrogazione intelligente che selezioni un sottoinsieme di peer rilevanti per una data richiesta.

I protocolli di instradamento moderni, come Chord [12, 60] e CAN [51], sono basati sull'idea di utilizzare tabelle hash distribuite per un instradamento dell'interrogazione efficiente. Putroppo, però, anche in tali sistemi è stato fatto solo un piccolo sforzo nei confronti dell'evoluzione dalla ricerca basata su parole chiave verso interrogazioni basate su rappresentazioni semantiche dei meta-dati.

Esistono molti meccanismi per la ricerca di informazioni in un sistema P2P ognuno con i propri vantaggi e difetti. Queste soluzioni possono essere classificate in tre categorie: meccanismi senza indici, ricerca centralizzata (meccanismo con indici solo ad alcuni specifici nodi) e ricerca distribuita (meccanismo con indici ad ogni nodo). Gnutella utilizza un meccanismo dove i nodi non hanno un indice e le interrogazioni sono propagate da un nodo all'altro sinché non viene trovato un insieme di documenti che soddisfi l'interrogazione. Sebbene tale approccio sia semplice e robusto, ha lo svantaggio di rallentare la rete ogni volta che viene generata un'interrogazione.

Sistemi di ricerca centralizzati usano, invece, nodi speciali per mantenere un indice dei documenti disponibili nel sistema P2P. Per trovare un documento, l'utente interroga un nodo indice per identificare i nodi che possiedono i documenti con il contenuto di interesse. Questi indici centralizzati possono essere costruiti attraverso la cooperazione dei nodi (ad esempio i nodi in Napster forniscono una lista dei file disponibili al momento della registrazio-

ne) oppure attraverso l'ispezione della rete P2P (come accade nei motori di ricerca per il Web).

Il vantaggio di una ricerca centralizzata è l'efficienza (è necessario un singolo messaggio per risolvere un'interrogazione). Tuttavia, un sistema centralizzato è vulnerabile ad attacchi ed è difficile mantenere gli indici aggiornati. In un sistema di indicizzazione distribuito gli indici sono mantenuti da ogni nodo e pertanto devono essere di piccole dimensioni. Quelli più adatti sono gli indici di instradamento (*Routing indices*) che forniscono la "direzione" attraverso la quale è possibile recuperare un documento piuttosto che fornire la sua locazione attuale. Usando i *Routing indices* la dimensione dell'indice è proporzionale al numero dei peer vicini ad ogni peer.

Solo una minima parte dei sistemi P2P esistenti si preoccupa di instradare le richieste tenendo conto della loro semantica e della semantica delle risorse condivise dai vari peer ed i pochi sistemi che se ne occupano sono dei prototipi sviluppati per scopo di ricerca [50, 63, 25, 10, 62, 28, 35]. In altre parole, la minima parte dei sistemi P2P fa proprie le problematiche che caratterizzano il Semantic Web. In accordo alla definizione di Berners-Lee, Hendler e Lassila [4]:

> Il Semantic Web è un'estensione dell'attuale Web in cui l'informazione è presentata con un significato ben definito, meglio comprensibile dai computer e dalle persone che lavorano in cooperazione.

Uno degli strumenti più utili per fornire una semantica alle applicazioni di rete sono le ontologie. Le relazioni esistenti fra ontologie ed il Semantic Web sono chiaramente illustrate da James Hendler, il principale scienziato del DARPA Information Systems Office ed il "program manager" responsabile per la computazione basata su agenti, il quale ha dato una sua personale visione del Semantic Web [26]:

> Prevedo un Web Semantico regolato dallo stesso tipo di anarchia che regola il resto del Web. Invece di ontologie consistenti e complesse che condividono un gran numero di utenti, vedo un grande numero di piccole componenti ontologiche che si referenziano tra loro. Gli utenti del Web svilupperanno queste componenti nello stesso modo in cui verranno creati i contenuti Web. Nei prossimi anni ogni compagnia, università, agenzia governativa o gruppi di interesse ad hoc vorranno collegare le loro risorse Web a ontologie in quanto esisteranno strumenti potenti per l'utilizzo di tali risorse. Le informazioni verranno scambiate liberamente fra le applicazioni permettendo ai programmi di collezionare e processare

> i contenuti del Web. Utilizzando questa infrastruttura la computazione basata sugli agenti diventerà più pratica. I programmi su computer distribuiti interagendo con risorse non locali basate sul Web, potrebbero diventare il modo principale con cui i computer interagiscono con gli umani.

Attualmente il Semantic Web fornisce una struttura comune che consente la condivisione e il riutilizzo dei dati da parte di applicazioni e di sistemi industriali. Uno dei problemi che affligge il Semantic Web è che il recupero delle informazioni basato sul contesto è ancora poco considerato. La combinazione fra il P2P e la tecnologia del Semantic Web sarà un percorso interessante per muoversi da soluzioni per la gestione della conoscenza più centralizzate ad un approccio decentralizzato. Gli scenari P2P aprono la via per derivare concettualizzazioni consensuali, ad esempio fra gli impiegati di un'azienda, e rendono le soluzioni per la gestione della conoscenza più vicine ad essere libere da un'amministrazione centralizzata, in modo da poter essere usate da chiunque, inclusi privati e piccole compagnie facenti parte, ad esempio, di una compagnia virtuale.

Negli ultimi anni l'utilizzo della parola ontologia è diventato sempre più frequente nell'ambito dell'Informatica. Nonostante continui a mancare una definizione universalmente riconosciuta di cosa sia un'ontologia, quella fornita da Tom Gruber meglio caratterizza l'essenza di un ontologia rispetto alle altre [24]:

> Un'ontologia è una specifica formale ed esplicita di una concettualizzazione condivisa.

Soltanto alcuni dei sistemi P2P proposti considera, però, la possibilità di avere un'ontologia comune condivisa dai peer per rappresentare le informazioni riguardanti le risorse utilizzando un "linguaggio" comune comprensibile da tutti i partecipanti al sistema [16]. Una minima parte dei sistemi proposti considera che non è sempre possibile avere un accordo generale tra i peer della rete sul significato dei concetti dell'ontologia.

Oltre al problema dell'instradamento semantico delle richieste, i sistemi attuali ed i nuovi proposti dipendono dall'assunzione che quando i peer sono connessi alla rete sono sempre disponibili a rispondere alle interrogazioni da chiunque esse provengano. Tuttavia, i peer possono desiderare di configurare delle politiche sulla loro disponibilità che dipendono da diversi fattori quali condizioni temporali (ad esempio il periodo del giorno nel quale la richiesta arriva), condizioni sullo stato interno (ad esempio carico di lavoro) e sulla tipologia di connessione del peer ricevente e condizioni sui certificati posseduti dai peer mittenti (ossia caratteristiche del peer che sottomette la richiesta).

Il concetto di politica [48] è piuttosto nuovo nell'ambito del P2P e pochi sistemi prendono in considerazione la possibilità che i peer possano non essere sempre disponibili a condividere risorse e rispondere a richieste. Le politiche di personalizzazione associate ai peer sono un insieme di regole che stabiliscono ed influenzano il comportamento del sistema e dipendono da fattori quali la collocazione degli utenti nella rete, le loro preferenze, i certificati ad essi associati, la loro appartenenza ad un gruppo, condizioni temporali, condizioni relative al tipo di connessione, etc. [59]. A partire dal comportamento di base del sistema le politiche di personalizzazione consentono agli utenti di personalizzare il comportamento della porzione del sistema con cui interagiscono come più preferiscono. La maggior parte degli approcci che tengono in considerazione l'utilizzo di politiche di personalizzazione le integrano nelle componenti della rete. Tale scelta rende questi approcci efficienti ma risulta non facile l'eventuale modifica delle politiche e la loro comprensione. Tra i vari approcci per la rappresentazione di politiche si preferisce quello dichiarativo, più flessibile e comprensibile, dove le politiche vengono descritte attraverso regole. In questo campo ci sono, comunque, ancora molte quastioni aperte [39].

L'obiettivo che questa tesi si prefigge è quello di trovare una soluzione al problema dell'instradamento delle interrogazioni ed al recupero delle informazioni rilevanti per una richiesta dati tramite l'utilizzo di ontologie che forniscono semantica ai dati. Nel sistema proposto l'instradamento semantico delle richieste dati è guidato sia dalla pubblicizzazione delle comptenze dei peer sia dalla rilevanza delle risposte fornite dai peer a richieste precedenti. Inoltre il sistema integra le nozioni di certificato e politica al fine di realizzare un sistema P2P che includa un meccanismo di condivisione delle risorse più flessibile, allontanandosi dall'assunzione che quando i peer sono connessi alla rete sono sempre disponibili a rispondere alle interrogazioni da chiunque esse provengano.

In questa tesi viene proposto SINAPSY P2P (A Semantic Information Retrieval Advertisement and Policy based System for a P2P network) un sistema basato su ontologie per il recupero di informazioni in una rete P2P. Il sistema si basa su un'architettura P2P pura in cui i peer reclamizzano le loro competenze, impongono politiche di disponibilità e controllo dei certificati dei peer che sottomettono una richiesta. L'unica fase che si presuppone centralizzata è quella di registrazione dove un "peer speciale" funge da server tenendo traccia di tutti i peer registrati e li restituisce quando un peer si

collega per la prima volta al sistema o su esplicita richiesta di un peer che non riesce a comunicare con nessuno dei peer di cui è a conoscenza. Durante il funzionamento del sistema i peer comunicano direttamente gli uni con gli altri senza passare per il "peer speciale". Il "peer speciale" fornisce, inoltre, l'ontologia globale condivisa da tutti i peer, rappresentata tramite un file RDFS oppure OWL, che fornisce la struttura su cui costruire la base di conoscenza di ogni peer ossia l'insieme delle istanze che rappresentano le risorse che l'utente desidera condividere nel sistema. Mentre l'ontologia globale è comune a tutti i peer, la base di conoscenza varia da peer a peer ossia le istanze dell'ontologia non sono condivise da tutti i peer.

I peer utilizzano tale ontologia condivisa per avere un linguaggio comune comprensibile da tutti i peer con cui reclamizzare le loro competenze nella rete. Un messaggio pubblicitario contiene, infatti, informazioni sul tipo di risorse che un peer intende condividere. In tal modo è possibile venire a conoscenza delle competenze di altri peer appartenenti al sistema ed, in particolare, di quelli con competenze affini alla propria. Tale meccanismo consente un instradamento semantico delle richieste dati in quanto nell'invio e nell'inoltro delle richieste privilegia, ove possibile, i peer con competenze affini a quelle dei concetti contenuti nell'interrogazione aumentando la probabilità di ottenere risposte attinenti ed adeguate.

Durante il funzionamento del sistema un peer viene, quindi, a conoscenza delle competenze di un certo numero di peer attraverso il meccanismo di pubblicizzazione delle competenze appena descritto. Quando l'utente sottomette un'interrogazione al sistema il suo peer, basandosi sulla conoscenza acquisita riguardo ad altri peer, può instradarla in modo mirato a quei peer che con più probabilità sono in grado di rispondere. Per evitare che la conoscenza dei peer resti limitata, l'interrogazione viene inoltrata anche ad un numero prefissato di peer scelto in modo casuale fra quelli conosciuti ma con competenze dissimili da quelle della richiesta. Attraverso tale meccanismo è possibile venire a conoscenza di porzioni di rete altrimenti irraggiungibili.

Nell'instradamento semantico viene comunque considerato il comportamento tenuto da un peer nelle interazioni precedenti ossia il grado di rilevanza che i peer hanno acquisito nelle precedenti risposte. Nell'invio e nell'inoltro di un'interrogazione si fanno infatti le seguenti assunzioni [63]: (i) un'interrogazione è posta ad un peer che si presuppone sappia rispondere (quindi di cui si conoscono le competenze) e che abbia risposto "bene" alle richieste dati precedenti sull'argomento dell'interrogazione attuale; (ii) un'assunzione generale è quella che se un utente è ben informato su un dominio specifico sarà, probabilmente, ben informato anche su un dominio più generale riguardante lo stesso argomento; (iii) in generale le persone possono essere più o meno esperte in modo indipendente dal domino ossia si presuppone che un

utente che abbia acquisito un grado di rilevanza alto in più di un argomento sia probabilmente una persona colta in generale e, pertanto, da privilegiare nel caso non si conoscano esperti nell'argomento della richiesta dati.

Una delle caratteristiche peculiari del sistema è quella delle politiche di disponibilità. I peer, infatti, possono imporre politiche di disponibilità che specificano sotto quali condizioni sono disponibili a condividere le risorse ed a rispondere a richieste di interrogazione da parte di altri peer nel sistema. Tali politiche possono essere specificate dall'utente in fase di registrazione e vengono controllate ogni volta che viene chiesto al peer di rispondere ad una interrogazione.

Per fornire un meccanismo più flessibile che consideri le caratteristiche dei peer che sottomettono le richieste, il sistema sfrutta il concetto di certificato che consiste in un insieme di proprietà che qualificano un utente in un'organizzazione (nel nostro caso la rete) [68]. Ogni utente può possedere uno o più certificati che asseriscono determinate sue proprietà (essere studente, essere maggiorenne, essere iscritto ad un'associazione). Tali proprietà sono utili al fine di consentire al peer di specificare politiche di condivisione delle risorse che si basino sulle proprietà dell'utente che invia la richiesta. Tali certificati vengono ottenuti dall'utente in base alle proprietà da lui specificate in fase di registrazione o in una fase successiva e vengono controllati prima di fornire risposte ad una interrogazione posta da un peer.

La tesi è strutturata nel modo seguente.

Nel **Capitolo 1** vengono analizzati i campi del WWW che necessitano maggiormente del Semantic Web e le tecnologie attualmente disponibili per realizzarlo. Verranno approfondite le motivazioni che hanno portato al Semantic Web, a partire dai problemi del WWW e verranno discussi i colli di bottiglia rispetto alla gestione della conoscenza e del commercio elettronico, sia in ambito di applicazioni B2C (Business to Consumer) sia B2B (Business to Business). Vengono mostrati, inoltre, i nuovi servizi che il Semantic Web può offrire e vengono trattate le tecnologie è necessario applicare per realizzare tali servizi. In particolare vengono descritti i linguaggi che consentono di aggiungere semantica al Web (cioè ontologie e linguaggi formali per lo sviluppo di ontologie) e gli strumenti per poter aggiungere tale semantica.

Nel **Capitolo 2** viene fornito un background sui sistemi P2P. In particolare viene specificato cosa si intende per P2P, quali classi di sistemi si possono ritenere tali e quali sono i fattori che influenzano la scelta di un tale approc-

cio (ad esempio miglioramento della scalabilità e dell'affidabilità, aumento di autonomia, aggregazione di risorse ed interoperabilità, supporto per comunicazioni e collaborazioni ad-hoc, etc.). Sono presentate una classificazione dei sistemi P2P dal punto di vista della computazione ed una classificazione dei sistemi P2P dal punto di vista delle possibili applicazioni. In particolare vengono descritte le quattro aree di applicazione principali del modello P2P (computazione distribuita, condivisione di documenti, applicazioni collaborative e piattaforme) e qualche esempio di sistema esistente (Napster, Gnutella, SETI@home, KaZaA, JXTA). Vengono, inoltre, presentate le caratteristiche generali dei sistemi P2P ponendo l'attenzione su quelle maggiormente coinvolte nell'integrazione di tali sistemi con le tecnologie del Semantic Web. Oltre ai requisiti di sicurezza, trasparenza, prestazioni e tolleranza ai guasti, un sistema P2P deve presentare caratteristiche di decentralizzazione, scalabilità ed efficienza della rete, interoperabilità ed auto-organizzazione particolarmente rilevanti per questa tesi. Infine, viene fornita una panoramica degli algoritmi P2P per il recupero delle informazioni in cui sono presentate alcune tecniche di ricerca per il recupero dell'informazione basato su parole chiave (keyword).

Nel **Capitolo 3** vengono analizzate le principali limitazioni, gli aspetti P2P specifici e gli argomenti principali che emergono per la combinazione e l'integrazione del P2P con la tecnologia del Semantic Web. Vengono inoltre introdotti alcuni sistemi P2P in cui è stata integrata la tecnologia del Semantic Web: (i)il progetto SWAP [50, 15] in cui è stata investigata la tecnologia del Semantic Web ed è stato indagato come l'uso di descrizioni semantiche di sorgenti di dati, memorizzatei nei peer, e descrizioni semantiche dei peer stessi, aiutino nella formulazione di interrogazioni tali da poter essere comprese da altri peer, aiutino nell'amalgamazione delle risposte ricevute da altri peer e nell'instradamento delle richieste attraverso la rete; (ii) un modello per la selezione dei peer basata sull'expertise [25] (competenza) in cui i peer usano un'ontologia condivisa per pubblicizzare le loro competenze in una rete P2P; (iii) un sistema P2P in cui i nodi sono collegati fra loro attraverso le relazioni semantiche che intercorrono fra i loro contenuti. Tali connessioni formano quelle che vengono definite Semantic Overlay Network (SON) [10].

Nel **Capitolo 4** vengono presentate e definite formalmente le componenti fondamentali del sistema SINAPSY P2P per il recupero dell'informazione in una rete P2P basato su ontologie che considera la disponibilità dei peer nel fornire risposte a richieste provenienti da altri peer. In particolare vengono

descritte l'organizzazione dell'ontologia, i concetti di certificato e politica, le nozioni di pubblicità e richiesta dati (interrogazione) e tutti i tipi di messaggi che i peer possono scambiarsi all'interno del sistema.

Nel **Capitolo 5** viene descritta l'architettura dei singoli peer nelle sue componenti funzionali e strutture dati. Viene descritta la fase di registrazione di un utente necessaria al fine di poter partecipare al sistema ponendo interrogazioni ed, eventualmente, publicizzando le proprie competenze. Vengono, infine, dettagliati i processi di invio, ricezione ed inoltro di un messaggio pubblicitario e di una richiesta dati di cui vengono inoltre forniti gli algoritmi.

Infine il **Capitolo** delle conclusioni chiude il lavoro e delinea alcune sue possibili estensioni future.

Capitolo 1

Semantic Web

Il World Wide Web (WWW) ha cambiato drasticamente la disponibilità delle informazioni elettroniche accessibili in Internet, ma il suo successo e la sua crescita esponenziale hanno reso difficile il ritrovamento, l'accesso, la presentazione ed il mantenimento delle informazioni utilizzate da una grande quantità di utenti. Attualmente le pagine sul Web vengono scritte tramite linguaggi di markup come HTML ed utilizzano protocolli che consentono ai browser di presentare le informazioni in un formato leggibile dall'utente umano. Linguaggi come HTML non consentono, però, di distinguere fra struttura dell'informazione e sua rappresentazione. Questo ha un grosso impatto nella ricerca di informazione, nella gestione di informazioni ridondanti e in contesti di manipolazione automatica dell'informazione come, ad esempio, il commercio elettronico.

- *Ricerca informazioni.* La ricerca è imprecisa e spesso restituisce riferimenti a migliaia di pagine. Quindi, un utente è costretto a leggere tutti i documenti restituiti per estrarre l'informazione desiderata che può risultare, pertanto, difficile da trovare e talvolta nascosta. Potrebbe, quindi, essere utile che sulla pagina Web restituita, la medesima porzione di informazione fosse presentata in contesti diversi e fosse adattata alle esigenze di diversi utenti. Tuttavia, il Web manca di strumenti per la traduzione automatica che permettono all'informazione di essere automaticamente trasformata tra contesti e formati di rappresentazione differenti.

- *Ridondanza informazioni.* La memorizzazione di informazioni ridondanti che siano consistenti e corrette è una caratteristica difficilmente supportata dagli attuali strumenti (tool) per il Web. Questo conduce ad una pletora di siti con informazioni inconsistenti e/o contraddittorie.

- *Commercio elettronico.* L'automatizzazione del commercio elettronico è seriamente ostacolata dal modo in cui l'informazione viene attualmente presentata. Gli agenti usano euristiche per estrarre le descrizioni dei prodotti da un'informazione testuale debolmente strutturata. Questo provoca un alto costo per lo sviluppo ed il mantenimento ed una limitazione nei servizi forniti.

È nata, pertanto, la consapevolezza che fornire la soluzione a questi problemi richieda una semantica comprensibile alla macchina per tutte le informazioni presenti nel WWW. Tale semantica deve permettere un accesso intelligente ad informazioni eterogenee e distribuite, permettendo a prodotti software (come gli agenti) di mediare fra le necessità dell'utente e le risorse di informazione disponibili.

> *Il Semantic Web è un'estensione dell'attuale Web in cui l'informazione è presentata con un significato ben definito, meglio comprensibile dai computer e dalle persone che lavorano in cooperazione [4].*

Realizzare un tale Semantic Web richiede:

- lo sviluppo di linguaggi per esprimere meta informazioni comprensibili dalla macchina e lo sviluppo di tecnologie che usino tali linguaggi, rendendo il tutto disponibile sul Web;

- lo sviluppo di strumenti e nuove architetture che usino tali linguaggi al fine di fornire un supporto per la ricerca, l'accesso, la presentazione ed il mantenimento delle risorse di informazione;

- la realizzazione di applicazioni che forniscano un nuovo livello di servizio agli utenti umani del Semantic Web.

Lo scopo di questo capitolo è quello di analizzare i campi nel WWW che necessitano maggiormente del Semantic Web e le tecnologie attualmente disponibili per realizzarlo.
Il capitolo è organizzato come segue. Nella Sezione 1.1 verranno approfondite le motivazioni che hanno portato al Semantic Web, a partire dai problemi del WWW. Verranno mostrati, inoltre, i nuovi servizi che il Semantic Web può offrire. Infine, nella Sezione 1.2 verrà spiegato come tali servizi sono sviluppati.

1.1 Obiettivi del Semantic Web

Il Web ha portato nuove possibilità nell'accesso alle informazioni e nel commercio elettronico ed è stata la sua semplicità ad agevolarne l'enorme crescita, semplicità che ha però seriamente ostacolato lo sviluppo di applicazioni più avanzate. In questa sezione verranno discussi i colli di bottiglia rispetto alla gestione della conoscenza (Sezione 1.1.1) e del commercio elettronico, sia in ambito di applicazioni B2C (Sezione 1.1.2) sia B2B (Sezione 1.1.3) (vedere [17] per maggiori dettagli). Il termine B2C (Business to Consumer) è usato per denotare aspetti di commercio elettronico che includono relazioni fra aziende e consumatori. Consiste nello scambio di prodotti, servizi o informazioni tra azienda e consumatore; corrisponde principalmente alla parte di commercio elettronico che riguarda la vendita al dettaglio su Internet. È spesso in contrasto con il termine B2B (Business to Business), usato per denotare aspetti di commercio elettronico che coinvolgono solo aziende o all'associazione fra organizzazioni. Consiste nello scambio di prodotti, servizi, o informazioni tra aziende piuttosto che fra azienda e consumatore.

1.1.1 Gestione della conoscenza

Con Knowledge Management (KM) si intende il processo di acquisizione, mantenimento e accesso alla conoscenza di un'organizzazione. Tale processo mira a sfruttare i "beni intellettuali" delle organizzazioni per migliorarne la produttività ed incrementarne la competitività. Attualmente le organizzazioni hanno la loro conoscenza distribuita su diverse sorgenti e l'esistenza di tali sorgenti, rese disponibili online, ha spinto gli sviluppatori alla creazione di sistemi per la loro gestione. Tali sistemi hanno però gravi debolezze per quanto concerne:

- la ricerca delle informazioni,
- l'estrazione delle informazioni,
- il mantenimento delle informazioni,
- la gestione automatica delle informazioni (ossia, siti Web che abilitano una riconfigurazione dinamica delle informazioni in accordo con i profili utente).

Gli approcci per la KM sono basati sul processo di integrazione della conoscenza, in cui forme eterogenee di sorgenti di conoscenza devono essere integrate attraverso un'interfaccia di ricerca comune che consenta di trovare la soluzione corretta ad un dato problema. La condizione necessaria, affinché

tale integrazione possa avvenire, consiste nella creazione di una descrizione unificata del contenuto delle sorgenti di conoscenza, ovvero di un unico formato e di un vocabolario comune.
La tecnologia del Semantic Web deve consentire la definizione strutturale e sintattica di documenti creando così nuove opportunità per la ricerca intelligente al posto della ricerca basata sulle parole chiave, la risposta certa alle interrogazioni al posto dell'applicazione di tecniche approssimate tipiche dell'Information Retrieval e la definizone di viste personalizzate sui documenti.

1.1.2 Commercio via Web

Il commercio elettronico (B2C) è un'area che sta riscontrando molto successo perché riduce i costi ed estende gli usuali canali di distribuzione creando un nuovo modello di commercio. Nello sviluppo del commercio elettronico sono stati creati agenti che visitano i diversi negozi online, estraggono le informazioni sui prodotti e le presentano al cliente come una vista generale del negozio. Le funzionalità degli agenti sono fornite attraverso piccoli software (*wrappers*) scritti e personalizzati per ogni negozio, che basano la ricerca di un prodotto sul matching delle parole chiave assumendo una certa regolarità nel formato di presentazione dei siti Web per negozi. Questa tecnologia ha però grosse limitazioni.

- Occorre definire un *wrapper* per ogni negozio online. Questa è un'attività che richiede tempo per lo sviluppo degli agenti e le modifiche nei layout (interfacce) dei negozi. Inoltre, occorre un alto livello di manutenzione per mantenere aggiornati gli agenti.

- Le informazioni sui prodotti, estratte tramite gli agenti, sono spesso limitate, inclini ad errore ed incomplete.

Questi problemi sono dovuti al fatto che nei siti Web, la maggior parte delle informazioni sui prodotti è espressa in linguaggio naturale ed il riconoscimento automatico del testo è un'area di ricerca con molti problemi ancora irrisolti. Per risolvere questi problemi occorre una semantica delle informazioni chiara, che possa essere compresa e processata da una macchina. La situazione cambierà drasticamente quando sarà disponibile un formalismo di rappresentazione standard per la struttura e la semantica dei dati. Potranno essere costruiti agenti software in grado di capire le informazioni sui prodotti fornite dai siti Web. Il basso livello di programmazione usato per scrivere gli attuali software per l'estrazione del testo sarà sostituito dal mapping semantico che tradurrà i differenti formati usati per rappresentare i prodotti e sarà usato per il recupero automatico delle informazioni richieste.

1.1.3 Commercio elettronico

Il commercio elettronico nel campo B2B non è un fenomeno nuovo e gli standard per descrivere le transazioni sono ancora troppo procedurali ed ingombranti. Questo rende la programmazione delle transazioni costosa, incline ad errori e con alti costi di mantenimento. Usare l'infrastruttura di Internet ha migliorato in modo significativo questa situazione. Tuttavia lo scambio di dati è attualmente ostacolato dal fatto che HTML non fornisce i mezzi necessari alla rappresentazione dei dati arricchiti con la loro sintassi e semantica. XML è stato creato per risolvere questo problema nell'attuale tecnologia di Internet e fornisce una sintassi standard per definire la struttura e la semantica dei dati, consentendo, quindi, di rappresentare la semantica delle informazioni come parte della definizione della loro struttura. Tuttavia, XML non supporta strutture dati e terminologie standard per descrivere i processi del business e per lo scambio dei prodotti.
La nuova tecnologia del Semantic Web giocherà un ruolo fondamentale nell'abilitare XML al commercio elettronico purché:

- si definiscano linguaggi ricchi di primitive e con un modello dei dati ben definito, per fornire un supporto nella definizione, mappatura e scambio dei dati "prodotto";
- si sviluppino ontologie standard per coprire diverse aree di commercio;
- si definiscano servizi di traduzione efficienti nelle aree in cui non esistono ontologie standard o un cliente desidera adoperare una propria terminologia.

Questo supporto permetterà di estendere in modo significativo l'automatizzazione dello scambio di dati e creerà un nuovo modello di commercio.

1.2 Realizzazione del Semantic Web

Nella Sezione 1.1 sono stati descritti i nuovi servizi che potranno essere forniti dallo sviluppo del Semantic Web. In questa sezione tratteremo, invece, le tecnologie che occorrerà applicare per realizzare tali servizi. Nella Sezione 1.2.1 descriveremo i linguaggi che consentono di aggiungere semantica al Web. Nella Sezione 1.2.2 descriveremo gli strumenti per poter aggiungere semantica al Web.

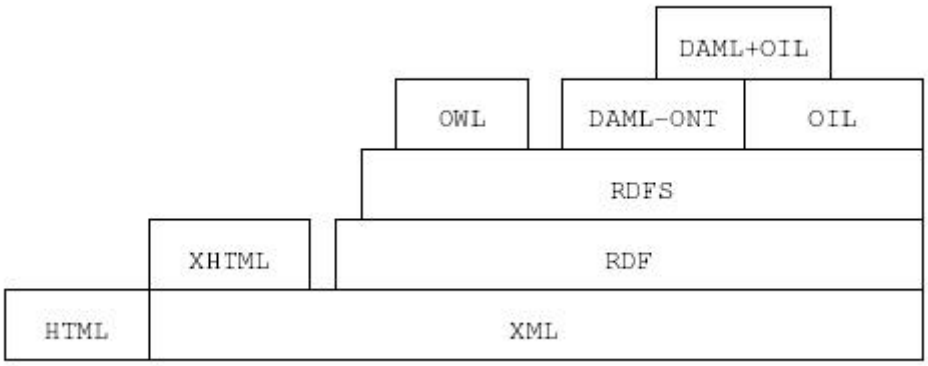


Figura 1.1: *Modello a "strati" dei linguaggi per il WWW*

1.2.1 Linguaggi

Nella definizione di linguaggi per il Semantic Web occorre garantire le seguenti proprietà:

1. un vocabolario standard che si riferisca alla semantica del mondo reale, consentendo ad agenti automatici ed umani di condividere informazioni e conoscenza,

2. una sintassi e una semantica formale per consentire di processare il loro contenuto in modo automatico.

Molto lavoro è stato svolto in questi anni per garantire tali proprietà. Nel seguito della sezione verrà presentato il lavoro svolto nell'attribuire semantica ai vocaboli e concetti (attraverso le ontologie) e quello svolto nell'ambito dei linguaggi formali (usati per sviluppare ontologie).

Ontologie

Le ontologie sono state sviluppate nel campo dell'intelligenza artificiale per facilitare la condivisione ed il riutilizzo della conoscenza e trovano applicazione in campi quali l'ingegnerizzazione della conoscenza, l'elaborazione del linguaggio naturale, la rappresentazione della conoscenza, l'integrazione intelligente dell'informazione, i sistemi di informazione cooperativi, il recupero dell'informazione, il commercio elettronico e la gestione della conoscenza.
La ragione per cui le ontologie sono diventate così popolari ha a che fare, in larga misura, con quello che promettono ossia un'intesa comune e condivisa

su alcuni domini specifici che può essere comunicata tra persone e applicazioni. Poiché lo scopo delle ontologie è quello di ottenere un accordo consensuale sulla conoscenza riguardo un certo dominio, il loro sviluppo è spesso un processo di cooperazione fra diverse persone.
Sono state date molte definizioni di ontologia, ma quella che meglio caratterizza l'essenza di un ontologia è quella fornita da Tom Gruber:

> *Un'ontologia è una specifica formale ed esplicita di una concettualizzazione condivisa [24].*

Concettualizzazione si riferisce ad un modello astratto di alcuni fenomeni nel mondo, che identifica i concetti rilevanti di tale fenomeno. *Esplicita* significa che i tipi di concetti usati ed i vincoli sul loro uso sono esplicitamente definiti. *Formale* si riferisce al fatto che dovrebbe essere comprensibile alla macchina (ci sono diversi livelli di formalismo). *Condivisa* riflette la nozione che un'ontologia cattura un'idea di conoscenza consensuale, cioè non ristretta a qualche individuo ma accettata da un gruppo.

Linguaggi formali per lo sviluppo di ontologie

In origine, il Web è cresciuto principalmente intorno al linguaggio HTML, perché è lo standard per strutturare documenti in modo che possano essere tradotti in modo canonico dai browser. Come già evidenziato in precedenza, HTML ha favorito la crescita smisurata del WWW ma la sua semplicità ha impedito la creazione di applicazioni Web più avanzate. Questa è stata la ragione per cui è stato definito il linguaggio XML [67], che consente di definire estensioni specifiche del dominio e delle applicazioni. XML è diventato la base per tutti gli standard che consentono di attribuire una semantica ai dati. In Figura 1.1 viene presentato il legame fra i linguaggi. Si noti come anche HTML sia stato ridefinito a partire dalla semantica di XML (XHTML) e non sia considerato quando si parla di semantica dei dati. Conseguentemente, è apparso logico definire il Semantic Web come un'applicazione XML.
Il primo passo in questa direzione è stato fatto con il Resource Description Framework (RDF) [33] che definisce una convenzione sintattica ed un modello dei dati semplice per rappresentare una semantica dei dati processabile dalla macchina. RDF è uno standard per metadati per il Web, sviluppato dal World Wide Web Consortium (W3C), utile per descrivere qualunque risorsa Web e, come tale, fornisce l'interoperabilità fra applicazioni che si scambiano informazioni processabili dalla macchina sul Web. RDF è un'applicazione XML e aggiunge un semplice modello dei dati, fornendo gli oggetti, le proprietà e i valori di proprietà applicabili a certi oggetti.
Un secondo passo è stato fatto con l'RDF Schema (RDFS) [7] che definisce

le primitive di base per modellare ontologie ed è basato sull'RDF. Consente la definizione di classi (cioè concetti), gerarchie di ereditarietà per le classi, proprietà, domini e intervalli di valori per proprietà.
Un linguaggio di modellazione che usa l'RDFS come punto di partenza e lo estende con un linguaggio completo per modellare ontologie, è l'Ontology Inference Layer (OIL) [19]. OIL unifica tre importanti aspetti forniti da diverse comunità: epistemologicamente è ricco di primitive di modellazione come la comunità dei Frame [64], è ricco di semantica formale e supporto efficiente per il ragionamento come fornisce la Logica Descrittiva [41, 3] ed è ricco di notazioni sintattiche standard per lo scambio come la comunità Web.
Un altro linguaggio di modellazione per ontologie basato sul Web è DAML-ONT. Il DARPA Agent Markup Language (DAML) [13] è una delle principali iniziative mirate a portare le ontologie nel Web. Tale linguaggio eredita molti degli aspetti di OIL. Entrambe le iniziative cooperano nel comitato Joint EU/US ad hoc Agent Markup Language per ottenere un linguaggio unificato chiamato DAML+OIL.
DAML+OIL è stato usato come punto di partenza dal W3C Web Ontology Working Group per la definizione di OWL [47], il cui scopo è quello di essere il linguaggio standard più largamente usato per il Semantic Web. Il Web Ontology Language (OWL) comprende una sintassi standard per lo scambio di ontologie e specifica la semantica del linguaggio, cioè come le strutture sintattiche devono essere interpretate.

1.2.2 Strumenti

Un lavoro efficiente ed effettivo con il Semantic Web deve essere supportato da strumenti avanzati che consentano l'applicazione di tutta la potenza di questa tecnologia. In particolare vengono richieste le seguenti caratteristiche:

- Linguaggi formali per esprimere e rappresentare ontologie (abbiamo già discusso di questi nella Sezione 1.2.1).

- Editor e costruttori semiautomatici per la definizione di nuove ontologie che supportino la definizione di gerarchie di concetti, la definizione di attributi per i concetti e la definizione di assiomi e vincoli. Tali editor devono supportare lo sviluppo dell'ontologia ed il successivo mantenimento e, per essere utili in questo contesto, devono essere forniti di un'interfaccia grafica ed essere conformi agli standard esistenti per lo sviluppo software basato sul Web. Uno degli editor che soddisfa tutti questi criteri è Protégé [23] sviluppato dall'università di Stanford.

- Ambienti per ontologie che aiutino a creare nuove ontologie attraverso

il riutilizzo di quelle esistenti. Le operazioni necessarie per combinare ontologie sono l'inclusione di ontologie, la restrizione di ontologie ed il raffinamento polimorfico di ontologie. Un tale ambiente per ontologie è Chimaera, sviluppato all'università di Stanford, che fornisce supporto per la fusione di ontologie multiple e per la diagnosi (e l'evoluzione) di ontologie [38].

- Istanze e schemi di inferenza che consentano un servizio di *query answering* (il calcolo delle risposte certe ad una *query*) avanzato ed un supporto per la creazione di ontologie che aiuti il processo di mapping fra termionlogie differenti ("servizi di ragionamento"). I servizi di ragionamento possono essere divisi in: ragionamento su istanze di ontologie utilizzato per rispondere a interrogazioni riguardanti la conoscenza implicita ed esplicita specificata attraverso l'ontologia (ad esempio Ontobroker [18]), e ragionamento sui concetti di un'ontologia utilizzato per derivare automaticamente la giusta posizione di un nuovo concetto nella gerarchia. Un sistema con tali capacità, FaCT (Fast Classification of Terminologies [27]) sviluppato all'università di Manchester, può essere usato per derivare gerarchie di concetti automaticamente.

- Strumenti per annotazioni che consentano di collegare sorgenti di informazioni non strutturate o semistrutturate a meta-dati.

- Strumenti per l'accesso e la navigazione di informazioni che permettano una consultazione intelligente da parte dell'utilizzatore umano.

- Servizi di traduzione e integrazione tra differenti ontologie che permettano un interscambio multistandard di dati e la definizione di viste multiple.

Capitolo 2

Peer to Peer

Il termine P2P è stato introdotto per identificare la possibilità di collaborare e condividere risorse migliorando la scalabilità (evitando la dipendenza da punti centralizzati), eliminando la necessità di un'infrastruttura specifica e consentendo l'aggregazione di risorse. Esistono varie definizioni di P2P che differiscono a seconda del contesto applicativo cui si fa riferimento. Quella che meglio evidenzia la natura di tale modello è fornita da Clay Shirky della O'Reilly e associati [56]:

> *"il P2P è una classe di applicazioni che si avvantaggia delle risorse disponibili su Internet. Poiché accedere a tali risorse decentralizzate significa operare in un ambiente di connettività instabile e con indirizzi IP impredicibili, i nodi P2P devono operare fuori dal sistema DNS ed hanno una significativa o totale autonomia dai server centrali".*

Il P2P è un modo per implementare sistemi basati sulla nozione di aumento della decentralizzazione del sistema, delle applicazioni o degli algoritmi e per trarre vantaggio dai computer distribuiti in tutto il mondo sfruttandone la potenza di calcolo, la quantità di memoria e la connettività. Si basa sul principio che il mondo sarà connesso e largamente distribuito e che non sarà possibile né desiderabile degradare la sua infrastruttura ad una gestione centralizzata. Concettualmente, è un'alternativa ai modelli di computazione centralizzato e client-server, costituiti tipicamente da un singolo o un piccolo gruppo di server e molti client. Nella sua forma pura, il modello P2P non possiede il concetto di server; piuttosto tutti i partecipanti sono peer ("pari"), hanno, cioè, le stesse capacità. Assumendo che un peer sia definito "come ogni altro", un sistema P2P è un sistema in cui i peer sono nodi di una rete, ognuno dei quali esiste in modo indipendente dagli altri e collabora con gli altri per il conseguimento di un obiettivo comune. Il modello P2P

è piuttosto esteso e potrebbe essere valutato da diverse prospettive. In termini di sviluppo, piattaforme come JXTA [29] forniscono un'infrastruttura per supportare applicazioni P2P. Inoltre, gli sviluppatori stanno iniziando ad esplorare il beneficio di implementare diverse tecnologie come computazioni distribuite, applicazioni collaborative e software per la condivisione di contenuti che usano il modello P2P piuttosto che i modelli client-server. Nel corso del capitolo verranno descritte queste tecnologie con maggiore dettaglio.
Il P2P è un tema di ricerca che ha sollevato molte discussioni tra gli operatori del settore. È una tecnologia ancora in via di sviluppo che ha ricevuto una grande attenzione sia dalle industrie, tra cui il P2P Working Group [65] che vanta partners come Intel, HP, Sony e Sun, sia dall'accademia, dove esistono un gran numero di eventi dedicati come l'International Workshop on P2P Computing [46], Global and P2P Computing on Large Scale Distributed Systems [45] e l'International Conference on P2P Computing [20]. Dal punto di vista della distribuzione della computazione vi è molto disaccordo sia nella comunità accademica sia in quella industriale. Da una parte si ritiene che i sistemi P2P siano i sistemi distribuiti che si studiano da decenni, dall'altra si ritiene che tali sistemi introducano nuovi requisiti che li contraddistinguono dai sistemi distribuiti tradizionali. Lo scopo di questo capitolo è quello di capire il significato di P2P, offrire un'analisi accurata della computazione P2P e analizzare il suo potenziale. Saranno, inoltre, discussi sia i sistemi che la comunità scientifica considera P2P (ad esempio SETI@home [55]), sia quelli che presentano solo qualche aspetto P2P (ad esempio .NET [42]). Nel tentativo di definire la natura del P2P, quello che è nuovo e quello che non lo è, analizziamo cosa riguarda e quali sono gli aspetti che include.

