Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

Peer to Peer DBMS: il sistema FOAF

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Giampiero Miccoli

Anno Accademico 2004-2005

Parole chiave:
Peer-to-Peer
P2PDBMS
Sicurezza
FOAF
Metadati

RINGRAZIAMENTI

Un primo, immenso e doveroso ringraziamento deve essere rivolto alla mia famiglia che mi è stata vicino e mi ha sostenuto in questi anni di università permettendomi di laurearmi.

Un particolare ringraziamento va alla Prof. Sonia Bergamaschi per tutto l'aiuto fornitomi durante la realizzazione della tesi.

Ringrazio infine tutti i miei compagni di corso con i quali ho condiviso gioie e fatiche in questi anni di studio.

Un sincero grazie a tutti Voi…

Giampiero Miccoli

INDICE

Capitolo 2: Le reti Peer-to-Peer

Capitolo 3: La sicurezza nelle applicazioni Peer-to-Peer

Capitolo 4: Il sistema FOAF

INDICE DELLE FIGURE

Introduzione

L'architettura client/server nelle sue varie implementazioni è stata padrona incontrastata della scena per diversi anni. Anche oggi, nella maggior parte dei casi, l'idea più naturale di rete è quella in cui alcune macchine particolarmente potenti erogano dei servizi a delle macchine meno potenti. Gli esempi possono essere tanti, e vanno dai vecchi mainframe con i loro "dumb terminals" ai file server dipartimentali o aziendali fino ad arrivare a servizi su rete geografica, uno per tutti, il World Wide Web.

In questa architettura gerarchica, si ha una grande quantità di risorse concentrata su uno o pochi nodi "centrali", i server, mentre poche o nessuna sono dislocate su nodi periferici che attingono alle risorse dei server: i client.

Ma era presente fin dagli esordi del TCP/IP un altro paradigma, in cui i ruoli di client e di server sono decisamente più sfumati, se non completamente assenti. Ma solo nell'ultimo decennio si è assistito all'esplosione di questo tipo di servizi. Il modello in discussione è il Peer-to-Peer (p2p).

Generalmente Peer-to-Peer sta per "condivisione di risorse fra pari", cioè si intende una rete di computer o qualsiasi rete che non possiede client o server fissi, ma un numero di nodi equivalenti (peer, appunto) che fungono sia da client che da server verso altri nodi della rete. Al più il server potrà svolgere una funzione di autenticazione iniziale dell'utente, nel momento in cui questi si collegherà al sistema.

Mediante questa configurazione qualsiasi nodo è in grado di avviare o completare una transazione. I nodi equivalenti possono differire nella configurazione locale, nella velocità di elaborazione, nella ampiezza di banda e nella quantità di dati memorizzati. L'esempio classico di p2p è la rete per la condivisione di file (File Sharing).

Molti software di file sharing oggi presenti in rete e dalla stessa liberamente scaricabili, tanto per citarne alcuni più noti, WinMX, KaZaA, Gnutella, Freenet, eMule, Morpheus, OpenNap vengono utilizzati dagli utenti di Internet per la trasmissione da un utente all'altro, appunto peer-to-peer, di svariati tipi di files, da quelli musicali ai filmati video, dagli spartiti agli appunti per gli esami, dal software (giochi e applicativi) a loghi e suonerie. Il file sharing si è affermato molto rapidamente perché facilitava la possibilità di condividere una grande

quantità di risorse, con la possibilità di replica degli stessi, e sua complice l'assenza di un server centrale che smisti ed indicizzi le richieste ed abbia una visione globale degli utenti e dei file condivisi.
Il prototipo del file sharing è stato per tutti, a livello planetario, Napster, ma fu preso di mira ed in parte sconfitto dalla RIAA (Recording Industry Association of America), la potente associazione dei discografici statunitensi, che ha intentato e vinto una serie di cause legali contro l'autore del software, considerato mezzo per la violazione del copyright, ad opera degli utenti connessi a Napster, sui brani musicali e sulle opere cinematografiche condivise tramite il sistema. Napster però non si basava sul sistema p2p puro, bensì su una contaminazione del più consolidato ed usuale sistema client/server.

Il testo è composto dai seguenti 4 capitoli:

1. **Il Peer-to-Peer**

   Questo capitolo è un'introduzione alle reti p2p: le sue "origini", le principali categorie, i vantaggi, il confronto con altre tipologie di reti ed infine l'analisi tra diverse reti p2p popolari, quali Naspter, Gnutella, eMule, BitTorrent, WinMX e le reti che permettono di far rimanere anonimi gli utenti che condividono le risorse, quali Freenet, MojoNation, Publius e Mute.

2. **Le reti Peer-to-Peer**

   Questo capitolo illustra le diverse architetture utilizzate, la classifica generazionale e i vantaggi/svantaggi legate all'utilizzo di un dato modello p2p.

3. **La sicurezza nelle applicazioni Peer-to-Peer**

   In questo capitolo sono elencate le esigenze di sicurezza che si riscontrano nelle applicazioni p2p, con lo scopo di sottolineare come queste siano diverse a seconda dell'applicazione considerata. Inoltre sono individuate le problematiche di sicurezza più generiche relative alle applicazioni p2p.

4. **Il sistema FOAF**

   Questo capitolo introduce il progetto FOAF (Friend Of A Friend), un sistema basato su descrizioni in XML, adottando le convenzioni del RDF e l'ontologia del OWL. Sono trattate applicazioni che permettono la realizzazione di descrizioni FOAF, di poterle verificare e alcuni modi per poterle pubblicarle sul Web. Inoltre sono provate ed illustrate

applicazioni che permettono di visualizzare tali descrizioni e di poter vedere i legami esistono tra vari utenti. È stata provata un'applicazione, denominata RDF Crawler, che permette la raccolta di dati RDF/XML presenti nel Semantic Web, collegati tra loro attraverso i link contenuti nelle descrizioni RDF. Infine vi è un'estensione di FOAF, denominata Semantic Campus, che mira a creare e a definire un vocabolario RDF per la descrizione dell'ambiente universitario, relativo alle risorse riguardo le università, i dipartimenti, i professori e gli studenti.

Capitolo 1

Il Peer-to-Peer

1.1 Descrizione

Internet oggi è una grande rete cooperativa, popolata da milioni di host sparsi in tutto il mondo. Fino a pochi anni fa il ruolo della maggior parte degli host era passivo: solo un ristretto numero di nodi possedeva le risorse, per cui chi voleva usufruirne non doveva far altro che richiederle ad uno di questi. Nel 1999 qualcosa è cambiato, o meglio, il panorama si è arricchito: ciò che un tempo funzionava in modo passivo ha cominciato ad agire attivamente.

Napster, l'applicazione di file sharing che consentiva a chiunque di condividere file musicali, ha scatenato una vera e propria rivoluzione, grazie alla quale i milioni di utenti connessi alla Rete hanno cominciato ad utilizzare i loro sempre più potenti computer per fare molto di più del semplice browsing e dello scambio di e-mail. È stato trovato il modo per connettere e far collaborare direttamente i computer tra loro, tanto da farli diventare dei veri e propri motori di ricerca.

Alcuni affermano che il peer-to-peer è esattamente ciò che Internet è ed è sempre stato: molti dei servizi della Rete sono peer-to-peer, come il trasferimento file, o il Telnet per la connessione remota.

Tuttavia, al di là delle critiche e degli elogi, è evidente che il peer-to-peer continua a diffondersi in modo sempre più capillare, perché offre l'opportunità di effettuare comunicazioni in tempo reale e di condividere le informazioni in un ambiente distribuito su larga scala.

1.2 Gli albori

Il peer-to-peer è sempre esistito, ma non era mai stato definito tale: i server con un indirizzo IP fisso o risolvibile hanno sempre avuto la possibilità di comunicare tra loro per accedere ai servizi. Tante applicazioni pre-p2p, come l'e-mail o il Domain Name System (DNS), sono state progettate proprio sulla base di questa capacità, ma una di queste, Usenet, si differenziava dalle altre.

Usenet è stata creata nel 1979 da due studenti del Nord Carolina, Tom Truscott e Jim Ellis, al fine di fornire ai computer un modo per scambiarsi informazioni nel periodo appena precedente la diffusione di Internet. La loro prima interazione consisteva nel collegarsi tra loro per ricercare nuovi file e scaricarli. Usenet presenta alcune caratteristiche che consentono di definirla probabilmente la prima applicazione p2p: non ha un'autorità di gestione centrale (la distribuzione dei contenuti è gestita da ogni singolo nodo) e i contenuti della rete Usenet sono replicati (completamente o in parte) attraverso i suoi nodi. Usenet non è un pezzo di software o una rete di server. Sebbene richieda del software e dei server per operare, questi elementi non definiscono realmente Usenet. Usenet è semplicemente un modo per far parlare tra loro le macchine al fine di scambiarsi messaggi attraverso la rete: sfruttando il Network News Transfer Protocol (NNTP), un numero indefinito di macchine possono partecipare indipendentemente per fornire dei servizi.

1.3 I vari tipi di rete

Dai tempi di Usenet, le più famose applicazioni p2p si sono distinte in tre principali categorie:

- *Scambio istantaneo di messaggi*;
- *Scambio di informazioni*;
- *Condivisione di potere computazionale*.

1.3.1 Scambio istantaneo di messaggi

Le applicazioni che consentono agli utenti di scambiarsi brevi messaggi di testo, messaggi vocali e file attraverso la rete sono denominate Instant Messaging (IM). Pensando a questo genere di applicazioni in alcuni casi si può forse parlare di precursori del p2p, infatti, con architetture orientate alla gestione di comunità distribuite, hanno gettato le basi dei protocolli e dei sistemi orientati ai peer.

L'utilizzo del cosiddetto Instant Messaging ha avuto un crescente successo negli ultimi anni.

Fra gli svantaggi maggiori che contraddistinguono questo tipo di reti è che la sicurezza non è ancora garantita, e quindi è possibile che vengano diffusi più facilmente virus e worms. Non solo: le informazioni viaggiano in chiaro quindi non è garantita nemmeno la privacy. Un ulteriore svantaggio sta nel fatto che i vari client di questo tipo di applicazione non comunicano fra loro.

Un grande vantaggio è che, paradossalmente, questo tipo di reti non sono ancora state coinvolte in procedure legali, nonostante la mole di file e risorse che circola in esse non sia di molto inferiore rispetto a quella che veniva scambiata attraverso Napster.
Inoltre la connessione fra due utenti risulta diretta, robusta, scalabile e veloce, in quanto i messaggi non necessitano di un passaggio attraverso un server centrale.
Esempi di client di Instant Messaging più popolari sono ICQ [1], MSN [2], AIM[3] e Jabber [4].

1.3.2 Scambio di informazioni

Le applicazioni di File Sharing permettono agli utenti di condividere servizi e risorse attraverso la rete. Diventati famosi con il precursore Napster, questo genere di applicativi rappresentano la parte più significativa degli attuali sviluppi. In effetti con piattaforme orientate alla condivisione di file sufficientemente aperte e generiche è possibile costruire layer di più alto livello in grado di rispondere a numerose esigenze (ad esempio file server distribuiti o applicazioni di cooperazione).

1.3.3 Condivisione di potere computazionale

Questo tipo di reti si differenzia dal precedente nel fatto che qui sono le risorse hardware che vengono condivise, mentre come avevamo visto per le reti precedenti erano risorse software che venivano messe a disposizione della "comunità" virtuale costituita dalle reti p2p.
La rete tra peer serve a realizzare un'architettura di calcolo distribuita, per concorrere, ad esempio, al raggiungimento di maggiori prestazioni in termini di calcolo: l'idea è quella di sfruttare la potenza di calcolo nei momenti di non utilizzo. Questa potenza di calcolo viene utilizzata per risolvere problemi complessi che un solo calcolatore non sarebbe capace di gestire (ricordiamo le applicazioni a sfondo di ricerca scientifica).

Tra le principali reti che si occupano del calcolo computazionale ricordiamo:

1. *GIMPS* [5]: Acronimo di Great Internet Mersenne Prime Search, ed è un progetto per la ricerca di grandi numeri primi. Con tale progetto è stato trovato anche il più grande numero primo attuale che ha 4 milioni di cifre;
2. *SETI@home* [6]: Sta per Search for Extraterrestrial Intelligence e analizza dati provenienti da radiotelescopi in cerca di segnali di origine extraterrestre. In totale 3,4

milioni di utenti hanno dedicato a questo compito più di 800.000 anni di tempo processore;

3. *Distributed.net* [7]: Ha decriptato messaggi usando ricerche a forza bruta nello spazio delle possibili chiavi di cifratura. Più di 100 miliardi di chiavi sono provate ogni secondo nel suo attuale progetto di decriptazione. Cerca anche insiemi di numeri che vengono utilizzati nella cifratura e nelle comunicazioni;
4. *FightAIDS@Home* [8]: I PC collegati alla rete vengono utilizzati per generare e testare i milioni di possibili farmaci contro l'AIDS;
5. *Intel/Devices cancer research project* [9]: Ricerca possibili farmaci antitumorali valutando tra 3,5 miliardi di molecole, quelle meglio conformate per legarsi ad una fra le otto proteine necessarie allo sviluppo di un tumore.

Questo tipo di applicazioni utilizzano i momenti in cui i processori dei PC, su cui sono installati, sono inattivi; quindi molto spesso vengono anche usati come screensaver: nei momenti in cui una porzione dei calcoli termina, si occupano le stesse applicazioni di inviare in background i risultati ottenuti.

1.4 I vantaggi

L'interesse per le soluzioni basate sul paradigma peer-to-peer si è fatto sempre più crescente dal momento in cui diversi anni fa due applicazioni di tipo p2p si sono largamente diffuse, diventando un vero e proprio fenomeno di massa: ICQ e Napster. La prima è un'applicazione di tipo IM (Instant Messaging), la seconda di tipo File Sharing.
Alcune reti e canali, come per esempio Napster, OpenNap o IRC (Internet Relay Chat) [10] usano il modello client-server per alcuni compiti (per esempio la ricerca) e il modello peer-to-peer per tutti gli altri. Reti come Gnutella o Freenet, vengono definite come il vero modello di rete peer-to-peer in quanto utilizzano una struttura peer-to-peer per tutti i tipi di transazione.
Quando il termine peer-to-peer venne utilizzato per descrivere la rete Napster, implicava che la natura a file condivisi del protocollo fosse la cosa più importante, ma in realtà la grande conquista di Napster fu quella di mettere tutti i computer collegati sullo stesso piano.

Le reti p2p presentano un insieme di vantaggi:

- Si basano sui protocolli http, ftp, ecc., quindi prevedono un libero scambio di informazioni attraverso i router ed i firewall;

- Non richiedono operazioni di autenticazione, e questo è fondamentale ai fini della garanzia di anonimità degli utenti della rete stessa;
- Danno la possibilità di implementare chat e di utilizzare e-mail;
- Non necessitano di amministratori di rete e sono particolarmente semplici da utilizzare. In particolare il costo dell'aggiornamento delle informazioni che circolano in rete è praticamente nullo;
- Sono estremamente tolleranti ai guasti poiché, in linea di massima, non dipendono da una struttura centrale, quindi la perdita di un nodo non porta ad un isolamento totale dalle informazioni cui si desidera accedere.

A fianco di tali vantaggi, notiamo però che:

- Non è sempre facile riuscire a reperire i dati cercati: questo è dovuto essenzialmente alla struttura della rete;
- È necessario concepire ed implementare nuovi ed efficienti algoritmi di ricerca per questo tipo di reti;
- In certi casi viene consumata una grande quantità di larghezza di banda per soddisfare le richieste degli utenti;
- La larghezza di banda impone delle restrizioni sul possibile numero di utenti che possono accedere alla rete. Ciascun nodo infatti ha la libertà di stabilire quanti nodi possono collegarsi ad esso: un numero troppo elevato di collegamenti porta ad un consumo eccessivo della sua larghezza di banda, mentre un numero esiguo di utenti porta ad una vera e propria restrizione sulla possibilità di accesso alla rete da parte di nuovi nodi;
- Spesso non si hanno garanzie sulla qualità e l'affidabilità dei dati che vengono forniti da un nodo. Alcune informazioni infatti possono essere fittizie e sono dovute ad attività di spamming mirate alla distruzione della rete.

Uno dei problemi fondamentali che si presenta per questo tipo di reti sta proprio nel fatto che il confine tra libertà di parola e abuso di tale libertà è decisamente sottile. La garanzia di anonimato nello scambio di informazioni fa sì che possano essere distribuiti materiali relativi alla pornografia o alle attività terroristiche o razziste.
D'altra parte però questo è un rischio che bisogna correre dato che i vantaggi di questo tipo di reti sono enormi, come la semplicità di ricerca e la condivisione di file.

1.5 Le tipologie di reti

Qualunque sia l'architettura fisica e logica di una rete, il funzionamento di base è sempre lo stesso: una connessione si instaura nel momento in cui un nodo della rete richiede una connessione ad un altro nodo ed esso la accetta, aprendo un canale di comunicazione. Questa richiesta si esplicita in maniera diversa a seconda dei protocolli utilizzati, e nell'architettura client/server denota quasi sempre chi è client e chi è server: il server attende le connessioni e i client gliele richiedono. Anche nel caso del p2p le connessioni non nascono spontaneamente, ma devono essere iniziate da una delle parti in causa. La differenza sostanziale dal paradigma client/server sta nel livello gerarchico dei due nodi: chi dà inizio alla connessione non è necessariamente ad un livello inferiore di chi la accetta.
In una rete p2p i casi possibili sono:

- Non esistono client o server;
- Esiste un server centrale che svolge funzioni di coordinamento dei client, ma i client hanno la possibilità di stabilire connessioni fra loro senza che il server debba intervenire;
- Tutti possono essere contemporaneamente client e server.

Il servizio talk di UNIX è un esempio del primo caso, Napster del secondo, le reti con risorse condivise e Gnutella del terzo. Esaminiamoli brevemente.

1.5.1 Il talk

Il talk è una delle tante utility che UNIX deve a BSD. La prima versione di talk fece infatti la sua comparsa nello UNIX 4.2BSD sviluppato dall'Università della California di Berkeley (UCB), pubblicato nel 1983. Per inciso, in 4.2BSD esordivano quelle che oggi sono pietre miliari del networking: fu infatti il primo sistema operativo che utilizzava integralmente il TCP/IP per le connessioni di rete, e in esso fecero la loro comparsa i socket e la memoria virtuale.
Talk è una utility di UNIX che consente a due utenti, localizzati sulla stessa macchina o su due diverse, di "chattare" fra loro attraverso una tty (ad esempio una finestra di terminale). Il meccanismo di funzionamento è semplice: se si vuole parlare con un utente connesso ad un altro host, si esegue il comando: *talk utente@host*
Dall'altra parte l'utente si vedrà comparire su un terminale un messaggio che lo avverte della richiesta di connessione e gli indica quale comando digitare per accettarla. Una volta digitato

il comando, le schermate dei terminali si divideranno in due metà orizzontali: nella metà superiore ogni utente vedrà ciò che scrive lui, in quella inferiore ciò che scrive l'interlocutore.
Gli utenti possono scrivere contemporaneamente: il contenuto del terminale è aggiornato in sincrono e la comunicazione viene chiusa quando uno dei due utenti chiude il programma con la sequenza di interruzione definita dal sistema o con la pressione contemporanea di CTRL e C dalla tastiera.
Talk è un programma un po' arcaico, ma dà l'idea di un protocollo in cui, una volta che la connessione è stabilita, nessuno dei partecipanti può dirsi server o client: i due host sono gerarchicamente alla pari.

1.5.2 Napster

Napster [11] è stato sviluppato nel 1999 dal lavoro di un giovane studente universitario americano della North-easterm University di Boston, Shawn Fanning.
Napster offriva agli utenti la possibilità di scambiarsi file musicali; permetteva la ricerca dei brani attraverso il nome dell'artista e/o il titolo del brano.
Essendo un software libero, il primo del suo genere, Napster ha avuto una diffusione capillare all'interno della Rete fino a quando è poi tramontato a causa delle implicazioni legali relative all'infrazione del copyright.

Napster impiegava una soluzione p2p ibrida [§2.2.3] strutturalmente molto simile a IRC, ma con alcune differenze importanti alla base: IRC ha l'obbiettivo di consentire ad un numero potenzialmente illimitato di utenti di chattare in canali comuni, avere comunicazioni private e scambiarsi file; Napster, oltre a fornire un servizio di chat funzionalmente analogo a IRC, è fortemente orientato allo scambio di file mp3; IRC è un protocollo client/server, Napster è peer-to-peer (in realtà non lo è completamente, ma la parte di comunicazione fra i client è marcatamente peer-to-peer).
Nel caso di IRC il server media tutte le comunicazioni fra i client. Nel caso di Napster il server non interviene nella comunicazione fra due client, ma svolge solo una funzione di collettore di informazioni sui file mp3 messi a disposizione dai client collegati e una funzione di motore di ricerca su tali informazioni.
Quando un utente ha individuato il file mp3 che intende scaricare, lo seleziona attraverso l'interfaccia di Napster; il programma richiede al server i dati che gli sono necessari a stabilire la connessione con il peer che ospita quel file e, una volta stabilita la connessione, il client

inizia il download e informa dell'evento il server, così come lo informa quando il download giunge a termine per qualunque motivo. Si noti dunque che un server Napster non partecipa allo scambio dei file, che avviene direttamente fra i client.

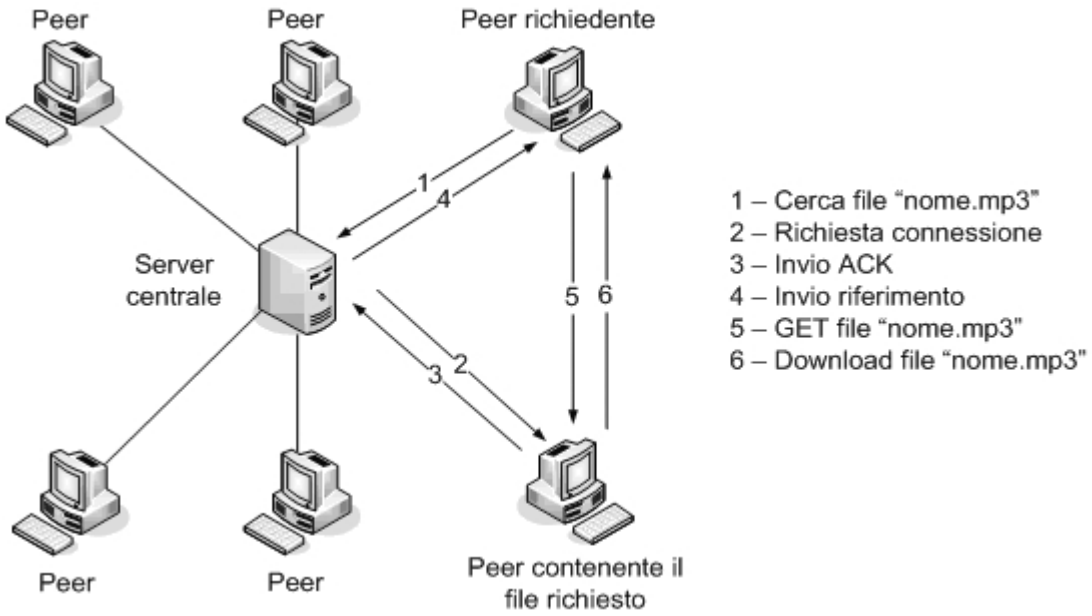


FIGURA 1 – Esempio di una comunicazione Napster

Quando si utilizza Napster, tutte le azioni intraprese possono essere rintracciate. Infatti, quando si avvia Napster, oppure quando si invia un comando di connessione dopo essersi disconnessi, viene contattato il server centrale di Napster, il quale ridirige l'utente verso un server membro vicino a lui (la distanza è calcolata in base all'indirizzo IP e al numero di passaggi che lo separano dai server disponibili); a quel punto, Napster collega l'utente al server in questione.

Ogni server membro tiene traccia di chi si collega e dei loro indirizzi IP. Dato che Napster funziona stabilendo una connessione diretta tra singoli utenti, i pacchetti di file trasferiti non passano attraverso il sistema Napster (ecco perché è normalmente possibile continuare a scaricare file quando l'utente che li condivide passa ad un server Napster diverso, ma non quando esce da Napster). Napster può tenere traccia, però delle ricerche, degli upload e dei download effettuati dagli utenti, mentre l'ISP (Internet Service Provider) può collegare un indirizzo IP a un account utente in base alla data e all'ora. Napster può inoltre tenere traccia di tutti i file che vengono condivisi.

Napster è tuttora presente nel modo del p2p, ma ha tramutato il concetto di accesso alle risorse: attraverso la partnership con la Bertelsmann AG, una impresa multinazionale tedesca del settore delle comunicazioni, è stato sviluppato un servizio basato sulla sottoscrizione che permette agli utenti di accedere a pagamento al catalogo musicale.

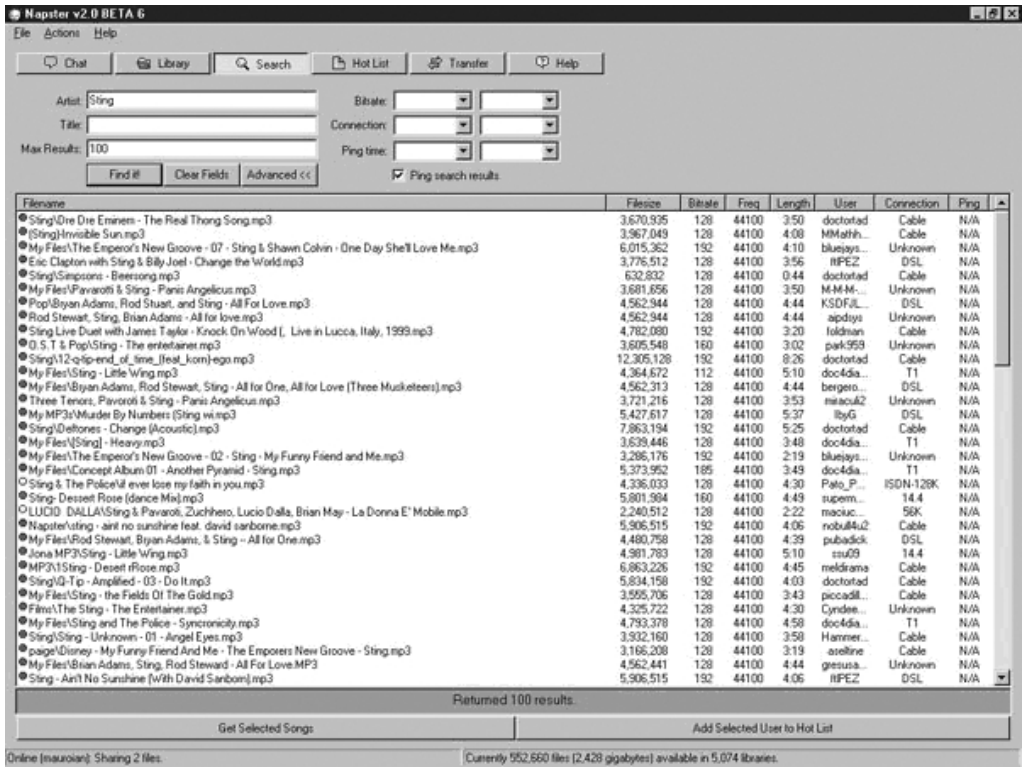


FIGURA 2 – L'interfaccia di Napster

1.5.3 Le reti con risorse condivise

Le reti con risorse condivise, essendo largamente diffuse e utilizzate, forse sono l'esempio più immediato di una architettura in cui i servizi vengono erogati in peer-to-peer.

Ogni host facente parte di una rete locale ha la possibilità di condividere delle risorse, dove con risorsa si intende un qualsiasi dispositivo fisico connesso al computer (una stampante, uno scanner, un'unità CD-ROM o DVD, un disco…) o una risorsa "logica" (una partizione di un disco, una cartella, ecc.) e di utilizzare le risorse che gli altri mettono a disposizione. In un ufficio è abbastanza comune avere dei posti di lavoro che condividono e usano un'unica stampante collegata ad una delle macchine e una cartella condivisa comune dove vengono depositati i file e i documenti che devono essere disponibili a tutti gli utenti per la consultazione; il tutto senza che le macchine che rendono disponibili quelle risorse siano

specificamente dedicate allo svolgimento di quel compito e con la possibilità di sfruttare le risorse condivise da altre macchine sulla rete.

Si noti come questo caso sia nettamente diverso dal precedente. Mentre nel caso del talk le definizioni di client e server non hanno nessun senso, qui un host può essere, anche contemporaneamente, client quando utilizza risorse condivise e server quando le mette a disposizione.

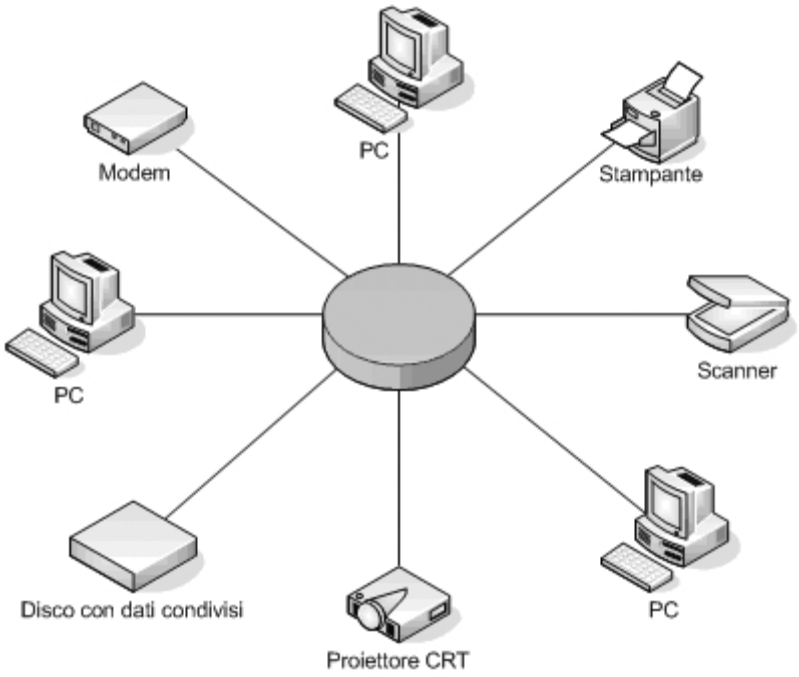


FIGURA 3 – Esempio di una rete con risorse condivise

1.5.4 Gnutella

Gnutella [12] è un protocollo peer-to-peer, uno dei più famosi, dove ciascun nodo può funzionare contemporaneamente da client e da server; la sua storia, così come quella di altri programmi simili, è piuttosto travagliata.