- Il P2P riguarda:
  - l'evoluzione storica della computazione in generale e di Internet in particolare (ad esempio SETI@home e altri sistemi di computazione distribuita),
  - aspetti sociologici per la condivisione dei contenuti (ad esempio Napster [40] e altri sistemi per la condivisione di file e contenuti),
  - miglioramenti tecnologici al progresso delle reti e della comunicazione (ad esempio reti wireless, dispositivi mirati a consentire una migliore collaborazione e comunicazione),
  - architetture software P2P (ad esempio JXTA o .NET),
  - sviluppo di algoritmi P2P (ad esempio Gnutella [22], FreeNet [21]).
- I nuovi aspetti del P2P includono:

  - Requisiti tecnologici: quantità di computer impiegati, connettività ad-hoc, sicurezza;
  - Requisiti architetturali: scalabilità dei futuri sistemi world-wide, privacy e anonimia.

Il capitolo è organizzato come segue. Nella Sezione 2.1 verrà fornito un background sui sistemi P2P. Nella Sezione 2.2 sarà presentata una classificazione dei sistemi P2P dal punto di vista della computazione. La Sezione 2.3 classificherà i sistemi P2P dal punto di vista delle applicazioni possibili. La Sezione 2.4 descriverà le caratteristiche generali dei sistemi P2P. La Sezione 2.5 fornirà una panoramica degli algoritmi P2P per il recupero delle informazioni.

2.1 I sistemi P2P

Il termine Peer to Peer (P2P) si riferisce ad una classe di sistemi e applicazioni che impiegano risorse distribuite per eseguire una funzione critica in modo decentralizzato. In tali sistemi tutti i peer sono definiti allo stesso modo e sono autonomi ovvero dipendono gli uni dagli altri ma non sono controllati da nessuna autorità centrale. La scelta di un tale approccio è spesso influenzata da molti dei seguenti fattori.

- *Riduzione dei costi.* I sistemi centralizzati che servono molti clienti, tipicamente fanno sopportare la maggior parte dei costi al server. Quando i costi imputati al server diventano troppo alti, un'architettura P2P può aiutare a distribuirli su tutti i peer. Una buona parte del processo di condivisione dei costi è realizzata tramite l'uso e l'aggregazione di risorse inutilizzate. A causa della tendenza dei peer ad essere autonomi è però di fondamentale importanza che la suddivisione dei costi avvenga in modo equo.

- *Miglioramento della scalabilità e dell'affidabilità.* Con la mancanza di un'autorità centrale, uno degli obiettivi principali dei sistemi P2P è quello di ottenere un miglioramento della scalabilità e dell'affidabilità del sistema. Questi requisiti possono essere ottenuti tramite le innovazioni algoritmiche nell'area della scoperta di risorse e della ricerca di informazioni che verrano analizzate in dettaglio nella Sezione 2.5.

- *Aggregazione di risorse ed interoperabilità.* Un approccio decentralizzato si presta in modo naturale all'aggregazione di risorse. In un sistema P2P ogni nodo possiede determinate risorse quali spazio di memoria e

potenza di calcolo. Le applicazioni, che possono usufruire largamente di tali risorse (come simulazioni di computazioni intensive o file system distribuiti), sono inclini in modo naturale all'uso di una struttura P2P per l'aggregazione al fine di risolvere problemi molto onerosi.

- *Aumento di autonomia.* In molti casi, gli utenti di un sistema distribuito sono poco disposti a contare su fornitori di servizi centralizzati. Preferiscono, piuttosto, che tutti i loro dati ed i loro lavori vengano svolti localmente. I sistemi P2P supportano questo livello di autonomia semplicemente perché permettono al peer locale di lavorare a nome del prorio utente.

- *Privacy.* Collegata al requisito di autonomia è la nozione di privacy. In generale, un utente può desiderare che nessuno sappia di un suo coinvolgimento nel sistema. Con un server centrale è difficile assicurare l'anonimato perché, tipicamente, un server vorrà identificare i suoi client o, per lo meno, il loro indirizzo Internet. Con l'impiego di una struttura P2P in cui le attività vengono eseguite localmente, gli utenti non sono tenuti a fornire informazioni su loro stessi.

- *Dinamismo.* I sistemi P2P assumono che l'ambiente di computazione sia altamente dinamico permettendo quindi ai nodi di connettersi e disconnettersi continuamente. In applicazioni per la comunicazione, le cosidette "buddy list" (liste di amici) sono usate per informare gli utenti su quando le persone con cui desiderano comunicare diventano disponibili. Se un tale supporto non fosse fornito, gli utenti sarebbero costretti a inviare periodicamente messaggi per poter comunicare con chi desiderano.

- *Supporto per comunicazioni e collaborazioni ad-hoc.* Collegata al dinamismo c'è la nozione di supporto di ambienti ad-hoc. Con ad-hoc si intendono ambienti in cui i membri vanno e vengono basandosi sulla loro locazione fisica o sui loro attuali interessi. Un sistema P2P è adatto a questo tipo di applicazioni perché prevede, in modo naturale, la possibilità che un partecipante possa cambiare gruppo. Non prevede, infatti, un'infrastruttura prestabilita ma, tipicamente, ne costruisce una adatta a seconda delle necessità.

2.2 Classificazione dei sistemi P2P

Una possibile classificazione dei sistemi P2P è mostrata in Figura 2.1. Tutti i sistemi di elaborazione possono essere classificati in *centralizzati* e *distri-*

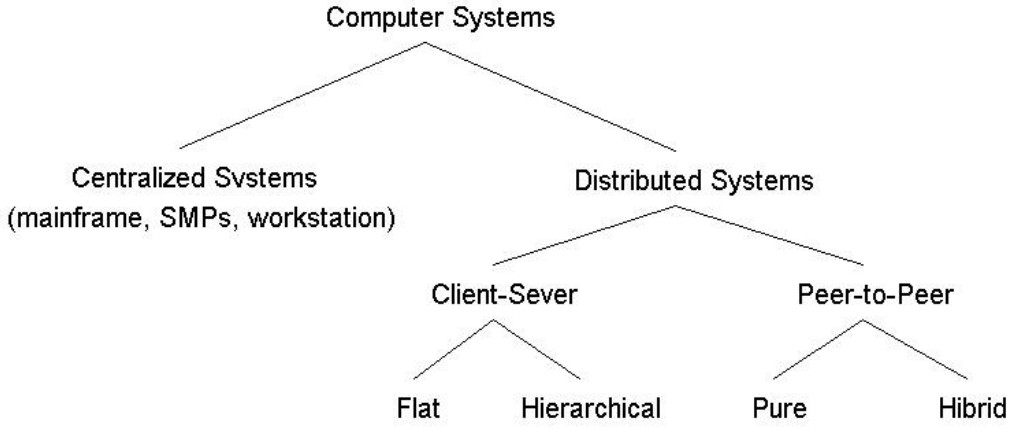


Figura 2.1: *Tassonomia dei sistemi*

buiti. I sistemi distribuiti possono ulteriormente essere classificati in *modelli client-server* e *modelli P2P*. Un modello client-server può essere *flat (piatto)*, nel qual caso i client possono comunicare solo con un singolo server, oppure *gerarchico*, per migliorare la scalabilità. Un sistema P2P, invece, può essere ulteriormente classificato in *puro* e *ibrido*. La differenza fra queste due tipologie di sistemi verrà discussa in questa sezione. Per meglio comprendere le differenze fra le tipologie di sistemi mostrate in Figura 2.1, riassumiamo in Tabella 2.1 le terminologie usate.

Abbiamo già specificato che i modelli P2P possono essere *puri* oppure *ibridi* (Figura 2.1). Vediamo nel dettaglio le differenze tra queste due tipologie.

- *Puri.* In un modello puro non esiste un server centralizzato e tutti i peer ricoprono lo stesso ruolo nel sistema. La ricerca delle informazioni sulla rete avviene inviando domande ai peer vicini i quali, se non sono in grado di rispondere, inoltrano tali domande ai loro vicini e così via. Esempi di tale modello sono Gnutella, che verrà analizzato in dettaglio nella Sezione 2.3.2, e Freenet.

- *Ibridi.* In un modello ibrido un server centralizzato contiene metadati riguardanti l'identità dei peer e le informazioni che memorizzano. Il server si comporta come un "broker" e fa da intermediario in una comunicazione fra due peer. All'inizio la comunicazione avviene tra il server ed il richiedente per ottenere la locazione o l'identità di un peer che contiene l'informazione ricercata. Da questo momento in poi, la comunicazione procede direttamente fra il peer richiedente ed il peer

Termine	**Descrizione**
Sistemi Centralizzati	Rappresentano soluzioni ad unità singola che includono macchine mono e multi processore
Sistemi Distribuiti	Sono sistemi in cui i componenti sono computer posti in una rete che coordinano le loro azioni attraverso lo scambio di messaggi
Client	Entità che pone richieste ma non è in grado di soddisfare una richiesta
Server	Entità che può adempiere a richieste di altre entità (in genere non pone richieste)
Modello Client-Server	Rappresenta l'interazione di entità con ruoli di client e server
Peer	Entità con capacità simili alle altre entità che compongono un sistema
Modello Peer-to-Peer	Consente ai peer di condividere risorse con una possibile ma limitata interazione con un server centrale

Tabella 2.1: *Terminologia*

che contiene l'informazione ricercata. Un esempio di tale modello è Napster che verrà analizzato in dettaglio nella Sezione 2.3.2.

Esistono anche soluzioni intermedie ai due modelli sopra descritti come quella dei *coordinatori* (*SuperPeers*). Un esempio di tale modello è KaZaA [31] che verrà analizzato in dettaglio nella Sezione 2.3.2. In questo modello i coordinatori hanno alcune delle informazioni che gli altri peer non possiedono e vengono usati come punto di riferimento per i peer che non riescono a trovare altrove tali informazioni.

2.3 Aree di applicazione del modello P2P

In questa sezione descriveremo le quattro principali aree di applicazione del modello P2P e qualche esempio di sistema realmente esistente. Mentre la Sezione 2.3.1 tratta le categorie di applicazione in generale, la Sezione 2.3.2 presenta dei casi di studio appartenenti ad alcune delle categorie descritte.

2.3.1 Classificazione delle applicazioni

Elenchiamo di seguito le principali applicazioni emerse nell'area P2P.

Computazione distribuita

Il problema che deve essere risolto viene suddiviso in piccole parti indipendenti e distribuito fra un gran numero di PC connessi ad Internet. L'elaborazione di ogni parte (attraverso l'uso di *fork* e *join*) è eseguita da ogni PC individualmente ed i risultati vengono raccolti in un server centrale, responsabile della distribuzione dei lavori ai PC su Internet. Ognuno dei PC è equipaggiato con un software client che si avvantaggia dei periodi di inattività del PC per eseguire alcune computazioni richiestegli dal server. Terminata la computazione, il risultato viene restituito al server e un nuovo lavoro viene affidato al client.
Questa tipologia di sistemi richiede tipicamente che le applicazioni vengano eseguite attraverso un controllo centrale e, per questo motivo, alcuni ritengono che tali sistemi non siano P2P. Il problema risiede nel fatto che, per controllare le risorse offerte dai PC, è richiesto un server centrale, i PC non possono agire come server e non avviene alcuna comunicazione fra di essi. Tuttavia una parte significativa del calcolo è eseguita sui peer, rendendoli altamente autonomi. Per questo motivo è possibile far rientrare la computazione distribuita tra le applicazioni P2P.
Esempi di aree di applicazione per questa tipologia di sistemi sono la finanza e la biotecnologia. Istituzioni finanziarie, come banche e compagnie di credito, eseguono simulazioni complesse per previsioni di mercato in tempo reale. Nel settore della biotecnologia, invece, è richiesta un grande potenza di calcolo, ad esempio, per la ricerca di sequenze complesse del genoma umano. Una delle applicazioni più famose della computazione distribuita è SETI@home che verrà analizzato in dettaglio nella prossima sezione.

Condivisione di documenti

La memorizzazione e lo scambio di contenuti è una delle aree dove la tecnologia P2P ha riscontrato maggior successo. Napster e Gnutella sono stati usati dagli utenti di Internet per aggirare le limitazioni di banda che, con l'utilizzo dei meccanismi classici, rendono inaccettabile il trasferimento di file molto grandi (come quelli con contenuti multimediali). Analizzeremo dettagliatamente nella prossima sezione il funzionamento di questi due sistemi. I sistemi di computazione distribuita hanno sfruttato le infrastrutture esistenti per offrire i seguenti servizi.

- *Scambio di file.* I file vengono memorizzati nella comunità P2P e vengono resi disponibili a tutti i peer. Per poter recuperare un determinato file, un peer deve conoscerne il riferimento. Sistemi come FreeNet, Gnutella e KaZaA rientrano in questa categoria.

- *Grande disponibilità di memoria.* Le politiche di duplicazione e ridondanza offrono memoria virtuale dove poter duplicare file ritenuti critici. Esempi di un tale sistema sono OceanStore [32] e Chord [12, 60].

- *Anonimia.* Alcuni sistemi assicurano con certezza matematica che i documenti pubblicati preservino l'anonimia su autori ed editori, consentendo comunque l'accesso a tali documenti.

- *Maneggiabilità.* I sistemi P2P consentono un recupero semplice e veloce dei dati distribuendoli su cache poste in nodi alla periferia della rete.

Sino ad ora, comunque, i sistemi P2P si sono focalizzati sullo scambio e la condivisione di piccoli oggetti come file. Tuttavia ci si aspetta che, in futuro, vengano estesi ad ogni tipo di contenuto come audio, video, software e documenti.

Applicazioni collaborative

Lo scopo di tali applicazioni è quello di consentire una collaborazione fra gli utenti, in tempo reale e senza contare su un server centralizzato per collezionare e catalogare le informazioni. La natura inerentemente ad-hoc della tecnologia P2P si presta molto bene alla realizzazione di applicazioni collaborative a livello utente. Queste applicazioni variano dall'instant messaging e le chat, ai giochi online, alle applicazioni condivise per il business. Le applicazioni collaborative sono, generalmente, basate su eventi. I peer formano un gruppo ed iniziano una data attività. Il gruppo può includere solo due peer che collaborano direttamente, o può essere più numeroso. Quando un peer cambia il suo stato viene generato un evento ed inviato al resto del gruppo. Sfortunatamente, devono ancora essere risolti e gestiti un gran numero di problemi prima che un'implementazione del P2P collaborativo possa diventare possibile.

- *Localizzazione dei peer.* Alcuni sistemi si basano sull'uso di una directory centralizzata contenente una lista di tutti i peer che sono online. I peer che vogliono formare un nuovo gruppo consultano tale directory e selezionano i peer che vogliono includere nel gruppo.

- *Tolleranza ai guasti.* Nelle applicazioni distribuite i messaggi devono essere consegnati in modo affidabile per garantire che tutti i peer condividano la stessa vista dell'informazione. La soluzione nelle applicazioni P2P è quella di mettere in coda i messaggi inviati ma non ancora consegnati; non appena il peer a cui erano rivolti torna online, avviene la consegna.

- *Esecuzione in tempo reale.* È forse uno dei requisiti fondamentali per le applicazioni collaborative. In un ambiente collaborativo gli utenti si trovano al livello più esterno e, come tale, ogni ritardo viene immediatamente percepito. Sfortunatamente, in questo caso, il collo di bottiglia non è dovuto al P2P ma alla rete sottostante.

Piattaforme

I sistemi operativi costringono le applicazioni a essere specifiche per l'ambiente run-time nel quale si trovano ad operare. L'implicazione più importante risulta essere la necessità di riscrittura del codice nel caso si debba utilizzare la stessa applicazione su diversi sistemi operativi. Per rendere l'applicazione indipendente dal sistema operativo si introduce un livello di astrazione, chiamato middleware, che si frappone fra le due entità e si fa carico di confinare in sé tutte le cause che implicherebbero la riscrittura dell'applicazione nel caso di cambio di ambiente. Soluzioni middleware come la Java Virtual Machine o i browser ed i server Web, sono gli ambienti di interesse principali sia per gli utenti sia per gli sviluppatori delle applicazioni. È probabile che i sistemi futuri dipenderanno fortemente da alcuni altri tipi di piattaforme che saranno il comun denominatore per utenti e servizi connessi al Web o in una rete ad-hoc. Le piattaforme, più di ogni altro sistema P2P, forniscono il supporto per le principali componenti P2P: naming (processo di assegnazione di un nome ad un elemento o ad un gruppo di elementi), ricerca, comunicazione, sicurezza e aggregazione delle risorse. Esistono diversi candidati che competono per il ruolo di future piattaforme P2P; due delle principali sono .NET e JXTA che verrà analizzato nella prossima sezione.

2.3.2 Prototipi

In questa sezione verranno presentati alcuni dei principali sistemi P2P esistenti appartenenti ad alcune categorie descritte nella Sezione 2.3.1. La scelta di quali sistemi presentare è dovuta alla loro diffusione ed al loro utilizzo effettivo. Presentiamo:

- SETI@home come rappresentante della categoria computazione distribuita;
- Napster, Gnutella e KaZaA come rappresentanti della la categoria condivisione di documenti. Un altro candidato ad essere presentato in tale categoria è Chord [12, 60] ma poiché presenta caratteristiche comuni ai sistemi appena citati si è scelto di non descriverlo;
- JXTA come rappresentante della categoria piattaforme.

2.3.2.1 SETI@home

SETI (Search for Extraterrestrial Intelligence) è una collezione di progetti di ricerca il cui scopo è quello di scoprire civiltà aliene. Uno di questi progetti, SETI@home [55], usa le emissioni radio ricevute dallo spazio attraverso il telescopio Arecibo e le analizza utilizzando la potenza di calcolo di milioni di computer inutilizzati collegati ad Internet. SETI@home è un progetto di ricerca scientifico mirato alla costruzione di un enorme computer virtuale basato sull'aggregazione della potenza di calcolo offerta da PC connessi ad Internet durante i loro periodi di inattività. Il progetto usa principalmente due componenti: un database server altamente scalabile e solido (più di tre milioni di utenti sono registrati) e i client. Il software lato client è disponibile come un modulo screen saver non vincolato ad alcuna tecnologia e, pertanto, utilizzabile su qualunque piattaforma. Uno dei punti forti di questo sistema è la tolleranza ai guasti. Poiché una elaborazione può richiedere molte ore per essere completata, è necessario assicurare il recovery (ripristino) sia quando un utente si collega alla macchina (fermando quindi il client SETI@home) sia quando questa viene spenta. L'elasticità del client si basa sul meccanismo di *checkpointing*, che salva l'insieme dei dati su hard disk ogni dieci minuti. Questo significa che, ogni volta in cui la computazione viene interrotta (dagli utenti o dal sistema), verrà recuperato l'ultimo insieme di dati salvato e l'elaborazione ripartirà da lì. Il valore maggiore del progetto SETI@home, comunque, a parte i risultati scientifici, è stata la prova che la tecnologia può essere applicata a situazioni reali.

2.3.2.2 Napster

Napster [40] è la prima applicazione nata nell'area P2P ed è originariamente stato sviluppato per superare il problema di copiare e condividere file musicali in Internet. Questo sistema utilizza una directory centralizzata per mantenere una lista di file musicali che possono essere aggiornati e rimossi dagli utenti che si connettono al sistema. Per ricercare un determinato file,

gli utenti inviano una richiesta basata sulle keyword “titolo”, “artista”, etc. Sebbene il meccanismo di ricerca sia centralizzato, il meccanismo di condivisione dei file è decentralizzato dal momento che il trasferimento dei file avviene direttamente fra i peer. Il modello a directory centralizzata limita inevitabilmente la scalabilità del sistema. Ad esempio, la disponibilità di banda di un utente può essere tremendamente ridotta dagli utenti che scaricano canzoni dal suo computer. D’altra parte, però, la centralizzazione semplifica il problema di ottenere un namespace e permette di realizzare in modo semplice meccanismi di sicurezza.

2.3.2.3 Gnutella

Gnutella [22] è un protocollo di comunicazione decentralizzato il cui scopo è quello di fornire una soluzione totalmente distribuita per la ricerca e la condivisione di file tra gli utenti. Le applicazioni che implementano tale protocollo consentono agli utenti di eseguire un software per condividere, cercare e scaricare file da altri utenti connessi ad Internet. Inoltre, Gnutella fornisce l’anonimia mascherando l’identità del peer che ha generato la richiesta e fornisce un meccanismo attraverso cui è possibile formare una rete ad hoc senza un controllo centrale. Poiché nella rete Gnutella non esiste alcun server centrale, per collegarsi al sistema un utente inizialmente si connette a diversi peer conosciuti sempre disponibili (sebbene non forniscano la condivisione di file) che restituiscono informazioni quali l’indirizzo IP e la porta di altri peer Gnutella. Una volta collegati alla rete, i peer interagiscono gli uni con gli altri scambiandosi messaggi. I messaggi forniti sono:

- *Ping Message*: essenzialmente un messaggio diretto ad un peer per verificare se è connesso;
- *Pong Message*: messaggio di risposta ad un Ping Message contenente informazioni sul peer come il suo IP, la sua porta, il numero dei file che condivide e la dimensione totale dei suoi file;
- *Query Message*: messaggio di richiesta (query), per la ricerca di un dato file, inoltrato all’intera rete. Viene identificato in modo univoco ma la sua sorgente resta sconosciuta;
- *Query Response Message*: è la risposta al Query Message ed include informazioni necessarie per scaricare il file (IP, porta ed altre informazioni sulla locazione). Le risposte contengono un ID univoco associato al peer richiedente. Questi messaggi vengono propagati indietro lungo lo stesso percorso fatto originariamente dal Query Message;

- *Get/Push Message*: il messaggio Get è semplicemente una richiesta per un file restituito come risposta ad una richiesta (query). Il peer richiedente si connette al peer fornitore direttamente e richiede il file. Spesso sui peer sono installati dei firewall che impediscono la risposta diretta ad una richiesta per un file da parte di un peer. Per tale ragione il protocollo Gnutella include anche il messaggio Push. Tale messaggio richiede al peer fornitore di iniziare la connessione verso il peer richiedente per consentirgli di scaricare il file. Tuttavia se entrambi i peer possiedono un firewall la comunicazione diventa impossibile.

Il protocollo Gnutella fornisce diverse caratteristiche che prevengono la ritrasmissione infinita dei messaggi attraverso la rete. Una di queste prevede che i messaggi contengano un campo Time-to-Live (TTL). Ogni peer che riceve un messaggio decrementa il TTL ad esso associato prima di inoltrarlo. Quando un peer vede un messaggio con TTL uguale a zero, lo elimina e non lo ritrasmette.
La prima applicazione di questa tecnologia è stata la condivisione di file musicali. Mentre questo continua ad essere il principale motivo di applicazione per le compagnie che sviluppano un software compatibile con Gnutella, un certo numero di aziende ha utilizzando il protocollo Gnutella per software aziendali che includono applicazioni per la gestione di progetti.

2.3.2.4 KaZaA

KaZaA [31] è diventato un sistema popolare e largamente utilizzato con oltre 120 milioni di file scaricati e una media di 3 milioni di utenti online al giorno. L'interfaccia utente (UI) di tale sistema è semplice: l'utente inserisce una richiesta (query) in un campo testuale (textbox) e dai risultati seleziona i file che desidera scaricare. Se la condivisione è abilitata, i file che l'utente ha selezionato vengono automaticamente condivisi sulla rete. Purtroppo sono disponibili solo pochi dettagli riguardanti tale sistema. È un esempio di sistema P2P per la condivisione di file che usa coordinatori (SuperPeer) come centri di ricerca locale. Questi sono nodi a connessione veloce che vengono eletti dinamicamente e formano una rete scarsamente strutturata. I peer si connettono ad uno o più coordinatori per comunicare le informazioni che desiderano condividere e per eseguire ricerche sulla rete. Una volta che il contenuto desiderato viene localizzato, KaZaA usa il protocollo HTTP per trasferirlo direttamente tra il fornitore del contenuto ed il nodo che ha inoltrato la richiesta. Il sistema cerca i file automaticamente e li scarica attraverso la connessione disponibile più veloce, ripristina automaticamente i trasferimenti che non sono andati a buon fine e scarica, spesso, i file da

diverse sorgenti simultaneamente per accelerarne il download. KaZaA usa un sistema per scaricare informazioni in modo intelligente per migliorare la velocità e l'affidabilità, scaricando più frammenti dello stesso file in parallelo da diversi utenti fornitori. Quando i file sono importati, il sistema estrae automaticamente alcuni meta-dati dal loro contenuto, rendendo più veloce ed accurata la ricerca.

2.3.2.5 JXTA

Il progetto JXTA [29] è stato sviluppato da Sun e il suo scopo è quello di fornire una piattaforma per la collaborazione aperta ed innovativa, che supporti un gran numero di applicazioni distribuite. Lo scopo del gruppo JXTA è quello di creare un sistema P2P per il Web rendendo la ricerca su una rete distribuita altamente efficiente. È importante notare che il progetto JXTA propone un'infrastruttura interamente nuova che non ha nessuna relazione con quelle già esistenti per i sistemi P2P. Usando i protocolli forniti dalla piattaforma, i peer possono formare gruppi auto-gestiti e auto-configurati senza l'ausilio di nessuna infrastruttura centralizzata e indipendentemente dalla loro posizione nella rete. I gruppi di peer, cuore dell'infrastruttura JXTA, sono essenzialmente una partizione del mondo dei peer utile alla comunicazione, sicurezza, prestazione, locazione logica dei peer ed affidabilità. Un singolo partecipante può appartenere a più di un gruppo simultaneamente. Nello sviluppo di JXTA si è cercato di creare un insieme di protocolli che non richiedano eccessive risorse in termini di tempo, con poche assunzioni sulla rete di trasporto sottostante e sull'ambiente dei peer, e che siano in grado di supportare una grande quantità di applicazioni e servizi P2P in un ambiente di rete altamente inaffidabile e mutabile.

2.4 Caratteristiche dei sistemi P2P

In questa sezione vengono presentate le caratteristiche principali di un sistema P2P, ponendo l'attenzione su quelle maggiormente coinvolte nell'integrazione di un tale sistema con le tecnologie del Semantic Web, in accordo con [57]. Oltre ai requisiti di *Sicurezza*, *Trasparenza*, *Performance* (*Prestazioni*) e *Tolleranza ai guasti*, un sistema P2P deve presentare le seguenti caratteristiche particolarmente rilevanti per questa tesi.

- *Decentralizzazione.* Una delle caratteristiche più rilevanti della decentralizzazione è la possibilità dell'utente di avere pieno controllo dei dati e delle risorse. In un sistema completamente decentralizzato ogni peer

partecipa al sistema con uguali competenze. Questo aspetto rende particolarmente difficoltosa l'implementazione del modello P2P, dal momento che non esiste un server centralizzato con una vista globale di tutti i peer nella rete o dei file che essi forniscono. Questa è la ragione per cui molti file system P2P (ad esempio Napster) scelgono di adottare il modello ibrido dove c'è una directory centralizzata di file ed i nodi scaricano tali file direttamente dai peer. D'altra parte, in file system completamente decentralizzati (ad esempio Freenet e Gnutella) trovare la rete diventa difficile. In tali sistemi, infatti, i nuovi peer devono conoscere gli indirizzi degli altri peer o usare un "host list" contenente gli indirizzi IP dei peer conosciuti. I peer si collegano alla rete stabilendo una connessione con almeno uno dei peer attualmente nella rete. A questo punto, è possibile per tali peer scoprire gli altri e memorizzare il loro indirizzo IP localmente.

- *Scalabilità ed efficienza della rete.* Con scalabilità si intende l'abilità della rete e dei peer di mantenere le loro caratteristiche di funzionamento anche se cambiano in dimensione e/o volume.
  La scalabilità del sistema dipende da due fattori: la scalabilità della rete e dei peer. La scalabilità della rete dipende, in gran parte, dal protocollo di instradamento (*routing*) usato dai peer per inoltrare domande e ricevere risposte. La scalabilità dei peer dipende, invece, dall'equilibrio tra le risorse fornite dal e necessarie al peer.
  In sistemi centralizzati, come Napster e SETI@home, un server gioca il ruolo di intermediario e coordina i processi. In questo caso l'uso della rete è molto efficiente ma anche vulnerabile rispetto a problemi dovuti ad un server centralizzato.
  In sistemi decentralizzati come Gnutella e FreeNet, ogni peer deve mandare le sue richieste a molti peer in modo "cieco", provocando la ricerca del documento da parte dei peer che hanno ricevuto la richiesta. Una conseguenza di questo meccanismo è che il tempo di recupero di un documento può diventare infinito. Inoltre, può accadere che la ricerca fallisca anche quando l'oggetto cercato esiste, rendendo il comportamento del sistema non deterministico.
  Da ciò si deduce che, nel caso di sistemi decentralizzati, diventa necessario migliorare l'efficienza della rete riducendo il traffico ridondante e riducendo il carico dei nodi.
  I sistemi P2P più recenti (ad esempio Chord e OceanStore) dettano un mapping consistente fra un oggetto ed un nodo, quindi un oggetto può sempre essere ritovato sinché possono essere contattati i nodi che lo ospitano. I nodi in questi sistemi compongono una rete virtuale, posta

sopra quella fisica, in cui ogni nodo mantiene informazioni su un piccolo sottoinsieme dei nodi del sistema. Questo limita l'ammontare degli stati che devono essere mantenuti e, quindi, aumenta la scalabilità.

- *Interoperabilità.* Con il termine interoperabilità si intende la capacità di un sistema di garantire consistenza nell'interazione fra diverse piattaforme, applicazioni e linguaggi di programmazione. Alcuni dei requisiti per l'interoperabilità includono:
  - come i sistemi determinano con chi devono interoperare,
  - quale protocollo deve essere usato per la comunicazione,
  - come i sistemi devono scambiarsi richieste e dati ed eseguire compiti quali lo scambio e la ricerca di file,
  - come i sistemi determinano se sono compatibili con il protocollo usato ad alto livello, cioè come può un sistema dipendere da un altro per cercare correttamente una determinata porzione di informazione,
  - come i sistemi mantengono lo stesso livello di sicurezza ed affidabilità.
- *Auto-organizzazione.* I sistemi P2P possono variare in dimensione in modo non predicibile in termini di numero di peer e di numero di utenti e questo richiede una frequente riconfigurazione del sistema centralizzato. Questa significante variazione di scala conduce ad un'alta probabilità di guasti che richiede un'auto-manutenzione ed un'auto-riparazione da parte del sistema. È, inoltre, difficile per ogni configurazione predefinita rimanere intatta per lunghi periodi di tempo a causa della connessione e disconnessione intermittente dei peer. Per questo motivo è richiesto un alto livello di adattamento per poter gestire le modifiche dovute alla continua connessione e disconnessione dei peer dal sistema.

2.5 Recupero di informazioni in una rete P2P

Uno dei problemi più importanti in una rete P2P è consentire la ricerca efficiente dei contenuti degli altri peer. Assumiamo che ogni peer possieda una collezione di documenti (audio, video, testo) che desidera condividere nella rete: questo rappresenta la sua base di conoscenza. Un nodo che vuole cercare delle informazioni invia una interrogazione, contenente un insieme di parole chiave, ai suoi peer. Un peer che riceve tale messaggio calcola la somiglianza dell'interrogazione con la sua collezione di documenti, tipicamente

cercando quei documenti che contengono l'insieme delle parole chiave della richiesta (query). Se la valutazione ha successo, il peer genera un messaggio di risposta che contiene i puntatori ai documenti risultati affini. Il requisito principale di questo processo è l'esecuzione veloce ed efficiente, attraverso la diminuzione del numero di messaggi inviati per ogni richiesta (query) pur mantenendo un alto grado di qualità nei risultati della ricerca. In questa sezione verranno presentate alcune tecniche di ricerca in una rete P2P per il recupero dell'informazione basato su parole chiave (keyword). Tutte le tecniche presentate sono applicabili al modello P2P puro, eccezion fatta per l'approccio Centralizzato che si applica al modello P2P ibrido.

Breadth First Search (BFS) "naive"

È una tecnica largamente utilizzata nelle applicazioni di condivisione di file come Gnutella. Il protocollo di ricerca BFS in una rete P2P lavora nel modo seguente. Un nodo q genera un messaggio `Query` che viene propagato a tutti i suoi vicini (peer di cui è a conoscenza). Quando un peer p riceve una richiesta `Query`, inoltra la richiesta a tutti i peer, oltre che al mittente, e cerca nel suo archivio (repository) locale. Se qualche nodo r riceve la richiesta ed ha un match (risposta), genera un messaggio `QueryHit` per trasmettere il risultato. Il messaggio `QueryHit` contiene informazioni sul numero di documenti ritrovati e sulla connettività del peer che sta rispondendo. Quando il nodo q riceve un messaggio `QueryHit` da più di un peer, può decidere di scaricare il file dal peer con la migliore connessione di rete. I messaggi `QueryHit` vengono inviati lungo lo stesso percorso che ha trasportato il messaggio `Query`; quindi, nel messaggio di risposta trasmesso, non è necessario memorizzare alcuna informazione sul percorso. Questa tecnica sacrifica le prestazioni e l'utilizzo della rete in favore della sua semplicità in quanto ogni richiesta utilizza la rete e le risorse in modo eccessivo poiché viene propagata lungo tutti i cammini (compresi i nodi con alta latenza). Quindi un nodo con poca disponibilità di banda può facilmente diventare un collo di bottiglia. Una tecnica per evitare l'intasamento dell'intera rete con messaggi è associare ad ogni richiesta un "tempo di vita" (TTL), che determina il massimo numero di passaggi (hop) che una data richiesta può effettuare per essere inoltrata.

Random Breadth First Search (RBFS)

In questo approccio, un peer q inoltra un messaggio di richiesta (query) solo ad una frazione dei suoi vicini selezionata in modo casuale (random). Il vantaggio della RBFS è che non richiede una conoscenza globale: un nodo è in grado di prendere decisioni locali in modo veloce, poiché necessita

di selezionare solo una porzione dei peer da lui conosciuti. D'altra parte questo algoritmo è probabilistico e, pertanto, una larga parte della rete può risultare non rintracciabile poiché un nodo non è in grado di capire che un link particolare potrebbe codurre la richiesta ad un ampio segmento della rete non considerato. Tale tecnica migliora, comunque, in modo significativo l'approccio "naive" e viene proposta e valutata in [30].

Intelligent Search Mechanism (ISM)

L'ISM è un meccanismo nuovo per il recupero dell'informazione in una rete P2P. L'obiettivo di tale algoritmo (proposto in [30]) è quello di aiutare il peer che pone la richiesta a trovare le risposte più rilevanti velocemente ed in modo efficiente. La chiave per migliorare la velocità e l'efficienza del meccanismo di recupero dell'informazione risiede nel minimizzare i costi di comunicazione, cioè il numero di messaggi inviati fra i peer, e nel minimizzare il numero di peer a cui è inoltrata l'interrogazione per ogni richiesta di ricerca. Per poter ottenere questo, i peer stimano, per ogni interrogazione, quali sono i peer che più probabilmente sapranno rispondere e propagano la richiesta solo a tali peer. L'ISM consiste di due componenti:

1. Un *Profile Mechanism* che un peer q usa per costruire un profilo per ognuno dei suoi peer vicini. Tale meccanismo viene utilizzato per mantenere le interrogazioni più recenti, i corrispondenti `QueryHit` ed il numero di risultati. Sebbene logicamente si possa considerare ogni profilo come una lista di interrogazioni distinte, nell'implementazione viene utilizzata una singola tabella *Queries* di dimensione nell'ordine `T*d`, che contiene le ultime `T` interrogazioni per ogni `d` vicini.

2. Un *RelevanceRank (RR)*, meccanismo che usa i profili dei peer per selezionare i vicini che contengono le risposte più rilevanti per una data richiesta. La funzione RR è usata da un peer al fine di eseguire una classificazione (ranking) online dei suoi peer vicini per determinare a quale inoltrare una data interrogazione.

Uno dei problemi di questo approccio è quello che i messaggi di ricerca possono trovarsi chiusi in cicli e, conseguentemente, fallire l'esplorazione di altre parti della rete. Per risolvere questo problema è possibile scegliere un piccolo sottoinsieme di peer e aggiungerlo all'insieme dei peer rilevanti per ogni interrogazione. Con un'alta probabilità, il meccanismo esplorerà un'ampia parte di rete e apprenderà il contenuto dei peer aggiunti.

BFS e l'euristica del miglior risultato nel passato (>RES)

In [70] è presentata una tecnica dove ogni peer inoltra una interrogazione ad un sottoinsieme dei peer basandosi su qualche statistica. Gli autori confrontano diverse euristiche sull'instradamento (*routing*) delle interrogazioni ed affermano che l'euristica del miglior risultato nel passato (>RES) ha le prestazioni più soddisfacenti. Un'interrogazione è soddisfatta se sono restituiti Z (costante fissata a priori), o più risultati. In >RES, un peer q inoltra un messaggio di ricerca ai k peer che hanno restituito i risultati migliori nelle ultime m interrogazioni. La tecnica >RES è simile alla tecnica ISM ma usa un'informazione più semplice sui peer. Lo svantaggio principale di questa tecnica, rispetto alla tecnica ISM, è che essa non gestisce l'esplorazione di nodi contenenti informazioni relative all'interrogazione. L'approccio >RES può essere considerato più quantitativo che qualitativo.

Random Walkers

L'algoritmo Random Walkers è presentato in [36]. L'idea chiave è che ogni nodo inoltra un messaggio di interrogazione, chiamato *walker*, in modo casuale ad uno dei suoi peer. Per ridurre il tempo di ricezione dei risultati, l'idea del walker viene estesa a *k-walker* che dopo T passi ci si aspetta abbiano raggiunto approssimativamente lo stesso numero di nodi di *1-walker* dopo $k*T$ passi. Un'altra tecnica simile ai Random Walker è l'algoritmo *Adaptive Probabilistic Search (APS)* [66]. In *APS* ogni nodo impiega un indice locale, che cattura la probabilità relativa di ogni vicino di essere scelto come il prossimo passaggio (hop) per qualche futura richiesta. La principale differenza con il Random Walker risiede nel fatto che, nell'APS, un nodo utilizza il feedback dalla precedente ricerca per guidare in modo probabilistico i walker futuri, piuttosto che inoltrare i walker in modo casuale. È provato che l'APS migliori le prestazioni rispetto al Random Walker.

Gossiping

In PlanetP [11] viene proposto un approccio per la costruzione di un servizio di condivisione di documenti basato sull'uso di una directory globale, di cui ogni peer apparetente alla rete possiede una copia locale, che sfrutta la tecnica del *gossiping* per monitorare lo stato globale di una communità non strutturata di peer. Il framework descritto è basato su un *inverted index* globale parzialmente costruito da ogni nodo appartenente alla rete dei peer. Ogni nodo n_k costruisce un *Bloom filter* ([6]) b_k del proprio indice locale, che memorizza i temini estratti dai documenti che il nodo condivide nella rete, e

lo propaga al resto della rete attraverso la tecnica del *gossiping*. L'*inverted index* globale è una collezione di tutti i b_i. Dal momento che ogni nodo n_k mantiene una lista parzialmente consistente di coppie (n_i, b_i), può eseguire una ricerca locale per derivare quali nodi possiedono i termini cercati ed inoltrare quindi l'interrogazione solo a tali nodi. Una volta che l'interrogazione raggiunge un nodo n_j, tale nodo può o eseguire una *ricerca esaustiva* oppure usare una *ricerca selettiva* (come in [60]) usando il modello del *vector space rank*. Il cuore di PlanetP è costituito dall'algoritmo di *gossiping*. Questo sistema usa il *gossiping* per replicare una directory globale che include la lista dei peer, il loro indirizzo IP ed il *Bloom filter* di ognuno di essi. Gli eventi che provocano un cambiamento nella directory globale e che richiedono, conseguentemente, *gossiping* includono il collegamento di un nuovo membro alla comunità, il collegamento di un membro temporaneamente off-line e la modofica di un *Bloom filter*. Non viene monitorato lo scollegamento (temporaneo o definitivo) di un peer. Ogni peer scopre che un'altro peer è off-line quando prova a comunicare con tale peer e la comunicazione fallisce. In questo caso il peer che non risponde viene marcato off-line nella directory locale del peer che ha iniziato la comunicazione ma tale informazione non viene propagata via *gissiping*. Quando un peer x torna on-line la sua presenza potrà eventualmente essere comunicata via *gossiping* all'intera comunità; ogni peer che ha marcato x come off-line nella propria directory, cambia lo stato di x ad on-line. Lo svantaggio principale del sistema PlanetP è la mancanza di scalabilità.

Ricerca tramite l'uso di indici di instradamento locali

In [9] viene presentata una tecnica ibrida per la costruzione ed il mantenimento di indici locali relativi ai documenti condivisi dai peer nella rete. Vengono presentate tre diverse tecniche, *Compound Routing Indexes (CRI)*, *Hop-Count Routing Index (HRI)* e *Exponentially aggregated RI (ERI)*, e valutate su diverse topologie di rete. L'idea chiave è quella che un nodo conosce attraverso quali peer dovrà condurre la richiesta dei documenti che desidera ma non l'esatto path (cammino) di tali documenti. In *CRI* un nodo q mantiene per ogni peer vicino alcune statistiche che indicano quanti documenti sono raggiungibili attraverso tale peer. Poiché *CRI* non tiene in considerazione il numero di salti (hop) richiesti per raggiungere i documenti, in *HRI* un nodo q mantiene un *CRI* per k hop. *HRI* ha, però, un costo proibitivo in termini di utilizzo di memoria per grandi valori di k e, per tale motivo, è stato proposto *ERI*. *ERI* indirizza questo problema attraverso l'aggrgazione degli indici *HRI* sulla base di una determinata funzione di costo. Gli esperimenti rivelano che *ERI* e *HRI* offrono miglioramenti significativi oltre a

quello di non utilizzare alcun indice di indirizzamento e, allo stesso tempo, mantengono bassi i costi di aggiornamento. Poiché la tecnica del Local Index è essenzialmente una distribuzione di modifiche effettuate da ogni nodo che invia ai peer che conosce informazioni riguardanti i suoi documenti, essa è complementare all'approccio ISM descritto in precedenza.