Gnutella fu rilasciato al pubblico da Slashdot [13] il 14 Marzo 2000 da Justin Frankel e Tom Pepper, i fondatori della Nullsoft (ricordiamo il lettore WinAmp, software freeware che consente di riprodurre ed organizzare file multimediali digitali).

Gnutella è il nome di una tecnologia, e non è né un programma né un unico sito Web: è piuttosto un modo di concepire la Rete. Per accedere ai servizi è necessario un servant (server più client), cioè un software per cercare e scaricare.

Gnutella fa parte di quella categoria di programmi che vengono classificati come "sistemi distribuiti per la condivisione di file" (Distributed systems for file sharing).

Esistono molte applicazioni (dette Gnutella client) che utilizzano il protocollo Gnutella, e che permettono quindi la creazione di una rete molto più estesa e robusta. Dobbiamo evidenziare anche la grandissima versatilità dei client Gnutella; questi infatti sono disponibili per i sistemi operativi principali quali Windows (BearShare, Gnucleus, LimeWire, Morpheus, Phex, Shareaza, Xolox), Linux/Unix (Gtk-Gnutella, LimeWire, Mutella, Phex, Qtella), Macintosh (LimeWire, Phex). Gnutella quindi da la possibilità a tutti i suoi utenti di mettersi in comunicazione e lo fa in modo che chiunque possa far parte della sua rete, a prescindere dal sistema operativo.
Una rete Gnutella è formata da un certo numero di nodi (gnodes) che condividono file di qualsiasi tipo. Il protocollo Gnutella non prevede la presenza di un server centrale che raccoglie informazioni su quanti e quali gnode compongono la rete; invece tutti i gnode cooperano pubblicando le informazioni in loro possesso ogniqualvolta ciò sia necessario con un meccanismo tutto sommato semplice.
In Gnutella le ricerche sono effettuate in maniera sostanzialmente anonima, in quanto le query (richieste di risorse) vengono propagate in flooding (cioè inondando i nodi vicini) e non contengono l'indirizzo del mittente e utilizza il TTL per evitare di congestionare la rete.
Ma come può un nodo isolato a prendere contatto con una rete esistente senza conoscere l'indirizzo di almeno un altro gnode la risposta è… non può; e in questo è del tutto simile al talk. Per ovviare a questo problema esistono dei cache server, che hanno il solo compito di tenere contatti con il più alto numero di client possibile e propagare le informazioni che raccolgono.

Sposando l'esigenza di realizzare soluzioni sempre più decentralizzate, il progetto Gnutella ha rafforzato il concetto di file sharing proposto da Napster, eliminando la necessità del server centrale per fornire le funzionalità di ricerca. L'indipendenza da un server di rete, combinata con la possibilità di scambiare file di qualunque tipo, fa di Gnutella uno degli esempi più potenti di tecnologia p2p. I "pari" nella rete Gnutella sono responsabili non solo di fornire i file, ma anche di rispondere alle richieste e ai messaggi di routing degli altri peer: essi operano dunque sia da client che da server e sono denominati "servent".
Il protocollo Gnutella è molto studiato ed è in continua evoluzione. La versione originaria, denominata 0.4, si basava su un'architettura completamente decentralizzata, ma oggi si tende all'introduzione dei "supernodi", cioè punti della rete con maggiori funzionalità rispetto agli altri.

1.5.4.1 L'architettura

Come mostrato in figura, ogni servent della rete è connesso ad un numero limitato di altri servent (tipicamente compreso tra 1 e 15) in modo completamente decentralizzato. Tali connessioni non sono fisiche, ma logiche, in virtù del fatto che la rete Gnutella si appoggia, ma non rispecchia, la topologia reale di Internet. Tale architettura però non è scalabile.

Dato che nessun servent assume un ruolo privilegiato, quindi non c'è un punto d'accesso principale alla rete, la connessione a Gnutella richiede che un peer conosca l'indirizzo IP di un host già connesso. Per risolvere questo problema, relativo alla prima connessione, è stato stabilito che un certo numero di peer con indirizzi IP statici o risolvibili fungano da starting point per l'accesso alla rete da parte degli altri peer: l'elenco di questi indirizzi IP, relativi ai nodi attivi in un certo istante, viene pubblicato su un sito noto in una lista aggiornata in tempo reale. Una volta connesso ad un nodo, un peer può ricevere la lista dei nodi attivi in quel momento in rete ed eventualmente aumentare il suo numero di connessioni.

Originariamente tutti i nodi di Gnutella erano connessi gli uni agli altri in maniera casuale ed erano in questo assolutamente uguali. Questo meccanismo funziona bene con le connessioni a broadband ma non per gli utenti con modem lenti. Quel problema può essere risolto organizzando la rete in una forma più strutturata.

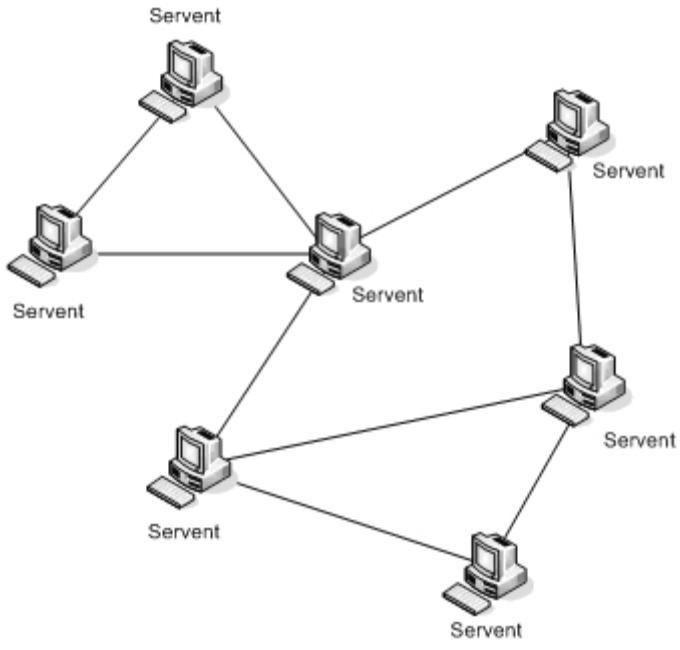


FIGURA 4 – Architettura di Gnutella 0.4

Il sistema degli Ultrapeer è stato trovato efficace per risolvere questo problema. Esso consiste nel dare una struttura gerarchica alla rete Gnutella dividendo i nodi sulla rete in *leaf* e *Ultrapeer*. Un leaf mantiene solo un piccolo numero di connessioni aperte e queste connessioni vengono stabilite con altri nodi di tipo UltraPeer. Un Ultrapeer invece agisce un poco come un proxy verso la rete Gnutella per i leaf connessi ad esso. Questo riduce il numero di nodi coinvolti nella manipolazione e nella circolazione dei messaggi sulla rete Gnutella e allo stesso tempo riduce il traffico reale tra i vari nodi.
Gli ultrapeer sono connessi gli uni agli altri e sono connessi anche ai "normali" nodi Gnutella (ovvero quei nodi che non implementano il sistema degli Ultrapeer magari perché usano client Gnutella che non fanno uso di questo schema).

1.5.4.2 Gestione dei messaggi

La comunicazione e lo scambio di informazioni tra i nodi connessi avviene tramite messaggi che possono essere di tre tipi:

- *Multicast*: questi messaggi vengono inviati da un nodo della rete verso tutte le connessioni attive e servono sia per effettuare le ricerche (*Query*) che per segnalare la propria presenza in rete (*Ping*), cercando contemporaneamente di scoprire gli altri host connessi. Un nodo che riceve un messaggio di questo tipo deve propagarlo a tutti i nodi ai quali è connesso, ad eccezione di quello da cui l'ha ricevuto;
- *Unicast*: questi messaggi servono per la comunicazione tra due nodi connessi alla rete Gnutella attraverso la rete stessa. Tipicamente si tratta di risposte a messaggi multicast (*QueryHit* in risposta a Query e *Pong* in risposta a Ping), ma c'è anche un altro messaggio (*Push*) che serve a mettere in comunicazione diretta nodi protetti da firewall;
- *Diretto*: questi messaggi consentono a due nodi di comunicare con connessione IP diretta senza utilizzare la rete Gnutella. Servono essenzialmente per gestire e trasferire le risorse.

Tutti i messaggi sono caratterizzati da un contatore di Hop, ovvero di passaggi da un nodo ad un altro, e da un Time to Live (TTL), che definisce dopo quanti hop il messaggio deve essere eliminato: infatti ogni nodo attraversato decrementa il valore di TTL e, quando raggiunge il valore zero, il nodo che ha ricevuto il messaggio è tenuto ad eliminarlo. Il TTL è impostato su 7 livelli.

Il routing dei messaggi nella rete Gnutella avviene in modo tale che vengano soddisfatti tre requisiti:

- Il messaggio deve essere integro;
- Il valore di TTL deve essere maggiore di zero;
- Il messaggio non deve essere stato precedentemente inoltrato da quel nodo.

Questa ultima condizione è posta per evitare che si creino dei loop nella rete e viene verificata tenendo traccia degli identificatori dei messaggi che sono passati, relativamente ad ogni connessione. Oltre all'ID, il nodo tiene traccia anche del tipo di messaggio e della connessione a cui si riferisce: questo permette ai messaggi di risposta di ritrovare la strada da cui sono pervenute le rispettive richieste.
Una volta ricevuta la risposta ad una richiesta e una volta selezionata la risorsa da scaricare, si stabilisce tra i due servent (richiedente e offerente) una connessione diretta di tipo HTTP: il file non viene mai trasferito attraverso la rete logica Gnutella, ma fluisce in modo diretto da un servent all'altro.

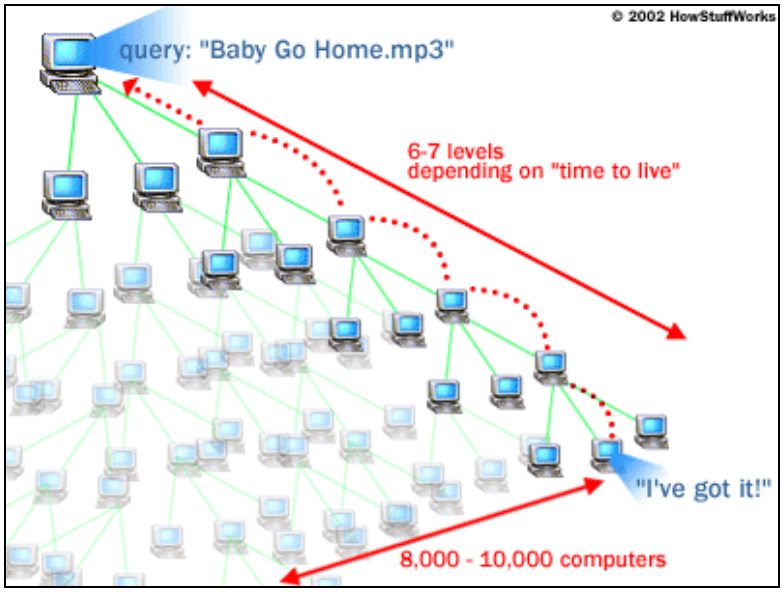


FIGURA 5 – Esempio di flooding nella rete Gnutella

Gnutella è ad oggi uno degli esempi più significativi di rete p2p pura, ma l'eliminazione del server centrale ha chiaramente introdotto altre problematiche:

- Come fanno i peer a distribuirsi i messaggi senza inondare la rete?
- Come fanno i peer a fornire contenuti in modo sicuro e anonimo?
- Come può la rete incoraggiare la condivisione delle risorse?

Gnutella non fornisce una risposta a queste domande, ma nonostante queste limitazioni funziona ed è largamente utilizzata in Internet. Altre varianti del file sharing, come Freenet e MojoNation, hanno cercato di venire incontro ad alcune di queste esigenze, in particolare Freenet fornisce un sistema di anonimato decentralizzato per pubblicare i dati e limitarne l'alterazione attraverso una forte crittografia e MojoNation utilizza una moneta artificiale, chiamata Mojo, per rinforzare la condivisione delle risorse.

1.6 I protocolli e le applicazioni

Per quanto riguarda lo scambio di informazioni esistono svariati tipi applicazioni per raggiungere tale scopo. Alla base di queste reti stanno spesso motivazioni non solo meramente utilitaristiche, ma anche ideologiche: la libertà di parola, la libertà di comunicazione, lo scambio di idee ed opinioni, il libero flusso di informazione.

Formato:

- rete/protocollo
    - applicazione che usa la rete

- BitTorrent protocollo
    - BitTorrent
    - BitTorrent++
    - ABC
    - Azureus
    - BitAnarch
    - SimpleBT
    - BitTorrent.Net
    - Shareaza
- Direct Connect rete
    - NeoModus Direct Connect
    - DC++
    - BCDC++
    - CZDC++
- eDonkey rete
    - eDonkey2000
    - eMule
    - mlDonkey
    - Shareaza
- FastTrack protocollo
    - KaZaA, KaZaA Lite, K++ Diet KaZaA
    - Grokster
    - iMesh
    - giFT
- MANOLITO/MP2P network
    - Blubster
    - Piolet
    - RockItNet
- Napster network
    - OpenNap
    - WinMX
    - Napigator
    - FileNavigator
- WPNP network
    - WinMX
- Joltid PeerEnabler
    - Joltid
    - Altnet
    - Bullguard
    - Sharman Networks (KaZaA)
- altri networks
    - Akamai
    - Alpine
    - Ares Galaxy
    - Audiogalaxy network
    - The Bridge
    - Carracho
    - Chord
    - The Circle

- Freenet rete
  - Freenet
  - Frost
  - Entropy (con una sua propria rete)
- Gnutella rete
  - Acquisition (Mac OS)
  - BearShare
  - Gnucleus
  - Limewire (Java)
  - Morpheus
  - Phex
  - Swapper
  - Shareaza
  - XoloX
  - gtk-gnutella (Unix)
- Gnutella2 rete
  - Shareaza
  - MLDonkey
  - Gnucleus
  - Morpheus
  - Adagio
  - Dexter
  - EarthStation 5 network
  - Evernet
  - FileTopia
  - GNUnet
  - Grapevine
  - Groove
  - Hotwire
  - IRC @find and XDCC.
  - JXTA
  - konspire2b
  - MojoNation Mnet
  - MUTE
  - OpenFT
  - Overnet network
  - Scribe
  - SongSpy network
  - Soulseek
  - SquidCam
  - Swarmcast
  - Waste
  - Winny (Giapponese)
- Kademlia tipologia di rete (le varie implementazioni non sono compatibili tra loro)
  - eMule la usa in aggiunta alla rete eDonkey
  - Azureus lo usa per i torrent tracker-less (o quando il tracker è down)

Ecco di seguito descritti alcuni sistemi di scambio più popolari e/o che si sono distinti per alcune proprietà: Gnutella2, FastTrack, Morpheus, eDonkey, eMule, BitTorrent, OpenNap, WinMX e Direct Connect.

1.6.1 Gnutella2

Gnutella2 [14] è una piattaforma di terza generazione, moderna ed efficiente, progettata per offrire delle fondamenta solide a servizi distribuiti globali come la comunicazione punto a punto, trasferimento dati e altri servizi futuri.

Il nome Gnutella2 deriva sai due componenti di cui racchiude il significato : "*Gnutella2 lo Standard*" e "*Gnutella2 la rete*".

Gnutella 2 è la rete principale usata dall'applicazione Shareaza [15] e da altri client compatibili.

Gnutella2 è una rete esclusiva rispetto alle attuali reti p2p sotto diversi punti di vista:

- Molte delle attuali reti di successo sono "chiuse" ovvero sono di proprietà di una singola entità con restrizioni più o meno marcate. Questo non è un modello

percorribile per realizzare una rete aperta e soprattutto che abbia uno scopo generico. Gnutella2 ha una architettura aperta dove ognuno può partecipare e contribuire. La rete è stata progettata per consentire una tale diversità senza la necessità di comprometterne l'integrità;

- La maggior parte delle reti sono dedicate esclusivamente ad uno scopo, spesso la condivisione di files. Questo è si un'applicazione molto popolare per una tecnologia peer-to-peer ma non è certamente l'unica possibile. Gnutella2 è progettata come rete generica che può essere un solido punto di partenza per un gran numero di applicazioni peer-to-peer diverse - certamente il file sharing ma anche strumenti di comunicazione e altri idee che saranno concepite in futuro.

La *Rete Gnutella2* è forse la componente più facilmente riconoscibile. E' una nuova architettura di rete peer-to-peer ad alta performance sulla quale possono essere costruite un gran numero di applicazioni distribuite come le applicazioni per il file sharing.
Lo *Standard Gnutella2* è un insieme di specifiche richieste per costruire applicazioni che operino sulla rete Gnutella2 in diversi modi. Specifica il minimo livello di compatibilità richiesta per essere riconosciuta come applicazione compatibile con Gnutella2.

1.6.1.1 La rete

La rete Gnutella2 è una collezione auto-gestita di nodi interconnessi che cooperano per svolgere produttive attività distribuite.
Non tutti i nodi che partecipano al sistema sono uguali: ci sono due principali tipi di nodi, i nodi *hubs* (centro) e i nodi *leaves* (foglia). Lo scopo è di massimizzare il numero di leaves e di minimizzare il numero di hubs, tuttavia a causa della natura limitata delle risorse il rapporto praticabile tra leavers ed hubs è limitato. Questa quantità è definita *densità di leaf*.

- **Nodi leaf:** i nodi leaf sono i nodi più comuni; non hanno responsabilità speciali e non costituiscono una parte operosa dell'infrastruttura di rete. I nodi con risorse limitate devono operare come nodi leaf: per risorse limitate si intende banda limitata, CPU o RAM inadeguati, tempo in cui si resta on-line scarso, ed incapacità ad accettare connessioni TCP e UDP in ingresso;
- **Nodi hub**: i nodi hub d'altra parte costituiscono una parte importante, attiva dell'infrastruttura di rete, organizzando i nodi circostanti, filtrando ed indirizzando il traffico. I nodi hub sacrificano una parte delle loro risorse alla rete, e quindi la loro

capacità di partecipare a funzioni di rete di alto livello è limitata. Sono i nodi più capaci vengono eletti al grado di hub, in base a specifiche regole.

Oltre a questi fattori interni, bisogna anche considerare la necessità per la rete di avere altri hubs. Senza un punto centrale di autorità la necessità di hubs addizionali non può essere determinata con assoluta certezza; tuttavia può essere approssimata considerando lo stato dei nodi vicini e in particolare lo stato degli hubs nel cluster di hub locale (cluster = gruppo). I nodi che funzionano da hubs per Gnutella 2 hanno un insieme di responsabilità. Gli hubs sono fortemente interconnessi, a formare una *rete di hub*, o una *ragnatela di hub*, dove ogni hub è connesso ad altri 5-30 vicini hubs.
Il numero di interconnessioni di hub può crescere al crescere delle dimensioni della rete; ogni hub accetta anche connessioni da una numerosa collezione di nodi leaf, tipicamente 300-500 in dipendenza dalle risorse disponibili. I nodi leaf si connettono simultaneamente a 3 hub, tuttavia dal punto di vista degli hubs, ogni leaf viene visto come un collegamento finale.

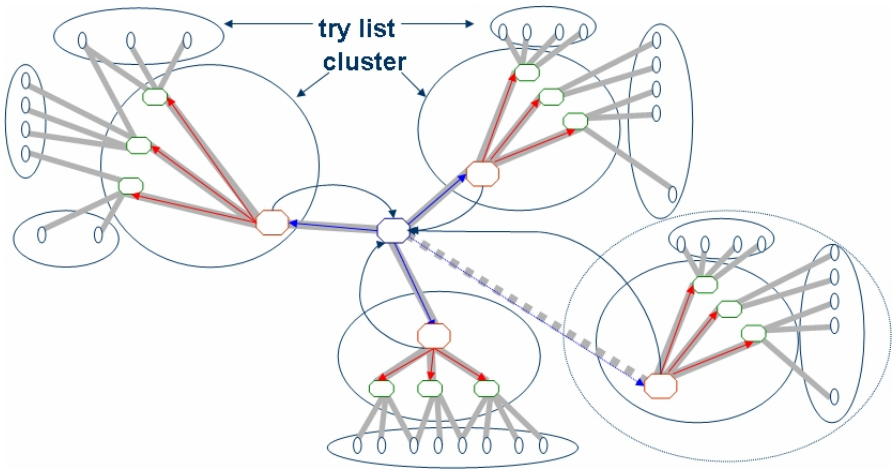


FIGURA 6 – Esempio di comunicazione Inter-Hub

Il gruppo di hubs all'interno della rete, che comprende l'hub locale e i suoi vicini, viene chiamato hub cluster e rappresenta una modalità di raggruppamento importante. Gli hub cluster conservano una comunicazione costante con gli altri cluster, condividendo informazioni sul carico della rete e statistiche, scambiando riferimenti di cache e filtrando le tavole degli hash. L'hub cluster rappresenta la più piccola unità di rete in cui è possibile svolgere delle ricerche sulla rete.

Tra le responsabilità degli hubs ci sono:

- mantenere aggiornate le informazioni relative agli altri hubs nel cluster, e i loro hubs vicini, fornendo gli aggiornamenti;
- conservare un indice delle risorse condivise da tutte le connessioni che l'hub stesso stabilisce, ovvero le risorse condivise dagli hubs vicini e soprattutto dai leaf connessi all'hub;
- monitorare lo stato delle connessioni locali per decidere se declassarsi allo stato di leaf, e mantenere aggiornati i servizi di connessione (ad esempio le GWebCaches);
- una done list (i nodi a cui abbiamo inviato una query più i loro vicini);
- una try list (i vicini dei vicini dei nodi a cui abbiamo inviato una query).

Gli ultimi due punti devono essere "mantenuti" per ogni query.
Questi due tipi di nodo, hub e leaf, durante la connessione iniziale ad un altro nodo, devono dichiarare il proprio tipo, cioè se nodo di tipo hub o leaf, e fornire informazioni riguardo le proprie capacità.

1.6.1.2 La comunicazione

L'architettura di Gnutella2 fa largo uso di una compressione di "deflazione". Il supporto dei collegamenti TCP compressi non è uno standard Gnutella2, ma è fortemente raccomandato. L'UDP (User Datagram Protocol) è una risorsa inestimabile nei sistemi p2p poiché fornisce uno strumento a basso costo (basso overhead) per inviare piccoli messaggi irregolari a richiesta ad un numero elevato di nodi. Stabilire una connessione TCP con un nodo semplicemente per trasferire un singolo pacchetto di informazione è uno spreco in termini di dati e di tempo per i nodi. Quando si ha a che fare con un grande numero di nodi questi costi diventano insostenibili. L'UDP fornisce una soluzione e rende possibile questo tipo di interazione.
Tuttavia l'uso di pacchetti UDP non è affidabile: i pacchetti possono andare persi per strada per diverse ragioni. Spesso questo comportamento è desiderabile, per esempio, quando la connessione del nodo di destinazione è molto congestionata i pacchetti UDP saranno molto probabilmente scartati. Se il contenuto non è critico questa perdita è appropriata poiché le risorse dell'host sono effettivamente non disponibili. In altri contesti purtroppo la mancanza di affidabilità è un problema: spesso o il mittente ha bisogno di assicurarsi che il destinatario ha ricevuto il messaggio o deve sapere che il destinatario non è al momento disponibile.

La rete Gnutella2 ha risolto questo problema implementando un nuovo selezionabile strato affidabile al di sopra del protocollo UDP di base. Questo strato affidabile ha alcune funzioni simili al TCP, ma non fornisce uno stato circa la connessione guadagnando così in efficienza. Questo permette a Gnutella 2 di selezionare il mezzo di comunicazione ottimale per ogni tipo di trasmissione:

- Se bisogna scambiare una grande quantità di dati, o se vengono inviati ad uno stesso destinatari più dati in successione, viene stabilita una connessione di tipo TCP;
- Se deve essere trasferito un volume piccolo di dati in un'unica operazione o in modo irregolare, viene utilizzato l'UDP affidabile;
- Se deve essere trasferito un piccolo volume di dati non importanti in un'unica operazione o in modo irregolare, viene utilizzato l'UDP non affidabile.

Le ricerche all'interno del cluster sono eseguite utilizzando il protocollo TCP, mentre le ricerche fra cluster utilizzano quello UDP.

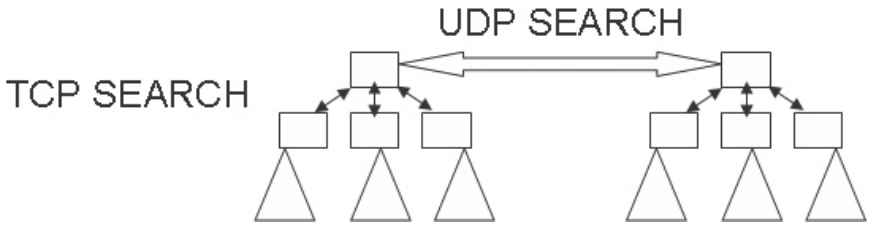


FIGURA 7 – Le ricerche nei cluster

1.6.1.3 La ricerca

Ogni nodo Gnutella2 deve conservare un elenco non completo di hubs conosciuti a livello globale e un elenco completo degli hubs che partecipano nei clusters di hub adiacenti. Il meccanismo di ricerca di oggetti distribuiti sulla rete è una componente importante di Gnutella2. Consente ad un client di localizzare oggetti distribuiti sulla rete in modo ottimo, richiedendo e ricevendo un sottoinsieme di tutte le informazioni conosciute su quel oggetto. Il meccanismo di ricerca degli oggetti su Gnutella 2 viene descritto come una successione iterativa di tentativi con una serie di importanti ottimizzazioni ricavate dalla topologia e dai componenti di rete:

- Un client contatta iterativamente gli hub noti inviando loro la sua ricerca;
- Gli hub utilizzano le parole di ricerca che hanno ricevuto dai loro vicini;
- Quando viene trovata una corrispondenza, la ricerca viene propagata agli altri hub;

- La ricerca si estende almeno ad un cluster, che è l'unità base di ricerca;
- I nodi, che ricevono la richiesta di ricerca filtrata, la processano ed inviano i risultati direttamente al client che aveva avviato tale ricerca.

1.6.2 FastTrack

FastTrack [16] rappresenta la seconda generazione dei protocolli p2p con architettura semi-centralizzata ed è utilizzato dai programmi di file sharing come KaZaA, Grokster ed iMesh..
Si basa sul protocollo Gnutella e lo evolve attraverso il concetto di supernodo e migliorandone la scalabilità, ma non è Open Source. La funzionalità dei supernodi è costruita attorno al client: un computer molto potente avente una linea veloce e che si connetta con un programma client diventa automaticamente un supernodo, in pratica agisce come un indice provvisorio per gli utenti più lenti.
Funziona in questo modo: al primo collegamento il programma si incorpora (hardcoded) una lista di supernodi, sotto forma di numeri IP. Il client individuerà il supernodo più vicino funzionante e riceverà una lista dei supernodi attivi e correnti da usare per futuri tentativi di connessione. Il client, ottenuto un supenodo come sua "sorgente" (upstream) invierà una lista di file con l'intento di condividerli a quel supernodo, e una richiesta di ricerca. Il supernodo comunica con altri supernodi allo scopo di soddisfare la richiesta di ricerca. Poi il client si collega direttamente (peer) per iniziare lo scaricamento (download) del file. Questo trasferimento viene eseguito utilizzando il protocollo HTTP. Per permettere lo scaricamento di files da sorgenti multiple, FastTrack impiega un algoritmo di hashing chiamato "UUHash". Questo algoritmo ha la capacità di decodificare file molto grandi, ma ha dei difetti che causano il danneggiamento dei files stessi senza la possibilità di accorgersene. Molti, tra cui la RIAA, hanno sfruttato questa vulnerabilità per inviare sulla rete file corrotti.

Il protocollo FastTrack utilizza una crittografia dei dati non documentata dal suo creatorore, così come lo era il software del primo client. Programmatori Open Source sono riusciti a decompilare la parte di protocollo che si occupa della comunicazione con il client-supernodo; la parte di protocollo che permette la comunicazione tra supernodi rimane in gran parte sconosciuta.
I dati di inizializzazione per effettuare la codifica dei dati mediante l'algoritmo vengono inviati in chiaro senza l'utilizzo di chiavi pubbliche codificate. È in questo modo che è stato possibile effettuare una decompilazione relativamente semplice di questa parte del protocollo.

1.6.3 Morpheus

Morpheus [17] è comparso per la prima volta on-line nella primavera del 2001 utilizzando la rete Gnutella, e successivamente anche Gnutella2, per le ricerche. È stato facile per Morpheus conquistarsi una così vasta popolarità. Il programma in questione ha ripreso le migliori caratteristiche di Napster e, per quanto fosse possibile, le ha addirittura migliorate. Innanzitutto con Morpheus è possibile scambiare con gli utenti connessi file di ogni genere: oltre a quelli audio, infatti sono disponibili video, immagini e addirittura interi programmi per PC. Un passo avanti niente male rispetto a Napster, soprattutto per gli amanti dei videoclip (ce ne sono a migliaia, ma "pesano" molto per poter garantire una qualità accettabile) e per chi del proprio artista preferito vuole veramente avere tutto. Con l'opzione "Everything" infatti il motore di ricerca elenca tutti i file disponibili legati alla parola chiave inserita.