Approccio centralizzato

I sistemi centralizzati mantengono un *inverted index* su tutti i documenti nella collezione degli host partecipanti. Tali sistemi possono essere sistemi commerciali di recupero delle informazioni, come motori di ricerca Web (ad esempio Google, Yahoo) oppure sistemi di indicizzazione P2P centralizzati [69]. Ad esempio, Napster usa un archivio centrale al quale ogni peer A collegato attacca un indice di tutti i suoi documenti condivisi. Un nodo richiedente B è in grado di cercare i documenti di A attraverso l'archivio centrale. Una volta che il documento cercato viene localizzato, B può comunicare direttamente con A usando, ad esempio, il protocollo HTTP.

Uso di identificatori di oggetti

Sistemi distribuiti di indicizzazione dei file, come Chord e OceanStore, consentono ai peer di eseguire una ricerca efficiente usando gli identificatori degli oggetti piuttosto che le parole chiave. Entrando più nello specifico, essi usano una struttura hash che permette ai peer di eseguire operazioni di osservazione sull' oggetto per ottenere l'indirizzo (ad esempio IP) del nodo contenente l'oggetto stesso. Questi sistemi sono stati sviluppati per ottimizzare il recupero dell'oggetto minimizzando il numero di messaggi richiesti per il suo ritrovamento. Lo svantaggio è che considerano solo il problema di ricerca delle chiavi e non catturano la rilevanza dei documenti memorizzati nel sistema. FreeNet è un altro sistema che memorizza e recupera informazioni in modo distribuito, che usa un meccanismo intelligente di *Depth First Search (DFS)* per localizzare le chiavi degli oggetti nel sistema.

Recupero di informazioni distribuite

Il problema principale nel recupero di informazioni distribuite è, assumendo di voler sottomettere un'interrogazione ad un sottoinsieme delle basi di dati disponibili, decidere quali più probabilmente contengono i documenti maggiormente rilevanti. Sono stati proposti un certo numero di algoritmi [34, 49] che assumono che la parte che sottopone l'interrogazione abbia una qualche conoscenza statistica (ad esempio la frequenza delle parole nei documenti)

sui contenuti di ogni base di dati e abbia, perciò, una visione globale del sistema. Inoltre, la maggior parte delle tecniche assume che l'ambiente sia sempre disponibile. È stato, inoltre, dimostrato che le prestazioni migliorano notevolmente se le collezioni di documenti sono concettualmente separate.

Capitolo 3

Peer to Peer e Semantic Web

Nell'attuale economia basata sulla conoscenza, la competitività delle aziende e la qualità del lavoro sono legate in modo diretto all'abilità di creare e condividere efficacemente la conoscenza all'interno e fra le organizzazioni. Le soluzioni P2P emergenti si adattano particolarmente bene ad incrementare la natura decentralizzata delle organizzazioni di oggi, siano esse una singola azienda o una rete dinamica di organizzazioni. Tali soluzioni rendono possibile a diversi partecipanti (organizzazioni, individui, dipartimenti in un'organizzazione) mantenere viste differenti del mondo mentre si scambiano informazioni. Allo stesso tempo, però, poiché adoperano una ricerca delle informazioni basata sulle parole chiave piuttosto che tecniche per la rappresentazione della conoscenza, le attuali soluzioni P2P sono ancora estremamente limitate. Esse non possono supportare facilmente l'introduzione di nuovi concetti, rendono difficile determinare se due termini sono equivalenti e, generalmente, possono supportare solo livelli molto limitati di automazione, tutti tipi di funzionalità fornite dalla tecnologia del Semantic Web.
La grande distribuzione dei sistemi P2P è anche la causa di un certo numero di nuovi problemi:

- la mancanza di un singolo schema coerente per organizzare le risorse di informazione nella rete dei peer ostacola la formulazione delle richieste di ricerca;
- la duplicazione di informazioni nella rete dei peer provoca la restituzione di risposte duplicate per un singola interrogazione;
- la risposta ad una interrogazione spesso richiede l'integrazione di informazioni residenti su peer differenti, indipendenti e non coordinati;
- l'instradamento delle iterrogazioni e la topologia della rete (quali peer

connettere a e a quali peer inviare/inoltrare le interrogazioni) sono due dei problemi più significativi.

Nel progetto SWAP [50, 15] è stato indagato come l'uso di descrizioni semantiche di sorgenti di dati, memorizzate attraverso i peer, e descrizioni semantiche dei peer stessi, aiutino nella formulazione di interrogazioni tali da poter essere comprese da altri peer, aiutino nell'amalgamazione delle risposte ricevute da altri peer e nell'instradamento delle richieste attraverso la rete. In particolare l'uso di ontologie e delle tecnologie del Semantic Web è stato identificato come il più promettente per i sistemi P2P [25].
Nell'ambito dell'instradamento semantico delle interrogazioni, A. Crespo e H. Garcia-Molina hanno proposto in [10] un sistema P2P in cui i nodi sono collegati fra loro attraverso le relazioni semantiche che intercorrono fra i loro contenuti. Tali connessioni formano quelle che vengono definite *Semantic Overlay Network (SON)*. Grazie alla creazione di queste reti virtuali, le interrogazioni vengono instradate alle *SON* appropriate incrementando le possibilità di ottenere rapidamente le risposte corrette ed evitando il coinvolgimento di nodi con contenuti non rilevanti.
Il capitolo è organizzato come segue. Nella Sezione 3.1 verranno analizzate le principali limitazioni che emergono nel modello P2P quando lo si vuole combinare con la tecnologia del Semantic Web. Nella Sezione 3.2 verranno analizzati gli aspetti P2P specifici e gli argomenti principali per la combinazione e l'integrazione con la tecnologia del Semantic Web. Nella Sezione 3.3 verrà introdotto specificamente il progetto SWAP in cui è stata investigata la tecnologia del Semantic Web per alleviare i problemi dovuti al P2P. Nella Sezione 3.4 verrà introdotto un modello in cui i peer usano un'ontologia comune per publicizzare le loro competenze in una rete P2P. Infine, nella Sezione 3.5 verranno descritte le *SON*, cioè un sistema in cui i nodi di una rete P2P sono semanticamente collegati in base al loro contenuto.

3.1 Limitazioni del P2P per il Semantic Web

La combinazione fra il P2P e la tecnologia del Semantic Web sarà un percorso interessante per muoversi da soluzioni per la gestione della conoscenza più centralizzate, attualmente implicate dalle soluzioni basate sulle ontologie, ad un approccio decentralizzato. Gli scenari P2P aprono la via per derivare concettualizzazioni consensuali, ad esempio fra gli impiegati di un'azienda, e rendono le soluzioni per la gestione della conoscenza più vicine ad essere libere da un'amministrazione centralizzata, in modo da poter essere usate da chiunque, incluse persone e piccole compagnie facenti parte, ad esempio,

di una compagnia virtuale. Sfortunatamente, le soluzioni P2P attuali hanno alcune limitazioni importanti che dovrebbero essere risolte quando le si vuole combinare con la tecnologia del Semantic Web.

- In molti progetti, il P2P è descritto come una soluzione a livello di protocollo e come un mezzo per distribuire spazio disco. Tuttavia, questo è un aspetto di minor importanza per il miglioramento del servizio di gestione della conoscenza. Qui è la condivisione di informazioni e conoscenza che deve essere supportata e non l'organizzazione dello spazio disco o del traffico di rete.

- Le soluzioni esistenti, come Napster e Gnutella, forniscono un supporto limitato alla condivisione dell'informazione e della conoscenza. Napster supporta solo ricerche, basate sulle parole chiave, di titoli di canzoni e nomi di autori, e Gnutella non definisce alcun supporto predefinito per la ricerca. Ogni client Gnutella è, infatti, completamente libero su come interpretare le interrogazioni.

- Gli sforzi fatti dalle industrie, come JXTA di Sun Microsystems, sono limitati ai servizi P2P di match delle stringhe e non viene fornito alcun supporto per condividere ontologie. Le interrogazioni sono specificate in un formato XML arbitrario e non viene fatto uso alcuno di opportunità quali l'RDF o l'RDF Schema per esprimere vocabolari comuni. Infine, JXTA limita il processo di *query answering* (risposta alle interrogazioni) ad usare risorse in una locazione singola, mentre, di fatto, molte interrogazioni richiederebbero la combinazione di informazioni recuperate da diverse risorse in differenti locazioni.

- La selezione dei peer non è, allo stato attuale, realmente basata sul loro contenuto. Questo aspetto necessita di essere migliorato, per consentire ad una interrogazione di essere instradata verso il peer meglio informato più vicino piuttosto che ad uno arbitrario.

Questi difetti rendono, quindi, il P2P inadeguato per gli scopi del processo di condivisione della conoscenza. La chiave del successo nella combinazione tra soluzioni P2P e la tecnologia del Semantic Web è l'uso delle Emergent Semantics [2]. Le Emergent Semantics sono costruite su ontologie heavyweight e/o lightweight che diversi individui, dipartimenti o organizzazioni hanno creato. Considerano la sovrapposizione tra le definizioni di diverse ontologie e l'uso di concetti e relazioni con dati concreti, al fine di estrarre ontologie condivise per un insieme di individui o gruppi di persone. Tool intelligenti useranno, poi, tali definizioni per assicurare che la conoscenza venga

strutturata in modo appropriato, così da poter essere recuperata facilmente ed in modo preciso. La gestione della conoscenza potrà, quindi, avvenire in modo distribuito senza l'overhead dovuto ad un tipo di amministrazione centralizzato.

3.2 Tecniche per combinare P2P e Semantic Web

Questa sezione descrive le questioni specifiche nella combinazione fra i sistemi P2P e la tecnologia del Semantic Web.

- *Servizio di selezione dei peer.* Per ricevere le risposte corrette senza intasare con interrogazioni la rete dei peer, è necessario inoltrare la richiesta al peer "giusto". Per ottenere un tale servizio, il meccanismo di selezione dei peer basato sulle ontologie necessita di sfruttare la somiglianza fra le ontologie.

- *Variazione di ontologie e mancanza di precisione ontologica.* Il problema riguarda l'utilizzo di ontologie differenti, sebbene parzialmente sovrapposte, e non conformi le une rispetto alle altre da parte di diversi peer. Le ontologie vengono infatti prodotte attraverso l'interazione di vari utenti e sono rese disponibili in molte forme differenti; possono, ad esempio, essere rappresentate attraverso classificazioni in cartelle oppure tramite l'utilizzo di meta-dati ma la loro definizione è spesso imprecisa ed incompleta. I processi di allineamento e di mapping[1], utili al fine di rendere conformi ontologie differenti, e gli strumenti per la visualizzazione devono, quindi, essere in grado di trattare diverse ontologie anche senza la specifica definizione di criteri per stabilire collegamenti fra esse. Alcuni dei criteri di allineamento e di mapping fra ontologie possono esere trovati attraveso l'analisi della conoscenza dei peer adoperando i metodi suggeriti dal campo appena emerso dell'Emergent Semantics (ad esempio lo stesso file catalogato per differenti concetti indica l'allineamento di tali concetti). Un motore inferenziale per queste ontologie deve, pertanto, essere in grado di porre domande e

[1] I processi di allineamento e di mapping consentono di stabilire la corrispondenza fra due o più ontologie e di determinare l'insieme di concetti che si sovrappongono (coincidono), l'insieme dei concetti con stesso significato ma nomi e strutture differenti e l'nsieme dei concetti che sono unici in ogni ontologia. Nel processo di allineamento le ontologie sorgenti devono rimanere separate ma devono essere rese coerenti e consistenti le une rispetto alle altre.

fornire risposte alle interrogazioni dei peer in modo robusto e scalabile e, spesso, deve risiedere localmente al peer.

- *Evoluzione dell'ontologia.* In un ambiente P2P, non è possibile aspettarsi che avvenga alcuna manutenzione sulle ontologie (infatti, gli utenti spesso non conoscono qual'è il contenuto delle ontologie che risiedono sulla loro macchina). Come conseguenza, è necessario sviluppare meccanismi che permettano alle ontologie di modificarsi autonomamente, per far fronte al problema della loro evoluzione. Le ontologie dovranno adattare le proprie definizioni, basandosi sulle richieste e sulle relative risposte che circolano all'interno di tutta la rete P2P (quindi non solo sulle interrogazioni e le risposte locali a dove l'ontologia è contenuta).

3.3 SWAP - Semantic Web and Peer to Peer

SWAP combina due tecnologie di gran successo quali il Semantic Web ed il P2P. Questo progetto dimostra che la potenza della computazione P2P e quella della tecnologia del Semantic Web possono essere combinate per supportare ambienti decentralizzati dove ai partecipanti è consentito mantenere viste individuali del mondo mentre condividono conoscenza, in modo tale da mantenere bassi i costi di amministrazione e rendere semplice la condivisione ed il ritrovamento della conoscenza. SWAP propone un modello dei meta-dati (*meta-data model*), basato su RDF(S), che combina le strutture ontologiche con le informazioni necessarie per allineare, far evolvere ed usare tali strutture per l'elaborazione delle interrogazioni. Nella Sezione 3.3.1 vengono esposti i requisiti necessari per lo sviluppo del modello dei meta-dati. La Sezione 3.3.2 si focalizza sull'ambiente SWAP in cui il modello dei meta-dati verrà utilizzato. Il modello viene introdotto nella Sezione 3.3.3. Infine, nella Sezione 3.3.4 vengono descritti i metodi per applicare il modello dei meta-dati.

3.3.1 Requisiti ed obiettivi del modello dei meta-dati

Il modello dei meta-dati deve soddisfare alcuni requisiti che nascono dalla necessità dei peer di condividere con altri peer informazioni residenti in sorgenti di conoscenza esterne. I più imporanti sono:

- sorgenti multiple di informazione;
- trattamento uniforme delle sorgenti interne ed esterne;
- viste multiple sull'informazione disponibile;

- supporto per la risposta e l'instradamento delle interrogazioni;
- distribuzione delle informazioni nella rete.

Il modello dei meta-dati, che verrà introdotto nella Sezione 3.3.3, rifletterà questi requisiti e dovrà raggiungere gli obiettivi elencati di seguito.

- *Integrazione.* Ogni frammento di conoscenza richiede un meta-dato riguardante la sua origine. Per recuperare informazioni esterne, il meta-dato deve catturare informazioni riguardanti il luogo di provenienza del frammento di informazione. Tali informazioni consentiranno di identificare un peer e localizzare le risorse nel suo database (repository).
- *Eterogeneità delle informazioni.* Poiché ogni peer utilizza la propria ontologia locale, le informazioni distribuite sono inerentemente eterogenee. È pertanto richiesto un processo di mapping, ossia andare oltre l'etichettatura eterogenea per poter identificare uno stesso oggetto.
- *Inconsistenza delle informazioni.* Poiché, in un repository locale, le informazioni sono aggiunte da un certo numero di peer, può verificarsi una certa inconsistenza nei dati. È pertanto necessario assegnare un certo grado di attendibilità alle informazioni, in modo che il sistema sia in grado di gestire l'eterogeneità e fornire un'informazione utilizzabile. In modo simile, può essere assegnato un livello di fiducia ai peer per modellare la loro affidabilità.
- *Sicurezza.* Alcune informazioni possono essere di natura privata e non dovrebbero essere visibili agli altri peer. Altre informazioni possono essere ristrette ad un insieme predefinito di peer. Il modello dei meta-dati deve fornire i mezzi per poter esprimere queste politiche di sicurezza.
- *Uso di cache (Caching).* All'interno dei sistemi P2P, non è sempre garantita la disponibilità degli altri peer. Inoltre alcuni peer possono avere una connettività migliore, in termini di disponibilità di banda, rispetto al resto della rete dei peer. Può, quindi, essere utile memorizzare le informazioni all'interno di cache per migliorare l'efficenza della rete. Il meccanismo di caching deve essere trasparente all'utente ma deve essere catturato dal modello dei meta-dati.

3.3.2 L'ambiente SWAP

L'ambiente SWAP è un'infrastruttura generica sviluppata per soddisfare i requisiti che un nodo deve possedere come specificato nella Sezione 3.3.1.

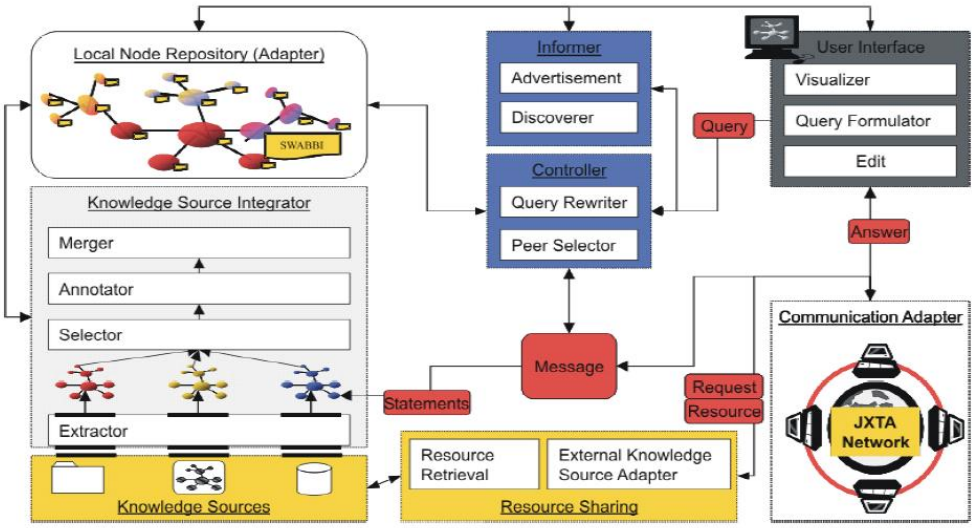


Figura 3.1: *Architettura astratta di un nodo SWAP*

L'architettura proposta è mostrata in Figura 3.1. Entriamo nel dettaglio delle singole componenti.

- *Knowledge Sources.* I peer possono avere informazioni locali come il file system locale, la directory delle email o la lista dei bookmark. Queste sorgenti di informazione locali rappresentano il cuore della conoscenza del peer così come il suo vocabolario di base e rappresentano le strutture in cui un peer può fisicamente memorizzare le informazioni (ad esempio documenti, pagine web) che desidera condividere nella rete.

- *Knowledge Source Integrator.* Il Knowledge Source Integrator è responsabile dell'estrazione e dell'integrazione delle sorgenti di conoscenza interne ed esterne nel Local Node Repository. Questo processo comprende (1) mezzi per l'accesso alle sorgenti locali di conoscenza e l'estrazione di una rappresentazione RDF della conoscenza memorizzata, (2) selezione degli *statement* RDF[2] che devono essere integrati nel Local Node Repository, (3) annotazione degli *statement* con i meta-dati e (4) incorporamento degli *statement* nell'ontologia dell'utente. Questi

[2]Uno *statement RDF* è composto da tre parti: un soggetto (risorsa), un predicato (aspetto specifico, caratteristica, attributo o relazione usata per descrivere la risorsa) e un oggetto (valore della proprietà). RDF prevede l'utilizzo di una sintassi basata su XML per specificare in maniera concreta gli statement RDF che vengono racchiusi dall'elemento `<rdf:RDF>`.

processi, che saranno descritti nella Sezione 3.3.4, utilizzano il modello dei meta-dati che verrà presentato nella Sezione 3.3.3.

- *Local Node Repository.* Il Local Node Repository gestisce un modello integrato della conoscenza disponibile ai peer in modo locale e remoto. I singoli elementi del modello sono annotati, in accordo con il modello dei meta-dati, con la loro sorgente ed un grado che rappresenta quanto il peer crede siano plausibili (la credenza del peer sulla loro plausibilità).

- *User Interface.* La User Interface del peer fornisce viste individuali sull'informazione disponibile sia nelle sorgenti locali sia nella rete. Le viste possono essere implementate usando differenti tecniche di visualizzazione (ad esempio gerarchie di argomenti, mappe tematiche, etc).

- *Informer.* Una delle caratteristiche principali di una rete P2P è quella che i peer, inizialmente, non sanno nulla gli uni degli altri. L'Informer fornisce un meccanismo per scoprire o reclamizzare la conoscenza attivamente.

- *Controller.* Il Controller è la componente coordinatrice che gestisce il processo di distribuzione delle richieste di interrogazione. Riceve le interrogazioni dalla User Interface e le distribuisce in accordo al loro contenuto. Quando il peer riceve una interrogazione da un altro peer, prova a rispondere oppure la inoltra (forward). La scelta del peer al quale inviare una data interrogazione, è presa dal Peer Selector e si basa sulla conoscenza posseduta dai peer. Le risposte ricevute vengono, infine, passate all'interfaccia grafica ed al Knowledge Source Integrator che integra i contenuti selezionati all'interno del Local Node Repository.

- *Communication Adapter.* Questa componente è responsabile della comunicazione fra i peer nella rete. Fornisce un livello di trasporto ad altre parti del sistema per l'invio e l'inoltro delle interrogazioni. Nasconde ed incapsula tutti i dettagli di comunicazione a basso livello al resto del sistema.

3.3.3 Il modello dei meta-dati

L'ambiente SWAP mira a fornire una vista generale della conoscenza che ogni peer possiede, facilitando l'accesso a diverse sorgenti di informazione e consente all'utente di avantaggiarsi della conoscenza degli altri peer. Un modello dei meta-dati fornisce, pertanto, la semantica per annotare dati siano

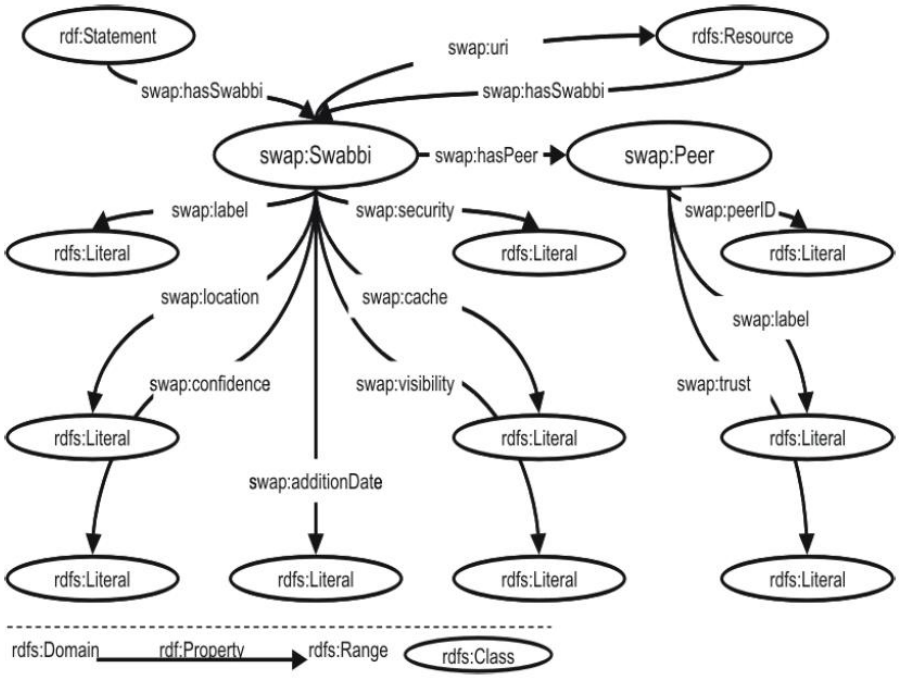


Figura 3.2: *Modello dei meta-dati di SWAP*

essi interni o esterni. L'informazione contenuta in diverse sorgenti e su diversi peer può essere integrata con un tale modello dei meta-dati, per consentire un successivo recupero dell'elemento di informazione desiderato. Consente, inoltre, di memorizzare informazioni nelle cache per rendere l'intera rete di lavoro più efficiente. Un altro scopo del modello dei meta-dati riguarda la gestione dell'eterogeneità delle informazioni, inerente nei sistemi P2P. Per rispondere agli obiettivi della Sezione 3.3.1, il modello dei meta-dati verrà definito in RDF(S). Una visione d'insieme del modello è data in Figura 3.2. (la definizione completa del modello è disponibile all'indirizzo `http://swap.semanticweb.org/2003/01/swap-peer#`).

Il modello consiste di due classi RDFS, cioè la classe "Swabbi" e la classe "Peer". Per queste classi vengono definite alcune proprietà utili al fine di raggiungere gli obiettivi menzionati nella Sezione 3.3.1.

- *Swabbi:* ogni porzione di conoscenza può essere annotata con un oggetto "Swabbi" che contiene meta-informazioni. Tale classe ha le seguenti proprietà:
  - *hasPeer:* questa proprietà è usata per tenere traccia di qual'è il peer a cui l'oggetto "Swabbi" è associato;

  - *uri:* ogni frammento di informazione è stato originariamente creato su un peer. Per tenere traccia dell'origine dell'informazione ed essere in grado di disambiguare l'indirizzo di un oggetto attraverso la rete, l'URI primario viene memorizzato esplicitamente con il meta-dato;
  - *location:* mentre l'URI identifica i meta-dati che rappresentano le risorse all'interno dell'ontologia del Local Node Repository, l'attributo *location* è un identificatore per accedere alle risorse fisiche, ad esempio un documento o un file;
  - *label:* l'attributo *label* è espresso in linguaggio naturale e memorizza come viene chiamata una data informazione all'interno del peer che l'ha originata. Poiché una risorsa può avere nomi diversi su peer diversi, tale proprietà viene aggiunta ad ogni oggetto "Swabbi" e non solo all'oggetto originale;
  - *confidence:* il valore di attendibilità indica, in una scala da 0 a 1, quanto è affidabile uno *statement* specifico. Un alto grado di attendibilità viene espresso con un valore alto; 1 significa che il peer è sicuro che lo *statement* sia vero, 0 significa che lo *statement* è definitivamente falso;
  - *security:* in un ambiente P2P aperto è richiesto qualche controllo sull'accesso alle informazioni per garantirne un uso appropriato;
  - *visibility:* piuttosto che rimuovere completamente gli oggetti, questi possono essere anche nascosti;
  - *additionDate:* questo attributo tiene traccia della data, relativa all'aggiunta o alla modifica, di una risorsa nel Local Node Repository. Tale attributo può essere utilizzato per determinare la attendibilità: informazioni vecchie possono diventare meno affidabili;
  - *cache:* per aumentare l'efficienza della rete è necessario un meccanismo di caching delle informazioni. L'informazione posta in cache viene annotata con questa proprietà e memorizzata con la data della sua inclusione.

- *Peer:* per ogni frammento di conoscenza è necessario ricordare qual'è il peer che l'ha originato. Il Local Node Repository memorizza, pertanto, differenti informazioni su ogni peer conosciuto. L'informazione viene classificata nell'oggetto "Peer". L'oggetto "Swabbi" è collegato al corrispondente oggetto "Peer".

- *peerID:* ad ogni peer è associato un ID unico affinché possa essere identificato in modo univoco. In questo progetto tale identificatore sarà un JXTA UID, poiché l'infrastruttura di comunicazione utilizzata è JXTA;
- *peerLabel:* questo attributo memorizza l'etichetta del peer che è una descrizione del peer in linguaggio naturale, leggibile e comprensibile dall'utente umano;
- *peerTrust:* alcuni peer possono essere più affidabili di altri. Questo attributo viene utilizzato per misurare l'attendibilità su una scala da 0 a 1, dove 0 significa nessuna fiducia sul peer e 1 fiducia assoluta.

3.3.4 Applicazione del modello dei meta-dati

Per poter utilizzare il modello descritto nella Sezione 3.3.3 per lo scambio di informazioni basato sulla semantica, è necessario fornire un insieme di metodi per costruire il repository in accordo al modello dei meta-dati e per valutare la conoscenza che vi è memorizzata all'interno. Nel seguito della sezione vengono descritti:

1. i metodi che sono stati sviluppati per creare ed integrare il contenuto del repository, automaticamente dalle sorgenti di informazione locali e remote;

2. i metodi stimare le informazioni nel repository basandosi sulla fiducia che si ha circa la loro affidabilità.

1. **Integrazione delle sorgenti di conoscenza.**
   Il processo di integrazione delle sorgenti di conoscenza consiste di quattro sottoprocessi:

   (a) *Estrazione di una rappresentazione RDF(S) dalle sorgenti di conoscenza.* Come già menzionato in precedenza, il sistema SWAP fornisce un modello integrato di diverse strutture che esiste localmente sul peer e in modo remoto sugli altri peer nella rete. Tali strutture sono, ad esempio, file system, email, database, ontologie. Per poter includere le diverse sorgenti di informazione che sono state selezionate dall'utente per essere condivise, in questo modello si estrae una rappresentazione RDF(S) per esse. L'esempio in Figura 3.3 mostra un frammento di una struttura cartella (directory) estratta (una risorsa di tipo Folder con etichetta "Project"

```
<rdf:RDF xmlns:rdf =
        http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:rdfs =
        http://www.w3.org/2000/01/rdf-schema#
  xmlns:swapcommon =
       "http://swap.semanticweb.org/2003/01/swap-common#"
  xmlns:swap =
       "http://swap.semanticweb.org/2003/01/swap-peer#">
 <swapcommon:Folder rdf:about =
       "swap://123456.jxta#project">
   <rdfs:label xml:lang = "en">Project</rdfs:label>
   <swapcommon:location>
     filefolder://windows/c:/Project
   </swapcommon:location>
 </swapcommon:Folder>
</rdf:RDF>
```

Figura 3.3: *Esempio di estrazione di una struttura Folder*

```
<rdf:Description rdf:about="swap://123456.jxta#SWAP">
  <rdf:type rdf:resource="swap://123456.jxta#project"/>
</rdf:Description>
```

Figura 3.4: *Esempio di descrizione dei un'informazione estratta*

e la locazione originale nella struttura directory). In un secondo momento, questa informazione estratta potrà essere arricchita semanticamente, ossia si potrà rendere esplicita la semantica che era precedentemente implicita nelle strutture. Per il momento assumiamo che si possa determinare il significato, così come inteso dall'utente, ed il tipo di informazione in un senso ontologico, ossia che si possa descrivere come un concetto, un'istanza o una proprietà. Se si estrae una directory "Project" dal file system e tale directory ha una sotto directory "SWAP", si assume che l'utente abbia posto lì le sue informazioni, dal momento che SWAP è un'istanza del progetto. Tale aspetto viene modellato come mostrato in Figura 3.4.

(b) *Selezione degli statement da includere nel Local Node Repository.* Il processo di selezione può essere descritto come un filtro su uno *statement* in ingresso. Bisogna decidere quale informazione è utile e dove deve essere inclusa nel Local Node Repository. I meta-dati

precedentemente raccolti, riguardanti la fiducia negli altri peer, possono dimostrarsi utili nella definizione del processo di selezione. L'informazione selezionata verrà annotata con i meta-dati per essere inclusa nel Local Node Repository, come descritto nel prossimo punto.

(c) *Annotazione degli statement con meta-dati.* Dopo aver costruito una rappresentazione delle sorgenti di informazione, in questo passo del processo vengono aggiunti gli oggetti "Swabbi". È stata fatta una distinzione fra le informazioni originate dal processo di estrazione sul peer locale e le informazioni ricevute da altri peer. Per le strutture estratte, viene creato un nuovo oggetto "Swabbi" per ogni risorsa o *statement*. Per collegare un oggetto "Swabbi" ad uno *statement* viene utilizzato il costrutto di reificazione dell'RDF. Le proprietà sono riempite in accordo a quanto specificato nella Sezione 3.3.3 e viene stabilito, attraverso la relazione "hasSwabbi" (Figura 3.2), un collegamento tra la risorsa e l'oggetto "Swabbi". Le altre principali sorgenti di informazione sono i peer. Il processo di selezione, descritto al passo precedente, ha già selezionato quali sono le informazioni da mantenere. Tali informazioni vengono aggiunte al Local Node Repository. Inoltre, l'informazione riguardante il peer che le ha originate viene inclusa usando la proprietà "hasPeer" dell'oggetto "Swabbi". L'esempio in Figura 3.5 mostra uno *statement* reificato con la sua informazione "Swabbi": il primo elemento descrive lo *statement* stesso, il secondo descrive l'oggetto "Swabbi" associato e l'ultimo descrive il peer corrispondente.

(d) *Incorporamento degli statement nel modello di conoscenza dell'utente.*
Uno degli scopi del progetto SWAP è quello di mantenere un modello unico. Attraverso l'aggiunta di risorse e *statement* da altri peer, lo stesso oggetto può essere presente nel repository locale ma con nomi differenti. Il sistema deve poter identificare questi oggetti attraverso misure di somiglianza. Nell'approccio descritto in [53] sono presentati tre metodi che in SWAP vengono estesi con un quarto:

- Word Matching: controllando le etichette è possibile determinare se gli oggetti sono simili.
- Feature Matching: se un'entità ha gli stessi predicati e oggetti di un'altra entità, questa è un'indicazione forte che l'oggetto è lo stesso in entrambi i casi.

```
<rdf:Statement rdf:about="swap://123456.jxta#statement01">
  <rdf:subject rdf:resource="swap://123456.jxta#project"/>
  <rdf:predicate rdf:resource="rdfs:subclassOf"/>
  <rdf:object rdf:resource="swap://123456.jxta#thing"/>
  <swap:hasSwabbi rdf:resource=
      "swap://123456.jxta#swabbiObjectNo01"/>
</rdf:Statement>

<swap:Swabbi rdf:about=
    "swap://123456.jxta#swabbiObjectNo01">
  <swap:hasPeer rdf:resource=
      "swap://123456.jxta#knownPeers0001"/>
  <swap:label>Project</swap:label>
  <swap:uri rdf:resource="swap://123456.jxta#project"/>
  <swap:location>
   filefolder://windows/c:/Project
  </swap:location>
</swap:Swabbi>

<swap:Peer rdf:about="swap://123456.jxta#knownPeers0001">
  <swap:peerId>123456</swap:peerId>
  <swap:peerLabel>Christoph</swap:peerLabel>
</swap:Peer>
```

Figura 3.5: *Esempio di statement RDF*

- Semantic-Neighborhood Matching: questo metodo controlla la vicinanza (contestuale) fra due entità ed è descritto in [37].
- Instance Matching: ci si occupa delle istanze e di dove vengono classificate nella struttura. Questo indica una certa somiglianza nelle rispettive classi.

Il primo prototipo del sistema SWAP utilizza la tecnica Word Matching basata sulle etichette delle entità. La tecnica Instance Matching viene realizzata confrontando gli identificatori univoci (ad esempio un codice hash nel caso di file e l'indirizzo email nel caso di persone). L'ultimo passo è quello di effettuare la fusione nel Local Node Repository. Mentre in RDF(S) non è definita una relazione esplicita di uguaglianza, in questo progetto è stata derivata una relazione di uguaglianza definita in OWL.

2. **Modello di stima dei contenuti.** Gli *statement* creati dai peer possono essere incompleti, vaghi e, a volte, persino falsi. Per questa ragione, gli *statement* non vengono accettati come una verità assoluta, ma vengono giudicati sulla base dell'esperienza precedente con il peer che li ha forniti come risposta ad una interrogazione [58]. Per esempio se il mittente riferisce al ricevente informazioni riguardanti gli ostelli della gioventù ed il ricevente sa che il mittente è un esperto di hotel, allora può derivare che, dal fatto che entrambi i concetti (hotel ed ostelli della gioventù) hanno una distanza semantica molto bassa, il mittente probabilmente saprà anche qualcosa circa gli ostelli, molto più di un utente medio comunque. Per formalizzare la competenza vengono introdotte delle misure di attendibilità, cioè *meta-statement* posti nell'oggetto Swabbi" che indicano l'affidabilità di un dato *statement*. Verranno ora descritti differenti aspetti dei metodi di misura dell'attendibilità:

   - *Assegnazione di misure di confidenza da un peer ad uno statement.* In questo caso bisogna distinguere fra gli *statement* derivati dagli algoritmi di estrazione precedentemente descritti e gli *statement* ricevuti da peer esterni. Nel primo caso si assume che l'utente abbia fiducia negli *statement* derivati ai quali perciò assegnerà un alto grado di attendibilità. Nel secondo caso, i valori di fiducia sono calcolati in base agli *statement* precedentemente ricevuti dal mittente. Quando un peer a riceve informazioni da un peer b e b non è conosciuto da a, lo *statement* da b avrà un basso valore di fiducia iniziale. Se invece b ha già fornito *statement* ad a, viene calcolato un nuovo valore di attendibilità. Il valore si ottiene

attraverso una media pesata, dove il fattore peso è determinato dalla distanza semantica fra un vecchio *statement* ed il nuovo. La misura di somiglianza usata è stata adattata da quella fornita in [52].

- *Modifica dei valori di attendibilità.* Se altri peer, diversi dal mittente originale, confermano lo *statement* ripetendolo, allora tale *statement* acquisisce un'alta attendibilità. L'aumento dipende dall'attendibilità della sorgente che conferma.
- *Determinazione degli esperti.* Quando una interrogazione viene ricevuta, chi la riceve prova prima a rispondere e, se non possiede una risposta soddisfacente, prova a trovare un esperto sull'argomento dell'interrogazione basandosi sulle informazioni ricevute da altri peer. Il sistema prova a trovare esperti sull'argomento che abbiano un distanaza semantica bassa rispetto all'argomento dell'interrogazione, adoperando le misure di somiglianza descritte in precedenza.
- *Meccanismi per svalutare le misure di confidenza nel tempo.* Un peer SWAP può recuperare un vasto insieme di *statement* da quelli generati dall'estrattore di ontologie e dagli altri peer. Per rendere il Local Node Repository scalabile, è utilizzato un meccanismo, basato sull'età, per rimuovere gli *statement* troppo obsoleti e con un basso valore di attendibilità.

3.4 Selezione dei peer con topologia semantica in una rete P2P

In [25] viene presentato un modello per la selezione dei peer basata sulla *expertise* (competenza) in cui viene creata una topologia semantica fra i peer attraverso la pubblicizzazione della loro *expertise*, indipendente dalla topologia della rete fisica sottostante. Nel modello proposto i peer usano un'ontologia condivisa per publicizzare le loro competenze in una rete P2P. Se un peer riceve una interrogazione può decidere di inoltrarla solo a quei peer di cui conosce che le competenze sono simili al soggetto dell'interrogazione. Il vantaggio di un tale approccio risiede nel fatto che le interrogazioni non saranno inoltrate a tutti i peer o ad un sottoinsieme casuale di essi ma solo a quelli che con buona probabilità sapranno rispondere.
Nel seguito della sezione gli esempi inerenti al modello generale verranno istanziati su uno scenario bibliografico di condivisione della conoscenza in

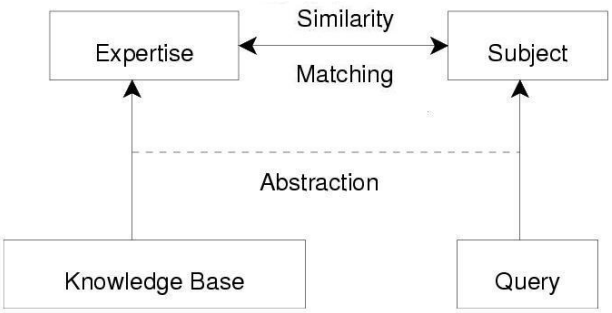


Figura 3.6: *"Matching" basato sull'expertise*

una rete P2P, in cui i peer condividono le descrizioni bibliografiche delle pubblicazioni. Lo scenario considerato sarà quello di una comunità di ricercatori che desiderano condividere meta-dati bibliografici attraverso un sistema P2P.

3.4.1 Modello di selezione dei peer basata sull'*expertise*

Nel modello proposto, i peer reclamizzano le loro competenze nella rete. La selezione dei peer è basata sulla corrispondenza fra il soggetto dell'interrogazione e l'*expertise* in accordo alla loro affinità semantica. La Figura 3.6 mostra l'idea del modello. Nel seguito della sezione verrà introdotto un modello per descrivere semanticamente le competenze dei peer e verrà descritto come i peer promuoveranno le loro competenze con messaggi pubblicitari nella rete. Inoltre, verrà descritto come la ricezione delle pubblicità consentirà di selezionare, per una data interrogazione, i peer in base alla corrispondenza fra l'interrogazione e la descrizione della loro *expertise*. Infine, verrà descritto il processo in base al quale, attraverso la pubblicizzazione dell'*expertise* dei peer, potrà essere formata la topologia semantica.

Descrizione semantica dell'expertise

- *Peer.* La rete P2P consiste di un insieme di peer P. Ogni peer $p \in P$ possiede una base di conoscenza che contiene la conoscenza che desidera condividere.