L'altra innovazione fondamentale di Morpheus è rappresentata dal metodo in cui viene effettuato il download di un file. Con Napster era quasi impossibile portare a termine un download se il collegamento alla rete veniva interrotto oppure se l'user a cui apparteneva il file si disconnetteva improvvisamente. Grazie alla tecnologia denominata SmartStream, Morpheus è invece in grado di risolvere questo problema permettendo ai suoi utenti di scaricare il file desiderato da più fonti simultaneamente. Così facendo viene incrementata la velocità del download e non c'è mai il rischio di perdere un download in corso se l'utente si disconnette. A quel punto, infatti, si continua il download da un altro user che ha nel suo archivio lo stesso brano.

1.6.4 eDonkey

eDonkey [18] (in breve ed2k) è un'applicazione più recente di Gnutella, la cui caratteristica principale è la particolare architettura: essa infatti ha un'architettura semi-centralizzata che si basa su un insieme di server sparsi in tutto il mondo. Tale rete si basa fondamentalmente sul protocollo Multisource File Transfer Protocol (MFTP).

Il MFTP è basato sul familiare protocollo FTP per il trasferimento dei files. Il vantaggio principale dell'MFTP è che permette di scaricare lo stesso file da diverse fonti contemporaneamente consentendo quindi di raggiungere alte velocità di trasferimento. In questo modo gli utenti che uplodano lentamente, possono essere raggruppati insieme fino a fornire velocità di download accettabili.

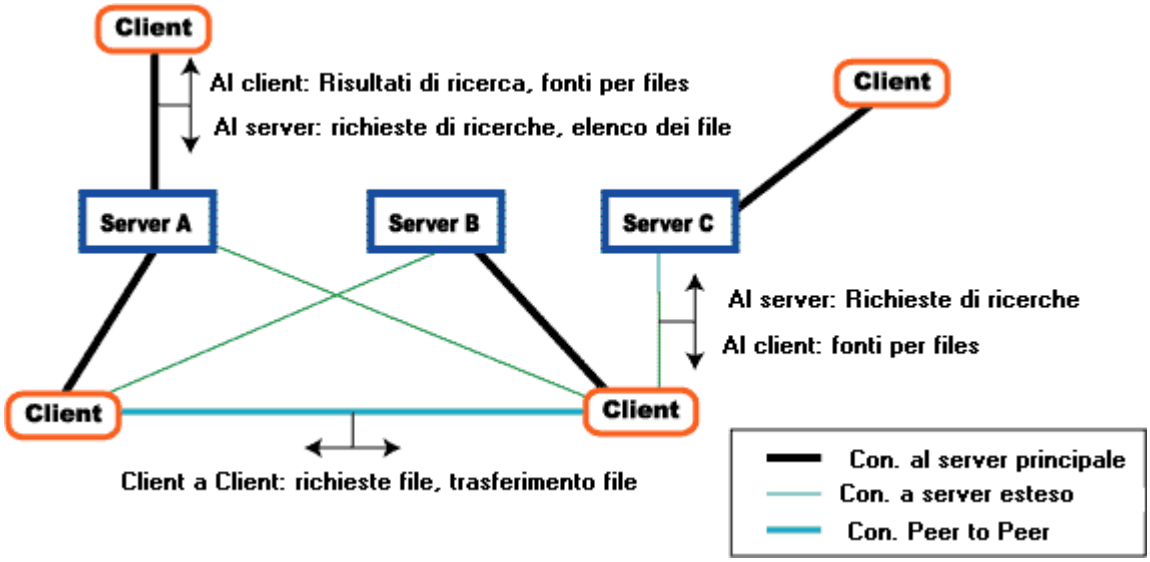


FIGURA 8 – Le connessioni di una rete ed2k

I server sono gestiti dagli utenti stessi dell'applicazione, i quali mettono a disposizione alcuni KByte della loro connessione Internet per servire centinaia o migliaia di utenti e svolgono solo servizi di directory: forniscono una lista sempre aggiornata dei client ad essi connessi e delle risorse tra loro condivise e un'ulteriore lista di server vicini per permettere ai client di estendere le ricerche. I server gestiscono tutte le operazioni di ricerca e comunicano fra loro solo per segnalare la propria presenza, mentre il trasferimento delle risorse avviene con connessione diretta client-client.

Le richieste ed i risultati vengono spediti via UDP per limitare la banda e l'overhead di connessione per i servers.

La figura seguente riporta alcune delle caratteristiche avanzate del meccanismo di download della rete ed2k.

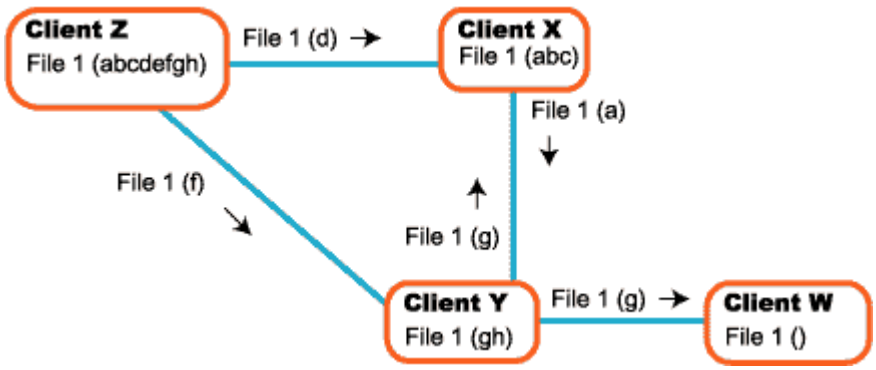


FIGURA 9 – Esempio di trasferimento di dati nella rete ed2k

Il Client Z ha tutte le parti del File 1 (le lettere in minuscolo rappresentano le parti del file).
I Client W,X e Y vogliono tutti scaricare il File 1. Poiché i Client X e Y hanno parti diverse del File 1 possono non solo ottenere il file da Z, ma anche cominciare ad inviarsi parti di files tra di loro. Questo consente al file di diffondersi in modo più veloce senza utilizzare solo la banda del Client Z. Il Client W può cominciare a scaricare il file anche se la fonte del file (Client Z) non ha più banda sufficiente per inviare il file.

Vi sono diversi client Open Source e free che sfruttano la rete di eDonkey: eMule è un client Windows Open Source, xMule è un'implementazione di eMule per diversi sistemi operativi, aMule è un client per Mac, Windows e Linux, Solaris e la piattaforma BSD, MediaVAMP, l'unico client coreano, mentre MLDonkey è un client a software libero che può essere eseguito in diversi sistemi operativi.

1.6.5 eMule

Tra i software di file sharing più utilizzati, un posto di rilievo spetta di diritto ad eMule [19].
Nasce nell'estate 2002 per iniziativa del programmatore tedesco Merkur e si afferma rapidamente come client di punta sulla rete, grazie alle sue caratteristiche innovative rispetto ai client già presenti sulla rete eDonkey.

Il suo strano nome trae origine da un gioco di parole utilizzato dal creatore del software intenzionato a realizzare un client peer-to-peer alternativo, in grado di viaggiare sulla rete eDonkey. È così che al più anziano eDonkey (letteralmente "asino elettronico") si affiancò un altrettanto efficiente ed infaticabile "mulo". Merkur creò il suo software seguendo la filosofia dell'Open Source, permettendo a tutti di visionare e modificare il codice sorgente del programma. E la chiave del successo che il programma continua a riscuotere è dovuto proprio alla collaborazione di decine di sviluppatori sparsi per il pianeta, che in pochi anni sono riusciti a potenziare il software con moduli aggiuntivi e versioni ottimizzate.

Ha una piacevole interfaccia grafica ed una vasta comunità di utenti, con tante possibilità di supporto. Inoltre supporta un vasto elenco di linguaggi ed offre la possibilità di chattare con gli altri utenti connessi alla rete attraverso il protocollo IRC.

eMule permette la condivisione e lo scambio di files attraverso le reti:

- *ed2k*: è la rete principale utilizzata da eMule e consta di numerosi server. Quando si stabilisce una connessione, l'elenco dei file condivisi viene inviato al server che aggiunge tali informazioni al database;
- *kad:* acronimo per Kademlia, è una rete priva di server nella quale ci si connette direttamente ai computer di altri utenti, raggiungibili solo se si conosce il loro indirizzo IP.

Tra le sue caratteristiche si segnalano:

- l'I.C.H. e l'A.I.C.H. (Intelligent Corruption Handling ed Advanced Intelligent Corruption Handling) che in presenza di corruzione dei dati ricevuti permette al client di effettuare il controllo e l'eventuale download di piccoli blocchi di dati;
- la compressione "al volo" dei dati, molto efficace nello scambio di file non compressi.

Su eMule non esistono server che contengano i files da scaricare, né esiste un unico server centrale. Su eMule i files risiedono esclusivamente nel disco rigido degli utenti connessi, ed i numerosi server, kad e ed2k appunto, servono solo per "indicizzare" i files condivisi dagli utenti stessi. L'oscuramento di un server non comporta gravi perdite in termini di risorse.
È un programma interamente Open Source sviluppato sotto GNU-GPL, i cui sorgenti sono disponibili affinché chiunque possa migliorarlo o personalizzarlo. In termini giuridici significa che non c'è un unico responsabile di eMule, ma una lunga lista di collaboratori, spesso anonimi, che hanno dedicato tempo e risorse al miglioramento di questo portentoso client p2p.
eMule offre numerose impostazioni e funzioni, ed è proprio per questo che ad un primo impatto può sembrare un programma ostico e complesso, ma in realtà permette ad ognuno di configurarlo in maniera ottimale; inoltre permette di scaricare files da più utenti contemporaneamente.

1.6.6 BitTorrent

BitTorrent [20] è un protocollo progettato per trasferire files. È fondamentalmente p2p perché gli utenti si connettono gli uni agli altri per inviare e ricevere parti di files. Tuttavia c'è un server centrale (chiamato Tracker) che coordina l'azione di questi utenti (detti anche "peers").

Il tracker si occupa solo delle connessioni, non ha idea di ciò che viene scambiato, e perciò possono essere serviti un gran numero di utenti con relativamente poco consumo di banda da parte del tracker.

La filosofia alla base di BitTorrent è che un utente deve *Uplodare*, ovvero trasmettere dati all'esterno nello stesso tempo in cui scarica il file. In questo modo viene utilizzata la banda della rete in maniera quanto più efficiente possibile. BitTorrent è stato progettato per lavorare meglio al crescere del numero delle persone che vogliono un certo file diversamente da altri protocolli di trasferimento file. Ogni utente che scarica il file diventa a sua volta fornitore dello stesso per tutti il tempo del download e così via a cascata generando quel torrente di bit da cui deriva il nome stesso del programma.
Questa soluzione permette di arrivare a velocità di download spesso molto vicine al proprio limite massimo di banda. Inoltre BitTorrent supporta il resume dei file riprendendo dal download parziale su disco, e per evitare il download di file corrotti usa un Hashing Crittografico (SHA1) di tutti i dati.

Creato da Bram Cohen, conosciuto per essere l'organizzatore del CodeCon (raduno di hacker ed esperti di sviluppo di applicazioni peer-to-peer e di sicurezza on-line, fortemente orientato alla tutela delle libertà individuali). Tra i client ricordiamo: BitTorrent Client, TorrentSpy, BitTorrent++ e Shareaza.

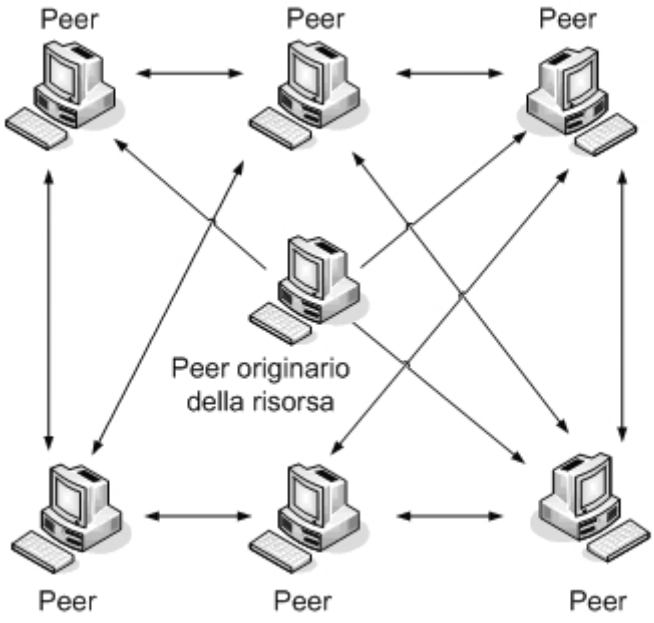


FIGURA 10 – Esempio di collaborazione alla diffusione della risorsa su BitTorrent

1.6.7 OpenNap

L'OpenNap [21] è discendente diretto di Napster. La rete di Napster era composta dai client e da circa 50 server centrali. Un server centrale di Napster indicizzava le cartelle condivise rendendole in questo modo disponibili virtualmente a chiunque si trovasse collegato alla sua rete. Ai tempi di Napster questa costituiva una tecnologia p2p all'avanguardia. La tecnologia dei suoi server fu sottoposta ad un'opera di "reverse engineering" e il codice sorgente divenne Open Source: da qui nacque l'OpenNap protocol.

Molte persone si sono interessate a costruire reti OpenNap. All'inizio il progetto OpenNap ha avuto un notevole successo ed una miriade di server di piccole, medie e grandi dimensioni popolarono il panorama del file sharing. Tra le reti più memorabili non possiamo non citare MusicCity e DJNap. Tuttavia la RIAA guardò tale successo come una minaccia e durante la primavera del 2001 fece chiudere molti server OpenNap, intervenendo legalmente direttamente presso gli Internet Service Provider (ISP).
Da allora l'OpenNap si è riorganizzato e si è ulteriormente evoluto. Il punto di forza della rete OpenNap è il fatto che è generalmente buona per trovare materiale raro e cose che possono non essere disponibili su altre reti, e forniscono anche ricerche più veloci e consistenti poiché in questo caso è il server centrale OpenNap che conserva le liste di tutti i files presenti sulla rete. Il rovescio della medaglia è rappresentato dalla difficoltà a volte di reperire questo materiale sia perché ci vuole spesso molto tempo per scaricarlo, sia perché qualche volta risulta completamente impossibile da scaricare. Una ragione per cui capita questo è il fatto che ci sono molti client differenti sulla rete che non sono sempre compatibili tra loro.

È possibile collegarsi a server OpenNap semplicemente usando un client OpenNap. Una volta collegati al server assistiamo ad un trasferimento attraverso protocollo TCP della nostra lista files. Questo permetterà ai server di indicizzare le risorse che abbiamo a disposizione e renderle disponibile nelle ricerche. I Server fungono solo da tramite: nel momento in cui decidiamo, in seguito ad una ricerca, di scaricare un determinato file il server ci metterà in contatto con gli utenti che hanno quel determinato file; il trasferimento del file avverrà mediante un collegamento peer-to-peer, ovvero i due client si metteranno in contatto tra di loro e cominceranno a trasferire i dati.