  Esempio 3.1 *Un ricercatore è rappresentato da un peer $p \in P$. Ogni peer ha una base di conoscenza RDF che consiste di un insieme di meta-dati bibliografici classificati in accordo alla gerarchia di argomenti dell'ACM* [3]*. In Figura 3.7 è mostrato un frammento di esempio di un*

[3] `http://www.cs.vu.nl/~heiner/public/SW@VU/classification.daml`

```
<rdf:RDF
xmlns =
  "http://www.semanticweb.org/ontologies/swrc-onto.daml#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:acm ="http://daml.umbc.edu/ontologies/topic-ont#">
 <Publication rdf:about ="dblp:persons/Codd81">
   <title>
     The Capabilities of
     Relational Database Management Systems.
   </title>
   <acm:topic rdf:resource =
     "http://daml.umbc.edu/ontologies/classification#
ACMTopic/Information_Systems/Database_Management" />
   <!- - ... - ->
 </Publication>
</rdf:RDF>
```

Figura 3.7: *Base di conoscenza di un peer*

argomento bibliografico basato sul Semantic Web Research Community Ontology (SWRC) [4]. □

- *Ontologia comune.* I peer condividono un'ontologia O che fornisce una concettualizzazione comune del loro dominio. L'ontologia viene usata per descrivere l'*expertise* dei peer ed il soggetto delle interrogazioni.

  Esempio 3.2 *L'ontologia O condivisa da tutti i peer è la gerarchia degli argomenti dell'ACM. La gerarchia contiene un insieme, T, di 1287 argomenti nel dominio dell'*informatica *e due relazioni fra essi:* SubTopic *e* seeAlso. □

- *Expertise.* La descrizione di una *expertise* $e \in E$ è una descrizione semantica ed astratta della base di conoscenza del peer basata sull'ontologia comune O. Questa *expertise* può essere estratta dalla base di conoscenza in modo automatico o spcificata in qualche altro modo.

  Esempio 3.3 *La gerarchia di argomenti dell'ACM è la base per per il modello basato sull'*expertise*. L'*expertise E *è definita come $E \subseteq 2^T$ dove ogni $e \in E$ denota un insieme di argomenti ACM per i quali un peer fornisce istanze classificate.* □

[4] `http://ontobroker.semanticweb.org/ontos/swrc.html`

```
CONSTRUCT {pub} <swrc:title> FROM
{Subject} <rdf:type> {<swrc:Publication>};
  <swrc:title> {title};
  <acm:topic>
  {<topic:ACMTopic/Information_Systems/Database_Management>}
USING NAMESPACE
swrc =<!http://www.semanticweb.org/ontologies/
swrc-onto.daml#>,
rdf =<!http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
acm =<!http://daml.umbc.edu/ontologies/topic-ont#>,
topic =<!http://daml.umbc.edu/ontologies/classification#>
```

Figura 3.8: *Interrogazione in SeRQL*

- *Advertisement (pubblicità).* Le pubblicità $A \subseteq P \times E$ sono usate per promuovere le descrizioni delle *expertise* dei peer nella rete. Una pubblicità $a \in A$ associa un peer p ad un'*expertise* e. I peer decidono in modo autonomo, senza controllo centrale, a chi inviare una pubblicità e quali pubblicità accettare. Questa decisione può essere basata sull'affinità semantica tra le descrizioni delle diverse *expertise*.

  Esempio 3.4 *Le pubblicità sono associate ai peer e alla loro* expertise*: $A \subseteq P \times E$. Una singola pubblicità, quindi, consiste di un insieme di argomenti ACM su cui i peer sono esperti.* □

Selezione dei peer

- *Interrogazioni.* Le interrogazioni $q \in Q$ vengono poste dall'utente e sono valutate sulle basi di conoscenza dei peer. Per prima cosa il peer valuta l'interrogazione sulla sua base di conoscenza e poi decide a quale peer deve essere inoltrata. I risultati dell'interrogazione sono restituiti al peer che ha originariamente posto l'interrogazione.

  Esempio 3.5 *Per esprimere le interrogazioni sulla base di conoscenza RDF dei peer viene usato il linguaggio SeRQL [8]. L'esempio in Figura 3.8 mostra un'interrogazione che richiede i titoli delle pubblicazioni ACM con argomento* Information Systems/Database Management. □

- *Soggetti.* Un soggetto $s \in S$ è un'astrazione di una data interrogazione q espressa in termini dell'ontologia comune. Il soggetto può essere visto come un complemento alla descrizione dell'*expertise* poiché specifica l'*expertise* richiesta per rispondere ad una interrogazione.

Esempio 3.6 *In modo analogo all'*expertise*, un soggetto $s \in S$ è un'astrazione di un'interrogazione q. Nello scenario introdotto per gli esempi, ogni s è un insieme di argomenti ACM perciò $s \subseteq T$. Ad esempio, il soggetto estratto dall'interrogazione dell'Esempio 3.5 è* Information Systems/Database Management. □

- *Funzione di similarità.* La funzione di similarità $SF : S \times E \rightarrow [0, 1]$ produce l'affinità semantica fra un soggetto $s \in S$ ed una descrizione di *expertise* $e \in E$. Un valore alto indica alta affinità. Se il valore è 0, s ed e non sono simili, se il valore è 1, combaciano esattamente. SF è usata per determinare a quali peer deve essere inoltrata un'interrogazione. In modo analogo, può essere definita una funzione $E \times E \rightarrow [0, 1]$ per determinare l'affinità fra le *expertise* di due peer.

  Esempio 3.7 *Nello scenario introdotto per gli esempi, la funzione di similarità SF è basata sull'idea che gli argomenti che nella gerarchia si trovano più vicini sono più simili di quelli con una grande distanza. Ad esempio, un esperto sull'argomento dell'ACM* Information Systems/Information Storage and Retrieval *ha una possibilità maggiore di rispondere in modo corretto ad un'interrogazione sull'argomento* Information Systems/Database Management *piuttosto di un esperto sull'argomento meno affine* Hardware/Memory Structures*. Per poter definire l'affinità fra l'*expertise *di un peer ed il soggetto di un'interrogazione, entrambe rappresentate come un insieme di argomenti, definiamo prima la similarità fra due singoli argomenti:*

$$S(t_1, t_2) = \begin{cases} \exp^{-\alpha l} \cdot \frac{\exp^{\beta h} - \exp^{-\beta h}}{\exp^{\beta h} + \exp^{-\beta h}} \\ 1 \end{cases}$$

  In questa formula l è la lunghezza del cammino minimo fra gli argomenti t_1 e t_2 nel grafo formato tramite la relazione SubTopic*, h è il livello nell'albero della classe più specifica da cui discendono i sottoalberi contenenti t_1 e t_2.*
  $\alpha \geq 0$ e $\beta \geq 0$ sono parametri che bilanciano il contributo rispettivamente del cammino minimo l e della profondità h. L'intuizione dietro alla scelta di adoperare la profondità della classe più specifica da cui discendono i sottoalberi contenenti gli argomenti da confrontare, deriva dall'osservazione che gli argomenti ai livelli più alti della gerarchia semantica sono più generali e sono semanticamente meno simili rispetto a quelli ai livelli inferiori.

```
subject := ExtractSubject(query)
rankedPeers := ∅
for all ad ∈ A do
  peer := Peer(ad)
  rank := SF(Expertise(ad), subject)
  if rank > γ then
    rankedPeers := (peer, rank) ∪ rankedPeers
return rankedPeers
```

Figura 3.9: *Algoritmo di selezione dei peer*

Una volta definita la funzione per il calcolo dell'affinità fra due singoli argomenti, è possibile definire SF come:

$$SF(s,e) = \frac{1}{|s|}\sum_{t_i \in s}(\max_{t_j \in e} S(t_i, t_j))$$

□

- *Algoritmo di selezione dei peer.* L'algoritmo di selezione dei peer restituisce un insieme ordinato di peer. Il grado di rilevanza è uguale al valore di affinità fornito dalla funzione di similarità. Da questo insieme ordinato di peer è possibile, ad esempio, selezionare gli n peer migliori, oppure tutti i peer il cui grado di rilevanza di risulta essere al di sopra di una data soglia. Il codice relativo all'algoritmo di selezione dei peer è mostrato in Figura 3.9, dove A è l'insieme delle pubblicità disponibili sul peer e γ è l'affinità minima tra l'*expertise* di un peer e gli argomenti dell'interrogazione.

  Esempio 3.8 *L'algoritmo di selezione dei peer ordina i peer conosciuti in accordo alla funzione di similarità descritta nell'Esempio 3.7. Quindi, i peer che hanno* expertise *più simile al soggetto dell'interrogazione avranno un grado di rilevanza alto. Dall'insieme ordinato dei peer, si considera un algoritmo che seleziona solo i migliori n.* □

Topologia semantica

La conoscenza dei peer circa l'*expertise* degli altri peer costituisce la base per la topologia semantica. È importante precisare che la topologia semantica è indipendente dalla topologia della rete fisica sulla quale non viene fatta alcuna assunzione particolare.
La topologia semantica può essere descritta attraverso la relazione seguente:

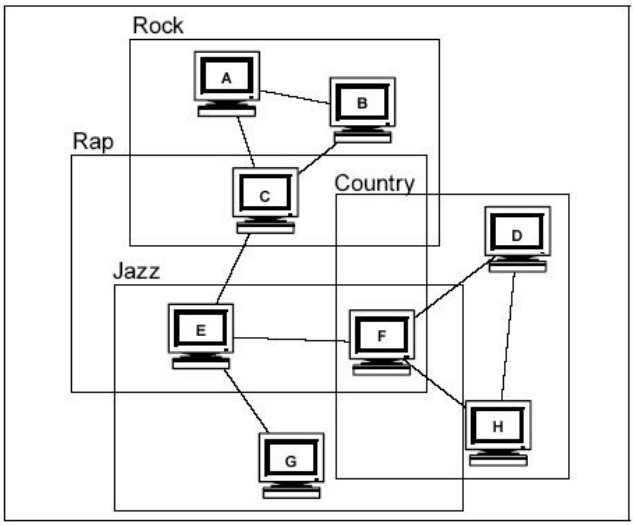


Figura 3.10: *Semantic Overlay Network*

$Knows \subseteq P \times P$, dove $Knows(p_1, p_2)$ significa che p_1 conosce l'*expertise* di p_2.

La relazione $Knows$ è stabilita attraverso la selezione dei peer a cui un peer invia le proprie pubblicità. Tuttavia, i peer possono decidere di accettare le pubblicità oppure di scartarle. La topologia semantica in combinazione con la selezione dei peer basata sull'*expertise* è la base per il processo di instradamento intelligente delle interrogazioni.

3.5 SON - Semantic Overlay Network

In questa sezione viene presentato il modello delle *Semantic Overlay Network (SON)* [10] basato sulla creazione di un'organizzazione di rete flessibile, che migliora le prestazioni di risposta ad un'interrogazione mantenendo un'alto grado di autonomia dei peer ed è basata sulle relazioni semantiche esistenti fra essi. I principi fondamentali su cui si basa questo approccio sono:

- il raggruppamento dei peer in base al loro contenuto;
- la possibile sovrapposizione di tali gruppi (*cluster*) dovuta al fatto che i peer possono contenere differenti contenuti e, quindi, appartenere a diversi gruppi;
- la distribuzione delle interrogazioni solo ai gruppi rilevanti e l'inoltro solo ai peer rilevanti all'interno del gruppo;

- l'esclusione dei gruppi irrilevanti nell'inoltro di una richiesta.

Quindi per creare una rete di questo tipo è necessario costruire un sistema di reti alternative costruite sopra quella fisica. Tali reti possono essere chiamate *Semantic Overlay Network* e devono contenere solo i peer rilevanti gli uni per gli altri. Ogni *SON* rappresenta un livello virtuale, astratto e indipendente di gruppi di peer precedentemente creati e classificati. Queste reti giocano il ruolo di mediatori tra le interrogazioni ed i peer, sono responsabili della comprensione del significato delle interrogazioni, di stabilire le relazioni semantiche fra le interrogazioni ed i peer e di implementare l'instradamento delle interrogazioni solo ai peer rilevanti riducendo, in tal modo, il sovraccarico della rete fisica.

Esempio 3.9 *Si consideri la Figura 3.10 che mostra otto nodi, da A ad H, connessi (attraverso le linee continue). Quando si utilizza una* SON*, i nodi sono connessi solo a quelli con contenuti semanticamente simili. Ad esempio, i nodi A, B e C riguardano canzoni "Rock"; in modo simile i nodi C, E ed F riguardano canzoni "Rap". Va sottolineato che i nodi appartenenti alla stessa* SON *non devono necessariamente essere collegati in modo diretto (ad esempio C ed F).* □

3.5.1 Definizione formale di SON

Ci sono due tipi di definizioni per una SON:

1. *Definizione basata sulla struttura "link" (collegamento).*
   In questo modello il sistema è visto come un insieme di nodi $\mathcal{N}$ dove ogni nodo $n_i \in \mathcal{N}$ mantiene un insieme di documenti D_i. Ogni nodo è logicamente collegato ad un insieme, relativamente piccolo, di nodi (detti *vicini*) i quali, a loro volta, sono collegati a più nodi.
   Un *link* è una tripla $(n_i,\ n_j,\ l)$ dove n_i ed n_j sono i peer connessi ed l è una stringa, nome del concetto nel contesto del quale i nodi sono collegati. Un'*overlay network* è rappresentata come un insieme di nodi $ON_l = \{n_i \in \mathcal{N} | \exists\ un\ link(n_i,\ n_j,\ l)\}$.
   Inoltre, per generare una overlay network, devono essere implementate le seguenti funzioni:

   - $Join(n_i,\ l)$ provoca la creazione di uno o più link della forma $(n_i,\ n_j,\ l)$ (dove $n_j \in ON_l$);
   - $Search(r,\ l)$ restituisce un insieme di peer in ON_l con un *match* per la richiesta r;

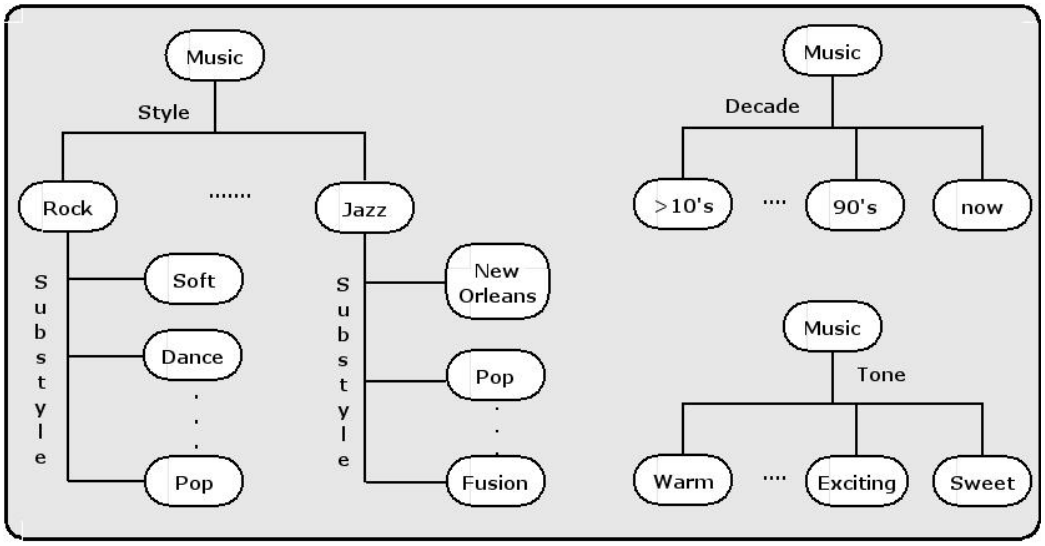


Figura 3.11: *Gerarchie di classificazione*

- *Leave*$(n_i,\ l)$ provoca la cancellazione di tutti i *link* in ON_l che includono n_i.

L'implementazione di queste funzioni potrà essere fatta da ogni peer o da un sottoinsieme dei peer o fornita da un computer esterno alla rete e varierà da sistema a sistema.
Le richieste di documenti avvengono attraverso la diffusione di una interrogazione (*query*) q e di alcune informazioni addizionali dipendenti dal sistema (ad esempio l'orizzonte della richiesta ossia a quanti nodi ogni nodo dovrà inoltrarla). Inoltre, anche le interrogazioni sono dipendenti dal sistema e possono essere semplicemente rappresentate attraverso l'identificatore di un documento o una lista di parole chiave, oppure in modo più complesso possono essere scritte in linguaggio *SQL*.
In questo modello un *match* (corrispondenza) tra un documento d e una interrogazione q è dato dalla funzione $M(q,\ d)$ che ritorna 1 se c'è un *match* e 0 altrimenti. Quindi il numero di successi per una interrogazione q in un nodo n_i è data dalla funzione $H(q,\ n_i) = \sum_{d \in D_i} M(q,\ d)$.
In modo simile, il numero di successi in un'*overlay network* è dato dalla funzione $H(q,\ ON_l) = \sum_{n_i \in ON_l} H(q,\ n_i)$.
Le interrogazioni possono essere *esaustive*, nel caso in cui il sistema restituisca tutti i documenti che soddisfano la richiesta, o *parziali*, nel caso in cui associato all'interrogazione vi sia il numero minimo di risultati che devono essere restituiti.

2. *Definizione basata sul concetto di gerarchia di classificazione.*
   In questa definizione viene usata una gerarchia di classificazione come base per la formazione delle *overlay network*.
   Una gerarchia di classificazione H è un albero di concetti.

   Esempio 3.10 *In Figura 3.11 sono mostrate tre possibili gerarchie di classificazione per documenti riguardanti musica. Nella prima i documenti di musica sono classificati in accordo al loro stile (rock, jazz, etc.) ed al loro sottostile (soft, dance, etc.); nella seconda sono classificati per decade; infine, nella terza, sono classificati per tono (warm, exciting, etc.).* □

   Ogni documento ed ogni interrogazione viene classificato in uno o più concetti che risiedono nelle foglie nella gerarchia. Concettualmente la classificazione dei concetti e delle interrogazioni è fatta attraverso due funzioni, rispettivamente $C_q^*(q)$ e $C_d^*(d)$, che restituiscono uno o più concetti foglia in H. Queste funzioni sono scelte in modo tale che se $M(q,d) = 1$ allora $C_q^*(q) \cap C_d^*(d) \neq \emptyset$. In pratica, però, le procedure di classificazione possono essere imprecise poiché non è sempre possibile determinare esattamente a quale concetto appartiene un documento o una interrogazione. In questo caso funzioni di classificazione imprecise, $C_q(q)$ e $C_d(d)$, possono restituire concetti che non risiedono nelle foglie della gerarchia. Quando ciò accade significa che il documento o l'interrogazione appartengono ad uno o più discendenti di un concetto che non risiede nelle foglie ma il classificatore non sa determinare quale sia.

   Esempio 3.11 *Quando si usa la gerarchia di classificazione più a sinistra in Figura 3.11, un documento riguardante il "Pop" può essere classificato come "Rock" se il classificatore non può determinare a quale sottostile ("Pop", "Dance" o "Soft") appartiene effettivamente il documento.* □

   Precisamente, se $c \in C_q^*(q)$ allora $\exists c' \in C_q(q)$ tale che $c' \geq c$ e se $c \in C_d^*(d)$ allora $\exists c' \in C_d(d)$ tale che $c' \geq c$, dove $c' \geq c$ significa che c' è uguale a c oppure che c' è un antenato di c in H.
   Questa definizione e la definizione di C^* implicano che se $M(q,d) = 1$ allora $\exists\ c_q \in C_q(q)$ ed $\exists\ c_d \in C_d(d)$ tali che $c_q \geq c_d$ oppure $c_d \geq c_q$.
   I classificatori possono anche incorrere in errori restituendo il concetto sbagliato per un'interrogazione o un documento. Precisamente, si verifica un errore di classificazione quando $M(q,d) = 1$ ma $\nexists(c_q \in C_q(q) \wedge c_d \in C_d(d))$ tali che $c_q \geq c_d \vee c_d \geq c_q$.

Sino ad ora sono stati considerati solamente i documenti ma in un sistema P2P i documenti sono mantenuti dai nodi. Pertanto, è utile raggruppare semanticamente anche i peer della rete. Formalmente, si definisce una *Semantic Overlay Network* come una *overlay network* associata ad un concetto di una gerarchia di classificazione. Quindi i documenti di ogni peer devono essere assegnati ai concetti di una data tassonomia affinché il peer possa essere assegnato alla corrispondente *SON*. Chiamiamo una "*SON* associata al concetto c" semplicemente "la *SON* di c" oppure "SON_c".

Esempio 3.12 *Nella gerarchia di classificazione più a sinistra in Figura 3.11 si definiscono 9 SON: 6 associate ai nodi foglia ("soft", "dance", "pop", "New Orlenas", etc.), una associata a "rock", un'altra associata a "jazz" ed infine una associata a "music".* □

Per definire completamente una *SON* è necessario specificare come i nodi vengono assegnati alle *SON* e come decidono quale *SON* utilizzare per rispondere ad una interrogazione. Un nodo decide a quale *SON* collegarsi in base alla classificazione dei suoi documenti.

3.5.2 Assegnazione dei peer alle SON

È stato verificato che una buona classificazione dipende fortemente dalle strategie di assegnamento dei peer alle *SON*.

- La strategia più ovvia è quella **conservativa** in cui si pone un peer in una SON_c se qualche suo documento è classificato in c. L'inconveniente principale di questa strategia è quello di creare molte connessioni fra i peer e le *SON* che possono diventare difficilemente gestibili dai nodi.

- Una strategia **meno conservativa** porrà un peer nella SON_c solo se esso possiede un numero significativo di documenti riguardanti il concetto c. In questo caso è possibile:

  – ridurre il numero di peer per ogni *SON* e

  – ridurre il numero di *SON* alle quali i peer appartengono.

  L'inconveniente risiede nel fatto che è possibile non trovare alcun documento se i peer rilevanti non sono connessi.

- La soluzione finale suggerita dagli autori è quella di usare l'approccio più sofisticato delle **Layered SON's**. I nodi determinano a quale *SON*

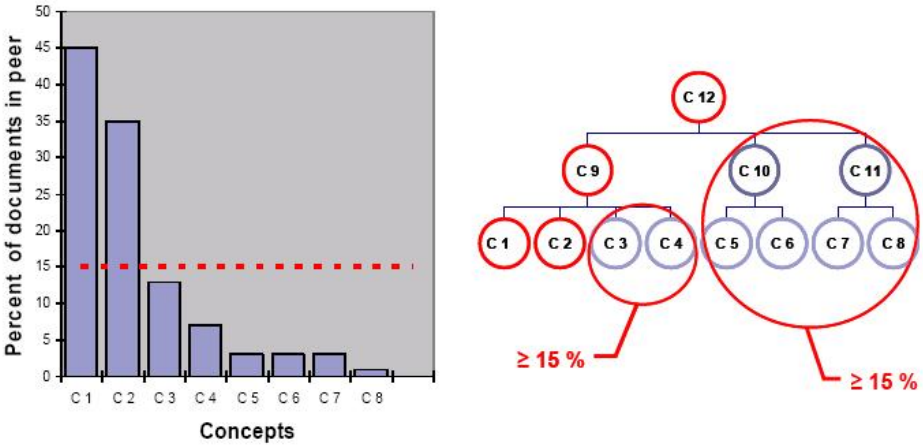


Figura 3.12: *Passo di generazione di una Layered SON*

connettersi sulla base del numero di documenti posseduti per ogni categoria. Per illustrare il principio di un tale approccio si consideri la Figura 3.12. Nella parte destra della figura è mostrata la gerarchia dei concetti in base alla quale il nodo stabilirà a quale *SON* collegarsi. Inoltre uno dei parametri di tale approccio è la percentuale minima di documenti che un nodo dovrebbe possedere in una categoria per appartenere alla *SON* associata. Nell'esempio tale percentuale è stata fissata al 15%, cioè il peer si può collegare alla *SON* del concetto c se ha almeno il 15% di documenti riguardanti il concetto c. Vediamo a quale *SON* il peer con istogramma (rappresentante la distribuzione dei suoi concetti) mostrato nella parte sinistra della figura si potrà collegare. Consideriamo per primi i concetti al livello più basso della gerarchia (da c_1 a c_8). Poiché c_1 e c_2 sono sopra il 15% il nodo si collega alla SON_{c_1} ed alla SON_{c_2}, mentre non si collega alle *SON* relative agli altri concetti di questo livello perchè sono tutti sotto il 15%. Consideriamo ora il secondo livello della gerarchia (c_9, c_{10} e c_{11}). La combinazione di tutti i concetti di c_9 non assegnati (c_3 e c_4) è maggiore del 15% e quindi il peer si collega anche alla SON_{c_9}. Non si collega, invece, alla $SON_{c_{10}}$ poiché la combinazione di c_5 e c_6 è minore del 15%. In modo simile, non si collega alla $SON_{c_{11}}$ poiché la combinazione di c_7 e c_8 è minore del 15%. Infine, il peer si collega alla *SON* associata al concetto c_{12} ($SON_{c_{12}}$) poiché la combinazione di c_5, c_6, c_7, e c_8 supera il 15%.
Si può osservare come la strategia *conservativa* sia equivalente all'approccio *Layered SON's* con soglia pari allo 0%.

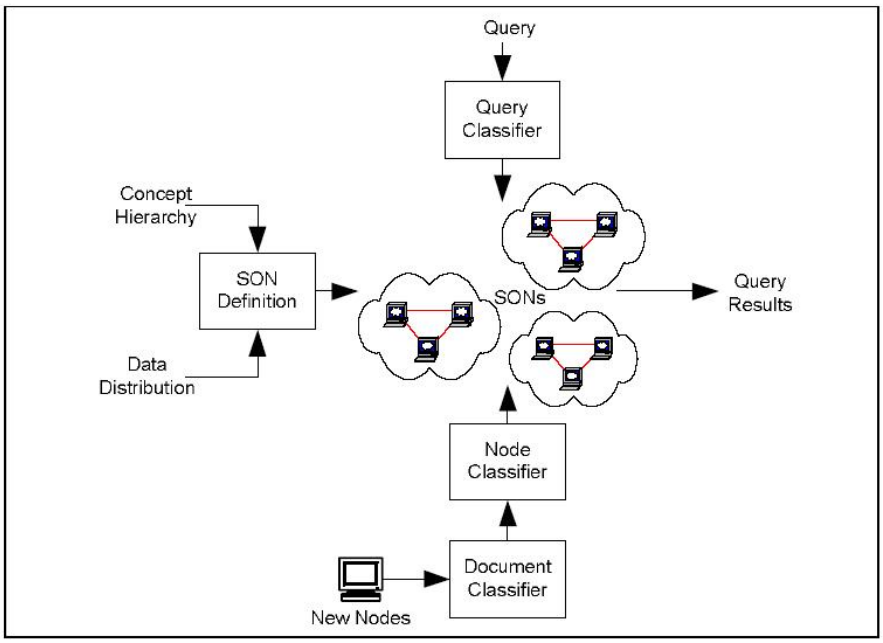


Figura 3.13: *Passo di generazione di una Semantic Overlay Network*

3.5.3 Processo di costruzione ed utilizzo di una SON

Il processo di generazione ed utilizzo delle *SON* è mostrato in Figura 3.13 e consiste delle seguenti fasi:

- *Inizializzazione.*
  Consiste nella valutazione e determinazione delle potenziali gerarchie di classificazione adatte alla distribuzione attuale dei dati nei nodi. Questa gerarchia sarà memorizzata da alcuni nodi e sarà usata per definire la *SON*.

- *Collegamento alla SON.*

  – Un peer che si collega al sistema per prima cosa richiede la gerarchia di classificazione in modo simile a come avviene in Gnutella;

  – esegue, poi, un programma per classificare tutti i suoi documenti in base alla gerarchia ottenuta;

  – un classificatore di peer assegna il peer alla sua *SON*;

  – i peer si collegano ad ogni *SON* cercando i peer che già vi appartengono. Anche questo può essere fatto alla maniera di Gnutella o usando una directory centrale.

- *Risposta ad una query.*
  - per prima cosa un utente pone una interrogazione;
  - il peer classifica l'interrogazione e costruisce la lista delle *SON* a cui deve essere distribuita;
  - la ricerca della *SON* appropriata avviene in modo simile alla ricerca della *SON* cui connettersi;
  - il peer manda l'interrogazione alle *SON* appropriate;
  - dopo che la richiesta è stata inviata alle *SON* appropriate, i peer che ne fanno parte provano a trovare una risposta.

Capitolo 4

Le componenti di SINAPSY P2P

In questo capitolo presentiamo le componenti di SINAPSY P2P, un sistema per il recupero dell'informazione in una rete P2P basato su ontologie che considera la disponibilità dei peer nel fornire risposte a richieste provenienti da altri peer. Il sistema da noi proposto si fonda sui seguenti principi:

- *Uso di ontologie.* In SINAPSY P2P distinguiamo fra i concetti di ontologia globale al sistema ed ontologia locale ad un singolo peer. In fase di registrazione un utente scarica l'ontologia globale del sistema ed in base alle sue competenze ed alle risorse che desidera condividere nel sistema estrae da essa la propria ontologia locale. I concetti contenuti nell'ontologia locale saranno quelli che il peer utilizzerà per creare i messaggi pubblicitari descriventi le sue competenze. La descrizione dettagliata dell'ontologia (sia globale sia locale) e di come è integrata nel sistema viene fornita nella Sezione 4.1.1.

- *Uso di certificati.* Ogni utente può possedere uno o più certificati per garantire sicurezza nell'accesso alle risorse e per certificare determinate sue proprietà (essere studente, essere maggiorenne, essere iscritto ad un'associazione). Tali proprietà sono utili al fine di consentire al peer di specificare politiche di condivisione delle risorse che si basino sulle proprietà dell'utente che invia la richiesta. Tali certificati vengono ottenuti dall'utente in base alle proprietà da lui specificate in fase di registrazione e vengono controllati prima di fornire risposte ad una interrogazione posta da un peer. Presupponiamo che il nostro sistema sia integrato con un sistema per rilasciare certificati che siano autenticati ed autenticabili tramite un meccanismo di firma digitale e che non

siano falsificabili. I certificati vengono definiti e discussi nella Sezione 4.1.2.

- *Uso di politiche.* Ogni peer può specificare delle politiche riguardanti la propria disponibilità a condividere risorse e gli utenti abilitati ad usufruirne. Tali politiche vengono specificate dall'utente in fase di registrazione e vengono controllate ogni volta che viene chiesto al peer di rispondere ad una interrogazione. Le politiche vengono definite e discusse nella Sezione 4.1.3 e determinano le regole in base alle quali i peer possono interagire gli uni con gli altri.

- *Uso di pubblicità.* Il concetto di *adv* (messaggio pubblicitario, volantino) viene utilizzato per promuovere la descrizione delle competenze di ogni peer nella rete. Un *adv* contiene, quindi, informazioni sul tipo di risorse che un peer intende condividere. In tal modo un peer può venire a conoscenza delle competenze di altri peer appartenenti al sistema e, in particolare, di quelli con competenze affini alla propria. Ricevuto un messaggio pubblicitario un peer potrà decidere se memorizzarlo, assieme all'identificatore del mittente, e/o inoltrarlo ad un sottoinsieme dei peer conosciuti. Tale meccanismo consente di porre interrogazioni a quei peer che si ritiene siano in grado di rispondere in base alle loro competenze e di effettuare un inoltro mirato delle richieste. Consente, cioè, un invio mirato anziché casuale delle richieste e aumenta la probabilità di ottenere risposte attinenti ed adeguate. Le definizioni formali di pubblicità e di messaggio pubblicitario vengono fornite rispettivamente nelle Sezioni 4.1.4 e 4.2.1 mentre nel Capitolo 5 verranno descritti in dettaglio i processi di invio e ricezione di un *adv*.

- *Instradamento semantico delle interrogazioni.* Durante il funzionamento del sistema un peer viene a conoscenza delle competenze di un certo numero di peer attraverso il meccanismo di pubblicizzazione delle competenze. Quando l'utente sottomette un'interrogazione al sistema il suo peer, basandosi sulla conoscenza acquisita riguardo ad altri peer, può instradarla in modo mirato a quei peer che con più probabilità sono in grado di rispondere. Per evitare che la conoscenza dei peer resti limitata, il messaggio di richiesta dati viene inoltrato anche ad un numero prefissato di peer scelto in modo casuale fra quelli conosciuti ma con competenze dissimili da quelle della richiesta. Attraverso tale meccanismo è possibile venire a conoscenza di porzioni di rete altrimenti irraggiungibili. Nell'instradamento semantico viene anche tenuto conto del comportamento tenuto da un peer nelle interazioni

precedenti (numero di risposte ricevute e rilevanza di tali risposte relativamente ai concetti presenti nella richiesta). Le definizioni formali di richiesta dati (interrogazione) e di messaggio di richiesta dati vengono date rispettivamente nelle Sezioni 4.1.5 e 4.2.2 mentre i meccanismi di instradamento dell'interrogazione e selezione dei peer a cui inoltrarla verranno dettagliati nel Capitolo 5.

Assumiamo per semplicità di identificare ogni peer con un unico utente che si collega al sistema sempre dallo stesso terminale rendendo l'indirizzo IP statico (cioè sempre lo stesso). In questo modo la corrispondenza fra peer ed utente è uno a uno e gli identificatori di peer e di utente possono essere usati in modo interscambiabile.
In questo capitolo vengono descritte e definite formalmente le componenti fondamentali del sistema. In particolare l'organizzazione dell'ontologia, i concetti di certificato e politica, le nozioni di pubblicità e richiesta dati (interrogazione).
Il capitolo è strutturato come segue. Nella Sezione 4.1 vengono specificati e formalmente definiti tutti gli elementi fondamentali del sistema proposto. Nella Sezione 4.2 vengono specificati e definiti formalmente tutti i tipi di messaggi che i peer possono scambiarsi all'interno del sistema.

4.1 Componenti base

In questa sezione sono presentate le componenti fondamentali del sistema SINAPSY P2P. In particolare:

- è definito il concetto di ontologia, specificato come si intende utilizzare tale nozione nel sistema e vengono definite le relazioni di somiglianza tra ontologie (Sezione 4.1.1);
- sono specificate le nozioni di certificato e politica (rispettivamente Sezioni 4.1.2 e 4.1.3);
- è definita la nozione di pubblicità e specificata la relazione di somiglianza tra una pubblicità e un'ontologia (Sezione 4.1.4);
- è definita formalmente la nozione di richiesta dati (interrogazione) e sono specificate le relazioni di somiglianza tra una richiesta dati e un'ontologia e tra un'interrogazione e una pubblicità (Sezione 4.1.5);
- infine viene definita formalmente la struttura della base di conoscenza di ogni peer (Sezione 4.1.6).

4.1.1 Ontologie

In questa sezione presentiamo l'organizzazione delle ontologie in SINAPSY P2P. Nel sistema proposto vengono distinti due concetti di ontologia: l'ontologia globale del sistema $\mathcal{G}$ e l'ontologia locale ad ogni peer $\mathcal{L}$. L'ontologia $\mathcal{G}$ viene inizialmente definita dallo sviluppatore dell'ontologia che conosce il dominio di appartenenza delle risorse nella rete dei peer. Assumiamo, quindi, che l'ontologia sia tematica (un dominio specifico) e non generica (tutti i domini possibili). L'approccio scelto per definire le nostre ontologie è quello *ibrido* [61] in cui la semantica delle competenze (*expertise*) di ogni peer è descritta attraverso l'ontologia $\mathcal{L}$ locale ad ogni peer. Per rendere le ontologie possedute da ogni peer confrontabili queste vengono costruite a partire dall'ontologia globale $\mathcal{G}$ contenente i termini che definiscono il dominio. In particolare, la relazione esistente fra le due ontologie è $\mathcal{L} \subseteq \mathcal{G}$. In tal modo tutti i peer comunicano attraverso "lo stesso linguaggio" e non si incorre nelle problematiche evidenziate nel Capitolo 3.
Nel seguito della sezione vengono definite formalmente la nozione di ontologia (sia globale sia locale) e la relazione che intercorrente fra $\mathcal{G}$ ed $\mathcal{L}$. Viene inoltre definita la relazione di somiglianza fra un concetto ed un'ontologia locale rispetto all'ontologia globale (Sezione 4.1.1.1) necessaria al processo di confronto fra le competenze dei peer e fra la competenza di un peer e gli argomenti contenuti in una richiesta dati.
Prima di definire la nozione di ontologia è però necessario definire formalmente la nozione di concetto.

Definizione 4.1 *Un* concetto *c è un termine appartenente al linguaggio del dominio. L'insieme dei concetti appartenenti all'ontologia così definiti sarà indicato con* $\mathcal{C}$*. Il concetto* **THING** *(che denota il concetto più generico possibile) appartiene a* $\mathcal{C}$ *per ogni dominio.*

I concetti sono collegati da una relazione di specificità definita come segue.

Definizione 4.2 *È definita su* $\mathcal{C} \times \mathcal{C}$ *una relazione d'ordine parziale (riflessiva, antisimmetrica e transitiva)* $\preceq$*.*
Dati due concetti c_1,$c_2 \in \mathcal{C}$ *se* $c_1 \preceq c_2$*,* c_1 *viene detto "più specifico di"* c_2*. Per ogni concetto* $c \in \mathcal{C}$*,* $c \preceq$ **THING**.

Esempio 4.1 *In Figura 4.1 è mostrato un esempio di ontologia (creata utilizzando il tool Protégé) che modella il dominio degli* Intrattenimenti per bambini. *La relazione* $\preceq$ *è definita come segue:*

Autore $\preceq$ Persona $\preceq$ **THING**
Intrattenimento $\preceq$ **THING**
Favola $\preceq$ Libro $\preceq$ Intrattenimento

Illustrato $\preceq$ Libro $\preceq$ Intrattenimento
Cartone $\preceq$ Film $\preceq$ Intrattenimento
Animazione $\preceq$ Film $\preceq$ Intrattenimento
Sigla $\preceq$ Musica $\preceq$ Intrattenimento
Zecchino $\preceq$ Musica $\preceq$ Intrattenimento □

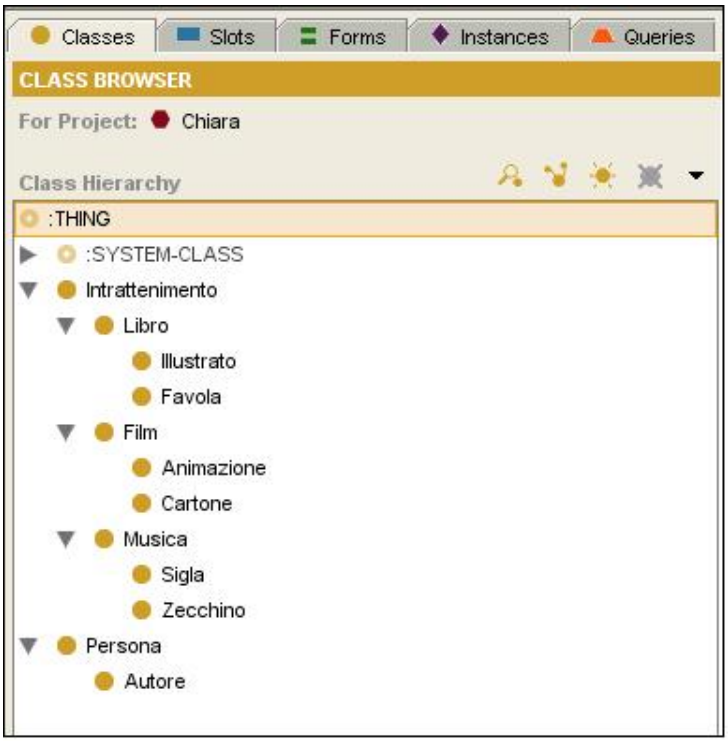


Figura 4.1: *Ontologia degli Intrattenimenti per bambini*

I concetti oltre ad essere organizzati nella gerarchia di specializzazione, sono caratterizzati da proprietà (o attributi). Definiamo innanzitutto la nozione di proprietà.

Definizione 4.3 *Una* proprietà *p è una stringa. L'insieme delle proprietà così definite sarà indicato con Prop.*

Esempio 4.2 *In Figura 4.2 sono mostrate le proprietà del concetto* Favola. □

Ogni concetto è caratterizzato da un insieme di proprietà i cui valori possono essere istanze di un tipo base o di un altro concetto (dominio della proprietà), come formalizzato dalle due definizioni seguenti.

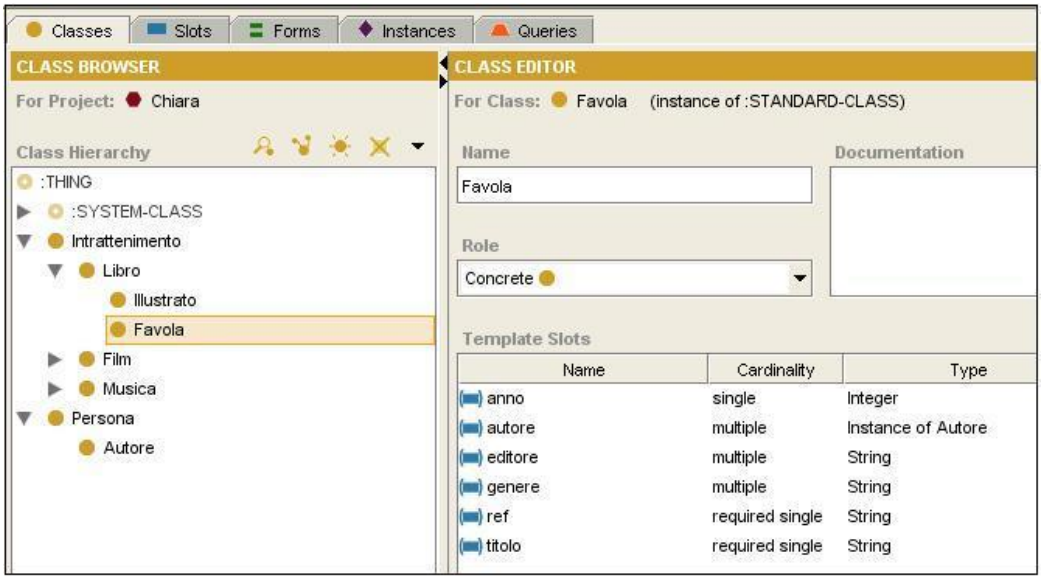


Figura 4.2: *Proprietà del concetto Favola*

Definizione 4.4 *È definita su* $\mathcal{C} \times Prop \times \mathcal{C}$ *una relazione* $\sim_P$ *che vale su* $(c_1, prop, c_2)$ *se prop è una proprietà del concetto* c_1 *i cui valori sono istanze del concetto* c_2*.*

Esempio 4.3 *Si consideri il concetto* Favola *in Figura 4.2. Tale concetto è legato attraverso la proprietà* autore *al concetto* Autore*. La relazione* $\sim_P$ *è pertanto definita come segue:*

$\sim_P$ (Favola, autore, Autore) □

Definizione 4.5 *Sia* $\mathcal{T}$ *l'insieme dei tipi base (interi, reali, booleani, caratteri, stringhe, etc.) utilizzati nell'ontologia. È definita su* $\mathcal{C} \times Prop \times \mathcal{T}$ *una relazione* $\approx_P$ *che vale su* $(c, prop, t)$ *se prop è una proprietà del concetto c i cui valori sono istanze del tipo t.*

Esempio 4.4 *Si consideri nuovamente il concetto* Favola *in Figura 4.2. La relazione* $\approx_P$ *è definita come segue:*

$\approx_P$ (Favola, anno, Integer)
$\approx_P$ (Favola, editore, String)
$\approx_P$ (Favola, genere, String)
$\approx_P$ (Favola, titolo, String) □

Alcuni concetti possono avere associata una proprietà *ref*, con dominio *String*, che indica il riferimento (ad esempio il path o la url) a cui è possibile reperire una risorsa (ad esempio un file) relativa al concetto che possiede tale

proprietà. Inoltre, in generale, una proprietà può essere richiesta come obbligatoria oppure opzionale e può essere definita anche la cardinalità.
Diamo ora la definizione formale di ontologia.

Definizione 4.6 *Un'*ontologia *è un grafo etichettato, orientato e connesso* (V, E)*, con* $E = E^{\preceq} \cup E^{\sim_P} \cup E^{\approx_P}$*, in cui:*

- *i nodi sono etichettati da elementi* $\in \mathcal{C} \cup \mathcal{T}$*;*
  - *sia* $V_{\mathcal{C}}$ *l'insieme dei nodi in* V *con etichette* $\in \mathcal{C}$ *e* $V_{\mathcal{T}}$ *l'insieme dei nodi in* V *con etichette* $\in \mathcal{T}$
- *la relazione d'ordine* $\preceq$ *definisce un sottografo* $(V_{\mathcal{C}}, E^{\preceq})$ *t.c. se* $c_2 \preceq c_1$ *allora* $\exists\ e \in E^{\preceq}$ *t.c.* $c_2 \rightarrow c_1$*; tale sottografo è un albero la cui radice è* **THING***;*
- *la relazione* $\sim_P$ *definisce un sottografo* $(V_{\mathcal{C}}, E^{\sim_P})$ *con archi etichettati su Prop tale che se* $\sim_P (c_1, p, c_2)$ *allora* $\exists\ e \in E^{\sim_P}$ *t.c.* $c_1 \xrightarrow{p} c_2$*;*
- *la relazione* $\approx_P$ *definisce un sottografo* $(V, E^{\approx_P})$ *con archi etichettati su Prop tale che se* $\approx_P (c, p, t)$ *allora* $\exists\ e \in E^{\approx_P}$ *t.c.* $c \xrightarrow{p} t$*.*

La formalizzazione di ontologia è un adattamento di [44] ed è conforme alla metodologia ivi specificata. Definita la nozione di ontologia, definiamo formalmente la nozione di ontologia locale ad ogni peer.

Definizione 4.7 *Sia* $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ *l'ontologia globale come definita in Definizione 4.6. Un'ontologia locale* $\mathcal{L}$ *è un sottografo* $(V_{\mathcal{L}}, E_{\mathcal{L}})$ *di* $\mathcal{G}$ *t.c.*

- $V_{\mathcal{L}} \subseteq V_{\mathcal{G}}$ *è l'insieme dei concetti di* $\mathcal{L}$*,*
- $E_{\mathcal{L}} \subseteq E_{\mathcal{G}}$ *è l'insieme degli archi di* $\mathcal{L}$*,*
- $\mathcal{L}$ *è un'ontologia in accordo alla Definizione 4.6.*

Esempio 4.5 *In Figura 4.3 evidenziata con un rettangolo è mostrata l'ontologia locale* Film *sottoinsieme di quella globale* Intrattenimenti per bambini. □

Tra i concetti è anche possibile definire una relazione semantica di *sinonimia* ($\sim_{SYN}$) cioè la relazione esistente fra due termini che hanno forma differente ma che condividono lo stesso significato (ovvero si definiscono sinonomi due termini che sono sostituibili in alcuni contesti senza modificarne il significato). Tale relazione è simmetrica.

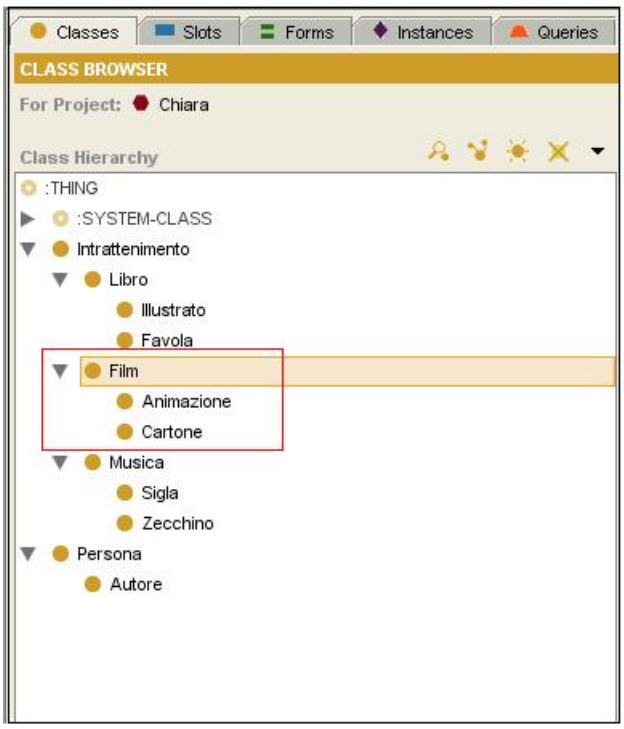


Figura 4.3: *Ontologia locale Film*

Esempio 4.6 *Consideriamo le due frasi:*

"come è grande l'aereo"
"come è grosso l'aereo"

in questo contesto "grande" e "grosso" sono sinonimi (grande $\sim_{SYN}$ *grosso). Se si cambia il contesto in cui vengono utilizzati:*

"Mara è la sorella più grande di Alice"
"Mara è la sorella più grossa di Alice"

questo non è più vero. □

Non abbiamo incluso tale relazione nella definizione di ontologia perché non è significativa nel nostro contesto di riferimento, in cui l'ontologia globale è predefinita e chi la definisce sceglie, tra i vari sinonimi, il termine privilegiato per denotare un concetto, che sarà poi quello usato in tutto il sistema. La nozione di ontologia può, invece, essere estesa per tenere conto di altre relazioni semantiche fra termini (ad esempio *aggregazione* e *part_of*).
Le ontologie all'interno del sistema sono rappresentate come documenti RDFS [7] oppure OWL [47] che possono essere automaticamente generati dagli strumenti per la costruzione di ontologie come quello utilizzato per gli esempi. In Appendice B vengono riportati i file RDFS e OWL corrispondenti all'ontologia di esempio. In questa tesi e per l'esempio di ontologia specificato le rappresentazioni RDFS ed OWL sono equivalenti. In generale, però, questo non è vero. RDFS ed RDF possiedono principalmente primitive riguardanti l'organizzazione gerarchica di un vocabolario: relazioni di sottoclasse e sottoproprietà, restrizioni sui domini e classi di valori ed istanze di classi. Mancano però alcune caratteristiche tra le quali la possibilità di specificare che due classi sono disgiunte, la possibilità di utilizzare le operazioni insiemistiche fra classi per definire una nuova classe in termini di quelle già esistenti oppure la possibilità di definire delle relazioni su e fra le proprietà ad esempio transitività ("maggiore di"), unicità ("madre di") oppure l'inverso di una peroprietà ("mangia" e "mangiato da"). Idealmente OWL è stato sviluppato come un'estensione di RDFS che utilizza lo stesso significato di classe e proprietà di RDFS aggiungendo primitive per supportare ed ottenere una maggiore potenza espressiva.

4.1.1.1 Similarità fra un concetto ed un'ontologia locale

L'assunzione principale da tener presente per il calcolo della funzione di similarità è quella che tutte le ontologie di cui calcoliamo la somiglianza sono

"sotto ontologie" della stessa ontologia globale $\mathcal{G}$. Pertanto, nel calcolo dell'attinenza di un concetto c rispetto ad un'ontologia locale $\mathcal{L}$ dovrà essere tenuto conto anche dell'ontologia $\mathcal{G}$.
Considerando la definizione di ontologia fornita (Definizione 4.6), due delle possibili metriche di attinenza da tenere in considerazione sono quelle indotte dalle definizioni di $\preceq$ che determina la struttura gerarchica dell'ontologia e di $\sim_P$. Pertanto nella definizione della funzione di similarità va tenuto conto della distanza del concetto c da tutti i concetti $c' \in \mathcal{C}_{\mathcal{L}}$ rispetto ai vari tipi di archi presenti in $\mathcal{G}$. Per dare il giusto peso al fatto che a diversi archi corrispondono diverse "distanze concettuali", si introduce la nozione di *"ontologia pesata"*.

Definizione 4.8 *Definiamo l'*ontologia pesata $\mathcal{G}^{\omega}$ *ottenuta dall'ontologia globale* $\mathcal{G}$ *nel modo seguente.*

- *I nodi di* $\mathcal{G}^{\omega}$ *sono gli stessi di* $\mathcal{G}$*;*
- *Sia* $E_{\mathcal{G}^{\omega}}$ *l'insieme degli archi pesati ed etichettati di* $\mathcal{G}^{\omega}$*.* $E_{\mathcal{G}^{\omega}}$ *è definito nel modo seguente:*
  - *per ogni arco* $e_{spec} \in E_{\mathcal{G}}^{\preceq}$*,* $e_{spec} \in E_{\mathcal{G}^{\omega}}$ *ed* $e_{spec^{-1}} \in E_{\mathcal{G}^{\omega}}$*, dove* $e_{spec^{-1}}$ *è l'arco inverso di* e_{spec}[1]*. Gli archi* e_{spec} *sono etichettati con "spec" e indicano che il concetto origine è più specifico del concetto destinazione, mentre gli archi* $e_{spec^{-1}}$ *sono etichettati con "*$spec^{-1}$*" e indicano che il concetto origine è meno specifico del concetto destinazione;*
  - *per ogni arco* $e_{prop} \in E_{\mathcal{G}}^{\sim_P}$*,* $e_{prop} \in E_{\mathcal{G}^{\omega}}$ *ed* $e_{prop^{-1}} \in E_{\mathcal{G}^{\omega}}$*, dove* $e_{prop^{-1}}$ *è l'arco inverso di* e_{prop}*. Gli archi* e_{prop} *sono etichettati con "prop" e indicano che il concetto origine ha come proprietà il concetto destinazione, mentre gli archi* $e_{prop^{-1}}$ *sono etichettati con "*$prop^{-1}$*" e indicano che il concetto origine è una proprietà del concetto destinazione.*
- *Il peso degli archi è determinato da una funzione* $\gamma : E_{\mathcal{G}^{\omega}} \rightarrow [0, 1]$*.*

I pesi degli archi possono essere determinati e fissati una volta per tutte al momento della creazione dell'ontologia da parte dello sviluppatore. Alternativamente è possibile definirli in fase di implementazione e di testing.

Esempio 4.7 *Supponiamo di definire la funzione* γ *come la funzione che associa* 0.8 *a tutti gli archi etichettati con spec e* $spec^{-1}$ *e* 0.5 *a tutti gli archi*

[1] Se $e_{spec} = c_1 \rightarrow c_2$, allora $e_{spec^{-1}} = c_2 \rightarrow c_1$.

etichettati con prop e prop^{-1}. L'ontologia pesata relativa all'ontologia introdotta nell'Esempio 4.1 è mostrata di seguito. Sulle frecce che rappresentano gli archi sono indicati sia il peso sia l'etichetta dell'arco.

THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Libro $\overset{0.8,spec^{-1}}{\longrightarrow}$ Illustrato
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Libro $\overset{0.8,spec}{\longleftarrow}$ Illustrato
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Libro $\overset{0.8,spec^{-1}}{\longrightarrow}$ Favola
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Libro $\overset{0.8,spec}{\longleftarrow}$ Favola
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Film $\overset{0.8,spec^{-1}}{\longrightarrow}$ Animazione
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Film $\overset{0.8,spec}{\longleftarrow}$ Animazione
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Film $\overset{0.8,spec^{-1}}{\longrightarrow}$ Cartone
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Film $\overset{0.8,spec}{\longleftarrow}$ Cartone
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Musica $\overset{0.8,spec^{-1}}{\longrightarrow}$ Zecchino
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Musica $\overset{0.8,spec}{\longleftarrow}$ Zecchino
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Musica $\overset{0.8,spec^{-1}}{\longrightarrow}$ Sigla
THING $\overset{0.8,spec}{\longleftarrow}$ Intrattenimento $\overset{0.8,spec}{\longleftarrow}$ Musica $\overset{0.8,spec}{\longleftarrow}$ Sigla
THING $\overset{0.8,spec^{-1}}{\longrightarrow}$ Persona $\overset{0.8,spec^{-1}}{\longrightarrow}$ Autore
THING $\overset{0.8,spec}{\longleftarrow}$ Persona $\overset{0.8,spec}{\longleftarrow}$ Autore
Libro $\overset{0.5,prop}{\longrightarrow}$ Autore
Libro $\overset{0.5,prop^{-1}}{\longleftarrow}$ Autore
Musica $\overset{0.5,prop}{\longrightarrow}$ Autore
Musica $\overset{0.5,prop^{-1}}{\longleftarrow}$ Autore □

Prima di definre la funzione che misura la similarità fra due concetti è necessario osservare che:

- un concetto è massimamente simile a se stesso;
- ogni arco del grafo fa "perdere" in somiglianza di un fattore pari al peso dell'arco;
- se fra due concetti ho cammini costituiti da più di un arco, ad ogni passo si "perde" il fattore γ relativo ad ogni arco. Per evidenziare il fatto che più due concetti sono lontani minore è la loro somiglianza, si utilizza il prodotto dei pesi degli archi;
- fra tutti i cammini possibili fra due concetti si sceglie quello che penalizza il meno possibile la somiglianza ossia quello il cui prodotto dei pesi è massimo.

Tali osservazioni derivano in parte e sono conformi a quanto fatto in [54]. Definiamo la funzione che misura la similarità di due concetti di un'ontologia globale $\mathcal{G}$ in termini della loro distanza.

Definizione 4.9 *Siano $\mathcal{G}$ l'ontologia globale e $\mathcal{C}_\mathcal{G}$ l'insieme dei suoi concetti. Siano $c, c' \in \mathcal{C}_\mathcal{G}$. Definiamo la funzione che stabilisce la similarità di c e c' rispetto a $\mathcal{G}$ nel modo seguente:*

$$SIM(c, c', \mathcal{G}) = \begin{cases} 1 & \text{se } c = c' \\ \max\{ \prod_{i=1\ldots n-1} \gamma\ (c_i \rightarrow c_{i+1}) \} & \text{altrimenti} \end{cases}$$

dove $c_1 \rightarrow c_2 \rightarrow \ldots c_{i-1} \rightarrow c_i \rightarrow \ldots c_n$ è un cammino di G^ω con $c_1 = c$ e $c_n = c'$.

Esempio 4.8 *Consideriamo l'ontologia introdotta nell'Esempio 4.1 e calcoliamo la similarità fra il concetto* Autore *ed il concetto* Musica *nell'ontologia pesata costruita a partire da quella iniziale come specificato nella Definizione 4.8. Supponiamo di definire la funzione γ come la funzione che associa* 0.8 *a tutti gli archi etichettati con spec e $spec^{-1}$ e* 0.5 *a tutti gli archi etichettati con prop e $prop^{-1}$. Fra i concetti specificati esistono due cammini possibili:*

1. (Autore $\overset{0.8,spec}{\longrightarrow}$ Persona $\overset{0.8,spec}{\longrightarrow}$ **THING** $\overset{0.8,spec^{-1}}{\longrightarrow}$ Intrattenimento $\overset{0.8,spec^{-1}}{\longrightarrow}$ Musica *la cui lunghezza è* $0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 = 0.4096$

2. (Autore $\overset{0.5,prop^{-1}}{\longrightarrow}$ Musica) *la cui lunghezza è* 0.5

Quindi l'affinità fra il concetto Autore *ed il concetto* Musica *è:*
$SIM(\text{Autore}, \text{Musica}, \mathcal{G}^\omega) = \max(0.4096, 0.5) = 0.5$ □

È possibile definire la funzione che misura l'attinenza di un concetto c rispetto ad un'ontologia locale $\mathcal{L}$ sull'ontologia globale $\mathcal{G}$ in termini della funzione appena definita.

Definizione 4.10 *Siano $\mathcal{G}$ l'ontologia globale e $\mathcal{C}_\mathcal{L}$ i concetti dell'ontologia locale $\mathcal{L}$. Sia $c \in \mathcal{C}_\mathcal{G}$. Definiamo la funzione che stabilisce l'attinenza di c rispetto ad $\mathcal{L}$ su $\mathcal{G}$ nel modo seguente:*

$$SCF(c, \mathcal{C}_\mathcal{L}, \mathcal{G}) = \max_{c_i \in \mathcal{C}_\mathcal{L}} \{ SIM(c, c_i, \mathcal{G}) \}$$

In generale un concetto c sarà affine ad un'ontologia $\mathcal{L}$ quanto più la funzione $SCF(c, \mathcal{C}_\mathcal{L}, \mathcal{G})$ sarà vicina al valore 1.

4.1.2 Certificati

I certificati rappresentano un mezzo per controllare l'accesso alle risorse e per subordinare la condivisione di risorse al fatto che l'utente abbia determinate

caratteristiche. Per consentire la specifica di politiche di condivisione basate non solo sull'identità del peer ma anche su alcune sue caratteristiche, ad ogni peer possono venire associati, al momento della registrazione, uno o più *certificati* contenenti le sue qualifiche (credenziali) [68]. Per semplicità non prevediamo la modifica e la gestione della scadenza di un certificato durante il funzionamento del sistema. Altri certificati, diversi da quelli eventualmente specificati in fase di registrazione, possono essere aggiunti in una fase successiva a quella della registrazione presso il sistema. Alcune delle caratteristiche memorizzate in un certificato possono riguardare, ad esempio, l'età dell'utente, la sua nazionalità, l'appartenenza ad un gruppo ed il pagamento di una quota di registrazione presso qualche ente.
Definiamo formalmente il concetto di proprietà che definiscono le caratteristiche di un utente. Tale nozione sarà necessaria alla definizione formale di certificato.

Definizione 4.11 *Siano $\mathcal{N}$ un insieme di nomi di proprietà e $\mathcal{V}$ un insieme di valori che possono essere assunti da tali proprietà. Dati $n \in \mathcal{N}$ e $v \in \mathcal{V}$, una* property *è una una coppia* $(n,\ v)$.

Esempio 4.9 *Si supponga che l'utente sia Samuele Ancona allora*

- $(nome,\ "Samuele")$,
- $(cognome,\ "Ancona")$,
- $(età,\ 3)$,
- $(sesso,\ "maschio")$,
- $(abitazione,\ "Piazza\ Cavour")$

sono proprietà atomiche di tale utente. □

Formalmente un certificato può essere definito come segue.

Definizione 4.12 *Un* certificato *è definito come una coppia:*

$$(n, Properties)$$

dove:

- *n è il nome del certificato;*
- *$Properties$ è un insieme di* property *in accordo alla Definizione 4.11 e rappresenta le proprietà dell'utente cui è riferito il certificato.*

```
( Tessera Disney, { (Nome, "Samuele"),
                    (Cognome, "Ancona"),
                    (Eta, 3),
                    (Abitazione, "Piazza Cavour"),
                    (Nazionalita, "Italiana") ....
                  }
 )
```

Figura 4.4: *Certificato dell'utente Samuele Ancona*

```
<xs:element name = "Certificate">
 <xs:complexType>
  <xs:sequence>
   <xs:element name = "Property" minOccurs = "0"
               maxOccurs = "unbounded">
    <xs:complexType>
     <xs:attribute name = "value" type = "xs:string"/>
     <xs:attribute name = "name" type = "xs:anyType"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name = "name" type = "xs:ID"/>
 </xs:complexType>
</xs:element>
```

Figura 4.5: *XML Schema Certificato*

Esempio 4.10 *Un esempio di certificato può essere la tessera di iscrizione al Disney Club. Consideriamo, nuovamente, l'utente Samuele Ancona. Il certificato associato è mostrato in Figura 4.4.* □

I *certificati* all'interno del sistema sono rappresentati come documenti XML e sono validi rispetto alla specifica XML Schema in Figura 4.5.

Esempio 4.11 *Riprendiamo l'esempio 4.10 utilizzando la specifica XML del certificato. Il documento XML relativo al certificato dell'utente Samuele Ancona è mostrato in Figura 4.6.* □

4.1.3 Politiche

I peer restringono la loro disponibilità a processare una richiesta attraverso l'uso di politiche (*policy*). Le *policy* sono espresse come una congiunzio-

```
<Certificate name = "Tessera Disney">
 <Property name = "Nome" value = "Samuele"/>
 <Property name = "Cognome" value = "Ancona"/>
 <Property name = "Eta" value = "3"/>
 <Property name = "Abitazione" value = "Piazza Cavour"/>
 <Property name = "Nazionalita" value = "Italiana"/>
  ......
</Certificate>
```

Figura 4.6: *Certificato dell'utente Samuele Ancona*

ne booleana di condizioni semplici. Le condizioni semplici possono essere classificate in quattro tipi differenti:

- condizioni *temporali* che specificano il periodo in cui una richiesta può essere ricevuta (esempi di questo tipo sono: "non in orario d'ufficio", "nei giorni lavorativi");
- condizioni sullo *stato interno* del peer ricevente (esempi di questo tipo sono: "se non ci sono più di cinque richieste da processare", "se il tempo di idle della CPU è almeno l'80%");
- condizioni *sulla tipologia di connessione* del peer sulla rete (un esempio è "se la connessione avviene attraverso LAN");
- condizioni sui *certificati* posseduti dal peer richiedente (un esempio è "se il peer richiedente ha pagato una quota di registrazione").

Ogni peer specifica una politica tramite una congiunzione delle condizioni precedenti. Preliminarmente alla definizione delle politiche viene descritta la gestione delle condizioni temporali contenute nelle politiche.

4.1.3.1 Condizioni temporali

Per rappresentare le condizioni temporali delle politiche è necessario un formalismo per denotare i "periodi di tempo". Il formalismo per rappresentare in forma simbolica le *periodic expression* è quello proposto da Niezette e Stevenne in [43] ed è basato sulla nozione di *calendario*.

Definizione 4.13 *Un* calendario *è definito come un insieme numerabile di intervalli consecutivi. Ogni intervallo di un calendario è numerato attraverso un numero intero, chiamato* indice *dell'intervallo, in modo tale che intervalli successivi siano numerati da interi successivi.*

Esempio 4.12 Giorni, Mesi *ed* Anni *sono esempi di calendari che rappresentano, rispettivamente, l'insieme di tutti i giorni, l'insieme di tutti i mesi e l'insieme di tutti gli anni.* □

Nuovi calendari possono essere dinamicamente costruiti da quelli esistenti. Il punto di partenza nella definizione di nuovi calendari è la definizione di un calendario base (il "tick" del sistema), a partire dal quale possono essere dinamicamente definiti nuovi calendari. Si assume che *Ore* sia il calendario base con "tick" indicizzato da 1. Un nuovo calendario C_1 è definito dal calendario esistente C_0 attraverso la funzione *"generate()"* [5].

Definizione 4.14 *Dati due calendari, C_1 e C_2, C_1 è detto* sottocalendario *di C_2 ($C_1 \sqsubseteq C_2$) se ogni intervallo di C_2 è esattamente coperto da un numero finito di intervalli di C_1.*

È facile vedere che la relazione *"sottocalendario di"* definisce un ordine parziale su *calendario* e che, considerando *Ore* come calendario base, $Ore \sqsubseteq C$ per ogni calendario C definito nel sistema. I calendari possono essere combinati per rappresentare insiemi di intervalli temporali più generali, non necessariamente contigui come "l'insieme di tutti i lunedì" oppure "la terza ora del primo giorno di ogni mese". Gli insiemi di intervalli temporali complessi, come quelli appena citati, sono rappresentati attraverso *periodic expression* che descrivono periodi di tempo nel modo seguente.

Definizione 4.15 *Dati i calendari C_d, $C_1, \ldots, C_n$ una* periodic expression *è definita come:*

$$P = \sum_{i=1}^{n} O_i.C_i \rhd r.C_d$$

dove $O_1 = all$, $O_i \in 2^{\mathbb{N}} \cup \{all\}$ e $C_i \sqsubseteq C_{i-1}$ per $i = 2, \ldots, n$, $C_d \sqsubseteq C_n$ e $r \in \mathbb{N}$.

Il simbolo $\rhd$ separa la prima parte dell'espressione, che identifica l'insieme degli intervalli di partenza, dalla specifica della durata di ogni intervallo in termini del calendario C_d.

Esempio 4.13 *La* periodic expression

$$all.Anni + \{3, 7\}.Mesi \rhd 2.Mesi$$

rappresenta l'insieme di intervalli che iniziano nello stesso istante il terzo ed il settimo mese di ogni anno aventi una durata di due mesi. □

Periodic expression	**Significato**
$Settimane + \{2, 6\}_{.}\, Giorni$	I lunedì ed i venerdì
$Mesi + 20.Giorni$	Il 20 di ogni mese
$Anni + 7.Mesi \rhd 3.Mesi$	Mesi estivi
$Settimane + \{2, \ldots, 6\}_{.}\, Giorni$	Giorni lavorativi
$Settimane + \{2, \ldots, 6\}_{.}\, Giorni + 10.Ore \rhd 4.Ore$	Fra le 9.00 e le 13.00 dei giorni lavorativi

Tabella 4.1: *Esempi di espressioni temporali*

O_i è omesso quando il suo valore è *all*, mentre è rappresentato da un singolo elemento quando è unico. $r.C_d$ è omesso quando è uguale a $1.C_n$. La Tabella 4.1 riporta un insieme di *periodic expression* con il loro significato intuitivo. Le *periodic expression* sono rappresentazioni di insiemi infiniti di intervalli temporali. L'insieme degli intervalli temporali corrispondenti alle *periodic expression* è formalizzata dalla funzione $\prod()$ definita come segue.

Definizione 4.16 *Sia*

$$P = \sum_{i=1}^{n} O_i.C_i \rhd r.C_d$$

una periodic expression*, allora* $\prod(P)$ *è un insieme di intervalli la cui durata comune è* $r.C_d$ *e il cui insieme* S *di punti iniziali e calcolato come segue:*

- *se* $n = 1$*,* S *contiene tutti i punti iniziali degli intervalli del calendario* C_1*;*
- *se* $n > 1$ *e* $O_n = \{n_1, \ldots, n_k\}$*, allora* S *contiene i punti iniziali degli intervalli* $n_1^{th}, \ldots, n_k^{th}$ *(tutti gli intervalli se* $O_n = all$*) del calendario* C_n *inclusi in ogni intervallo di*

$$\prod(\sum_{i=1}^{n-1} O_i.C_i \rhd 1.C_{n-1})$$

Esempio 4.14 *Se* P *è l'ultima espressione in Tabella 4.1 allora* $\prod(P)$ *è l'insieme degli intervalli temporali con durata quattro ore che iniziano con la decima ora (dalle 9.00 alle 10.00) del secondo, terzo, quarto, quinto e sesto giorno di ogni settimana.* □

È necessario anche un formalismo simbolico per esprimere i limiti di una *periodic expression*. Un tale formalismo deve garantire che ogni espressione

identifichi un solo istante nel calendario base o, possibilmente, i valori speciali $\pm\infty$. Per definire il formalismo viene scelta la notazione di *data*.

Definizione 4.17 *Una* date expression *ha la forma mm/gg/aa:oo, dove* $mm \in \{1, \ldots, 12\}$, $gg \in \{1, \ldots, 31\}$, $aa \in \{00, \ldots, 99\}$ *e* $oo \in \{01, \ldots, 24\}$.

Una *date expression* denota un "tick" singolo del calendario *Ore* in modo conforme alla semantica intuitiva. Si richiede che *begin* sia una *date expression* ed *end* sia una *date expression* oppure ∞. Siamo ora in grado di definire la nozione di *condizione temporale*.

Definizione 4.18 *Una* condizione temporale *è una coppia* $\langle [\ begin, end\ ], P \rangle$ *dove* P *è una* periodic expression *che denota un insieme infinito di intervalli di tempo e* $[\ begin, end\ ]$ *sono* date expression *che denotano il limite inferiore e superiore imposti sugli intervalli di tempo in* P.

Esempio 4.15 *Supponiamo di voler esprimere il vincolo che per tre anni lo stipendio di un impiegato verrà pagato il 27 di ogni mese. La condizione temporale che esprime questo vincolo è:*

$\langle\ [\ 1/1/05 : 01, 1/1/08 : 01\ ], Mesi + 27.Giorni\ \rangle$ □

4.1.3.2 Politiche e soddisfacibilità

Definiamo ora formalmente il concetto di condizione che definisce la disponibilità di un utente. Tale nozione sarà necessaria alla definizione formale di politica.

Definizione 4.19 *Siano* $\mathcal{N}$ *un insieme di nomi di proprietà,* $\mathcal{V}$ *un insieme di valori che possono essere assunti dalle proprietà e* $op \in \{\leq, \geq, <, >, =\}$. *Definiamo una* condizione *come un'espressione booleana della forma* (n op v).

Formalmente una politica può essere definita come segue.

Definizione 4.20 *Dati* $n \in \mathcal{N}$, $v \in \mathcal{V}$, $op \in \{\leq, \geq, <, >, =\}$, *definiamo una* politica *come una quadrupla:*

$$(id, period, InternalConditions, CertConditions)$$

dove:

- *id rappresenta l'identificatore della politica;*
- *period rappresenta il periodo di validità della politica ed è una* condizione temporale *come definita nella Definizione 4.18;*

$$
\begin{array}{l}
\text{Chiara} = \{(1,\ \langle\,[\,2/1/05:01,\ \infty\,],\ \mathit{Settimane} + \{2,\ldots,6\}.\,\mathit{Giorni}\,\rangle, \\
\qquad\qquad \{(\text{RichiesteInAttesa} \leq 15), \\
\qquad\qquad\ (\text{TempoIdleCPU} < 50\%)\}, \\
\qquad\qquad \{(\text{Eta} \leq 12)\}) \\
\qquad\quad (2,\ \langle\,[\,2/1/05:01,\ \infty\,],\ \mathit{Settimane} + \{1,7\}.\,\mathit{Giorni}\,\rangle,\ \emptyset, \\
\qquad\qquad\ \{(\text{Iscrizione} = \text{"Tessera Disney"})\})\}
\end{array}
$$

Figura 4.7: *Politiche dell'utente Chiara*

- *InternalConditions è un insieme che corrisponde alla congiunzione di* condizioni *come definite nella Definizione 4.19 e rappresenta le condizioni sullo stato interno e sullo stato della connessione del peer;*
- *CertConditions è un insieme che corrisponde alla congiunzione di* condizioni *come definite nella Definizione 4.19 e rappresenta le condizioni sui certificati.*

Esempio 4.16 *Supponiamo che l'utente Chiara desideri specificare che è disponibile a rispondere alle richieste durante i giorni lavorativi, per gli utenti di età inferiore ai 12 anni, se il numero delle richieste in attesa di risposta sono minori o uguali di 15 e se il tempo di inutilizzo della CPU è minore del 50%. Contrariamente, nei week-end è disponibile a rispondere alle richieste degli iscritti al Club Disney. L'insieme delle politiche di Chiara è mostrato in Figura 4.7.* □

Quando le condizioni di una politica sono verificate la richiesta viene processata, altrimenti viene rifiutata. Verificare le condizioni di una politica significa:

- verificare se la condizione temporale è soddisfatta;
- verificare se le condizioni su stato interno e connessione sono soddisfatte;
- verificare se le condizioni sul/i certificato/i posseduto/i dall'utente sono soddisfatte.

Diamo ora la definizione di soddisfacimento della condizione temporale di una politica.

Definizione 4.21 *Sia* $period = \langle[\ begin, end\ ], P\rangle$ *la condizione temporale di una politica* $p = (id, period, InternalCondition, CertConditions)$. *Denotiamo con* now *l'istante (inteso come tempo macchina) in cui viene verificata la politica. Diciamo che la condizione temporale è soddisfatta se*

$$\text{now} \in [\ begin, end\ ] \wedge \text{now} \in \prod(P)$$

Diamo ora la definizione di soddisfacimento delle *CertConditions* di una politica.

Definizione 4.22 *Dati un certificato* $c = (name, Properties)$ *ed una politica* $p = (id, period, InternalConditions, CertConditions)$*, siano* $CertConditions = \{(n_1\ op_1\ v_1), \ldots, (\ n_k\ op_k\ v_k\ )\}$ *e* $Properties = \{(\ nome_1,\ valore_1\ ), \ldots, (\ nome_m,\ valore_m\ )\}$.
Diciamo che un certificato soddisfa una politica se:

$$\forall\ i \in [1,k]\ \exists\ j \in [1,m]\ (\ n_i\ =\ nome_j\ ) \wedge (\ valore_j\ op_i\ v_i\ )$$

Viene omessa la definizione di soddisfacimento delle *InternalConditions* di una politica poiché il suo significato è quello intuituivo.
Mentre è chiara la semantica di un insieme di politiche le cui condizioni temporali si riferiscono a periodi disgiunti, va definito come valutare un insieme di politiche in cui le condizioni temporali hanno periodi con intersezione non vuota.

Esempio 4.17 *Le due condizioni temporali:*

$$Settimane + \{2, \ldots, 6\}_{.}\, Giorni$$
$$Settimane + Giorni + 10.Ore \rhd 4.Ore$$

che esprimono rispettivamente i giorni lavorativi e le ore dalle 9.00 alle 13.00 hanno intersezione non vuota. Nel caso siano presenti due politiche con tali condizioni temporali *va chiarito quale delle due considerare.* □

Non considerando l'ipotesi di valutare staticamente che questo non accada, la soluzione proposta è quella di tener conto dell'ordine in cui le diverse politiche sono specificate. Quindi dato un insieme di politiche:

- si valuta se il tempo corrente *now* soddisfa la *condizione temporale* della prima politica (come specificato nella Definizione 4.21);
- se la condizione precedente è verificata, si valutano le altre condizioni contenute nella prima politica; se sono soddisfatte la politica è soddisfatta e si elabora la richiesta altrimenti si passa alla seconda politica e così via;
- se nessuna politica è soddisfatta la richiesta viene respinta.

```
<xs:element name = "Policies">
 <xs:complexType>
  <xs:sequence minOccurs = "0" maxOccurs = "unbounded">
   <xs:element name = "policy">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref = "TempConstDef"/>
      <xs:element name = "InternalCondition" minOccurs = "0"
                  maxOccurs = "unbounded">
       <xs:complexType>
        <xs:attribute name = "type">
         <xs:simpleType>
          <xs:restriction base = "xs:string">
           <xs:enumeration value = "state"/>
           <xs:enumeration value = "connection"/
          </xs:restriction>
         </xs:simpleType>
        </xs:attribute>
        <xs:attribute name = "onProp" type = "xs:string"/>
        <xs:attribute name = "operation"
                      type = "OperationType"/>
        <xs:attribute name = "value" type = "xs:string"/>
       </xs:complexType>
      </xs:element>
      <xs:element name = "CertCondition">
       <xs:complexType>
        <xs:attribute name = "onProp" type = "xs:string"/>
        <xs:attribute name = "operation"
                      type = "OperationType"/>
        <xs:attribute name = "value" type = "xs:string"/>
       </xs:complexType>
      </xs:element>
     </xs:sequence>
     <xs:attribute name = "id" type = "xs:ID"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name = "name" type = "xs:ID"/>
 </xs:complexType>
</xs:element>
```

Figura 4.8: *XML Schema Politiche*

Le politiche all'interno del sistema sono rappresentate attraverso un documento XML.
Ogni documento è valido rispetto alla specifica XML Schema in Figura 4.8. I valori dell'attributo *type* del tag *InternalCondition* hanno il seguente significato:

- *state* si riferisce ad una condizione sullo stato interno del peer,
- *connection* si riferisce ad una condizione sullo stato della connessione alla rete del peer.

Le strutture di *TempConstDef* ed *OperationType* sono specificate nell'XML Schema globale in Appendice A.

Esempio 4.18 *Riprendiamo l'Esempio 4.16 specificato prima utilizzando la specifica XML delle politiche. Il documento XML relativo alle politiche dell'utente Chiara è mostrato in Figura 4.9.* □

4.1.4 Pubblicità

Gli *adv* vengono utilizzati dai peer per rendere pubbliche le loro competenze e consentono di venire a conoscenza di quali sono i peer nella rete e delle loro competenze e, in particolare, di quali sono quelli con competenze affini.
Una pubblicità deve contenere un identificatore univoco, un sottoinsieme dei concetti dell'ontologia locale che descrivono le competenze del peer e le politiche di disponibilità delle risorse relative a tali concetti che il peer desidera condividere.
Descriviamo ora formalmente la pubblicità di un peer.

Definizione 4.23 *Sia $\mathcal{C}$ l'insieme dei concetti dell'ontologia globale, sia $\mathcal{C}_{\mathcal{L}}$ l'insieme dei concetti dell'ontologia locale $\mathcal{L}$ al peer e sia $C' \subseteq \mathcal{C}_{\mathcal{L}}$ il sottoinsieme dei concetti che rappresentano gli argomenti pubblicizzati dal peer.*
Una pubblicità *è definita come una tripla:*

$$(AdvId, C', Policy)$$

dove:

- *AdvId è l'identificatore univoco della pubblicità;*
- *C' è una lista di concetti relativamente ai quali il peer dichiara di avere risorse da condividere;*

```
<Policies name = "Claudio">
 <policy id = "1">
  <TempConstDef name = "TC1">
    <IntervalExpr  name = "DaFeb1">
      <begin> 2/1/05 </begin>
    </IntervalExpr>
    <PeriodicTimeExpr name = "lavorativi">
      <StartTimeExpr>
       <Year> all </Year>
       <DaySet>
        <Day> 2 </Day> <Day> 3 </Day> <Day> 4 </Day>
        <Day> 5 </Day> <Day> 6 </Day>
       </DaySet>
      </StartTimeExpr>
    </PeriodicTimeExpr>
  </TempConstDef>
  <InternalCondition type = "state"
                     onProp = "Richieste In Attesa"
                     operation = "LE" value = "15"/>
  <InternalCondition type = "state"
                     onProp = "Tempo Idle CPU"
                     operation = "L" value = "50"/>
  <CertCondition onProp = "Eta" operation = "LE"
                 value = "12"/>
 </policy>
 <policy id = "2">
  <TempConstDef name = "TC2">
    <IntervalExpr  name = "DaFeb2">
      <begin> 2/1/05:01 </begin>
    </IntervalExpr>
    <PeriodicTimeExpr name = "weekend">
      <StartTimeExpr>
       <Year> all </Year>
       <DaySet> <Day> 7 </Day> <Day> 1 </Day> </DaySet>
      </StartTimeExpr>
    </PeriodicTimeExpr>
  </TempConstDef>
 <CertCondition onProp = "Iscrizione" operation = "EQ"
                value = "Tessera Disney"/>
 </policy>
</Policies>
```

Figura 4.9: *Politiche dell'utente Chiara*

$$
\begin{array}{l}
(\mathrm{AdvChi1},\ \{\mathrm{Film}\},\\
\qquad \{(1,\ \langle\, [\,2/1/05:01,\ \infty\,],\ \mathit{Settimane} + \{2,\ldots,6\}.\mathit{Giorni}\,\rangle,\\
\qquad\quad \{(\mathrm{RichiesteInAttesa} \leq 15),\\
\qquad\quad\ (\mathrm{TempoIdleCPU} > 0.5)\},\\
\qquad\quad \{(\mathrm{Eta} \leq 12)\})\\
\\
(\mathrm{AdvChi2},\ \{\mathrm{Favola}\},\\
\qquad \{(2,\ \langle\, [\,2/1/05:01,\ \infty\,],\ \mathit{Settimane} + \{1,7\}.\mathit{Giorni}\,\rangle,\ \emptyset)\})
\end{array}
$$

Figura 4.10: *Pubblicità dell'utente Chiara*

- *Policy è l'insieme delle politiche del peer sulla condivisione delle risorse relative ai concetti specificati in C' in accordo alla Definizione 4.20.*

Un peer ha la possibilità di inviare più di un messaggio pubblicitario al fine di diversificare la propria disponibilità in base al tipo di informazioni che è disposto a condividere.