La rete OpenNap è strutturalmente organizzata seguendo i protocolli:

- *Client-Server*: per quanto riguarda ricerche e altre semplici operazioni in cui viene chiamato in causa il server;
- *p2p*: nel trasferimento effettivo dei file che non transitano mai per il server ma che interessano solo ed esclusivamente i client/computer (peers) interessati allo scambio.

Per accrescere la possibilità di ricerche, le loro dimensioni virtuali, e quindi la possibilità di fornire risultati, i server OpenNap possono essere collegati (linkati) tra di loro. In questo modo collegandosi ad uno solo di questi server si avrà accesso alle risorse di tutti i server accrescendo enormemente il database di file e di utenti che è possibile contattare.
Un certo numero di server OpenNap collegati fra loro danno vita ad una Rete OpenNap. Per accedere ad una rete OpenNap è sufficiente collegarsi solo ad uno dei server attivi che la costituiscono.
Le reti OpenNap operano in maniera del tutto indipendente dalla rete WinMX Peer Network ed offrono un rete centralizzata basata sul vecchio protocollo di Napster dove tutti gli utenti si connettevano al server centrale.
Tra i programmi che possono essere utilizzati per potersi connettere ai server OpenNap sono, ad esempio, OpenNap client, WinMX, Lopster e Napigator.

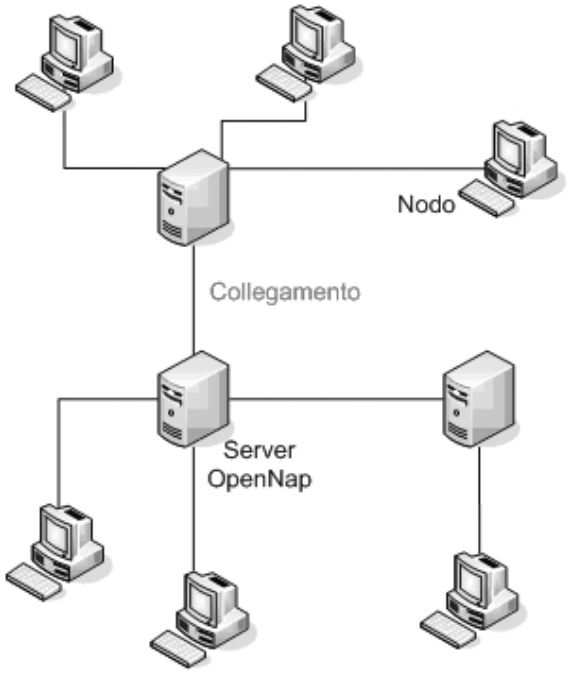


FIGURA 11 – Esempio di una rete con server OpenNap linkati

1.6.8 WinMX

WinMX [22] è una delle ultime reti nate nell'ambito della condivisione dei file. Ha una struttura decentralizzata come Gnutella ma risulta essere molto più efficiente in quanto il numero di utenti che vi si collegano è elevato e quindi la ricerca di un elemento risulta più esaustiva e soddisfacente.
Attraverso di esso ci si collega alla rete di server OpenNap, server che utilizzano il protocollo Napster, attraverso i quali viene realizzato lo scambio di qualunque tipo di file (anche se in realtà la rete WinMX viene utilizzata prevalentemente per file in formato audio e video).
Nelle ultime versioni ha introdotto il download da più sorgenti incrementando la possibilità che il download abbia successo. Sebbene la sua interfaccia non sia tra le più semplici, WinMX rappresenta una delle migliori reti p2p che uguaglia e va oltre quella di Napster.

In WinMX viene fatta una distinzione fra nodi di connessione primaria:

- direttamente connessi ai server;
- sono usati anche per il Routing;

e nodi di connessione secondaria:

- connessi solo ai nodi di connessione primaria;
- non si occupano di Routing.

Fu rilasciata nel 2000 dalla FrontCode Technologies e al contrario dei suoi predecessori infatti WinMX non era solo un semplice client OpenNap, ma aveva una propria rete p2p decentralizzata.
Con l'introduzione della WinMX PNP (Peer Networking Protocol) Network, questa architettura assunse i vantaggi tipici della decentralizzazione. Questa architettura di rete era decentralizzata come Gnutella, ma ha comunque approfittato dei supernodi grazie all'efficienza del network. In più era necessario un host cache server, necessario per la connessione del client al network. Il client WinMX si connette al server della host cache e ottiene una lista di IP per comunicare con i supernodi.
WinMX non supporta sul protocollo OpenNap alcune delle caratteristiche del protocollo WPNP che usa sulla WinMX Peer Network come ad esempio il download da più sorgenti (multisource downloading), la ricerca automatica delle sorgenti, la visualizzazione dei file incompleti; così sarà più difficile da utilizzare della rete WinMX Peer Network soprattutto quando si cercano di scaricare grandi file.

1.6.9 Direct Connect

Direct Connect [23] è sia un client che una rete di file sharing peer-to-peer creato da NeoModus. Non è di tipo decentralizzato come Gnutella o FastTrack, dato che usa hub connessi a gruppi di utenti, che rappresentano spesso aree di interesse particolare. Spesso, questi hub concentrici consentono l'accesso solo a quegli utenti che posseggono una determinata quantità di files da condividere. Nel programma è presente anche una chat, come mezzo di comunicazione tra utenti che formano piccole comunità di interesse, piuttosto che un puro file sharing anonimo.

Il client originale Neomodus, è stato ormai sostituito recentemente da altri client (DC++, DCPRO), ma viene ancora utilizzato. I client alternativi condividono delle caratteristiche comuni, come per esempio una bassa probabilità di essere infettati da virus, essendo Open Source. Le versioni ulteriormente modificate come BCDC++ e CZCD++, ODC, tutti basati su DC++, sono state sviluppate per comunità che hanno particolari interessi (per esempio comunità che condividono file musicali). Attualmente attorno a questo client c'è un grande interesse e una forte cooperazione nell'aggiungere nuove caratteristiche al protocollo Direct Connect (come statistiche, codifiche dati, pack di linguaggi, ecc.).

Direct Connect, a causa dell'eccessiva centralità del suo protocollo di comunicazione, ha delle difficoltà ha sostenere il traffico di più di 1000 utenti e la creazione di ciò che viene definito un multihub. Multihub è l'abbreviazione che viene usata per delineare un sistema di interconnessione fra più hub, in maniera tale da dare l'impressione che ci si trovi tutti contemporaneamente su un unico server.

Comunque le comunità che ruotano attorno a Direct Connect hanno tutto l'interesse a tenere basso il numero di persone che aderiscono al progetto, anche per evitare ripercussioni legali legate alla condivisione di materiale protetto da copyright.

Spesso questi network hanno delle capacità di clustering che reindirizzano gli utenti da hub troppo pieni (dove è stato superato il limite massimo di utenti imposto), ad altri hub dello stesso network.

1.7 Le applicazioni "anonime"

Con le reti di file sharing standard è difficile conservare la propria privacy poiché l'indirizzo Internet di tutti coloro che condividono files è molto facile da rintracciare. In analogia con il telefono, usare reti di condivisione standard è un po' come fare degli scherzi telefonici a qualcuno che ha il "riconoscimento degli ID" sul proprio telefono, ed è rischioso. Con il

largo uso del "riconoscimento ID chiamante" tutti quelli che fanno scherzi telefonici di questi tempi sanno che devono digitare prima di ogni chiamata un codice specifico per nascondere la propria identità.
La ragione per cui gli indirizzi Internet vengono resi disponibili nelle reti di file sharing standard è perché è necessario: non c'è modo per stabilire una connessione diretta verso un nodo per un download senza conoscere l'indirizzo Internet. Allo stesso modo non c'è modo per un nodo di accettare la connessione per un download senza essere capace di determinare l'indirizzo IP. La trasmissione dati su Internet funziona semplicemente in questo modo, e non esiste la funzione "blocco ID chiamante". L'unico modo per proteggere l'identità delle persone è cercare di costruire qualcosa al di sopra dei protocolli base di Internet per evitare le connessioni dirette tra chi scarica e chi invia i files, superando così la necessità di condividere gli indirizzi.
In seguito, tra tutte le applicazioni che permettono la filosofia del p2p anonimo, sono descritte: Freenet, MojoNation, Publius e Mute.

1.7.1 Freenet

Freenet [24] è un software elaborato dal Ian Clarke nel 1999, un informatico irlandese che opera all'interno della vasta comunità di sviluppatori Linux e dell'Open Source.
Il programma, scaricabile gratuitamente, basandosi appunto su un approccio peer-to-peer analogo a quello di Napster, consente di trasferire file di ogni genere (non solo musicali, dunque) da un PC ad un altro godendo del più totale anonimato, grazie all'utilizzo di una serie di tecniche crittografiche che rendono impossibile identificare e rintracciare origine e destinazione delle transazioni.
Freenet è un network peer-to-peer su larga scala, il quale condividendo la potenza dei computer che ne fanno parte in tutto il mondo, crea un grande deposito virtuale di informazioni aperto a chiunque per pubblicare o consultare informazioni liberamente.

Le caratteristiche di Freenet sono:

- *Alto tasso di sopravvivenza*: tutti i processi interni sono completamente anonimi e decentralizzati in tutta la rete, rendendo virtualmente impossibile per un attacker distruggere informazioni o prendere il controllo del sistema. Un attacker è un programma che ascolta il "traffico" della rete;

- *Privato*: Freenet rende estremamente difficile osservare le informazioni visionate, pubblicate e immagazzinate dagli utenti;
- *Sicuro*: le informazioni immagazzinate in Freenet sono protette attraverso la crittografia, in modo tale da evitare manomissioni e falsificazioni;
- *Efficiente*: Freenet copia e ridistribuisce le informazioni dinamicamente in base alla richieste per garantire un servizio efficiente ed l'utilizzo del minimo di banda, indipendentemente dal carico di lavoro. Infatti Freenet generalmente richiede *log(n)* tempo per fornire il risultato in una network composta da *n* elementi.

Freenet è un sistema aperto e democratico che non può essere controllato da nessuna persona, nemmeno dal suo creatore.

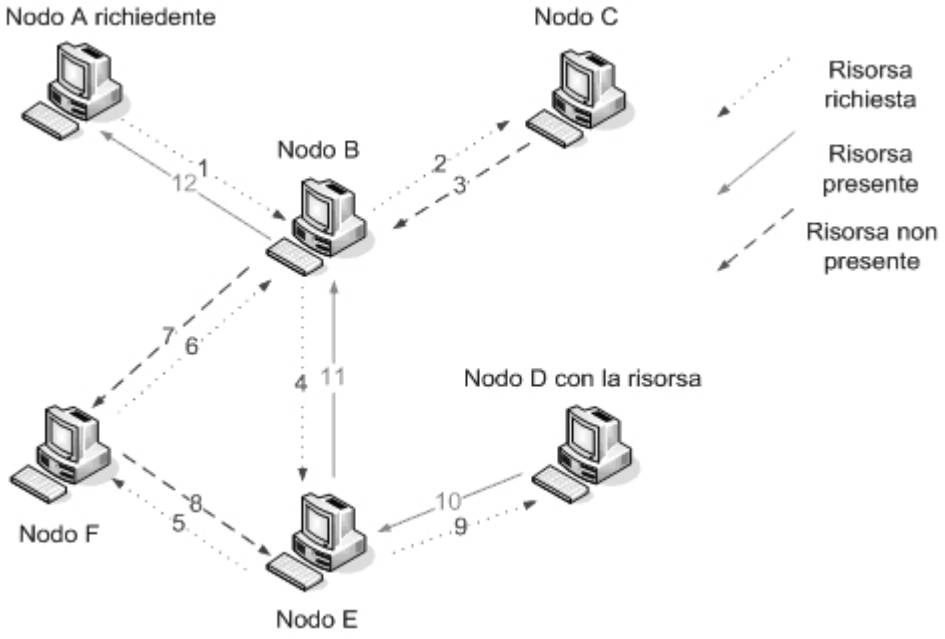


FIGURA 12– Esempio di routing di una rete Freenet

1.7.2 MojoNation

L'idea di utilizzare degli incentivi per favorire il corretto comportamento degli utenti nasce come tentativo di fronteggiare l'usanza, molto diffusa tra gli utenti delle reti p2p, di diffondere le risorse condivise al di fuori della community, cosa che d'altro canto favorisce la possibile violazione dei diritti di copyright.

Tuttavia, dato che gli "incentivi" si traducono in risorse di vario tipo da offrire agli utenti, questo meccanismo viene utilizzato anche per ricavare informazioni sulla reputazione, perché

la quantità di risorse posseduta da un utente della rete fornisce indicazioni sul suo comportamento passato.
È quello che succede in MojoNation [25], un'applicazione p2p di tipo file sharing realizzata da Jim McCoy, ex programmatore di Yahoo! e un anno amministratore delegato di Autonomous Zone Industries, che utilizza una moneta artificiale, chiamata Mojo, per rinforzare la condivisione delle risorse. In MojoNation ogni interazione tra gli utenti prevede un'offerta di Mojo: tutte le richieste, le risposte o le azioni effettuate tra due peer hanno un costo che non è di tipo economico, ma viene valutato in termini di banda, spazio su disco, risorse computazionali.

La tecnologia di MojoNation sfrutta appieno le caratteristiche delle reti p2p: l'affidabilità e la velocità con cui le risorse vengono distribuite vengono incrementate grazie al lavoro contemporaneo di più peer e non c'è un server centrale dal quale i peer dipendono. Lo stack protocollare di MojoNation prevede quattro livelli:

1. *Trasporto*: si preoccupa di instradare i dati attraverso TCP/IP;
2. *Cifratura e Autenticazione*: garantisce la privatezza e la sicurezza della comunicazione tramite cifratura e decifratura dei messaggi con crittografia a chiave pubblica;
3. *Comunicazione*: associa un messaggio di richiesta al messaggio di risposta corrispondente tramite un identificativo generato casualmente;
4. *Transazione*: si occupa dello scambio di Mojo; quando arriva un messaggio di richiesta con una corrispondente offerta di moneta, il peer ricevente invia una lista contenente i prezzi da lui richiesti per ogni servizio. Questo livello si preoccupa di controllare se il prezzo offerto è adeguato alle richieste del peer che ospita la risorsa.

I messaggi in uscita passano in sequenza dall'ultimo al primo livello del protocollo: un'offerta di Mojo viene valutata al livello transazione, poi il messaggio passa al livello comunicazione, dove viene associato alla sua controparte. Da qui, il messaggio viene cifrato al livello cifratura/autenticazione e infine passato al livello di trasporto.
Analogamente, i messaggi in ingresso passano in sequenza dal primo all'ultimo livello del protocollo, seguendo in senso opposto il procedimento appena visto.
Nella rete di MojoNation sono presenti dei particolari agenti, i Brokers, che supervisionano le operazioni di scambio di Mojo e le attività dei Tracker.

Questi ultimi sono anch'essi degli agenti e possono essere di tre tipi:

- I metatracker monitorizzano i Brokers che sono attualmente on line, insieme ai loro identificativi pubblici e alla lista di servizi che forniscono;
- I content tracker agiscono come motori di ricerca nella rete;
- I publication tracker inseriscono e distribuiscono i dati nella rete.