Esempio 4.19 *Supponiamo che l'utente Chiara desideri pubblicizzare il fatto che è esperta di film e voglia condividere tali risorse durante i giorni lavorativi, per gli utenti con età inferiore a 12 anni, se il numero delle richieste in attesa di risposta sono minori o uguali di 15 e se il tempo di inutilizzo della CPU è maggiore del 50%. Supponiamo, inoltre, che desideri anche pubblicizzare che è esperta di favole e voglia condividere tali risorse solo nei week-end. Le pubblicità relative alle competenze dell'utente Chiara sono mostrate in Figura 4.10.* □

Le pubblicità all'interno del sistema sono rappresentate come documenti XML. Ogni documento è valido rispetto alla specifica XML Schema in Figura 4.11. La struttura di *Policies* è stata specificata in Figura 4.8 nella Sezione 4.1.3 mentre lo schema XML che specifica la struttura di *Concept* è riportato in Appendice A.

Esempio 4.20 *Riprendiamo l'esempio 4.19 utilizzando la specifica XML delle pubblicità. I documenti XML relativi alle due pubblicità di Chiara sono mostrati in Figura 4.12 ed in Figura 4.13.* □

Similarità fra una pubblicità ed un'ontologia locale

La verifica dell'affinità fra un'ontologia locale ed i concetti contenuti in una pubblicità segue dalla definizione di affinità fra un concetto ed un'ontologia locale $\mathcal{L}$ rispetto all'ontologia globale $\mathcal{G}$.

```
<xs:element name = "Advs">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref = "Concept" maxOccurs = "unbounded"/>
   <xs:element ref = "Policies"/>
  </xs:sequence>
  <xs:attribute name = "id" type = "xs:ID"/>
 </xs:complexType>
</xs:element>
```

Figura 4.11: *XML Schema Pubblicità*

Definizione 4.24 *Dato l'insieme dei concetti $\mathcal{C}_\mathcal{L}$ appartenenti ad un'ontologia $\mathcal{L}$ ed una pubblicità $a = (AdvId, C', Policy)$ diciamo che a ed $\mathcal{L}$ sono simili se:*

$$\sum_{c_i \in C'} (SCF(c_i, \mathcal{C}_\mathcal{L}, \mathcal{G})) \geq threshold$$

dove threshold è un parametro che dipende dalle ontologie.

4.1.5 Richieste dati

Una richiesta dati deve contenere un identificatore univoco, le condizioni che descrivono il contenuto dell'interrogazione ed il certificato del mittente della richiesta. I certificati, contenenti le qualifiche dell'utente che pone l'interrogazione, vengono utilizzati per decidere se è possibile rispondere alla richiesta. Diamo alcune definizioni ausiliarie utili alla definizione formale di richiesta dati.

Definizione 4.25 *L'insieme delle* path expression *PE rispetto ad un'ontologia definita in modo conforme alla Definizione 4.6 è definito ricorsivamente come segue:*

- *un concetto $c \in \mathcal{C}$ è una* path expression *con origine in c;*
- *se $e \in PE$ è una* path expression *con origine in c_0 che denota un concetto $c \in \mathcal{C}$ e $(c, p, c') \in \sim_P$ allora $(e.p)$ è una* path expression *con origine in c_0 che denota il concetto $c' \in \mathcal{C}$;*
- *se $e \in PE$ è una* path expression *con origine in c_0 che denota un concetto $c \in \mathcal{C}$ e $(c, p, t) \in \approx_P$ allora $(e.p)$ è una* path expression *con origine in c_0 che denota il tipo base $t \in \mathcal{T}$.*

```
<Advs id = "AdvChi1">
 <Concept name = "Film"></Concept>
 <Policies name = "Chiara">
  <policy id = "1">
   <TempConstDef name = "TC1">
    <IntervalExpr  name = "DaFeb1">
      <begin> 2/1/05 </begin>
    </IntervalExpr>
    <PeriodicTimeExpr name = "lavorativi">
      <StartTimeExpr>
       <Year> all </Year>
       <DaySet>
        <Day> 2 </Day>
        <Day> 3 </Day>
        <Day> 4 </Day>
        <Day> 5 </Day>
        <Day> 6 </Day>
       </DaySet>
      </StartTimeExpr>
    </PeriodicTimeExpr>
   </TempConstDef>
   <InternalCondition type = "state"
                      onProp = "Richieste In Attesa"
                      operation = "LE" value = "15"/>
   <InternalCondition type = "state"
                      onProp = "Tempo Idle CPU"
                      operation = "L" value = "50"/>
   <CertCondition onProp = "Eta" operation = "LE"
                  value = "12"/>
   </policy>
 </Policies>
</Advs>
```

Figura 4.12: *Pubblicità dell'utente Chiara*

```
<Advs id = "AdvChi2">
 <Concept name = "Favola"></Concept>
 <Policies name = "Chiara">
  <policy id = "2">
   <TempConstDef name = "TC2">
    <IntervalExpr  name = "DaFeb2">
      <begin> 2/1/05:01 </begin>
    </IntervalExpr>
    <PeriodicTimeExpr name = "weekend">
      <StartTimeExpr>
       <Year> all </Year>
       <DaySet>
        <Day> 7 </Day>
        <Day> 1 </Day>
       </DaySet>
      </StartTimeExpr>
    </PeriodicTimeExpr>
   </TempConstDef>
  </policy>
 </Policies>
</Advs>
```

Figura 4.13: *Pubblicità dell'utente Chiara*

Esempio 4.21 *Esempi di* path expression *in riferimento all'ontologia in Figura 4.1 sono:*

- Libro.autore *con origine in* Libro *che denota il concetto* Autore*;*
- Favola.titolo *con origine in* Favola *e* Libro.autore.cognome *con origine in* Libro *che denotano il tipo base* String. □

I concetti "interrogabili" sono quelli che possiedono la proprietà *ref* ossia quelli a cui è associata una risorsa fisica (ad esempio un file) da poter restituire come risposta ad una richiesta dati.

Definizione 4.26 *Siano PE l'insieme delle path expression,* $\mathcal{V}$ *l'insieme dei valori assunti dalle path expression che denotano un tipo e* $op \in \{\leq, \geq, <, >, =\}$ *un operatore.*
Una queryCondition *è definita come:*
(*pe op v*)
dove

- $v \in \mathcal{V}$;
- *pe è una* path expression *con origine nel concetto interrogabile c che denota un tipo.*

Esempio 4.22 *Un esempio di* queryCondition *in riferimento all'ontologia in Figura 4.1 è:*
Libro.autore.cognome = *"Collodi"* □

Diamo ora la definizione formale di richiesta dati.

Definizione 4.27 *Sia $\mathcal{C}$ l'insieme dei concetti dell'ontologia globale $\mathcal{G}$.*
Una richiesta dati *è definita come una tripla:*

$$(QueryId, Query, Certificates)$$

dove:

- *QueryId è l'identificatore univoco della richiesta;*
- *Query è:*
  - *una* path expression *che denota un concetto interrogabile, oppure*
  - *un insieme di* queryCondition *con la stessa origine come definite nella Definizione 4.26 che vengono interpretate in congiunzione;*
- *Certificates è l'insieme dei certificati specificanti le qualifiche del mittente in accordo alla Definizione 4.12. Tale insieme può anche essere vuoto.*

Il linguaggio di specifica delle *Query* è facilmente estendibile anche alla disgiunzione ed alla negazione. Questi casi non sono trattati perchè esulano dallo scopo principale della tesi.

Esempio 4.23 *Supponiamo che l'utente Viviana desideri le favole di Walt Disney con titolo "La carica dei 101" e "Gli aristogatti", che desideri i libri di Collodi e che desideri tutte le canzoni. Supponiamo infine che Viviana non voglia far sapere niente di sè e non specifichi pertanto alcun certificato. Le richieste dati dell'utente Viviana sono mostrate in Figura 4.14.* □

Le richieste dati all'interno del sistema sono rappresentate come documenti XML e sono valide rispetto all'XML Schema in Figura 4.15. La struttura di *Certificate* è stata specificata in Figura 4.5 nella Sezione 4.1.2 mentre la struttura di *QueryCondition* è specificata nell'XML Schema globale in Appendice A.

Esempio 4.24 *Riprendiamo l'esempio 4.23 utilizzando la specifica XML delle richieste dati. Il documento XML relativo alla richiesta dati di Viviana è mostrato nelle Figure 4.16 e 4.17.* □

```
(QueryViv1, {(Favola.autore.cognome = "Walt Disney"),
             (Favola.titolo = "La carica dei 101")}, ∅)

(QueryViv2, {(Favola.autore.cognome = "Grimm"),
             (Favola.titolo = "Pollicino")}, ∅)

(QueryViv3, {(Libro.autore.cognome = "Collodi")}, ∅)

(QueryViv4, Musica, ∅)
```

Figura 4.14: *Richiesta dati dell'utente Viviana*

```
<xs:element name = "DataRequest">
 <xs:complexType>
  <xs:element ref="Query"/>
  <xs:element ref="Certificate" minOccurs="0"
              maxOccurs="unbounded"/>
  <xs:attribute name = "id" type = "xs:ID"/>
 </xs:complexType>
</xs:element>
```

Figura 4.15: *XML Schema Richiesta dati*

```
<DataRequest id = "QueryViv1">
  <Query>
    <QueryCondition operation = "EQ"
                    value = "Walt Disney">
      <PathExpression>
        <Concept name = "Favola">
         <Property name = "autore"/>
         <Property  name = "cognome"/>
        </Concept>
      </PathExpression>
    </QueryCondition>
    <QueryCondition operation = "EQ"
                    value = "La carica dei 101">
      <PathExpression>
       <Concept name = "Favola">
        <Property  name = "titolo"/>
       </Concept>
      </PathExpression>
    </QueryCondition>
  </Query>
</DataRequest>

<DataRequest id = "QueryViv3">
  <Query>
    <QueryCondition operation = "EQ" value = "Collodi">
      <PathExpression>
        <Concept name = "Libro">
          <Property name = "autore"/>
          <Property  name = "cognome"/>
        </Concept>
      </PathExpression>
    </QueryCondition>
  </Query>
</DataRequest>
```

Figura 4.16: *Richieste dati dell'utente Viviana*

```
<DataRequest id = "QueryViv2">
  <Query>
    <QueryCondition operation = "EQ"
                    value = "Grimm">
      <PathExpression>
        <Concept name = "Favola">
          <Property name = "autore"/>
          <Property  name = "cognome"/>
        </Concept>
      </PathExpression>
    </QueryCondition>
    <QueryCondition operation = "EQ"
                    value = "Pollicino">
      <PathExpression>
        <Concept name = "Favola">
          <Property name = "titolo"/>
        </Concept>
      </PathExpression>
    </QueryCondition>
  </Query>
</DataRequest>

<DataRequest id = "QueryViv4">
  <Query>
    <PathExpression>
      <Concept name = "Musica">
      </Concept>
    </PathExpression>
  </Query>
</DataRequest>
```

Figura 4.17: *Richieste dati dell'utente Viviana*

Similarità fra un'interrogazione ed un'ontologia locale

La verifica dell'affinità fra un'ontologia locale ed i concetti contenuti in una richiesta dati segue dalla definizione di affinità fra un concetto ed un'ontologia locale $\mathcal{L}$ rispetto all'ontologia globale $\mathcal{G}$.

Definizione 4.28 *Sia $\mathcal{C}_{\mathcal{L}}$ l'insieme dei concetti appartenenti ad un'ontologia $\mathcal{L}$, sia $q = (QueryId, Query, Certificates)$ una richiesta dati e sia $QC = \{c_1, \ldots, c_n\}$ l'insieme dei concetti contenuti in Query di q. Diciamo che q ed $\mathcal{L}$ sono simili se:*

$$\sum_{c_i \in QC} (SCF(c_i, \mathcal{C}_{\mathcal{L}}, \mathcal{G})) \geq threshold$$

dove threshold è un parametro che dipende dalle ontologie.

Similarità fra un'interrogazione ed una pubblicità

La verifica dell'affinità fra le competenze contenute in una pubblicità e fra i concetti contenuti in una richiesta dati segue dalla definizione di affinità fra un concetto ed un'ontologia locale $\mathcal{L}$ rispetto all'ontologia globale $\mathcal{G}$.

Definizione 4.29 *Siano $a = (AdvId, C', Policy)$ una pubblicità e $q = (QueryId, Query, Certificates)$ una richiesta dati e sia $QC = \{c_1, \ldots, c_n\}$ l'insieme dei concetti contenuti nelle Query di q. Diciamo che a e q sono simili se:*

$$SLF(QC, C', \mathcal{G}) = \sum_{c_i \in QC} (SCF(c_i, C', \mathcal{G})) \geq threshold$$

dove threshold è un parametro che dipende dalle ontologie.

4.1.6 Base di conoscenza

Una volta registratosi presso il sistema e presa visione dell'ontologia un utente è in grado di creare la sua base di conoscenza. Tale base di conoscenza consiste nella creazione delle istanze dell'ontologia ovvero nel collegamento delle risorse ai concetti che le riguardano. Tale base di conoscenza sarà quella utilizzata dal peer per fornire risposte alle richieste dati. Le istanze dell'ontologia all'interno del sistema sono rappresentate come documenti RDF [33] oppure OWL [47] che possono essere automaticamente generati dagli strumenti per la costruzione di ontologie come quello utilizzato per gli esempi. In Appendice C vengono riportati i file RDF e OWL corrispondenti alle istanze

dell'ontologia di esempio.
Prima di dare la definizione formale di risposta ad una richiesta dati, forniamo alcune definizioni preliminari.

Definizione 4.30 *Data un'ontologia $\mathcal{O}$ ed un concetto $c \in \mathcal{O}$ con proprietà $\{p_1, \ldots, p_n\}$, un'*istanza *$I \in KB$ è un insieme $\subseteq \{(p_i, v_i) |\ i \in [1, n]\}$, dove v_i è il valore associato alla proprietà p_i in modo conforme al dominio specificato per tale proprietà, che contiene tutte le proprietà obbligatorie del concetto c.*

Esempio 4.25 *Si consideri il concetto* Favola *che presenta le seguenti proprietà:* titolo, ref, anno, autore, editore, genere *(come mostrato in Figura 4.2). Di seguito vengono presentate alcune delle possibili istanze del concetto* Favola*:*

1. $I_F = \{$(titolo, $Cenerentola$),
   (ref, $''C : /chiara/condivisi/favole/cenerentola.html''$)$\}$
2. $I_{F1} = \{$(titolo, $Cenerentola$),
   (ref, $''C : /chiara/condivisi/favole/cenerentola.html''$),
   (genere, $''romantico''$)$\}$
3. $I_{F2} = \{$(titolo, $Pollicino$),
   (ref, $''C : /chiara/condivisi/favole/pollicino.html''$),
   (genere, $''avventura''$), (anno, 1918), (autore.cognome, $''Grimm''$)$\}$ □

Definizione 4.31 *La* base di conoscenza *di un peer è l'insieme delle istanze dei concetti appartenenti all'ontologia locale $\mathcal{L}$ come descritto nella Definizione 4.30.*

Definizione 4.32 *Date un'istanza I ed una proprietà p di un concetto c, la notazione $I.p$ denota il valore assunto da I per p.*

Esempio 4.26 *Si consideri l'istanza I_F del concetto* Favola *introdotta nell'Esempio 4.25.*

- I_F.titolo *denota il valore $''Cenerentola''$;*
- I_F.ref *denota il valore*
  $''C : /chiara/condivisi/favole/cenerentola.html''$. □

Definizione 4.33 *Date una* path expression *e ed un'*istanza *I del concetto c origine di e, si definisce la* valutazione *di e rispetto ad I come:*

$$[\![e]\!]_I = \begin{cases} I & \text{se } e = c \\ [\![e']\!]_I.p & \text{se } e = e'.p \end{cases}$$

Esempio 4.27 *Si consideri l'istanza I_{F1} del concetto* Favola *introdotta nell'Esempio 4.25 e si considerino le due* path expression*:*

1. $e =$ Favola
2. $e_1 =$ Favola.genere

Le valutazioni delle due path expression *rispetto all'istanza I_{F1} sono:*

- $[\![e]\!]_{I_{F1}} = [\![\text{Favola}]\!]_{I_{F1}}$ *che restituisce I_{F1};*
- $[\![e_1]\!]_{I_{F1}} = [\![\text{Favola}]\!]_{I_{F1}}$.genere *che restituisce "romantico".* □

Diamo ora la definizione formale di risposta ad una richiesta dati.

Definizione 4.34 *Date la* base di conoscenza KB *ed una* richiesta dati $q = (QueryId,\ Query,\ Certificates)$*, si definisce* risposta *alla richiesta dati q l'insieme delle istanze tali che valgono le seguenti condizioni:*

- *se $Query$ è una* path expression *che denota il concetto interrogabile c, le risposte sono le $I \in KB$ istanze di c;*
- *se $Query$ è l'insieme di queryCondition $\{cond_1, \ldots, cond_n\}$ con origine comune nel concetto interrogabile c e dove ogni $cond_i = e_i\ op_i\ v_i$, allora le risposte sono le $I \in KB$ istanze di c tali che $[\![e_i]\!]_I\ op_i\ v_i = true\ \forall\ i \in [1, n]$.*

Esempio 4.28 *Si considerino le richieste dati:*

- QueryViv2 *con* $Query = \{$(Favola.autore.cognome $=''$ *Grimm*$''$), (Favola.titolo $=''$ *Pollicino*$''$)$\}$
- QueryViv4 *con* $Query = \{$Musica$\}$

introdotte nell'Esempio 4.23.
La risposta *alla richiesta dati* QueryViv2 *è l'istanza I_{F2} introdotta nell'Esempio 4.25.*
La risposta *alla richiesta dati* QueryViv4 *sono, invece, le istanze seguenti:*

- $I_{M1} = \{$(ref, $''C : /chiara/condivisi/musica\_Z/coccodrillo.mp3''$), (titolo, $''Ilcoccodrillocomefa''$), $\ldots\}$
- $I_{M2} = \{$(ref, $''C : /chiara/condivisi/musica\_Z/44gatti.mp3''$), (titolo, $''Quarantaquattrogatti''$), $\ldots\}$

- $I_{M3} = \{(\text{ref}, \textit{"}C : /chiara/condivisi/musica\_Z/valzer.mp3\textit{"}), (\textit{titolo}, \textit{"}Valzerdelmoscerino\textit{"}), \ldots\}$

- $I_{M4} = \{(\text{ref}, \textit{"}C : /chiara/condivisi/musica\_S/harlock.mp3\textit{"}), (\text{titolo}, \textit{"}CapitanHarlock\textit{"}), \ldots\}$

- $I_{M5} = \{(\text{ref}, \textit{"}C : /chiara/condivisi/musica\_S/heidi.mp3\textit{"}), (\text{titolo}, \textit{"}Heidi\textit{"}), \ldots\}$ □

4.2 Messaggi fra peer

Sulla base delle componenti definite nella Sezione 4.1, vengono definiti i messaggi che i peer si scambiano. Tali messaggi possono essere di tre tipi: messaggi pubblicitari che contengono le pubblicità dei peer (Sezione 4.2.1), messaggi di richiesta dati che contengono le interrogazioni poste dai peer (Sezione 4.2.2) e messaggi di risposta alle richieste dati (Sezione 4.2.3). Prima di definire la struttura di questi tipi di messaggi diamo alcune nozioni ausiliarie utili al fine di non sovraccaricare la rete con un traffico eccessivo e per evitare la ritrasmissione infinita dei messaggi attraverso la rete.
I messaggi pubblicitari e di richiesta dati contengono due parametri:

- un campo *TTL (Time To Live)* che specifica la massima distanza fra il mittente del messaggio e l'ultimo ricevente; ogni peer che riceve un messaggio decrementa il *TTL* ad esso associato prima di inoltrarlo. Quando un peer riceve un messaggio con *TTL=0*, lo elimina e non lo inoltra. TTL deve essere un valore numerico intero positivo.

- un campo *BS (Broad Search)* che limita la frazione dei peer riceventi a cui inoltrare il messaggio; se *BS=1*, il messggio è inoltrato a tutti i possibili peer, mentre se *BS=0.5*, il messaggio è inoltrato al 50% dei possibili peer. BS deve essere un valore numerico reale nell'intervallo $[0, 1]$.

4.2.1 Messaggi pubblicitari

Ogni messaggio pubblicitario deve contenere un identificatore univoco, generato concatenando l'identificatore del peer mittente con un identificatore univoco del messaggio, utile per controllare che un messaggio non sia già stato ricevuto, l'identificatore del peer mittente (tipicamente il suo indirizzo IP), la pubblicità del peer mittente definita nella Sezione 4.1.4 ed i campi TTL e BS. Contiene inoltre anche la data in cui viene inviato il messaggio che

consente al peer di conoscere l'"anzianità" di una pubblicità e di decidere, in base alle sue politiche interne, quanto mantenerla e quando eliminarla.

Definizione 4.35 *Un* messaggio pubblicitario *è definito come una n-upla:*

$$(AdvMsgId, PeerId, Adv, TTL, BS, Date)$$

dove:

- *AdvMsgId è l'identificatore univoco del messaggio pubblicitario;*
- *PeerId è l'identificatore univoco del peer mittente del messaggio pubblicitario;*
- *Adv è la pubblicità che esprime le competenze del peer in accordo alla Definizione 4.23;*
- *TTL (Time To Live) è un contatore che specifica la massima distanza fra il mittente del messaggio e l'ultimo ricevente;*
- *BS (Broad Search) è un contatore che specifica la frazione dei peer riceventi a cui inoltrare il messaggio;*
- *Date è la data in cui viene inviato il messaggio pubblicitario.*

Il messaggio pubblicitario all'interno del sistema viene reppresentato come un documento XML ed è valido rispetto alla specifica XML Schema in Figura 4.18. La struttura di *Advs* è stata specificata in Figura 4.11 nella Sezione 4.1.4.

4.2.2 Messaggi di richiesta dati

Una volta registratosi un utente può sottomettere le interrogazioni al sistema che vengono trasmesse ai peer attraverso un messaggio di richiesta dati. Tale messaggio deve contenere un identificatore univoco, generato concatenando l'identificatore del peer mittente con un identificatore univoco della richiesta, utile per controllare che un messaggio non sia già stato ricevuto, l'identificatore del peer mittente della richiesta (tipicamente il suo indirizzo IP), utile al fine di comunicare direttamente al mittente il messaggio di risposta, la richiesta di dati definita nella Sezione 4.1.5 ed i campi TTL e BS.
Un peer può decidere, per ragioni di efficienza e per controllare di non aver già ricevuto una richiesta, di mantenere le interrogazioni ricevute per un certo periodo di tempo. Per questo motivo il messaggio di richiesta dati contiene anche un campo *Date* che specifica la data in cui viene posta l'interrogazione.

```
<xs:element name = "AdvsMsg">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref = "Advs"/>
  </xs:sequence>
  <xs:attribute name = "date" type = "xs:date"/>
  <xs:attribute name = "BS">
    <xs:simpleType>
     <xs:restriction base = "xs:float">
      <xs:minInclusive value = "0"/>
      <xs:maxInclusive value = "1"/>
     </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name = "TTL" type = "xs:integer"/>
  <xs:attribute name = "peerID" type = "xs:string"/>
  <xs:attribute name = "id" type = "xs:ID"/>
 </xs:complexType>
</xs:element>
```

Figura 4.18: *XML Schema messaggio pubblicitario*

Tale data consente al peer di conoscere l'"anzianità" di una richiesta dati e di decidere, in base alle sue politiche interne, quanto mantenerla e quando eliminarla.
Diamo la definizione formale di messaggio di richiesta dati.

Definizione 4.36 *Un* messaggio di richiesta dati *è definito come una n-upla:*

$$(QueryMsgId, PeerId, Query, TTL, BS, Date)$$

dove:

- *QueryMsgId è l'identificatore univoco del messaggio di richiesta di dati;*
- *PeerId è l'identificatore univoco del peer mittente della richiesta;*
- *Query è la richiesta di dati che esprime l'interrogazione del peer in accordo alla Definizione 4.27;*
- *TTL (Time To Live) è un contatore che specifica la massima distanza fra il mittente del messaggio e l'ultimo ricevente;*
- *BS (Broad Search) è un contatore che specifica la frazione dei peer riceventi a cui inoltrare il messaggio;*

```
<xs:element name ="DataRequestMsg">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref = "DataRequest"/>
  </xs:sequence>
  <xs:attribute name = "id" type = "xs:ID"/>
  <xs:attribute name = "date" type = "xs:date"/>
  <xs:attribute name = "BS">
   <xs:simpleType>
    <xs:restriction base = "xs:float">
     <xs:minInclusive value = "0"/>
     <xs:maxInclusive value = "1"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name = "TTL" type = "xs:integer"/>
  <xs:attribute name = "peerID" type = "xs:string"/>
 </xs:complexType>
</xs:element>
```

Figura 4.19: *XML Schema Messaggio di richiesta dati*

- *Date è la data in cui viene inviato il messaggio di richiesta dati.*

Il messaggio di richiesta di dati all'interno del sistema viene reppresentato come un documento XML ed è valido rispetto alla specifica XML Schema in figura 4.19. La struttura di *DataRequest* è stata specificata in Figura 4.15 nella Sezione 4.1.5.

4.2.3 Messaggi di risposta ad una richiesta dati

Ricevuto un messaggio di richiesta dati un peer, se sa rispondere, prepara il messaggio di risposta. Tale messaggio deve contenere un identificatore univoco della risposta, l'identificatore del peer che sta rispondendo (tipicamente il suo indirizzo IP), l'identificatore dell'interrogazione a cui sta rispondendo e l'insieme delle risorse relative alla richiesta dati corrispondenti alla risposta. Diamo la definizione formale di messaggio di risposta ad una richiesta dati.

Definizione 4.37 *Un* messaggio di risposta *è definito come una n-upla:*

$$(AnswerMsgId, PeerId, QueryMsgId, Resource)$$

dove:

```
<xs:element name = "DataResponseMsg">
 <xs:complexType>
  <xs:sequence>
   <xs:element name = "Resource" type = "xs:string"
               maxOccurs = "unbounded"/>
  </xs:sequence>
  <xs:attribute name = "id" type = "xs:ID"/>
  <xs:attribute name = "peer" type = "xs:string"/>
  <xs:attribute name = "queryID" type = "xs:string"/>
 </xs:complexType>
</xs:element>
```

Figura 4.20: *XML Schema Messaggio di risposta ad una richiesta dati*

- *AnswerMsgId è l'identificatore univoco del messaggio di risposta ad una richiesta dati;*
- *PeerId è l'identificatore univoco del peer che fornisce la risposta;*
- *QueryMsgId è l'identificatore univoco del messaggio di richiesta dati;*
- *Resource è l'insieme delle risorse fornite come risposta. Infatti nella creazione delle risposte (Definizione 4.34) non vengono precisamente restituite le istanze ma i valori dell'attributo* ref *di tali istanze (che è sicuramente presente dal momento che il concetto cui le istanze si riferiscono è interrogabile).*

Il messaggio di risposta ad una richiesta dati all'interno del sistema viene reppresentato come un documento XML ed è valido rispetto alla specifica XML Schema in Figura 4.20.

Capitolo 5

Architettura e funzionamento di SINAPSY P2P

Dopo aver introdotto gli elementi principali di SINAPSY P2P in questo capitolo presentiamo le funzionalità del sistema. Il capitolo è strutturato come segue. Nella Sezione 5.1 viene descritta l'architettura dei singoli peer. Nella Sezione 5.2 vengono dettagliate le strutture dati introdotte nella Sezione 5.1 e vengono definite alcune strutture ausiliarie (indici) utili al fine di migliorare l'efficienza degli algoritmi. Nella Sezione 5.3 viene descritta la fase di registrazione di un utente necessaria per poter partecipare al sistema ponendo interrogazioni ed, eventualmente, pubblicizzando le proprie competenze. Nella Sezione 5.4 vengono dettagliati i processi che coinvolgono i messaggi pubblicitari e vengono descritte le funzioni ausiliarie utilizzate dagli algoritmi descritti nelle Sezioni 5.4 e 5.5. Infine, nella Sezione 5.5 è definito il processo di recupero semantico delle risorse.

5.1 Architettura dei peer

L'architettura di ogni peer appartenente al sistema è mostrata in Figura 5.1. Ogni peer registrato (vedi Sezione 5.3) possiede le seguenti componenti funzionali e strutture dati.

- *Routing Engine:* è costituito da due sotto componenti:
  - *Adv Routing Engine* che si occupa dell'invio e dell'instradamento dei messaggi pubblicitari (*adv*) e della selezione dei peer a cui inviarli/inoltrarli;
  - *Query Routing Engine* che si occupa dell'invio e dell'instradamento delle interrogazioni ai peer con competenze affini all'argomento

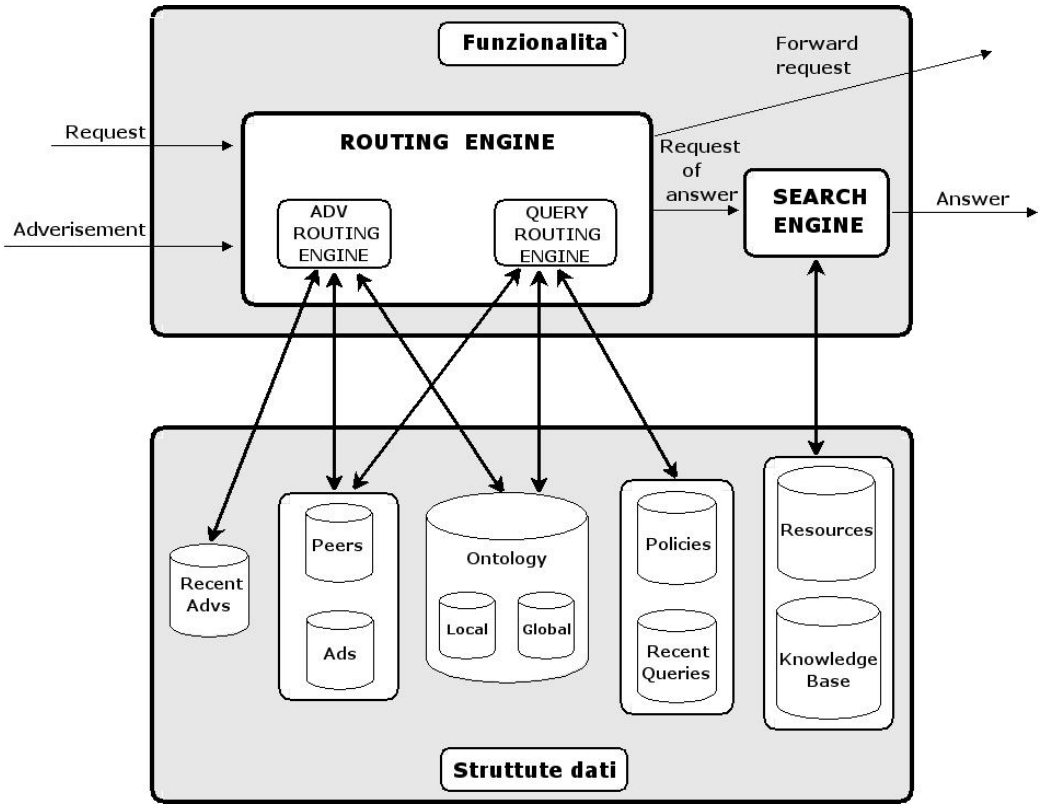


Figura 5.1: *Architettura di un peer*

dell'interrogazione e ad una parte dei peer scelta in modo casuale fra quelli conosciuti.

- *Global Ontology:* rappresenta l'ontologia globale inerente ai concetti rappresentanti il dominio di applicazione del sistema come definita nella Sezione 4.1.1. È un file RDFS o OWL che rappresenta lo schema dell'ontologia globale.

- *Local Ontology:* rappresenta l'ontologia locale, sottoinsieme di quella globale, inerente ai concetti legati alle competenze del peer come definita nella Sezione 4.1.1. È una porzione del file RDFS o OWL contenente lo schema dell'ontologia globale relativo ai concetti dell'ontologia locale;

- *Knowledge Base:* rappresenta la base di conoscenza di un peer ossia l'insieme delle istanze dell'ontologia locale. Dal punto di vista del sistema è rappresentata attraverso un file RDF oppure OWL.

- *Resources:* è la base di dati (collezione) delle risorse che il peer desidera condividere con gli altri peer.

- *Search Engine:* è la componente funzionale che si occupa della ricerca locale delle risorse che soddisfano una richiesta dati come specificato nella Sezione 4.1.6.

- *Peers:* struttura contenente i *PeerID* dei peer conosciuti unitamente ad alcune informazioni riguardanti la loro localizzazione nella rete, il grado di rilevanza che hanno ottenuto nelle risposte alle richieste dati in riferimento ai concetti in esse contenuti.

- *Ads:* è una struttura necessaria alla memorizzazione dei messaggi pubblicitari e dei peer che li hanno inviati. Le informazioni relative ai messaggi pubblicitari consistono nel *PeerID* del mittente, la lista dei concetti contenuti nelle pubblicità ricevute da tale peer unitamente alle politiche di condivisione relative ai concetti pubblicizzati.

- *Recent Queries:* memorizza le interrogazioni che il peer ha ricevuto nell'ultimo periodo. In questa struttura locale vengono memorizzati i messaggi di richiesta dati e la relativa data di ricezione unitamente all'identificatore del peer mittente.

- *Recent Advs:* memorizza i messaggi pubblicitari ricevuti dal peer nell'ultimo periodo. In questa struttura locale vengono memorizzati i messaggi pubblicitari e la relativa data di ricezione unitamente all'identificatore del peer mittente.

- *Policies:* memorizza l'insieme delle politiche che il peer intende seguire per rispondere alle interrogazioni poste da altri peer della rete (vedi Sezione 4.1.3) unitamente ai concetti dell'ontologia locale a cui si riferiscono tali politiche. Come già specificato nel Capitolo 4, infatti, gli utenti hanno la possibilità di diversificare le politiche relative alle risorse condivise.

5.2 Strutture dati

In questa sezione vengono specificate le strutture dati introdotte nella Sezione 5.1 e vengono inoltre presentate alcune strutture ausiliarie (indici) utili al fine di rendere più efficienti gli algoritmi sviluppati.

- Sulla struttura *Global Ontology* vengono costruite le seguenti strutture ausiliarie:
  - *index_spec* della forma (*concetto*, {*concetti*}), indicizzata sui concetti dell'ontologia globale, che associa ad ognuno di essi l'insieme dei concetti diretti più specifici;
  - *index_gen* della forma (*concetto*, *concettoG*), indicizzata sui concetti dell'ontologia globale, che associa ad ognuno di essi il concetto diretto più generale;
  - *index_prop* della forma (*concetto*, {*prop*}), indicizzata sui concetti dell'ontologia globale, che associa ad ogni concetto l'insieme delle sue proprietà;
  - *index_inv_prop* della forma (*prop*, {*concetti*}), indicizzata sulle proprietà dei concetti dell'ontologia globale, che associa ad ognuna di esse l'insieme dei concetti che la possiedono.
- Sulla struttura *Knowledge Base* vengono costruite le seguenti strutture ausiliarie:
  - *index_conc_res* della forma (*concetto*, [*istanze*]), indicizzata sui concetti dell'ontologia locale, che associa ad ognuno di essi le istanze di tale concetto;
  - per ogni concetto c e per le sue proprietà p [1], scelte localmente al peer al momento della creazione della base di conoscenza, viene definito un indice *index_cp_res* della forma (*valore*, [*istanze*]) che associa ad ogni valore di p la lista delle istanze corrispondenti a tale valore.

[1] Sicuramente quelle obbligatorie ad esclusione di *ref*.

- La struttura dati *Peers* è un insieme di triple della forma:

  $$(peerId,\ \{(ril_1, c_1), \ldots, (ril_n, c_n)\},\ rilTot)$$

  dove $peerId$ è l'identificatore del peer che ha fornito le risposte, le coppie (ril_i, c_i) rappresentano la rilevanza acquisita nel rispondere a richieste dati relative al concetto c_i e $rilTot = \sum_{i=1}^{n} ril_i$ è la rilevanza totale acquisita dal peer nel fornire risposte a tutte le richieste dati. Se un peer p_1 non ha fornito alcuna risposta la tripla associatagli sarà $(p_1,\ \emptyset,\ 0)$.
  Sulla struttura *Peers* vengono costruite le seguenti strutture ausiliarie (indici):

  - *index_rel_sim* della forma $((concetto,\ peer),\ rilevanza)$, indicizzata sui concetti e sui peer, che associa ad ogni coppia $(c_i,\ p_i)$ la rilevanza acquisita dal peer p_i relativamente alle risposte fornite per il concetto c_i come specificato nella Sezione 5.5.1;
  - *index_rel* della forma $(rilevanza,\ peer)$, indicizzata ed ordinata in ordine decrescente sulla rilevanza totale acquisita dai peer, che associa i peer alla rilevanza che hanno acquisito nel rispondere a tutte le richieste dati ricevute. Tale indice è ordinato per consentire un recupero efficiente dei peer in base alla loro rilevanza.

- La struttura dati *Ads* è un insieme di coppie della forma:

  $$(peerId,\ \{(c_1, p_1), \ldots, (c_k, p_k)\})$$

  dove $peerId$ è l'identificatore del peer mittente del messaggio pubblicitario e $\{(c_1, p_1), \ldots, (c_k, p_k)\}$ è l'insieme dei concetti contenuti nei messaggi pubblicitari ricevuti da tale peer associati alle politiche relative a tali concetti.
  Sulla struttura *Ads* viene costruita la struttura ausiliaria (indice) *index_adv* della forma $((concetto,\ peer),\ politica)$, indicizzata sui concetti e sui peer, che associa ad ogni coppia (c_i, p_i) le politiche di disponibilità del peer p_i relative alle risorse associate al concetto c_i.

- La struttura dati *Recent Queries* è un insieme di triple della forma:

  $$(QueryId,\ date,\ peerId)$$

  dove $QueryId$ è l'identificatore del messaggio di richiesta dati, *date* è la data di ricezione del messaggio e $peerId$ è l'identificatore del peer mittente del messaggio di richiesta dati.

- La struttura dati *Recent Advs* è un insieme di triple della forma:

  $$(AdvId,\ date,\ peerId)$$

  dove $AdvId$ è l'identificatore del messaggio pubblicitario, *date* è la data di ricezione del messaggio e $peerId$ è l'identificatore del peer mittente del messaggio pubblicitario.

- la struttura dati *Policies* è un insieme di coppie della forma:

  $$([c_1, \ldots, c_k],\ policy)$$

  dove $[c_1, \ldots, c_k]$ è la lista dei concetti relativamente ai quali si applicano le politiche di disponibilità specificate da *policy* (documento XML descritto nella Sezione 4.1.3).
  Sulla struttura *Policies* viene costruita la struttura ausiliaria (indice) *index_conc_enable* della forma (*concetto*, *politica*), indicizzata sui concetti, che associa ad ognuno di essi le politiche di disponibilità che il peer applica sulle risorse associate a tale concetto.