La richiesta di una risorsa da parte di un utente viene fatta al *Broker*, il quale interroga tutti i content tracker disponibili nel sistema: una volta individuata la risorsa corrispondente alla richiesta, ogni content tracker invia le informazioni ad essa relative all'utente richiedente, che a sua volta sceglie una delle risorse proposte da scaricare.
In questo sistema, la reputazione degli utenti viene valutata in termini di quantità di moneta posseduta: se un utente possiede molti Mojo, vuol dire che in passato ha effettuato molte transazioni sicure. Inoltre, visto che il download di una risorsa comporta il pagamento di una certa somma, gli utenti vengono stimolati a non diffondere le risorse al di fuori della comunità.

1.7.3 Publius

Il progetto Publius [26], sviluppato dai laboratori di ricerca dell'AT&T, promette di garantire l'anonimato di ognuno sul Web e di "resistere alla censura".
Publius, definito Censorship Resistant Publishing System (sistema di pubblicazione a prova di censura), trae il suo nome dallo pseudonimo usato oltre due secoli fa dai tre autori dei Federalist Papers (Alexander Hamilton, John Jay e James Madison), raccolta di articoli anti-britannici che incitarono la popolazione di New York a ratificare la Costituzione degli Stati Uniti d'America. In quella circostanza, l'anonimato era stato essenziale per la diffusione delle idee indipendentiste senza mettere a rischio la vita degli autori del testo.
Allo stesso modo, il Publius "virtuale", nasce per consentire la diffusione sicura e anonima soprattutto di testi (ma anche di pagine HTML, immagini, file PDF o Postscript e altro ancora), per di più cifrati e quindi spezzati in più parti e riprodotti sui server. Solo chi ha pubblicato il documento può rimuovere o modificare l'originale (quindi c'è anche un discorso di autenticazione). Ma allo stesso tempo, per chi scarica e legge lo stesso documento, l'autore rimane assolutamente anonimo.

Libertà di espressione, diritto all'anonimato, resistenza alla censura: tre principi sui quali si basano gli ispiratori del progetto Publius per offrire agli utenti Internet la possibilità di diffondere le loro risorse sul Web in tutta tranquillità.
La tecnologia messa a punto da due ricercatori del laboratorio di AT&T e uno studente del dipartimento di informatica dell'Università di New York, permette di cifrare le cartelle (testi, immagini, musica) e di dividerli in molti pezzi, come quelli di un puzzle, che saranno poi suddivisi su differenti server. Infatti Publius prevede l'utilizzo di un insieme di server in cui andare a postare i propri "pezzi" degli utenti, aggiornarli ed eventualmente cancellarli.
Solo Publius sarà in grado di riunire i frammenti, cosa che rende estremamente difficile, se non impossibile, ogni ricerca sull'autore della cartella.
Gli utenti possono accedere a questi dati usando un browser standard e un programma fornito dagli inventori stessi di Publius. Il software presenta una limitazione di 100 KByte per file, per scoraggiare il trading di file musicali e filmati ed invitare gli utenti a sfruttarlo per condividere testi e documenti piuttosto che altro.

1.7.4 Mute

Mute [27] è un software di scambio file di ultima generazione nato per assicurare la privacy a chi lo utilizza. Il programma, creato dall'americano Jason Rohrer, impedisce che i due estremi dello scambio di file (chi dà e chi riceve) si parlino e comunichino direttamente.
La prima versione è stata distribuita il 18 dicembre 2003. Allo stato attuale la rete Mute rende possibili trasmissioni a velocità molto basse, di prelevare la risorsa solamente da un'unica fonte (1 ad 1) e non permette il resume dei download interrotti.
Mute è software libero rilascitato sotto licenza GNU-GPL e può funzionare su Windows, GNU Linux e Mac OS X.

Lo scopo principale di Mute è permettere il file sharing mantenendo l'anonimato degli utenti, in modo che non sia possibile capire chi condivide o scarica un determinato file. Per renderlo possibile, vengono usati metodi di crittografia asimmetrica (RSA per scambiare chiavi segrete e AES per la trasmissione dei file) e il routing probabilistico.
Per proteggere l'identità di chi condivide i file, ad ogni nodo viene assegnato un indirizzo virtuale in modo che non sia necessario svelarne l'indirizzo IP, che identifica univocamente una macchina in Internet.

Tutti i messaggi e i download viaggiano sulla rete da nodo a nodo senza che vengano svelati gli indirizzi reali del mittente e del destinatario: ogni nodo svolge la funzione di client, server e router. Ogni nodo conosce soltanto l'identità di un numero limitato di nodi limitrofi (solitamente 5), ma non può sapere se i messaggi che gli arrivano da uno di questi provengano dal nodo stesso oppure da un suo nodo limitrofo. Quando un nodo deve inviare una richiesta, sua o di un nodo limitrofo, usa la tecnica del flooding. In questo modo è sicuro che il messaggio arriverà a destinazione anche se non sa quale sia la destinazione; un nodo che (dopo aver processato una richiesta) riceve il messaggio di risposta, lo invia al nodo limitrofo da cui aveva ricevuto la richiesta: in questo modo si costruisce il percorso mittente/destinatario senza che i nodi intermedi ne conoscano gli indirizzi reali (se non nel caso limite di un unico nodo intermedio).

L'unico modo per compromettere l'anonimato di un nodo è controllare fisicamente tutti i suoi nodi limitrofi, in modo da poter decifrare tutti i messaggi in uscita e controllare quelli in entrata. Se ad una richiesta in uscita non corrisponde una richiesta in entrata significa che il nodo ha generato la richiesta; se a una richiesta in entrata non corrisponde una richiesta in uscita significa che probabilmente il nodo possiede il file cercato (oppure la richiesta è scaduta). In ogni caso, la possibilità che associazioni come la RIAA o la SIAE riescano a controllare tutti i nodi limitrofi di un utente, è alquanto remota.

1.8 Il p2p è legale?

Esistono disposizioni di legge in Italia che vietino la condivisione di file via Internet? La risposta non è tanto semplice, né immediata. Anzi è piuttosto complessa.
Anzitutto va detto che il p2p in sé, come sistema di scambio e condivisione di file non è vietato dalla legge, né lo sono i software che consentono lo scaricamento (download) di file dal computer di un utente a quello di un altro. Tali software non vengono creati e distribuiti allo scopo di far condividere agli utenti materiali secondo modalità vietate per legge (brani musicali e opere cinematografiche duplicati senza l'autorizzazione del produttore, videogiochi e software piratati).

Quello che potrebbe rendere illegale un software di file sharing è la condivisione, da parte degli utenti, di materiale protetto dal copyright. Nel caso dei sistemi p2p, in particolare, l'assenza di un server centrale che smisti ed indicizzi le richieste e che abbia una visione

globale degli utenti e dei file contenuti negli hard disk degli stessi e, dunque, conoscenza dei file eventualmente illegali, impedisce di considerare il sistema illegale. Anche perchè nel caso dei più moderni applicativi di file sharing, come Freenet, la ricerca dei file e soprattutto la loro condivisione non passa dal server centrale, ma avviene direttamente attraverso i computer-nodi, peers, dei singoli utenti. Il server centrale, utilizzato per l'autentica iniziale dell'utente, non viene invece interessato dalle interrogazioni tra utenti e dal conseguente scambio di file. La condivisione ed il download passano solo ed esclusivamente per i computer degli utenti, (veri e propri client/server decentralizzati), diventando un fatto personale dei soggetti coinvolti.

Napster fu dichiarato illegale in quanto era incentrato su un sistema client/server ibrido. Per via di tale sistema, Napster fu giudicato responsabile di violazione delle norme sul diritto d'autore relativamente ai file musicali scambiati dagli utenti senza l'autorizzazione del discografico, secondo la formula del concorso nell'illecito e della responsabilità oggettiva: per concorso, in quanto avrebbe messo coscientemente a disposizione degli utenti un mezzo, il software, per violare il copyright, senza il quale mezzo tali violazioni non avrebbero potuto essere commesse; per responsabilità oggettiva o vicaria in quanto, avendo conoscenza degli utenti e dei file scambiati dagli stessi, non avrebbe impedito la trasmissione di file illeciti.
Al di là delle violazioni di copyright, uno dei motivi per cui la rete Napster fallì nel fornire una elevata qualità del servizio (QoS) ai suoi utenti è dovuto alla sua caratteristica di rete peer-to-peer centralizzata. Il QoS esprime la qualità di un sistema percepita dall'utente; può essere misurata in base a diverse metriche, come il numero dei risultati, il tempo di risposta, ecc.

Per i moderni sistemi di file sharing, incentrati sul modello p2p, il problema in cui è incappato Napster non si pone. E ciò per l'assenza di un server centrale detentore di informazioni, e perciò un potenziale controllo, sulla globalità degli utenti e dei file condivisi. I server centrali sono infatti utilizzati solo per l'autenticazione iniziale, mentre le ricerche dei file richiesti dagli utenti avvengono in maniera decentralizzata, sfruttando direttamente ed esclusivamente i computer degli utenti, e lasciando fuori il server centrale di sistema. Dunque Freenet, Gnutella, Imesh non sono illegali. È di tutta evidenza come i successori di Napster siano stati ideati e costruiti dai propri autori in maniera tale da evitare le motivazioni che costrinsero Napster alla chiusura.

1.9 Contese legali

Il tipi di file maggiormente condivisi nelle reti p2p sono i file musicali mp3 e i file video DivX, cioè filmati in formato digitale compressi tramite omonimo codec video. Questo ha portato molti, soprattutto le compagnie discografiche e i media, ad affermare che queste reti potevano diventare una minaccia contro i loro interessi e il loro modello industriale. Di conseguenza il peer-to-peer divenne il bersaglio legale delle organizzazioni che riuniscono queste aziende, come la RIAA e la MPAA (Motion Picture Association of America). Per esempio il servizio di Napster venne chiuso da una causa intentata dalla RIAA.

Sia la RIAA che la MPAA spesero ingenti quantità di denaro al fine di convincere i legislatori ad approvare restrizioni legali. La manifestazione più estrema di questi sforzi risale al Gennaio 2003 quando venne introdotto, negli U.S.A. un disegno di legge dal Californian Rapresentative Berman, nel quale si garantiva al detentore del copyright i diritti legali per fermare i computer che distribuivano materiale tutelato dai diritti d'autore.

Da quel momento in poi le reti peer-to-peer si espansero sempre di più, si adattarono velocemente alla situazione e divennero tecnologicamente più difficili da smantellare, spostando l'obiettivo delle major sugli utenti.

Qualcuno ha cominciato ad affermare che queste reti potevano diventare un modo per consentire a malintenzionati di nascondere la propria identità. Altri dicevano che per essere completamente immuni dalle major fosse necessario creare una rete wireless ad hoc in cui ogni unità o computer fosse connessa in modo equivalente (peer-to-peer sense) a quella vicina.

Casi Importanti:

- Sony Corporation of America. v. Universal City Studios Inc [28];
- MGM Studios Inc. v. Grokster Ltd [29].

1.10 Le prospettive dalla Computer Science Research

Tecnicamente, le applicazioni peer-to-peer dovrebbero implementare solo protocolli di peering che non riconoscono il concetto di server e di client. Tali applicazioni o reti "pure" [§2.2.2] sono in realtà molto rare. Molte reti e applicazioni che si descrivono come peer-to-peer fanno però affidamento anche su alcuni elementi "non-peer", come per esempio il DNS (Domain Name System). Inoltre, applicazioni globali utilizzano spesso protocolli multipli che fungono simultaneamente da client, da server, e da peer. Reti "peers" completamente

decentralizzate sono state utilizzate per molti anni, per esempio Usenet (1979) e FidoNet (1984).
La Sun MicroSystem aggiunse degli oggetti in linguaggio Java per velocizzare rapidamente lo sviluppo delle applicazioni p2p a partire dal 1990. In questo modo i programmatori poterono realizzare delle piccole applicazioni chat in real time, prima che divenisse popolare la rete di Instant Messaging. Questi sforzi culminarono con l'attuale progetto JXTA.
I sistemi e le applicazioni peer-to-peer suscitarono grande attenzione da parte dei ricercatori di computer science. Alcuni importanti progetti di ricerca comprendono il Chord lookup service [30], Arpanet, il PAST distributed storage utility [31], il CoopNet cooperative content distribution system [32] e l'OceanStore Project [33].

1.11 Il sistema JXTA

Differenti protocolli, differenti architetture, differenti implementazioni: questo descrive in sintesi le attuali soluzioni p2p. Come visto nei precedenti paragrafi, ad oggi gli sviluppatori utilizzano diverse metodologie e diversi approcci per creare applicazioni p2p. Gli standard e i protocolli, presenti nel mondo client/server, sono praticamente assenti nel mondo p2p: a questa carenza vuole far fronte un nuovo progetto: JXTA [34].

JXTA è il diminutivo di "juxtapose", che significa "parallelamente". Tale termine pone l'accento sul fatto che il peer-to-peer non è sostitutivo del paradigma client/server o del Web based computing, ma ne è un'alternativa. Il progetto JXTA è stato avviato da Sun MicroSystem [35] nel 2001 e si propone di fornire una piattaforma completamente Open Source per lo sviluppo di applicazioni peer-to-peer in qualunque ambiente di rete.
JXTA non è una libreria di codice; piuttosto, è un insieme di protocolli che può essere implementato in ogni linguaggio e su ogni rete per costruire applicazioni p2p.

Esso è stato concepito con lo scopo di raggiungere una serie di obiettivi tecnologici che puntano al superamento delle forti limitazioni legate alle applicazioni peer-to-peer attualmente esistenti:

- *Interoperabilità*: la tecnologia di JXTA si propone di facilitare la localizzazione, la comunicazione e la partecipazione alle attività di collaborazione da parte dei peer coinvolti nella rete, permettendo loro di offrire servizi in modo semplice attraverso le diverse comunità e i diversi sistemi p2p;

- *Indipendenza dalla piattaforma*: la tecnologia di JXTA risulta indipendente dal linguaggio di programmazione (C, Java, Perl), dal sistema operativo (Windows, Unix) e dalla piattaforma di rete (TCP/IP, Bluetooth);
- *Ubiquità*: la tecnologia di JXTA è pensata per essere implementata in qualsiasi dispositivo, anche di diversa tipologia, quali telefoni cellulari, PDA, PC, server, set-top box, ecc., consentendo ai dati e alle applicazioni di poter essere condivise e fatte girare in modo trasparente su ogni piattaforma che supporti Java.