5.3 Fase di registrazione

Quando un utente desidera partecipare al sistema per la prima volta si collega al sito conosciuto dove gli viene chiesto di registrarsi. In modo simile a come avviene in Gnutella, in tale sito risiede un "peer speciale", conosciuto e sempre disponibile, al quale i peer possono sempre connettersi e che interagisce nella fase di registrazione rendendo disponibile l'interfaccia di SINAPSY P2P e l'ontologia globale rappresentante il dominio di interesse del sistema. Poiché il sistema SINAPSY P2P prevede che i peer possano connettersi e disconnettersi dinamicamente dalla rete uno, dei compiti del "peer speciale", che non fornisce la condivisione di risorse, è quello di memorizzare i peer registrati e mettere a disposizione le loro informazioni(ad esempio l'indirizzo IP e la porta). Tali peer rappresentano i primi peer di cui l'utente (ed il rispettivo peer) viene a conoscenza e che verranno memorizzati come conosciuti nella struttura *Peers* (con entry iniziale $(peerID, \emptyset, 0)$).
Una volta che l'utente è collegato al sistema può procedere nell'eventuale creazione dei *certificati* contenenti alcune delle sue caratteristiche (qualifiche, credenziali) utili al fine di controllare l'accesso alle risorse condivise. Tali certificati sono creati attraverso il "peer speciale" e vengono rappresentati nel sistema come documenti XML memorizzati localmente al peer. La fase di creazione dei certificati non è obbligatoria e un peer che non possiede certificati o non vuole divulgarli (tutti o parte di essi) per una questione di

privacy, può decidere di non crearli senza precludersi la possibilità di registrarsi e partecipare al sistema. Inoltre è possibile che un utente acquisisca un certificato in una fase successiva a quella di registrazione; in tal caso si collegherà al sistema e farà precisa richiesta di creazione del suo nuovo certificato. In questo processo il "peer speciale" gioca il ruolo di "ente certificatore" della correttezza ed autenticità dei certificati creati [2].
Nella fase successiva all'utente appare l'interfaccia grafica contenente l'ontologia globale che può essere navigata al fine di visualizzare la descrizione testuale di ogni concetto ivi contenuto. Così facendo l'utente può determinare quali concetti meglio descrivono il contenuto delle risorse che desidera condividere. A questo punto l'utente può decidere di creare la sua ontologia locale e di istanziarla con le sue risorse oppure può decidere di fare queste operazioni off-line in un momento qualunque successivo alla fase di registrazione. In modo indipendente dal momento in cui tali operazioni verranno eseguite, il peer scaricherà dal "peer speciale" sia l'ontologia globale sia gli indici costruiti su di essa. L'operazione di istanziazione dell'ontologia locale dell'utente corrisponde alla creazione della base di conoscenza del peer. Per creare tale base di conoscenza l'utente deve selezionare i concetti corrispondenti agli argomenti delle sue risorse, associare tali risorse ai concetti selezionati fornendo i valori per le eventuali proprietà di tali concetti. Conseguenza di queste operazioni è la creazione di un file RDF oppure OWL contenente le istanze dell'ontologia locale in modo conforme a quanto specificato nella Sezione 4.1.6.
L'utente può a questo punto specificare le sue politiche di disponibilità di condivisione e il tipo di utenti abilitati ad accedere alle risorse da lui messe a disposizione. Come nel caso dei certificati, le politiche relative ai concetti corrispondenti alle competenze del peer vengono rappresentate nel sistema come un documento XML memorizzato localmente al peer.
Completata la fase di registrazione il peer è in grado di pubblicizzare le sue competenze e sottomettere interrogazioni al sistema in modo conforme a quanto verrà discusso nelle prossime sezioni.
Può accadere che al momento della registrazione non vi sia alcun peer registrato. In questo caso la base di dati dei peer conosciuti (*Peers*) dall'utente che si sta registrando è vuota e non è possibile per lui inviare alcun messaggio pubblicitario o sottomettere interrogazioni immediatamente dopo la fase di registrazione. Al momento del primo collegamento al sito, successivo alla

[2]Questa assunzione si basa sul fatto che non c'è uno standard di rappresentazione dei certificati per cui il "peer speciale" o riceve alcuni certificati rilasciati da diversi enti e li rende conformi alla rappresentazione interna al sistema o li crea direttamente. Se esistesse uno standard di rappresentazione dei certificati forniti da un ente esterno al sistema, il "peer speciale" potrebbe non essere coinvolto nella fase di creazione dei certificati.

Funzione	**Descrizione**
`|S|`	restuisce la cardinalità di `S`
`Extract_BS(F, S)`	estrae il valore dell'attributo `BS` dal file XML `F`, controlla qual è la percentuale specificata da tale valore rispetto a tutti i peer contenuti in `S` (`|S|`), trasforma tale percentuale in un intero e lo restituisce
`IsIn(Id, S)`	controlla se l'identificatore univoco `Id` è contenuto nella struttura `S`; restituisce un valore booleano che vale `TRUE` se `Id` è contenuto in `S` e `FALSE` altrimenti
`Send(F, peerID)`	invia il file XML `F` al peer con identificatore `peerID`
`Update_Peer_Database()`	richiede al "peer speciale" l'elenco dei peer registrati, controlla quali sono quelli che non appartengono alla struttura *Peers* e li inserisce in tale struttura

Tabella 5.1: *Funzioni ausiliarie*

fase di registrazione, il peer richiederà al "peer speciale" gli indirizzi dei peer attualmente registrati. A questo punto sarà in grado di pubblicizzare le sue competenze ed inviare le sue richieste alla rete dei peer.

5.4 Diffusione e ricezione di pubblicità

In questa sezione presentiamo il meccanismo di pubblicizzazione delle competenze di ogni peer. Diamo quindi le specifiche di funzionamento dell'*Adv Routing Engine*. Nella Sezione 5.4.1 viene analizzato il processo di invio dei messaggi pubblicitari inerenti le competenze del peer e la scelta dei peer ai quali trasmetterli. Nella Sezione 5.4.2 descriviamo i processi di ricezione, memorizzazione ed inoltro dei messaggi pubblicitari. Nella Tabella 5.1 vengono, inoltre, introdotte le funzioni ausiliarie utilizzate dagli algoritmi coinvolti nei processi di invio, ricezione ed inoltro sia delle pubblicità sia delle richieste dati che verranno descritti nel seguito del capitolo. Negli algoritmi faremo, inoltre, riferimento alle strutture dati *Ads*, *Peers*, *Policies*, *Recent Queries* e *Recent Advs* introdotte nella Sezione 5.2.
In Tabella 5.2 sono elencate le funzioni utilizzate dai processi di invio, ricezione ed inoltro di un messaggio pubblicitario. Il parametro `A` rappresenta il file XML, descritto nella Sezione 4.2.1, che corrisponde al messaggio pubblicitario.

Funzione	**Descrizione**
`Update_Ads(peerID, A)`	estrapola da `A` i concetti e le relative politiche ed aggiorna l'insieme associato al peer `peerID` nella struttura *Ads* con le nuove coppie estratte $(c_1, p_1), \ldots, (c_n, p_n)$ dove i p_i non sono necessariamente diversi gli uni dagli altri o da quelli precedenti mentre i c_i sono tutti diversi
`Insert_Ads(peerID, A)`	inserisce nella struttura *Ads* il peer con identificatore `peerID` e gli associa l'insieme delle coppie $(c_1, p_1), \ldots, (c_n, p_n)$ di concetti e politiche corrispondenti estrapolati da `A` dove i p_i non sono necessariamente diversi gli uni dagli altri mentre i c_i sono tutti diversi
`Extract_conceptsA(A)`	estrae dal messaggio pubblicitario `A` i concetti in esso contenuti (attributo *name* dei tag *Concept*) e li restituisce
`Extract_Similar(concepts)`	restituisce i `peerID` estratti dalla base di dati *Ads*, tramite l'utilizzo dell'indice *index_adv* creato su tale struttura, dei peer con competenze simili alla lista dei concetti specificata da `concepts`; la similarità è calcolata in modo conforme alla definizione data nella Sezione 4.1.4
`Random(S, n)`	restituisce una lista lunga `n` di *peerID* estratti in modo casuale fra 1 e $\|S\|$ contenuti nella struttura `S`

Tabella 5.2: *Funzioni dell'Adv Routing Engine*

5.4.1 Diffusione di pubblicità

Un peer che voglia rendere pubbliche le sue competenze invia uno o più messaggi pubblicitari (file XML con formato descritto nella Sezione 4.2) ai peer conosciuti (residenti nella base di dati *Peers* locale al peer). In tali messaggi potranno essere incluse le politiche (file XML con formato descritto nella Sezione 4.1.3) riguardanti la disponibilità del peer a condividere risorse ed i peer abilitati ad usufruire di tali risorse. In generale, poiché non tutte le risorse del peer dovranno sottostare alle stesse politiche, sarà possibile inviare differenti messaggi pubblicitari contenenti ciascuno le politiche relative al sottoinsieme delle competenze del peer in essi pubblicizzate (e quindi ad un sottoinsieme delle risorse condivise). Se il peer sarà invece sempre disponibile a condividere le sue risorse e non porrà limiti sugli utenti abilitati all'accesso, non specificherà alcuna politica. Il processo di diffusione delle competenze del peer può avvenire in qualunque momento. Una volta estratti gli indirizzi

```
procedure SendAdv(A)
 var i: integer
 begin
   if(Peers ≠ ∅) then
      for i := 1 to |Peers| do
         Send(A, Peers[i].peerID)
   else
     Update_Peer_Database()
     if(Peers ≠ ∅) then
       SendAdv(A)
 end
```

Figura 5.2: *Algoritmo di invio di un messaggio pubblicitario A*

dei peer dalla base di dati dei peer conosciuti (*Peers*), il messaggio pubblicitario creato verrà inoltrato a tutti i peer ivi contenuti. Il meccanismo di invio delle pubblicità è quello che consente ai peer di venire a conoscenza delle competenza degli altri peer della rete e sarà utile al momento di scegliere a chi porre le richieste dati e guidare il processo di instradamento (*routing*). Se al momento della diffusione dei messaggi pubblicitari il peer non fosse a conoscenza di nessuno degli altri peer della rete, potrà collegarsi al sistema e fare esplicita richiesta dei peer registrati al "peer speciale". Il motivo per cui si è scelto di inviare i messaggi pubblicitari a tutti i peer conosciuti è quello di creare maggiori possibilità di ricevere messaggi dagli altri peer e di venire pertanto a conoscenza del maggior numero di peer possibili a cui poter inviare le interrogazioni.
In Figura 5.2 viene descritto l'algoritmo in pseudocodice corrispondente all'invio di un messaggio pubblicitario.

5.4.2 Ricezione ed inoltro pubblicità

Un peer che riceve un messaggio pubblicitario contenente, eventualmente, le politiche del peer mittente controlla di non aver già ricevuto tale messaggio confrontando l'*AdvId* in esso contenuto con la base di dati dei *Recent Advs* (Figura 5.1). Se l'*AdvId* compare nella base di dati il messaggio viene eliminato e non inoltrato altrimenti il suo *AdvId* viene inserito, assieme alla data di ricezione e al *perId* del mittente, nella base di dati dei *Recent Advs*. A questo punto il peer esegue le azioni elencate di seguito.

1. **Ricezione di un messaggio pubblicitario.**
   In questa fase il peer:

- estrae i concetti (corrispondenti alle competenze del peer mittente) contenuti nel messaggio;
- controlla nella base di dati *Ads* il *peerId* del mittente per verificare di non aver già ricevuto una pubblicità da quel peer;
  - se il *peerId* non compare nella base di dati *Ads*, controlla se è presente nella base di dati *Peers* ed, in caso negativo, lo aggiunge a tale struttura ed aggiunge alla base di dati *Ads*, oltre al *peerId* del mittente, i concetti contenuti nel messaggio pubblicitario unitamente alle politiche applicate a tali concetti;
  - se, invece, il *peerId* compare nella base di dati *Ads* aggiorna solamente tale base di dati aggiungendo i concetti contenuti nell messaggio pubblicitario unitamente alle politiche applicate a tali concetti alla lista di quelli precedenti.

I messaggi pubblicitari ricevuti dal peer vengono tutti memorizzati. Se l'utente riterrà di non essere interessato ad alcuni di essi potrà rimuoverli dalla base di dati delle pubblicità in ogni momento attraverso l'interfaccia grafica del sistema. Inoltre, l'utente potrà decidere di ordinare i messaggi pubblicitari in base all'affinità semantica fra le competenze in essi pubblicizzate e le proprie. Se un utente è interessato solo ai messaggi pubblicitari riguardanti argomenti affini alle sue competenze potrà eliminare gli altri.

2. **Inoltro di un messaggio pubblicitario.** Dopo aver provveduto alla memorizzazione del messaggio pubblicitario, tale messaggio viene inoltrato in modo conforme ai campi TTL e BS in esso contenuti. Il numero di peer a cui inoltrare il messaggio è specificato dal campo BS mentre il livello di profondità a cui si vuole arrivare è specificato dal campo TTL che viene decrementato di 1 da ogni peer che riceve il messaggio. I peer che ricevono un *adv* con $TTL=0$ non ritrasmettono il messaggio ma si limitano a memorizzarlo. Si è scelto per semplicità di non decrementare anche il parametro BS. Inoltre, l'inoltro è parzialmente mirato ossia privilegia i peer con competenze affini a quelle contenute nel messaggio.

In Figura 5.3, 5.4 e 5.5 vengono riportati gli algoritmi in pseudocodice per la ricezione, la memorizzazione e l'inoltro di un messaggio pubblicitario.

```
procedure ReceiveAdv(A)
 begin
      StoreAdv(A)
      ForwardAdv(A)
 end
```

Figura 5.3: *Algoritmo di ricezione di un messaggio pubblicitario A*

```
procedure StoreAdv(A)
 begin
   if(IsIn(A.id, RecentAdvs)) then return
   else
     RecentAdvs := RecentAdvs ∪ (A.id, A.date, A.peerID)
     if(IsIn(A.peerID, Ads))
          Update_Ads(A.peerID, A)
     else
       if(¬IsIn(A.peerID, Peer))
         Peers := Peers ∪ (A.peerID, ∅, 0)
       Insert_Ads(A.peerID, A)
 end
```

Figura 5.4: *Algoritmo di memorizzazione di un messaggio pubblicitario A*

```
procedure ForwardAdv(A)
 var BS: integer
 var simPeers, similarPeers, otherPeers: array of peerID
 var concepts: array of string
 begin
   if(A.TTL ≠ 0) then
     A.TTL := A.TTL − 1
     concepts := Extract_conceptsA(A)
     simPeers := Extract_Similar(concepts)
     BS := Extract_BS(A, Peers)
     similarPeers := Random(simPeers, Fraction(BS))
     otherPeers:= Random(Peers, BS–Fraction(BS))
     for i=1 to |similarPeers|
       Send(A, similarPeers[i].peerID)
     for j=1 to |otherPeers|
       Send(A, otherPeers[j].peerID)
   else return
 end
```

Figura 5.5: *Algoritmo di inoltro di un messaggio pubblicitario A*

5.5 Recupero semantico di risorse

In questa sezione presentiamo i meccanismi di instradamento semantico (*routing*) e valutazione delle richieste (*query*) sui peer. Diamo quindi le specifiche di funzionamento del *Query Routing Engine* e del *Search Engine*. Nella Sezione 5.5.1 viene specificato il calcolo del grado di rilevanza associato ad un peer e viene descritto il processo di valutazione delle risposte alle interrogazioni sul peer. Nella Sezione 5.5.2 vengono analizzati i criteri di invio, ricezione ed inoltro delle interrogazioni e di scelta dei peer ai quali trasmetterle e vengono forniti gli algoritmi descriventi tali processi. Nelle Tabelle 5.3 e 5.4 sono elencate le funzioni utilizzate dai processi di invio, ricezione ed inoltro di una richiesta dati. Il parametro `Q` rappresenta il file XML, descritto nella Sezione 4.2.2, che corrisponde al messaggio di richiesta dati.

5.5.1 Rilevanza

Quando l'utente riceve le risorse come risposta ad una richiesta dati, le valuta specificando quali sono attinenti e quali no e può eventualmente fornire valutazioni aggiuntive su di esse.
Le valutazioni aggiuntive che l'utente fornisce rispetto alle risorse ottenute

Funzione	**Descrizione**
`Extract_conceptQ(Q)`	estrae dal messaggio di richiesta dati `Q` le condizioni della richiesta dati, estrapola da esse i concetti coinvolti in tale richiesta e li restituisce
`Relevance_Order()`	utilizza l'indice *index_rel* su *Peers*, ordina i peer in base alla loro rilevanza totale e restituisce i `peerID`
`Relevance_Similar_Order(concepts)`	utilizza l'indice *index_rel_sim* su *Peers* per ottenere i peer con competenze simili a quelle contenute in `concepts` utilizzando la funzione di similarità definita nella Sezione 4.1.5 e li restituisce ordinati in base alla relazione d'ordine $\leq_{concepts}$ definita nella Sezione 5.5.1 Definizione 5.3
`Extract_AdsPeers(concepts)`	utilizza l'indice *index_adv* su *Ads* per ottenere i `peerID` dei peer con concetti simili a `concepts` in modo conforme alla funzione di similarità definita nella Sezione 4.1.5 e che siano disponibili a ricevere un'interrogazione secondo le politiche di disponibilità associate ai loro messaggi pubblicitari
`Enable_Policy(Q, S)`	estrapola dalla richiesta dati `Q` gli eventuali certificati dell'utente, controlla quali peer nella struttura `S` sono disponibili a ricevere una richiesta (Sezione 4.1.3.2) utilizzando l'indice *index_adv* su *Ads* e restituisce la lista di tali peer
`Check_Policy(Q)`	utilizza l'indice *index_conc_enable* su *Policies* per ottenere le politiche associate ai concetti contenuti nella richiesta dati e controlla se il certificato del peer mittente (contenuto in `Q`) soddisfa tali politiche (in modo conforme a quanto specificato nella Sezione 4.1.3.2); restituisce un valore booleano che vale `TRUE` se le politiche consentono al peer di valutare la richiesta ed il certificato soddisfa le politiche e vale `FALSE` altrimenti
`Compare_Expertise(Q)`	confronta i concetti contenuti nella richiesta dati `Q` con i concetti appartenenti all'ontologia locale del peer ricevente (corrispondente alla sue competenze) in modo conforme alla funzione di similarità descitta nella Sezione 4.1.5; ritorna `TRUE` se le competenze sono simili e `FALSE` altrimenti
`Answer_query(peerId, resources)`	invia l'insieme delle risorse `resources` al peer con identificatore `peerId`

Tabella 5.3: *Funzioni dell'Query Routing Engine*

Funzione	Descrizione
`Create_Answer(Q)`	valuta le condizioni contenute in `Q`, estrae i concetti ed i valori delle proprietà dalle condizioni e attraverso gli indici *index_arg_res* e *index_cp_res* (rispettivamente sui concetti e sui valori delle proprietà associate ai concetti) su *Resources* recupera le risorse possedute dal peer ricevente come specificato nella Sezione 4.1.6

Tabella 5.4: *Funzioni del Search Engine*

contribuiscono al calcolo del grado di rilevanza da associare al peer che la ha fornite e possono essere diverse in riferimento a concetti diversi. In generale nel calcolo del grado di rilevanza tali valutazioni peseranno, comunque, meno rispetto a quelle relative al numero di risorse attinenti alla richiesta dati rispetto al totale di quelle ottenute. Le valutazioni aggiuntive fornite dall'utente sono rappresentate attraverso il parametro β che sarà specificato nella Definizione 5.1.
A seguito delle valutazioni dell'utente mittente della richiesta, il peer calcola per ogni risposta ottenuta e per ogni concetto contenuto nella richiesta dati il grado di rilevanza da associare al peer fornitore della risposta nella base di dati locale *Peers* come specificato nella Definizione 5.1.
Il grado di rilevanza acquisito da un peer a seguito delle risposte a richieste dati è influenzato da diversi fattori:

- dal rapporto di quante risorse l'utente che riceve la risposta accetta rispetto al totale di quelle restituite;
- dalle eventuali valutazioni (rappresentate dal parametro β) che l'utente può fornire sulla risposta ottenuta (sia positive, corrispondenti ad un valora di β prossimo o uguale ad 1, sia negative, corrispondenti ad una valore di β prossimo o uguale a 0).

In base ai fattori specificati è possibile definire il grado di rilevanza associato ad un peer rispetto alla risposta ad una richiesta dati.

Definizione 5.1 *Sia q una richiesta dati inviata dal peer p e sia $\{c_1, \ldots, c_n\}$ l'insieme dei concetti contenuti in q. $\forall c_i \in \{c_1, \ldots, c_n\}$ definiamo il* grado di rilevanza *associato dal peer p al peer p', fornitore di una risposta per q, come:*

$$Relevance_{(p',c_i)} = \frac{risorse\ memorizzate}{risorse\ ricevute} + \beta$$

dove:

- $\frac{risorse\ memorizzate}{risorse\ ricevute}$ *rappresenta la misura di quante sono le risorse che il peer p giudica rilevanti rispetto a tutte quelle ottenute dal peer* p'*; più tale rapporto si avvicina ad 1 e più le risorse ottenute sono attinenti alla richiesta dati di p;*

- $\beta \in [0,1]$ *è un fattore non prevedibile che corrisponde ad eventuali valutazioni aggiuntive fornite dall'utente e può essere diverso per concetti diversi; in generale* $\beta \leq \frac{risorse\ memorizzate}{risorse\ ricevute}$ *e* $\beta + \frac{risorse\ memorizzate}{risorse\ ricevute} \leq 1$

$Relevance_{(p',c_i)}$ *viene associato al peer* p' *per il concetto* c_i *nella struttura* Peers *e nel relativo indice* index_rel_sim*.*

Il calcolo e l'aggiornamento del grado di rilevanza associato ad un peer mittente della risposta ad una richiesta dati sono definiti nel modo seguente.

Definizione 5.2 *Sia q una richiesta dati inviata dal peer p e sia* $\{c'_1, \ldots, c'_m\}$ *l'insieme dei concetti contenuti in q. Sia* p' *un peer che fornisce una risposta per q. Si distinguono due casi.*

1. *Se esiste nella struttura dati* Peers *di p la tripla* $(p', R, rilTot)$*, con* $R = \{(ril_1, c_1), \ldots (ril_n, c_n)\}$*, l'aggiornamento del* grado di rilevanza *associato da p al peer* p' *per la risposta q è specificato nel modo seguente:*

   (a) *se* $\forall\ j \in [1,m]\ \exists\ i \in [1,n]$ *tale che* $c'_j = c_i$ *allora* $ril_i = ril_i + Relevance_{(p',c_i)}$ *e* $rilTot = rilTot + Relevance_{(p',c_i)}$*,*

   (b) *altrimenti* $R = R \cup \{(Relevance_{(p',c'_j)}, c'_j)\}$ *e* $rilTot = rilTot + Relevance_{(p',c_i)}$

2. *Se non esiste nella struttura dati* Peers *di p una tripla per* p'*, viene creata ed inserita in* Peers *la seguente tripla:*

$$(p', \{(ril'_1, c'_1) \ldots (ril'_m, c'_m)\}, rilTot')$$

con $rilTot' = \sum_{j=1}^{m} ril'_j$ *e dove ogni* $ril'_j = Relevance_{(p',c'_j)}$*.*

Esempio 5.1 *Consiederiamo l'ontologia introdotta nela Capitolo 4. Sia* p_1 *un peer che ha inviato una* richiesta dati *q contenente i concetti* {Film, Zecchino, Musica}*. Sia* p_2 *un peer che ha fornito una* risposta *per q ottenendo rilevanza pari a:*

- *0.6 rispetto al concetto* Film*,*

- *1 rispetto al concetto* Zecchino*,*

- *0.7 rispetto al concetto* Musica.

Consideriamo prima il caso in cui $(p_2, \{(0.5, \text{Musica}), (0.8, \text{Favola}), (0.9, \text{Sigla}), (1, \text{Zecchino})\}, 3.2) \in$ Peers. *In tal caso l'aggiornamento della rilevanza associata a* p_2 *risulta essere:*

$(p_2, \{(1.2, \text{Musica}), (0.8, \text{Favola}), (0.9, \text{Sigla}), (2, \text{Zecchino}), (0.6, \text{Film})\}, 8.7)$

Se, invece, $p_2 \notin$ Peers, *l'inserimento della tripla assocaita a* p_2 *risulta essere:*

$(p_2, \{(0.7, \text{Musica}), (1, \text{Zecchino}), (0.7, \text{Film})\}, 2.4)$ □

I peer appartenenti alla struttura *Peers* possono essere ordinati per rilevanza rispetto ad un insieme di concetti.

Definizione 5.3 *Sia* Peers $= \{(p_1, \{(ril_{11}, c_{11}), \ldots, (ril_{1s}, c_{1s})\}, rilTot_1), \ldots, (p_n, \{(ril_{n1}, c_{n1}), \ldots, (ril_{nt}, c_{nt})\}, rilTot_n)\}$ *e sia* $Concepts$ *un insieme di concetti. Si definisce la funzione che per ogni peer calcola la sua rilevanza rispetto ai concetti contenuti in* $Concepts$ *nel modo seguente:*

$$\begin{aligned} Rel(p_i, Concepts) &= \sum_{c \in Concepts \text{ t.c. per qualche } j,\ c = c_{ij}} ril_{ij} \\ &+ \alpha^d \cdot \sum_{c \in Concepts \preceq c' \text{ t.c. per qualche } k,\ c' = c_{ik}} ril_{ik} \end{aligned}$$

con $\alpha \in [0, 1]$ *e* d *distanza fra* c *e* c' *nella gerarchia. Per ottenere* $c \preceq c'$ *si utilizza iterativamente l'indice* index_gen *cha associa ad ogni concetto il concetto diretto più generale.*
Si definisce la relazione d'ordine $\leq_{concepts}$ *sui peer* $\in$ Peers *nel modo seguente:*

$$p_i \leq_{concepts} p_j \text{ sse } Rel(p_i, Concepts) \leq Rel(p_j, Concepts)$$

Esempio 5.2 *Consideriamo un sottoinsieme di* Peer *formato dai seguenti tre peer:*

- $(p_1, \{(0.5, \text{Favola}), (0.8, \text{Sigla}), (3, \text{Zecchino}), (1, \text{Musica})\}, 5.3)$
- $(p_2, \{(1.5, \text{Film}), (0.3, \text{Libro}), (2, \text{Animazione}), 3.8)$
- $(p_3, \{(0.8, \text{Libro}), (1.5, \text{Musica}), 2.3)$

e consideriamo l'insieme di concetti $C = \{\text{Libro}, \text{Zecchino}\}$. *Consideriamo per ogni peer la funzione che calcola la rilevanza rispetto a tali concetti:*

- $Rel(p_1,$ Libro$) = 0$ *poiché tra i concetti contenuti nelle risposte fornite precedentemente da* p_1*, non compare né* Libro *né nessun concetto più generale di* Libro*;*
  $Rel(p_1,$ Zecchino$) = 3 + \alpha^d$.
  Supponendo di fissare $\alpha = 1$ *e supponendo che il peso dell'arco tra Zecchino e Musica sia* 0.8*,* $Rel(p_1,$ Zecchino$) = 3 + 1^{0.8} = 4$*;*

- $Rel(p_2,$ Libro$) = 0.3$ *;*
  $Rel(p_2,$ Zecchino$) = 0$ *poiché tra i concetti contenuti nelle risposte fornite precedentemente da* p_1*, non compare né* Zecchino *né nessun concetto più generale di* Zecchino*;*

- $Rel(p_3,$ Libro$) = 0.8$ *;*
  $Rel(p_3,$ Zecchino$) = 1.3 \cdot \alpha^d$.
  Supponendo di fissare $\alpha = 1$ *e supponendo che il peso dell'arco tra Zecchino e Musica sia* 0.8*,* $Rel(p_3,$ Zecchino$) = 1.3 + 1^{0.8} = 2.3$.

Traendo le somme dei conti appena svolti si deriva che:

- $Rel(p_1,\ C) = 4$,
- $Rel(p_2,\ C) = 0.3$,
- $Rel(p_3,\ C) = 3.1$.

Per definizione di $\leq_{concepts}$*, l'ordinamento dei peer* $p_1,\ p_2,\ p_3$ *rispetto* C *è:*
$p_2 \leq_{concepts} p_3 \leq_{concepts} p_1$ □

5.5.2 Instradamento semantico delle interrogazioni

Il sistema proposto adotta un instradamento semantico delle richieste dati in quanto nell'invio delle richieste privilegia, ove possibile, i peer con competenze affini a quelle dei concetti contenuti nell'interrogazione. Se non si ha alcuna conoscenza di peer con competenze affini a quelle dell'interrogazione, il meccanismo di scelta dei peer a cui inoltrare la richiesta dati è casuale. In entrambe le situazioni proposte è, comunque, tenuto in considerazione il grado di rilevanza che i peer hanno acquisito nelle precedenti risposte. Nell'inviare una richiesta dati si fanno infatti le seguenti assunzioni [63]:

- una richiesta dati è posta ad un peer che si presuppone sappia rispondere (quindi di cui si conoscono le competenze) e che abbia risposto "bene" alle richieste dati precedenti sull'argomento dell'interrogazione attuale;

- un'assunzione generale è quella che se un utente (e quindi il peer corrispondente) è ben informato su un dominio specifico sarà, probabilmente, ben informato anche su un dominio più generale riguardante lo stesso argomento;

- in generale le persone possono essere più o meno esperte in modo indipendente dal domino ossia si presuppone che un utente (e quindi il peer corrispondente) che abbia acquisito un grado di rilevanza alto in più di un argomento sia probabilmente una persona colta in generale e, pertanto, da privilegiare nel caso non si conoscano esperti nell'argomento della richiesta dati.

5.5.2.1 Invio di un messaggio di richiesta dati

Quando un peer desidera sottomettere una richiesta dati alla rete dei peer (file XML con formato descritto nella Sezione 4.2.2), seleziona dalla base di dati dei messaggi pubblicitari (*Ads* in Figura 5.1) gli indirizzi dei peer con competenze affini a quelle della richiesta dati tenendo conto del grado di rilevanza acquisito da tali peer nelle precedenti risposte e considerando, inoltre, le loro politiche di disponibilità. Se il peer mittente non è a conoscenza di nessun peer con competenze affini a quelle della richiesta dati, l'invio è guidato solo dal grado di rilevanza associato ai peer. Unitamente ad una richiesta, il peer può decidere di inviare anche i suoi certificati (file XML con formato descritto nella Sezione 4.1.2) che rappresentano le sue qualifiche e che saranno verificati dai peer riceventi per constatare se l'utente è abilitato o meno a ricevere le risposte. Se al momento dell'invio di una richiesta dati il peer non fosse a conoscenza di nessuno degli altri peer della rete, potrà collegarsi al sistema e fare esplicita richiesta dei peer registrati al "peer speciale". Ottenuti gli indirizzi dei peer la richiesta dati viene inviata ed il peer si pone in attesa dei messaggi di risposta con annessi i relativi documenti.
Nelle Figure 5.6 e 5.7 viene presentato l'algoritmo in pseudocodice corrispondente all'invio di una richiesta dati. Il parametro `max_p` che determina il massimo numero di peer a cui inviare la richiesta dati viene fissato a priori e verrà determinato in fase di implementazione. Al fine di chiarificare il funzionamento dell'algoritmo di selezione dei peer a cui inviare una richiesta dati viene illustrato di seguito un esempio.

Esempio 5.3 *Siano:*

- enabledPeers *l'insieme dei peer* $\{p_7, p_9, p_{17}, p_2, p_4, p_{12}, p_{41}, p_{19}\}$,

- enabledsimilarPeers *l'insieme dei peer* $\{p_9, p_2, p_4, p_{19}, p_{21}, p_5\}$,

```
procedure SendQuery(Q, max_p)
 var i, j, dim_p: integer
 var relevancePeers, similarPeers: array of peerID
 begin
   if(Peers ≠ ∅) then
     similarPeers :=
             Relevance_Similar_Order(Extract_conceptsQ(Q))
     relevancePeers := Relevance_Order(Q)
     if(Ads = ∅) then
       for i := 1 to |similarPeers| do
         Send(Q, similarPeers[i].peerID)
       if(|similarPeers| < max_p)
         dim_p := max_p - |similarPeers|
         for j := 1 to dim_p do
           Send(Q, relevancePeers[j].peerID)
     else
       Send_Similar_Peers(Q, max_p)
   else
     Update_Peer_Database()
     if(Peers ≠ ∅) then
       SendQuery(Q, max_p)
 end
```

Figura 5.6: *Algoritmo di invio di una richiesta dati*

- enabledPeers1 *l'insieme dei peer* $\{p_7, p_{21}, p_{30}, p_{12}, p_5, p_6, p_1, p_{20}\}$,

e sia max_p = *12.*
L'algoritmo seleziona nell'ordine i seguenti insiemi di peer:

1. $\{p_9, p_2, p_4, p_{19}\}$ *sono gli* enabledPeers $\in$ enabledsimilarPeers
2. $\{p_7, p_{17}, p_{12}, p_{41}\}$ *sono gli* enabledPeers $\notin$ enabledsimilarPeers
3. $\{p_{21}, p_5\}$ *sono gli* enabledsimilarPeers $\notin$ enabledPeers
4. $\{p_{30}, p_6\}$ *sono gli* enabledPeers1 $\notin$ enabledPeers $\cup$ enabledsimilarPeers *ordinati per rilevanza totale.*

5.5.2.2 Ricezione ed inoltro delle richieste di dati

Un peer che riceve un messaggio di richiesta dati e gli eventuali certificati del peer mittente controlla di non aver già ricevuto tale messaggio confrontando

```
procedure Send_Similar_Peers(Q, num_p)
 var i, i1, i2, i3, j,
     num_p1, num_p2, np, np1, np2, rest: integer
 var relevancePeers, similarPeers, enabledsimilarPeers,
     enabledPeers, enabledPeers1: array of peerID
 begin
  similarPeers :=
            Relevance_Similar_Order(Extract_conceptsQ(Q))
  enabledPeers := Enable_Policy(Q, similarPeers)
  relevancePeers := Relevance_Order(Q)
  enabledsimilarPeers :=
                  Extract_AdsPeers(Extract_conceptsQ(Q))
  i, i1, i2, i3 := 1
  np, np1, np2 := 0
  while(num_p > np ∨ i ≤ |enabledPeers|)
    if(enabledPeers[i] ∈ enabledsimilarPeers) then
      Send(Q, enabledPeers[i].peerID)
      np++
    i++
  if(num_p > np) then
    num_p1 := num_p − np
    while(num_p1 > np1 ∨ i1 ≤ |enabledPeers|)
      if(enabledPeers[i1] ∉ enabledsimilarPeers)
        Send(Q, enabledPeers[i1].peerID)
        np1++
      i1++
    if(num_p1 > np1) then
      num_p2 := num_p1 − np1
      while(num_p2 > np2 ∨ i2 ≤ |enabledsimilarPeers|)
        if(enabledsimilarPeers[i2] ∉ enabledPeers) then
          Send(Q, enabledsimilarPeers[i2].peerID)
          np2++
        i2++
      if(num_p2 > np2)
        rest := num_p − np − np1 − np2
        enabledPeers1 := [peerID ∈
                    Enable_Policy(Q, relevancePeers) ∧
                 ∉ (enabledPeers ∪ enabledsimilarPeers)]
        while(rest ≥ i3 ∧ i3 ≤ |enabledPeers1|)
          Send(Q, enabledPeers1[i3].peerID)
          i3++
 end
```

Figura 5.7: *Algoritmo di selezione dei peer a cui inviare una richiesta dati*

il *QueryId* in esso contenuto con la base di dati delle *Recent Queries* (Figura 5.1). Se il *QueryId* compare nella base di dati la richiesta viene eliminata altrimenti viene valutata ed inoltrata. Una volta stabilito di non aver ricevuto il messaggio, inserisce la richiesta dati ricevuta nella base di dati delle *Recent Queries* assieme alla sua data di ricezione e applica le sue politiche di disponibilità per verificare se è in grado di rispondere alla richiesta e se l'utente ha le caratteristiche necessarie affinché gli possa essere restituita una risposta. Solo se l'applicazione delle sue politiche glielo consente valuta effettivamente la richiesta altrimenti la inoltra solamente. A questo punto il peer esegue le azioni elencate di seguito.

1. **Ricezione di un messaggio di richiesta dati.** In questa fase il peer:

   - valuta la richiesta dati controllando quali sono sono i concetti contenuti in tale messaggio;
   - se le sue competenze sono affini a quelle della richiesta (come specificato nella Sezione 4.1.5) e se le sue politiche glielo consentono:
     - recupera le risorse dalla sua base di dati *Resources*;
     - invia le risorse al peer mittente;
     - inoltra la richiesta dati come specificato al punto 2.

     I peer che possiedono una risposta, comunicano direttamente con il peer mittente della richiesta grazie al *PeerId* (del mittente) contenuto nella richiesta dati.
   - se le sue competenze non sono affini ai concetti contenuti nella richiesta dati e, pertanto, non è in grado di rispondere si limita ad inoltrare la richiesta dati come specificato al punto 2.

2. **Inoltro di un messaggio di richiesta dati.** Fornita o meno una risposta alla richiesta dati il peer provvede all'inoltro del messaggio.

   - inoltra la richiesta ai peer con competenze affini ai concetti contenuti nella richiesta dati (se ne conosce) in base alle politiche di disponibilità e al grado di rilevanza posseduto da tali peer;
   - inoltra la richiesta anche ad un sottoinsieme di peer scelti in modo casuale fra quelli con competenze non affini in base alle politiche di disponibilità e al grado di rilevanza posseduto da tali peer.

   L'inoltro avviene in modo conforme ai parametri TTL e BS specificati nel messaggio. Il numero di peer a cui inoltrare il messaggio è specificato dal campo *BS* mentre il livello di profondità a cui si vuole arrivare

```
procedure ReceiveQuery(Q)
 var resources: array of file
 begin
  if(IsIn(Q.id, RecentQueries)) then return
  else
    RecentQueries := RecentQueries ∪
                     (Q.id, Q.date, Q.peerID)
    if(¬Check_Policy(Q)) then return
    else
      if(Compare_Expertise(Q)) then
        resources := Create_Answer(Q.peerID)
        Answer_query(Q.peerID, resources)
      ForwardQuery(Q)
 end
```

Figura 5.8: *Algoritmo di ricezione di una richiesta dati*

```
procedure ForwardQuery(Q)
 var BS: integer
 begin
   if(Q.TTL ≠ 0) then
     BS := Extract_BS(Q, Peer)
     Q.TTL := Q.TTL − 1
     SendQuery(Q, BS)
 end
```

Figura 5.9: *Algoritmo di inoltro di una richiesta dati*

è specificato dal campo *TTL* che viene decrementato di 1 da ogni peer che riceve il messaggio. I peer che ricevono una richiesta con *TTL=0* non ritrasmettono il messaggio.

In Figura 5.8 e 5.9 vengono riportati gli algoritmi in pseudocodice per la ricezione e l'inoltro di un messaggio di richiesta dati.

Conclusioni e sviluppi futuri

In questa tesi è stato presentato SINAPSY P2P un sistema basato su ontologie per il recupero di informazioni in una rete P2P che utilizza un meccanismo di pubblicizzazione delle competenze dei peer e che tiene conto delle politiche di disponibilità imposte dai peer e dei certificati dei peer che sottomettono una richiesta.

Con l'avvento di Internet l'ammontare delle risorse disponibili è cresciuto drasticamente questo ha reso difficile il ritrovamento, l'accesso, la presentazione ed il mantenimento delle informazioni utilizzate da una grande quantità di utenti. Le tecniche usate nelle basi di dati distribuite, sviluppate per gestire i dati distribuiti su un numero limitato di server, hanno raggiunto i loro limiti nel contesto di Internet non supportando la scalabilità.

È quindi stata sviluppata una nuova classe di sistemi distribuiti, i sistemi Peer to Peer (P2P) [1], la cui essenza è basata sul fatto che i nodi nella rete sfruttano direttamente le risorse presenti su altri nodi della rete senza l'intervento di un server centrale.
Tali sistemi consentono di supportare migliaia di nodi fornendo soluzioni interessanti dal punto di vista della scalabilità.

Tuttavia, le soluzioni P2P odierne, come Napster [40] e Gnutella [22], supportano solo funzionalità limitate per quanto concerne la modifica, la ricerca ed il ritovamento delle risorse sui peer basate per lo più sul *matching* fra parole chiave.
Malgrado l'*Information Retrieval (IR)* sia stato progettato come una delle maggiori applicazioni dei protocolli P2P, questi difetti rendono gli attuali sistemi non utilizzabili per scopi di condivisione della conoscenza. In quasi tutti i tipi di architetture P2P la risoluzione delle interrogazioni è basata su algoritmi di *flooding* che consentono la propagazione da nodo a nodo ma tale principio ha qualche inconveniente [14] poiché trasmettere tutte le interrogazioni a tutti i peer non rende tali architetture scalabili.
È pertanto richiesto un instradamento dell'interrogazione intelligente che

selezioni un sottoinsieme di peer rilevanti per una data richiesta.

Oltre al problema dell'instradamento semantico delle richieste, i sistemi attuali ed i nuovi proposti dipendono dall'assunzione che quando i peer sono connessi alla rete sono sempre disponibili a rispondere alle interrogazioni da chiunque esse provengano. Tuttavia, i peer possono desiderare di configurare delle politiche sulla loro disponibilità.

Il contributo di questa tesi è stato la proposta di un sistema con le seguenti caratteristiche:

- integrazione delle nozioni di certificato e politica per la realizzazione di meccanismi di condivisione delle risorse più flessibili in un sistema P2P;
- instradamento semantico delle interrogazioni basato sull'utilizzo di ontologie. L'instradamento è guidato sia dalla pubblicizzazione delle competenze dei peer sia dalla rilevanza delle risposte fornite dai peer a richieste precedenti.

Del sistema proposto è stata dettagliata l'architettura generale e dei singoli peer, sono state specificate le strutture dati e definiti gli algoritmi in pseudocodice per l'invio, l'inoltro, la ricezione e la memorizzazione sia di un messaggio pubblicitario sia di una richiesta dati. Sono state definite formalmente le nozioni base per il funzionamento del sistema (ontologia, certificati, politiche, pubblicità, richieste dati, base di conoscenza e risposte a richieste dati) e ne è stata proposta una specifica in XML.

Il lavoro svolto offre molte oppotunità e spunti per una ricerca futura, tra cui segnaliamo:

- Creazione di un prototipo del sistema SINAPSY P2P basato sulla specifica elaborata in questa tesi.
- Rilassamento del vincolo imposto sulla corrispondenza fra l'IP ed il peer al fine di consentire una maggiore flessibilità nell'interazione con il sistema P2P. Può essere abbandonato il vincolo di staticità degli IP in favore di un modello più astratto che preveda la possibilità di creare, inviare ed autenticare profili in modo da poter riconoscere univocamente un utente e quindi il peer ad esso associato indipendentemente dall'informazione resa dal solo IP.

- Possibilità di allegare alle pubblicità meta-informazioni riguardanti il contenuto della base di conoscenza dei peer. In tal modo sarà possibile condividere anche dati, oltre che risorse, per consentire che le informazioni sui valori delle proprietà delle istanze possano essere scambiate fra i peer. Questo permette agli utenti che condividono nelle loro istanze le stesse proprietà di non inserire tali proprietà ma di riferirsi a quelle di un altro peer.

- Rilassamento delle condizioni contenute nella richiesta dati (interrogazione) nel caso in cui il peer ricevente non disponga, nella sua base di conoscenza, di alcune delle proprietà opzionali inserite nell'interrogazione. Tale meccanismo consiste nel valutare e fornire risposte a richieste dati considerando solo le condizioni sulle proprietà opzionali di cui il peer dispone (eventualmente nessuna).

- Estensione del linguaggio di formulazione delle richiesta dati.

- Associazione dei certificati (credenziali) anche alle risposte a richieste dati in modo da consentire al peer ricevente di specificare politiche sull'accettazione di tali risposte.

Bibliografia

[1] K. Aberer and M. Hauswirth. Peer-to-Peer Information Systems: Concepts and Models, state-of-the-art and Future Systems. In *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-9)*, pages 326–327. ACM Press, 2001.

[2] K. Aberer et al. Emergent Semantics Principles and Issues. In *Proceedings of 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, LNCS(2973), Jeju Island, Korea, 2004. Springer. `www.ipsi.fraunhofer.de/~risse/pub/`.

[3] F. Baader and W. Nutt. Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 2, pages 43–95. Cambridge University Press, 2003.

[4] T. Berners-Lee. *Weaving the Web*. London: Orion Business, 1999.

[5] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. *ACM Transactions on Information and Systems*, Vol. 23(3):231–285, September 1998.

[6] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, Vol. 7(13):422–426, 1970.

[7] D. Brickley and R. Guha. Resource Description Framework (RDF) Schema Specification 1.0 (candidate raccomandation). World Wide Web Consortium. Available from: `http://www.w3.org/TR/rdf-schema/`.

[8] J. Broekstra and A. Kampman. Serql: An RDF Query and Trasformation Language. In *International Semantic Web Conference (ISWC)*, 2004. `http://www.openrdf.org/doc/SeRQLmanual.html`.

[9] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *Proceedings of International Conference on Distributed Computing Systems*, Vienna, Austria, 2002.

[10] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical report, 2002.
`http://www-db.stanford.edu/~crespo/publications`.

[11] F.M. Cuenca-Acuna and T.D. Nguyen. Text-Based Content Search and Retrieval in ad hoc P2P Communities. In *International Workshop on Peer-to-Peer Computing*. Springer-Verlag, May 2002.

[12] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan. Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*. Schloss Elmau, Germany, May 2001.

[13] DAML. The DAML home page. `http://www.daml.org`.

[14] N. Daswani, B. Garcia-Molina, and B. Yang. Open Prpblems in Data-Sharing Peer-to-Peer Systems. In *Proceedings of the 9th International Conference on Database Theory (ICDT)*, 2003.

[15] M. Ehring, P. Haase, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer, and C. Tempich. The SWAP data and metadata model for semantics-based peer-to-peer systems. In *Multiagent Systems Technologies*, LNCS(2831), pages 144–155. Springer, 2004.

[16] D. Elenius and M. Ingmarsson. Ontology-based Service Discovery in P2P Networks. In *International Workshop on Peer-to-Peer Knowledge Management P2PKM*, 2004.

[17] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Berlin: Springer-Verlag, 2001.

[18] D. Fensel, J. Angele, S. Decker, M. Erdmann, H. P. Schnurr, R. Studer, and A. Witt. Lessons Learned from Applying AI to the Web. *Journal of Cooperative Information Systems*, Vol. 9(4):361–382, 2000.

[19] D. Fensel, I. Horrocks, F. Van Harmelen, D. McGuiness, and P. Patel-Schneider. OIL: Ontology Infrastructure to Enable the Semantic Web. *IEEE Intelligent Systems*, pages 38–45, March/April.
`http://www.ontoknowledge.org/oil/oilhome.shtml`.

[20] The Fourth IEEE International Conference on Peer-to Peer Computing, 2002. http://femto.org/p2p2004/.

[21] FREENET. The FreeNet home page, 2001. http://www.freenetproject.org.

[22] GNUTELLA. The GNUTELLA home page, 2001. http://www.gnutella.wego.com.

[23] W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, and M. A. Musen. Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000). In *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW-1999)*, 1999. http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-1999-0801.html.

[24] T. R. Gruber. A traslation approach to portable ontology specification. *Knowledge Acquisition*, Vol. 5(2):199–220, 1993.

[25] P. Haase, R. Siebes, and F. van Harmelen. Peer Selection in Peer-to-Peer Networks with Semantic Topologies. In *International Conference on Semantics of a Networked World (ICSNW)*, Paris, 2004.

[26] J. Hendler. Agent and the Semantic Web. *IEEE Intelligent Systems*, Vol. 16(2):30–37, 2001.

[27] I. Horrocks and P. F. Petel-Schneider. Optimizing Description Logic Subsumption. *Journal of Logic and Computation*, Vol. 9(3):267–293, 1999.

[28] S. Joseph. Neurogrid: Semantically Routing Queries in Peer-to-Peer Networks. In *In Proceedings of the International Workshop on Peer-to-Peer Computing*, 2002.

[29] JXTA. The JXTA home page, 2001. http://WWW.jxta.org.

[30] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In *In 11th International Conference on Information and Knowledge Management (CIKM'2002)*, November 4-9 2002.

[31] KAZAA. The KaZaA home page, 2001. http://www.kazaa.com.

[32] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, R. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.

[33] O. Lassila. Web Metadata: A Matter of Semantics. *IEEE Internet Computing*, Vol. 2(4):30–37, 1998.

[34] Z. Lu and K. S. McKinley. *The Effect of Collection Organization and Query Locality on Information Retrieval System Performance and Design*, chapter in Advances in Information Retrieval. Bruce Croft Editor, 2000.

[35] N. Lumineau and A. Doucet. Sharing Communities Experiences for Query Propagation in Peer-to-Peer Systems. In *International Database Engineering and Applications Symposium (IDEAS '04)*, pages 246–253, 2004.

[36] Q. Lv, P. Cao, E. Choen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *ACM International Conference on Supercomputing (ICS02)*, 2002. `http://www.tc.cornell.edu/ics02/`.

[37] A. Maedche, B. Moik, N. Silva, and R. Volz. Mafra - a mapping framework for distributed ontologies. In *Proceedings of 9th International EKAW (Knowledge Engineering and Knowledge Management)*, LNCS(2473). Springer, 2002.

[38] D. L. McGuinness, R. Fikes J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Rapresentation and Reasoning (KR-2000)*, pages 483–493, 2000.

[39] M. Mesiti. Personalization Policies for Web Applications. In Panel "Future of Internet Applications" (SAINT), 2005. `http://www.saint2005.org/`.

[40] NAPSTER. The NAPSTER home page, 2001. `http://www.napster.com`.

[41] D. Nardi and R.J. Brachman. An Introduction to Description Logics. In *The Description Logic Handbook: Theory, Implementation and Applications*, chapter 1, pages 5–44. Cambridge University Press, 2003.

[42] .NET. The .Net home page. `http://www.microsoft.com/net/`.

[43] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proceedings of First International Conference on Information and Knowledge Management*, 1992.

[44] N. F. Noy and D. L. McGuinness. Ontology development 101: a guide to creating your first ontology. Technical report, March 2001. Available from: `http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf`.

[45] Fifth International Workshop on Global and Peer to Peer Computing, 2005. `http://gp2pc.lri.fr/`.

[46] International Workshop on Peer-to Peer Computing, 2004. `http://www.elet.polimi.it/p2p/`.

[47] OWL. `http://www.w3.org/2004/OWL/`.

[48] POLICY '04. IEEE 5th International Workshop on Policies for Distributed Systems and Networks, 2004. `http://www.research.ibm.com/policy2004/`.

[49] A. L. Powell, J. C. French, J. Callan, M. Connell, and C. L. Viles. The Impact of Database Selection on Distributed Searching. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Developement in Information Retrival*, pages 232–239, 2000.

[50] SWAP Project. `http://swap.semanticweb.org/`.

[51] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the ACM Conference of the Special Interest Group on Data Communication (SIGCOMM)*, pages 161–172, August 2001.

[52] P. Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, Vol. (11):95–130, 1999.

[53] M. A. Rodriguez and M. J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engeneering*, Vol. 15:442–465, 2003.

[54] R. Schenkel, A. Theobald, and G. Weikum. Semantic Similarity Search on Semistructured Data with the XXL Search Engine. *Information Retrieval*, 2005.

[55] SETI@home, 2001. `http://setiathome.ssl.berkeley.edu`.

[56] C. Shirky. What is P2P... and what Isn't, 2001. `http://www.openp2p.com/lpt/a/p2p/2000/11/24/shirky1-whatisp2p.html`.

[57] R. Siebes. Peer to Peer solutions in the Semantic Web context: an overview. EU-IST Project IST-2001-34103 SWAP. `http://swap.semanticweb.org/public/deliverables.htm`.

[58] R. Siebes and F. van Harmelen. Ranking agent statement for building evolving ontologies. In *Proceedings of the AAAI-02 workshop on meaning negotiation*, Alberta, Canada, 2002.

[59] M. Sloman and E. Lupu. Policy Specification for Programmable Networks. In *Proceedings of the First International Working Conference on Active Networks (IWAN '99)*, pages 73–84. Springer-Verlag, 1999.

[60] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Look-up Service for Internet Applications. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2001.

[61] H. Stuckenschmidt and F. Van Harmelen. *Information Sharing on the Semantic Web*, volume XIX of *Advanced Information and Knowledge Processing*. Springer-Verlag, 2005.

[62] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient Information Retrieval in Peer-to-Peer Networks. Technical Report HPL-2002-198, HP Labs, 2002.

[63] C. Tempich, S. Staab, and A. Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 640–649. ACM Press, 2004.

[64] The Frame Problem in Artificial Intelligence. http://www.engin.umd.umich.edu/CIS/course.des/cis479/projects/frame/welcome.html.

[65] Peer to Peer Working Group. http://p2p.internet2.edu/.

[66] D. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search for Peer-to-Peer Networks. In *Proceedings of the Third IEEE International Conference on P2P Computing (P2P2003)*, 2003.

[67] World Wide Web Consortium (W3C). Extensible Markup Language (XML). http://www.w3.org/XML/.

[68] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Using digital credentials on the World Wide Web. *J. of Computer Security*, 5, 1997.

[69] B. Yang and H. Garcia-Molina. Comparing hybrid Peer-to-Peer systems. In *Proceedings of the 27th International Conference on Very Large Data Bases*, Rome, 2001.

[70] B. Yang and H. Garcia-Molina. Efficient Search in Peer-to-Peer Networks. In *Proceedings of International Conference on Distributed Computing Systems*, 2002.

Appendice A

XML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- edited with XMLSpy v2005 sp1 U (http://www.xmlspy.com)
     by Chiara (University of Genova) -->
<!--W3C Schema generated by XMLSpy v2005 sp1 U
      (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

 <xs:element name="Certificate">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Property" minOccurs="0"
                maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="value" type="xs:string"/>
      <xs:attribute name="name" type="xs:anyType"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="name" type="xs:ID"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="Advs">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Concept" maxOccurs="unbounded"/>
    <xs:element ref="Policies"/>
   </xs:sequence>
```

```
   <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="DataRequest">
  <xs:complexType>
   <xs:element ref="Query"/>
   <xs:element ref="Certificate" minOccurs="0"
               maxOccurs="unbounded"/>
   <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="AdvsMsg">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Advs"/>
   </xs:sequence>
   <xs:attribute name="date" type="xs:date"/>
   <xs:attribute name="BS">
    <xs:simpleType>
     <xs:restriction base="xs:float">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="TTL" type="xs:integer"/>
   <xs:attribute name="peerID" type="xs:string"/>
   <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="DataRequestMsg">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="DataRequest"/>
   </xs:sequence>
   <xs:attribute name="id" type="xs:ID"/>
   <xs:attribute name="date" type="xs:date"/>
   <xs:attribute name="BS">
    <xs:simpleType>
     <xs:restriction base="xs:float">
```

```
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="1"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="TTL" type="xs:integer"/>
   <xs:attribute name="peerID" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:element name="DataResponseMsg">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Resource" type="xs:string"
               maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID"/>
  <xs:attribute name="peer" type="xs:string"/>
  <xs:attribute name="queryID" type="xs:string"/>
 </xs:complexType>
</xs:element>

<xs:element name="TempConstDef">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="PeriodicTimeExpr" minOccurs="0"/>
   <xs:element ref="IntervalExpr" minOccurs="0"/>
   <xs:element ref="DurationExpr" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="xtcd_id" type="xs:ID"/>
 </xs:complexType>
</xs:element>

<xs:element name="IntervalExpr">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="begin" type="xs:date"/>
   <xs:element name="end" type="xs:date"/>
  </xs:sequence>
  <xs:attribute name="i_expr_id" type="xs:ID"/>
 </xs:complexType>
</xs:element>
```

```
<xs:element name="PeriodicTimeExpr">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="StartTimeExpr"/>
  </xs:sequence>
  <xs:attribute name="pt_expr_id" type="xs:ID"/>
 </xs:complexType>
</xs:element>

<xs:element name="DurationExpr">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="cal">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="Years"/>
      <xs:enumeration value="Months"/>
      <xs:enumeration value="Weeks"/>
      <xs:enumeration value="Days"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:element>
   <xs:element name="len">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
       <xs:minInclusive value="0"/>
      </xs:restriction>
    </xs:simpleType>
   </xs:element>
  </xs:sequence>
  <xs:attribute name="d_expr_id" type="xs:ID"/>
 </xs:complexType>
</xs:element>

<xs:element name="StartTimeExpr">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Year" minOccurs="0">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="all"/>
      <xs:enumeration value="odd"/>
      <xs:enumeration value="even"/>
```

```
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
    <xs:element name="MonthSet" minOccurs="0">
     <xs:complexType>
      <xs:sequence maxOccurs="12">
       <xs:element ref="Month"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="WeekSet" minOccurs="0">
     <xs:complexType>
      <xs:sequence maxOccurs="4">
       <xs:element ref="Week"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="DaySet" minOccurs="0">
     <xs:complexType>
      <xs:sequence maxOccurs="7">
       <xs:element ref="Day"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Month" type="xs:gYearMonth"/>

<xs:element name="Week">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="1"/>
   <xs:maxInclusive value="4"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>

<xs:element name="Day">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="1"/>
```

```
    <xs:maxInclusive value="7" />
   </xs:restriction>
  </xs:simpleType>
 </xs:element>

 <xs:element name="Policies">
  <xs:complexType>
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="policy">
     <xs:complexType>
      <xs:sequence>
       <xs:element ref="TempConstDef" />
       <xs:element name="InternalCondition" minOccurs="0"
                   maxOccurs="unbounded">
        <xs:complexType>
         <xs:attribute name="type">
          <xs:simpleType>
           <xs:restriction base="xs:string">
            <xs:enumeration value="state" />
            <xs:enumeration value="connection" />
           </xs:restriction>
          </xs:simpleType>
         </xs:attribute>
         <xs:attribute name="onProp" type="xs:string" />
         <xs:attribute name="operation"
                       type="OperationType" />
         <xs:attribute name="value" type="xs:string" />
        </xs:complexType>
       </xs:element>
       <xs:element name="CertCondition">
        <xs:complexType>
         <xs:attribute name="onProp" type="xs:string" />
         <xs:attribute name="operation"
                       type="OperationType" />
         <xs:attribute name="value" type="xs:string" />
        </xs:complexType>
       </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="name" type="xs:ID" />
```

```
 </xs:complexType>
</xs:element>

<xs:element name="Property">
 <xs:attribute name="name" type="xs:string"/>
</xs:element>

<xs:element name="Concept">
 <xs:attribute name="name" type="xs:string"/>
</xs:element>

<xs:element name="PathExpression">
 <xs:complexType>
  <xs:element ref="Concept"/>
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
   <xs:element ref="Property"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="QueryCondition">
 <xs:complexType>
  <xs:element ref="PathExpression" minOccurs="0"
              maxOccurs="unbounded"/>
  </xs:complexType>
  <xs:attribute name="operation" type="OperationType"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>

<xs:element name="Query">
 <xs:complexType>
  <xs:choice minOccurs="0" maxOccurs="unbounded">
   <xs:element ref="PathExpression"/>
   <xs:sequence>
    <xs:element ref="QueryCondition" minOccurs="0"
                maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:choice>
 </xs:complexType>
</xs:element>

<xs:simpleType name="OperationType">
 <xs:restriction base="xs:string">
```

```
    <xs:enumeration value="EQ" />
    <xs:enumeration value="GE" />
    <xs:enumeration value="G" />
    <xs:enumeration value="LE" />
    <xs:enumeration value="L" />
   </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Appendice B

Ontologia

B.1 RDFS

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
<!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<!ENTITY rdf_ 'http://protege.stanford.edu/rdf'>
<!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'> ]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:rdf_="&rdf_;"
         xmlns:rdfs="&rdfs;">

 <rdfs:Class rdf:about="&rdf_;Animazione"
             rdfs:label="Animazione">
        <rdfs:subClassOf rdf:resource="&rdf_;Film"/>
 </rdfs:Class>

 <rdfs:Class rdf:about="&rdf_;Autore" rdfs:label="Autore">
        <rdfs:subClassOf rdf:resource="&rdf_;Persona"/>
 </rdfs:Class>

 <rdfs:Class rdf:about="&rdf_;Cartone"
             rdfs:label="Cartone">
        <rdfs:subClassOf rdf:resource="&rdf_;Film"/>
 </rdfs:Class>

 <rdf:Property rdf:about="&rdf_;Chiara_Slot_5"
               rdfs:label="Chiara_Slot_5">
        <rdfs:range rdf:resource="&rdf_;Autore"/>
 </rdf:Property>
```

```
<rdf:Property rdf:about="&rdf_;Cognome"
              rdfs:label="Cognome">
      <rdfs:domain rdf:resource="&rdf_;Persona"/>
      <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdfs:Class rdf:about="&rdf_;Favola" rdfs:label="Favola">
      <rdfs:subClassOf rdf:resource="&rdf_;Libro"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Film" rdfs:label="Film">
      <rdfs:subClassOf
            rdf:resource="&rdf_;Intrattenimento"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Illustrato"
            rdfs:label="Illustrato">
      <rdfs:subClassOf rdf:resource="&rdf_;Libro"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Intrattenimento"
            rdfs:label="Intrattenimento">
      <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Libro" rdfs:label="Libro">
      <rdfs:subClassOf
            rdf:resource="&rdf_;Intrattenimento"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Musica" rdfs:label="Musica">
       <rdfs:subClassOf
             rdf:resource="&rdf_;Intrattenimento"/>
</rdfs:Class>

<rdf:Property rdf:about="&rdf_;Nome" rdfs:label="Nome">
      <rdfs:domain rdf:resource="&rdf_;Persona"/>
      <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdfs:Class rdf:about="&rdf_;Persona"
            rdfs:label="Persona">
      <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
```

```
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Sigla" rdfs:label="Sigla">
       <rdfs:subClassOf rdf:resource="&rdf_;Musica"/>
</rdfs:Class>

<rdfs:Class rdf:about="&rdf_;Zecchino"
            rdfs:label="Zecchino">
       <rdfs:subClassOf rdf:resource="&rdf_;Musica"/>
</rdfs:Class>

<rdf:Property rdf:about="&rdf_;anno" rdfs:label="anno">
       <rdfs:domain rdf:resource="&rdf_;Intrattenimento"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&rdf_;autore"
              rdfs:label="autore">
       <rdfs:range rdf:resource="&rdf_;Autore"/>
       <rdfs:domain rdf:resource="&rdf_;Libro"/>
       <rdfs:domain rdf:resource="&rdf_;Musica"/>
</rdf:Property>

<rdf:Property rdf:about="&rdf_;casa_discografica"
              rdfs:label="casa discografica">
       <rdfs:domain rdf:resource="&rdf_;Musica"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&rdf_;durata"
              rdfs:label="durata">
       <rdfs:domain rdf:resource="&rdf_;Film"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&rdf_;editore"
              rdfs:label="editore">
       <rdfs:domain rdf:resource="&rdf_;Libro"/>
       <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&rdf_;genere"
              rdfs:label="genere">
```

```
        <rdfs:domain rdf:resource="&rdf_;Film"/>
        <rdfs:domain rdf:resource="&rdf_;Libro"/>
         <rdfs:range rdf:resource="&rdfs;Literal"/>
 </rdf:Property>

 <rdf:Property rdf:about="&rdf_;interprete"
               rdfs:label="interprete">
        <rdfs:domain rdf:resource="&rdf_;Musica"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
 </rdf:Property>

 <rdf:Property rdf:about="&rdf_;ref"
               rdfs:label="ref">
        <rdfs:domain rdf:resource="&rdf_;Intrattenimento"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
 </rdf:Property>

 <rdf:Property rdf:about="&rdf_;titolo" rdfs:label="titolo">
        <rdfs:domain rdf:resource="&rdf_;Intrattenimento"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
 </rdf:Property>

</rdf:RDF>
```

B.2 OWL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:protege="http://protege.stanford.edu/plugins/
                    owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
    xml:base="http://www.owl-ontologies.com/unnamed.owl">

  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/
                               plugins/owl/protege"/>
  </owl:Ontology>
```

```
<owl:Class rdf:ID="Favola">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Libro"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Zecchino">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Musica"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Persona"/>

<owl:Class rdf:ID="Film">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Intrattenimento"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Illustrato">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Libro"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Autore">
  <rdfs:subClassOf rdf:resource="#Persona"/>
</owl:Class>

<owl:Class rdf:ID="Sigla">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Musica"/>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Libro">
  <rdfs:subClassOf rdf:resource="#Intrattenimento"/>
</owl:Class>

<owl:Class rdf:ID="Animazione">
  <rdfs:subClassOf rdf:resource="#Film"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Cartone">
  <rdfs:subClassOf rdf:resource="#Film"/>
</owl:Class>

<owl:Class rdf:about="#Musica">
  <rdfs:subClassOf rdf:resource="#Intrattenimento"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="autore">
  <rdfs:range rdf:resource="#Autore"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Libro"/>
        <owl:Class rdf:about="#Musica"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:ID="durata">
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Film"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/
                          owl#FunctionalProperty"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="anno">
  <rdfs:domain rdf:resource="#Intrattenimento"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/
                          owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#int"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="editore">
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Libro"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="interprete">
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Musica"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="casa_discografica">
  <rdfs:domain rdf:resource="#Musica"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/
                          owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#string"/>
  <rdfs:label>casa discografica</rdfs:label>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="ref">
  <rdfs:domain rdf:resource="#Intrattenimento"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/
                          owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="genere">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Libro"/>
        <owl:Class rdf:about="#Film"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
                            XMLSchema#string"/>
</owl:DatatypeProperty>

<owl:FunctionalProperty rdf:ID="Chiara_Slot_5">
  <rdfs:range rdf:resource="#Autore"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/
                          owl#ObjectProperty"/>
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="Nome">
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/
                              XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Persona"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/
                            owl#DatatypeProperty"/>
  </owl:FunctionalProperty>

  <owl:FunctionalProperty rdf:ID="titolo">
    <rdfs:range rdf:resource="http://www.w3.org/2001/
                              XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Intrattenimento"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/
                            owl#DatatypeProperty"/>
  </owl:FunctionalProperty>

  <owl:FunctionalProperty rdf:ID="Cognome">
    <rdfs:domain rdf:resource="#Persona"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/
                            owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/
                              XMLSchema#string"/>
  </owl:FunctionalProperty>

</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 1.3, Build 225.4)
     http://protege.stanford.edu -->
```

Appendice C

Istanziazione dell'ontologia

C.1 RDF

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY a 'http://protege.stanford.edu/system#'>
 <!ENTITY rdf_ 'http://protege.stanford.edu/rdf'>
 <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>

<rdf:RDF xmlns:rdf="&rdf;"
         xmlns:rdf_="&rdf_;"
         xmlns:a="&a;"
         xmlns:rdfs="&rdfs;">

<a:_instance_annotation rdf:about="&rdf_;Chiara_Instance_12"
         a:_creation_timestamp="2005.02.11_10:27:00.308 CET"
         a:_creator="1996s054"
         rdfs:label="Chiara_Instance_12"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_10"
             rdf_:Cognome="Avogadro"
             rdf_:Nome="O."
             rdfs:label="Chiara_Instance_10"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_10014"
             rdf_:Cognome="Zanin"
             rdf_:Nome="L."
             rdfs:label="Chiara_Instance_10014"/>
```

```
<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_10015"
             rdf_:Cognome="Della Giustina"
             rdf_:Nome="A."
             rdfs:label="Chiara_Instance_10015"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_10016"
             rdf_:Cognome="Migliacci"
             rdf_:Nome="F."
             rdfs:label="Chiara_Instance_10016"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_10017"
             rdf_:Cognome="Albertelli"
             rdf_:Nome="L."
             rdfs:label="Chiara_Instance_10017"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_11"
             rdf_:Cognome="Cesarini"
             rdf_:Nome="G."
             rdfs:label="Chiara_Instance_11"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_6"
             rdf_:Cognome="Walt Disney"
             rdfs:label="Chiara_Instance_6"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_7"
             rdf_:Cognome="Perrault"
             rdfs:label="Chiara_Instance_7"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_8"
             rdf_:Cognome="Collodi"
             rdfs:label="Chiara_Instance_8"/>

<rdf_:Autore rdf:about="&rdf_;Chiara_Instance_9"
             rdf_:Cognome="Grimm"
             rdfs:label="Chiara_Instance_9"/>

<rdf_:Favola rdf:about="&rdf_;Chiara_Instance_36"
             rdf_:anno="1918"
             rdf_:editore="Fatatrac"
             rdf_:genere="Avventura"
             rdf_:ref="C:/chiara/condivisi/favole/
                       pollicino.html"
```

```
            rdf_:titolo="Pollicino"
            rdfs:label="Pollicino">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_9"/>
</rdf_:Favola>

<rdf_:Favola rdf:about="&rdf_;Chiara_Instance_22"
            rdf_:anno="1955"
            rdf_:editore="Walt Disney"
            rdf_:genere="Avventura"
            rdf_:ref="C:/chiara/condivisi/favole/lilli.html"
            rdf_:titolo="Lilli e il vagabondo"
            rdfs:label="Lilli e il vagabondo">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_6"/>
</rdf_:Favola>

<rdf_:Favola rdf:about="&rdf_;Chiara_Instance_23"
            rdf_:anno="1950"
            rdf_:editore="Walt Disney"
            rdf_:genere="Romantico"
            rdf_:ref="C:/chiara/condivisi/favole/
                      cenerentola.html"
            rdf_:titolo="Cenerentola"
            rdfs:label="Cenerentola">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_7"/>
</rdf_:Favola>

<rdf_:Favola rdf:about="&rdf_;Chiara_Instance_24"
            rdf_:anno="1949"
            rdf_:editore="Rizzoli"
            rdf_:genere="Avventura"
            rdf_:ref="C:/chiara/condivisi/favole/
                      pinocchio.html"
            rdf_:titolo="Pinocchio"
            rdfs:label="Pinocchio">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_8"/>
</rdf_:Favola>

<rdf_:Animazione rdf:about="&rdf_;Chiara_Instance_25"
                 rdf_:anno="1995"
                 rdf_:durata="80"
                 rdf_:genere="Avventura"
                 rdf_:ref="C:/chiara/condivisi/film_A/
                           toystory.avi"
```

```
                    rdf_:titolo="Toy Story"
                    rdfs:label="Toy Story"/>

<rdf_:Animazione    rdf:about="&rdf_;Chiara_Instance_26"
                    rdf_:anno="1998"
                    rdf_:durata="83"
                    rdf_:genere="Musicale"
                    rdf_:ref="C:/chiara/condivisi/film_A/
                              Zformica.avi"
                    rdf_:titolo="Z la formica"
                    rdfs:label="Z la formica"/>

<rdf_:Animazione    rdf:about="&rdf_;Chiara_Instance_27"
                    rdf_:anno="2001"
                    rdf_:durata="89"
                    rdf_:genere="Comico"
                    rdf_:ref="C:/chiara/condivisi/film_A/
                              shrek.avi"
                    rdf_:titolo="Shrek"
                    rdfs:label="Shrek"/>

<rdf_:Cartone       rdf:about="&rdf_;Chiara_Instance_28"
                    rdf_:anno="1961"
                    rdf_:durata="79"
                    rdf_:genere="Avventura"
                    rdf_:ref="C:/chiara/condivisi/film_C/
                              carica101.avi"
                    rdf_:titolo="La carica dei 101"
                    rdfs:label="La carica dei 101"/>

<rdf_:Cartone       rdf:about="&rdf_;Chiara_Instance_29"
                    rdf_:anno="2003"
                    rdf_:durata="100"
                    rdf_:genere="Avventura"
                    rdf_:ref="C:/chiara/condivisi/film_C/
                              nemo.avi"
                    rdf_:titolo="Alla ricerca di Nemo"
                    rdfs:label="Alla ricerca di Nemo"/>

<rdf_:Cartone       rdf:about="&rdf_;Chiara_Instance_30"
                    rdf_:anno="1989"
                    rdf_:durata="82"
                    rdf_:genere="Romantico"
```

```
                rdf_:ref="C:/chiara/condivisi/film_C/
                          sirenetta.avi"
                rdf_:titolo="La sirenetta"
                rdfs:label="La sirenetta"/>

<rdf_:Zecchino rdf:about="&rdf_;Chiara_Instance_31"
                rdf_:anno="1993"
                rdf_:casa_discografica="EMI"
                rdf_:interprete="Angelo Ferri"
                rdf_:ref="C:/chiara/condivisi/musica_Z/
                          coccodrillo.mp3"
                rdf_:titolo="Il coccodrillo come fa"
                rdfs:label="Il coccodrillo come fa">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_10"/>
</rdf_:Zecchino>

<rdf_:Zecchino rdf:about="&rdf_;Chiara_Instance_32"
                rdf_:anno="1968"
                rdf_:casa_discografica="EMI"
                rdf_:interprete="Barbara Ferigo"
                rdf_:ref="C:/chiara/condivisi/musica_Z/
                          44gatti.mp3"
                rdf_:titolo="Quarantaquattro gatti"
                rdfs:label="Quarantaquattro gatti">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_11"/>
</rdf_:Zecchino>

<rdf_:Zecchino rdf:about="&rdf_;Chiara_Instance_33"
                rdf_:anno="1968"
                rdf_:casa_discografica="EMI"
                rdf_:interprete="Cristina D'avena"
                rdf_:ref="C:/chiara/condivisi/musica_Z/
                          valzer.mp3"
                rdf_:titolo="Valzer del moscerino"
                rdfs:label="Valzer del moscerino">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_10014"/>
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_10015"/>
</rdf_:Zecchino>

<rdf_:Sigla rdf:about="&rdf_;Chiara_Instance_34"
             rdf_:anno="1980"
             rdf_:casa_discografica="EMI"
             rdf_:interprete="La banda dei bucanieri"
```

```
            rdf_:ref="C:/chiara/condivisi/musica_S/
                      harlock.mp3"
            rdf_:titolo="Capitan Harlock"
            rdfs:label="Capitan Harlock">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_10017"/>
</rdf_:Sigla>

<rdf_:Sigla rdf:about="&rdf_;Chiara_Instance_35"
            rdf_:anno="1982"
            rdf_:casa_discografica="EMI"
            rdf_:interprete="Elisabetta Viviani"
            rdf_:ref="C:/chiara/condivisi/musica_S/heidi.mp3"
            rdf_:titolo="Heidi"
            rdfs:label="Heidi">
 <rdf_:autore rdf:resource="&rdf_;Chiara_Instance_10016"/>
</rdf_:Sigla>

</rdf:RDF>
```

C.2 OWL

```
<rdf:RDF
    xmlns:protege="http://protege.stanford.edu/plugins/
                    owl/protege#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/unnamed.owl#"
    xml:base="http://www.owl-ontologies.com/unnamed.owl">

  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://protege.stanford.edu/
                               plugins/owl/protege"/>
  </owl:Ontology>

<Favola rdf:ID="Chiara_Instance_24">
 <ref rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string">
    C:/chiara/condivisi/favole/pinocchio.html
 </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
```

```
                                XMLSchema#string">
    Pinocchio
  </titolo>
  <autore>
   <Autore rdf:ID="Chiara_Instance_8">
    <Cognome rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#string">
      Collodi
    </Cognome>
   </Autore>
  </autore>
  <anno rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
    1949
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#string">
    Avventura
  </genere>
  <editore rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#string">
    Rizzoli
  </editore>
</Favola>

<Favola rdf:ID="Chiara_Instance_36">
  <ref rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
    C:/chiara/condivisi/favole/pollicino.html
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#string">
    Pollicino
  </titolo>
  <autore>
   <Autore rdf:ID="Chiara_Instance_9">
    <Cognome rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#string">
      Grimm
    </Cognome>
   </Autore>
  </autore>
  <anno rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
    1918
```

```
 </anno>
 <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Avventura
 </genere>
 <editore rdf:datatype="http://www.w3.org/2001/
                         XMLSchema#string">
   Fatatrac
 </editore>
</Favola>

<Favola rdf:ID="Chiara_Instance_22">
 <ref rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string">
   C:/chiara/condivisi/favole/lilli.html
  </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Lilli e il vagabondo
 </titolo>
 <autore rdf:resource="#Chiara_Instance_6"/>
 <anno rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
   1955
 </anno>
 <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Avventura
 </genere>
 <editore rdf:datatype="http://www.w3.org/2001/
                         XMLSchema#string">
   Walt Disney
 </editore>
</Favola>

<Autore rdf:ID="Chiara_Instance_6">
 <Cognome rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Walt Disney
 </Cognome>
</Autore>

<Favola rdf:ID="Chiara_Instance_23">
 <ref rdf:datatype="http://www.w3.org/2001/
```

```
                        XMLSchema#string">
    C:/chiara/condivisi/favole/cenerentola.html
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Cenerentola
  </titolo>
  <autore rdf:resource="#Chiara_Instance_7"/>
  <anno rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#int">
    1950
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Romantico
  </genere>
  <editore rdf:datatype="http://www.w3.org/2001/
                         XMLSchema#string">
    Walt Disney
  </editore>
</Favola>

<Autore rdf:ID="Chiara_Instance_7">
  <Cognome rdf:datatype="http://www.w3.org/2001/
                         XMLSchema#string">
    Perrault
  </Cognome>
</Autore>

<Animazione rdf:ID="Chiara_Instance_27">
  <ref rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#string">
    C:/chiara/condivisi/film_A/shrek.avi
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Shrek
  </titolo>
  <anno rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#int">
    2001
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
```

```
                              XMLSchema#string">
    Comico
  </genere>
  <durata rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#int">
    89
  </durata>
</Animazione>

<Animazione rdf:ID="Chiara_Instance_26">
  <ref rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    C:/chiara/condivisi/film_A/Zformica.avi
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#string">
    Z la formica
  </titolo>
  <anno rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#int">
    1998
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#string">
    Musicale
  </genere>
  <durata rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#int">
    83
  </durata>
</Animazione>

<Animazione rdf:ID="Chiara_Instance_25">
  <ref rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    C:/chiara/condivisi/film_A/toystory.avi
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#string">
    Toy Story
  </titolo>
  <anno rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#int">
```

```
    1995
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Avventura
  </genere>
  <durata rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
    80
  </durata>
</Animazione>

<Cartone rdf:ID="Chiara_Instance_29">
  <ref rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#string">
    C:/chiara/condivisi/film_C/nemo.avi
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Alla ricerca di Nemo
  </titolo>
  <anno rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#int">
    2003
  </anno>
  <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Avventura
  </genere>
  <durata rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
    100
  </durata>
</Cartone>

<Cartone rdf:ID="Chiara_Instance_30">
  <ref rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#string">
    C:/chiara/condivisi/film_C/sirenetta.avi
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    La sirenetta
```

```
 </titolo>
 <anno rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
   1989
 </anno>
 <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Romantico
 </genere>
 <durata rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
   82
 </durata>
</Cartone>

<Cartone rdf:ID="Chiara_Instance_28">
 <ref rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#string">
   C:/chiara/condivisi/film_C/carica101.avi
 </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   La carica dei 101
 </titolo>
 <anno rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
   1961
 </anno>
 <genere rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
   Avventura
 </genere>
 <durata rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#int">
   79
 </durata>
</Cartone>

<Sigla rdf:ID="Chiara_Instance_35">
 <ref rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#string">
   C:/chiara/condivisi/musica_S/heidi.mp3
 </ref>
```

```
<titolo rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#string">
  Heidi
</titolo>
<interprete rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
  Elisabetta Viviani
</interprete>
<autore>
 <Autore rdf:ID="Chiara_Instance_10016">
  <Nome rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#string">
    F.
  </Nome>
  <Cognome rdf:datatype="http://www.w3.org/2001/
                         XMLSchema#string">
    Migliacci
  </Cognome>
 </Autore>
</autore>
<anno rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#int">
  1982
</anno>
<casa_discografica rdf:datatype="http://www.w3.org/2001/
                                 XMLSchema#string">
  EMI
</casa_discografica>
</Sigla>

<Sigla rdf:ID="Chiara_Instance_34">
 <ref rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string">
   C:/chiara/condivisi/musica_S/harlock.mp3
 </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
                       XMLSchema#string">
   Capitan Harlock
 </titolo>
 <interprete rdf:datatype="http://www.w3.org/2001/
                           XMLSchema#string">
   La banda dei bucanieri
 </interprete>
```

```
 <autore>
  <Autore rdf:ID="Chiara_Instance_10017">
   <Nome rdf:datatype="http://www.w3.org/2001/
                       XMLSchema#string">
     L.
   </Nome>
   <Cognome rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
     Albertelli
   </Cognome>
  </Autore>
 </autore>
 <anno rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#int">
   1980
 </anno>
 <casa_discografica rdf:datatype="http://www.w3.org/2001/
                                  XMLSchema#string">
   EMI
 </casa_discografica>
</Sigla>

<Zecchino rdf:ID="Chiara_Instance_33">
 <ref rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string">
   C:/chiara/condivisi/musica_Z/valzer.mp3
 </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
                       XMLSchema#string">
   Valzer del moscerino
 </titolo>
 <interprete rdf:datatype="http://www.w3.org/2001/
                           XMLSchema#string">
   Cristina D'avena
 </interprete>
 <autore>
  <Autore rdf:ID="Chiara_Instance_10014">
   <Nome rdf:datatype="http://www.w3.org/2001/
                       XMLSchema#string">
     L.
   </Nome>
   <Cognome rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
```

```
      Zanin
    </Cognome>
   </Autore>
  </autore>
  <autore>
   <Autore rdf:ID="Chiara_Instance_10015">
    <Nome rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
      A.
    </Nome>
    <Cognome rdf:datatype="http://www.w3.org/2001/
                           XMLSchema#string">
      Della Giustina
    </Cognome>
   </Autore>
  </autore>
  <anno rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#int">
    1968
  </anno>
  <casa_discografica rdf:datatype="http://www.w3.org/2001/
                                   XMLSchema#string">
    EMI
  </casa_discografica>
</Zecchino>

<Zecchino rdf:ID="Chiara_Instance_32">
  <ref rdf:datatype="http://www.w3.org/2001/
                     XMLSchema#string">
    C:/chiara/condivisi/musica_Z/44gatti.mp3
  </ref>
  <titolo rdf:datatype="http://www.w3.org/2001/
                        XMLSchema#string">
    Quarantaquattro gatti
  </titolo>
  <interprete rdf:datatype="http://www.w3.org/2001/
                            XMLSchema#string">
    Barbara Ferigo
  </interprete>
  <autore rdf:resource="#Chiara_Instance_11"/>
  <anno rdf:datatype="http://www.w3.org/2001/
                      XMLSchema#int">
    1968
```

```
 </anno>
 <casa_discografica rdf:datatype="http://www.w3.org/2001/
                                        XMLSchema#string">
    EMI
 </casa_discografica>
</Zecchino>

<Autore rdf:ID="Chiara_Instance_11">
  <Nome rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
    G.
 </Nome>
 <Cognome rdf:datatype="http://www.w3.org/2001/
                           XMLSchema#string">
    Cesarini
 </Cognome>
</Autore>

<Zecchino rdf:ID="Chiara_Instance_31">
 <ref rdf:datatype="http://www.w3.org/2001/
                       XMLSchema#string">
    C:/chiara/condivisi/musica_Z/coccodrillo.mp3
 </ref>
 <titolo rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
    Il coccodrillo come fa
 </titolo>
 <interprete rdf:datatype="http://www.w3.org/2001/
                             XMLSchema#string">
    Angelo Ferri
 </interprete>
 <autore>
  <Autore rdf:ID="Chiara_Instance_10">
   <Nome rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#string">
      O.
   </Nome>
   <Cognome rdf:datatype="http://www.w3.org/2001/
                             XMLSchema#string">
      Avogadro
   </Cognome>
  </Autore>
 </autore>
```

```
 <anno rdf:datatype="http://www.w3.org/2001/
                          XMLSchema#int">
    1993
 </anno>
 <casa_discografica rdf:datatype="http://www.w3.org/2001/
                                    XMLSchema#string">
    EMI
 </casa_discografica>
</Zecchino>

</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 1.3, Build 225.4)
     http://protege.stanford.edu -->
```