JXTA fornisce tutte le funzionalità di base richieste in una applicazione p2p, fra queste:

- *peer discovery;*
- *peer communication.*

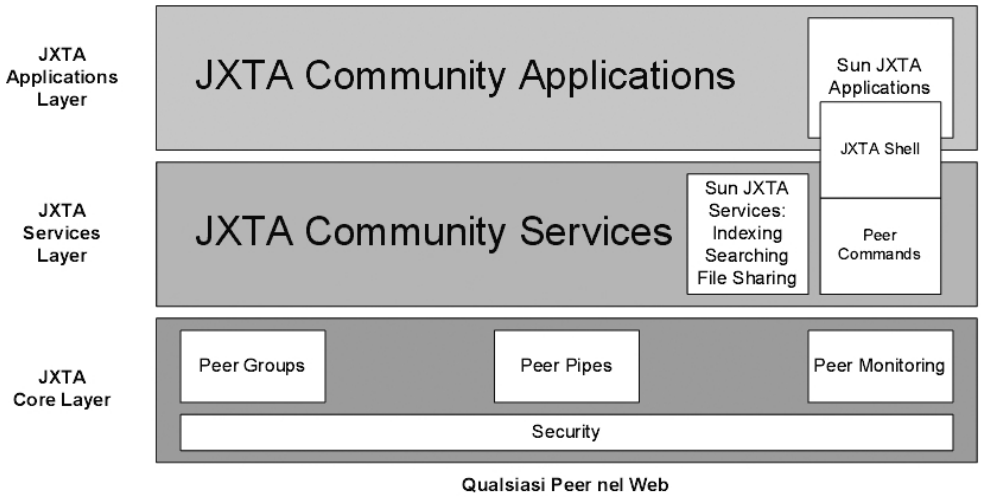


FIGURA 13 – L'architettura di JXTA

L'architettura logica di JXTA è organizzata su tre strati principali, ognuno dei quali realizza particolari funzionalità. Come mostrato in figura, questi sono: Platform Layer, Services Layer e Applications Layer.

- *Platform Layer (JXTA Core):* il platform layer racchiude le primitive minime ed essenziali comune alle reti p2p. Esso include dei blocchi di costruzione per rendere disponibili i meccanismi chiave per le applicazioni p2p, includendo la scoperta, il trasporto, la creazione di peer e gruppi di peer, e le primitive del sistema di sicurezza associato. Il livello Core include inoltre il set dei sette principali protocolli forniti da JXTA, ciascuno dei quali si occupa di un particolare aspetto del networking;

- *Services Layer:* il services layer include i servizi di rete che possono non essere assolutamente necessari affinché una rete p2p sia funzionante, ma sono di uso comune o desiderabili in un sistema p2p. Esempi di servizi di rete includono la ricerca e l'indicizzazione, servizi di directory, i sistemi di memorizzazione, la condivisione dei file, i sistemi di file distribuiti, l'autenticazione, e i servizi per il PKI (infrastruttura a chiave pubblica);
- *Applications Layer:* l'Applications layer include l'implementazione di applicazioni integrate, come lo scambio di messaggi, la condivisione di documenti e risorse, la gestione e la diffusione di servizi di intrattenimento multimediale, un sistema di posta elettronica p2p e molti altri.

La rete JXTA è costituita da una serie di nodi interconnessi, o peer. I peer si possono organizzare dentro dei gruppi di peer, che forniscono un set comune di servizi. In particolare, JXTA indica due peer particolari, chiamati: Rendezvous ed Edge. I peer Rendezvous possono essere usati per interconnettere gruppi diversi, separati in modo logico. Un peer Rendezvous agisce come una guida; tutti i peer Edge connessi ad un Rendezvous possono interagire, con peer non appartenenti al gruppo, soltanto spedendo e ricevendo messaggi attraverso un peer Rendezvous.

I peer Rendezvous naturalmente permettono di costruire una topologia controllata (quando un peer è posto come Rendezvous, deve essere specificata una lista di altri peer Rendezvous da usarsi per assegnare delle connessioni dirette) in grado di gestire tutte le comunicazioni necessarie per il sistema proposto. Inoltre, in JXTA viene fornita anche una sorta di capacità di tolleranza dell'errore; per esempio, se cade un Rendezvous, tutti i peer Edge connessi ad esso possono essere automaticamente rediretti ad un altro Rendezvous preservando il funzionamento del sistema. In JXTA è anche implementato un algoritmo per cercare delle richieste inoltrate attraverso la rete JXTA dei Rendezvous.

In particolare, una tipica configurazione di rete deve essere costituita da un peer Rendevous per ogni rete locale (LAN) e possibilmente da molti peer Edge locali. JXTA usa un protocollo multicast per scoprire altri peer in una LAN, mentre in una rete geograficamente distribuita i peer Rendezvous agiscono come meccanismo di instradamento permettendo ai nodi posizionati in differenti LAN di comunicare.

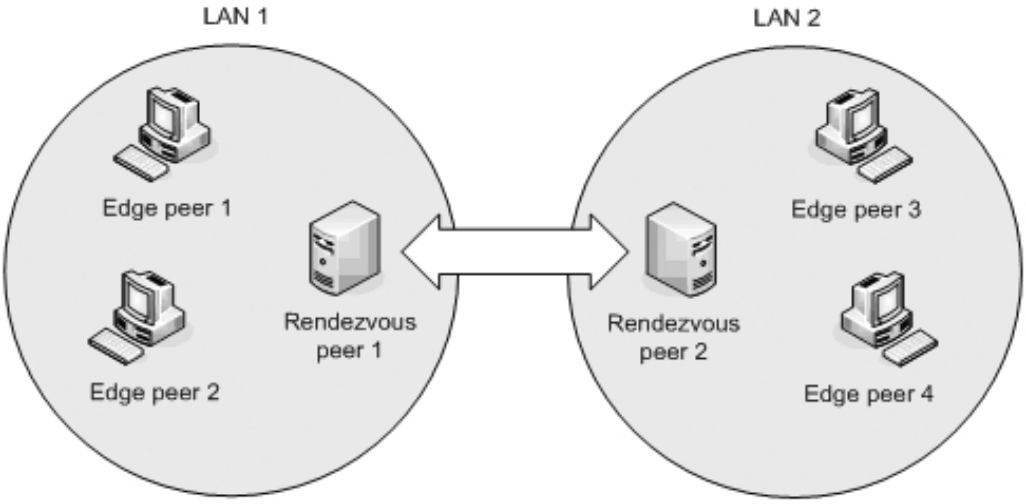


FIGURA 14 – Una tipica rete con Rendezvous ed Edge peer

I peer Edge possono interagire con peer non appartenenti al gruppo (un gruppo è guidato almeno da un peer Rendezvous) solamente spedendo e ricevendo messaggi attraverso un peer Rendezvous. Come mostrato in figura, gli Edge peer in LAN 1 possono comunicare con ogni altro e con Rendezvous 1, mentre per comunicare con il peer Edge 3 e il peer Edge 4, devono usare il Rendezvous 1 che, comunicando con il Rendezvous 2, inoltra i messaggi ai peer di destinazione. I peer Rendezvous e i peer Edge differiscono solo per una configurazione del layer JXTA sottostante, ma dal punto di vista dell'utente, essi presentano la stessa interfaccia. Nella configurazione più semplice, il peer Rendezvous è il nodo che inizia la sessione di visualizzazione distribuita, mentre gli altri utenti entrano in questa sessione connettendosi al sistema come peer Edge.

Attualmente molte aziende e svariati ricercatori in tutto il mondo collaborano alla realizzazione del progetto JXTA, per definire delle funzionalità non ancora completamente sviluppate, come quelle riguardanti la sicurezza.

Capitolo 2

Le reti Peer-to-Peer

L'evoluzione delle applicazioni peer-to-peer nel tempo ha visto l'adozione, da parte degli sviluppatori, di stili architetturali sempre più distaccati dal paradigma centralizzato, il quale, dopo la nascita di Internet, si era imposto come il modello di riferimento per lo sviluppo di applicazioni in Rete.
Oggi il modello p2p non pretende di sostituirsi al paradigma client/server, ormai consolidato e adeguato per tante tipologie di applicazioni, ma vuole piuttosto offrire un modo alternativo di creare applicazioni, sfruttando le potenzialità di Internet e basandosi su un'architettura quanto più possibile decentralizzata.

2.1 Classificazione delle reti p2p

Le applicazioni p2p sono costituite da tre fasi principali:

- *Boot*: permette a un peer di trovare la rete e di connettersi ad essa; (nessuno o quasi fa boot p2p);
- *Lookup*: permette ad un peer di trovare il gestore/responsabile di una determinata informazione; (pochi sono p2p);
- *Scambio di file*; (sono tutti p2p).

Parleremo di applicazioni:

- *p2p pure* se:
    - le fasi di boot, lookup e scambio di file sono p2p;
- *p2p* se:
    - le fasi di lookup e scambio di file sono p2p;
    - la fase di boot utilizza qualche server;
- *p2p ibride* se:
    - la fase di scambio dei file è p2p;
    - la fase di boot utilizza qualche server;
    - nella fase di lookup vengono usati Peer particolari.

In generale il p2p descrive un ambiente in cui i computer (gli host) sono connessi gli uni agli altri secondo un'architettura distribuita che non usa un punto di controllo centrale per collegare le macchine ed instradare il traffico dei dati. A differenza dell'architettura client/server, nella quale è presente un unico punto di erogazione del servizio, le reti p2p non hanno necessariamente un server centrale al quale delegare l'accesso, ed impiegano piuttosto un'architettura piatta, fortemente interconnessa. I nodi della rete fungono sia da fornitori che da utilizzatori di servizi, e la responsabilità del corretto funzionamento della rete viene distribuita tra tutti i "pari".

Esistono diversi tipi di architetture p2p. Queste possono differenziarsi una dall'altra dal punto di vista strutturale (ad esempio Napster sfrutta dei server di indicizzazione delle risorse, mentre in Gnutella anche il discovery delle risorse è distribuito) e/o per il diverso scenario d'utilizzo: le possibili applicazioni vanno dalle comunità di file sharing alle VPN (Virtual Private Network) aziendali, da sistemi di pubblicazione distribuita ad architetture per la condivisione delle risorse di calcolo.

Tutte le architetture p2p hanno in comune il fatto che il trasferimento dei dati avviene instaurando una connessione diretta tra il peer che offre la risorsa e il peer che la richiede, ma il reperimento della risorsa voluta può essere realizzato in tanti modi, per cui si possono individuare tre diverse architetture adottate dalle applicazioni p2p attualmente esistenti: *centralizzata*, *pura* ed *ibrida*. In tutti e tre questi sistemi il trasferimento e la memorizzazione dei file è p2p.

2.1.1 Sistemi ad architettura centralizzata

Un'architettura centralizzata utilizza uno schema di tipo client/server dove tutti i peer dipendono da un server centrale che gestisce l'elenco degli utenti connessi e le risorse messe a disposizione da questi ultimi. Napster, Audiogalaxy e Direct Connect sono un esempio di sistemi p2p con lookup centralizzata.

La richiesta di una risorsa viene inviata al server, il quale si preoccupa di cercare la risorsa, tipicamente in un suo database di file condivisi tra gli utenti della rete, e fornisce i risultati con gli indirizzi di chi mette a disposizione la risorsa ricercata. Un sistema del tutto simile ad un motore di ricerca, ma in questo caso sono i client a segnalare al server cosa forniscono e non il server a visitare il web ed indicizzare il contenuto.

Il risultato della ricerca che un utente inoltra non è altro che la lista dei nodi che dispongono della risorsa, a questo punto basta sceglierne uno e cominciare la trasmissione diretta. Il passaggio del file avviene direttamente tra i due utenti interessati e non coinvolge in nessuna maniera il server, che non memorizza su di sé alcun file. Il server mantiene anche una chat-room, che non è però coinvolta nei meccanismi di sharing.
Una volta trovata la risorsa, questa viene messa a disposizione direttamente tramite una connessione diretta tra il peer che la richiede e il peer che la ospita.

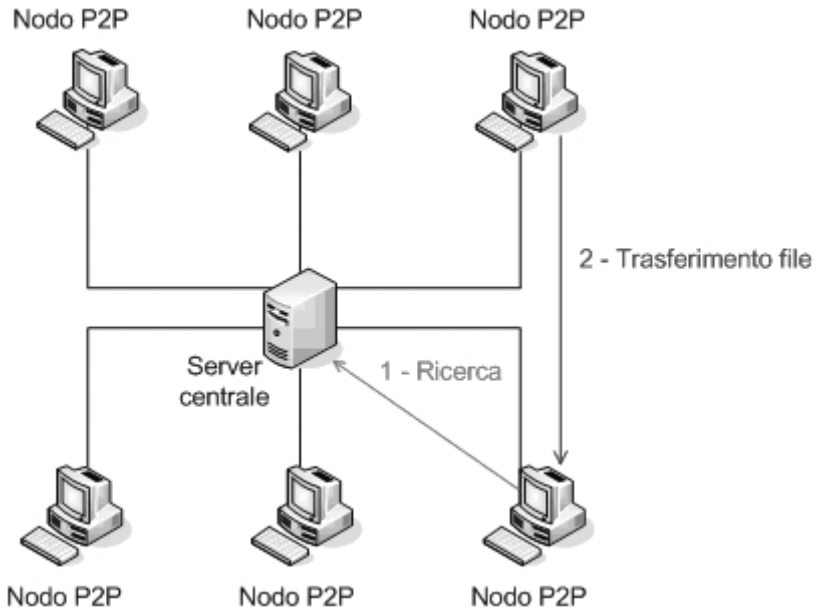


FIGURA 15 – Esempio di Architettura centralizzata

I vantaggi di questo tipo di reti p2p sono:

- La presenza nel server di un indice centrale che permette agli utenti di cercare e trovare i file desiderati in modo efficiente e rapido;
- La registrazione obbligatoria da parte degli utenti con cui si garantisce una ricerca più esaustiva possibile.

Gli svantaggi di tale tipo di reti p2p sono invece

- La presenza di uno o pochi punti di accesso, da cui dipende l'esistenza e la funzionalità dell'intera rete;
- L'eventuale fornitura di informazioni non aggiornate o di link non più validi a causa di un aggiornamento non immediato del database del server.

2.1.2 Sistemi ad architettura decentralizzata

In un'architettura p2p decentralizzata**,** detta anche "pura", non è presente alcun server centrale e ogni nodo connesso alla rete è considerato pari agli altri, cioè con gli stessi diritti, e prendono il nome di servant. Un servant è un'entità che è in grado di compiere le funzioni da server e da client. Un server centrale può essere eventualmente utilizzato solo per operazioni di registrazione alla rete, ma tutte le altre operazioni avvengono poi in modo completamente decentralizzato. Questo tipo di architettura supporta il DHT (Distributed Hash Table). FastTrack e Gnutella sono un esempio di sistemi p2p con lookup decentralizzato.

In figura è mostrato il meccanismo con cui avviene la ricerca e il trasferimento delle risorse: per poter entrare a far parte del circuito è necessario conoscere l'indirizzo di almeno un servant già attivo. Il primo compito è quello di comunicare al nodo servant conosciuto la nostra presenza all'interno della rete, il quale a sua volta comunicherà agli altri nodi, al quale è connesso, la nostra esistenza e lo stesso faranno questi ultimi e via così. Il processo non si perpetua all'infinito poiché il protocollo prevede un TTL (time to live), cioè un numero massimo di passaggi (hop), dopo il quale i server smettono di propagare la nostra esistenza.

Dopo essersi inseriti nella rete è possibile effettuare le ricerche presso i nodi direttamente collegati, i quali la inoltreranno a loro volta se non sono in grado di soddisfarla. Una volta che la richiesta raggiunge un peer che ospita la risorsa, questa viene messa a disposizione instaurando una connessione diretta tra il peer richiedente e quello offerente. In questa tipologia nessun nodo è vitale per l'esistenza della rete.

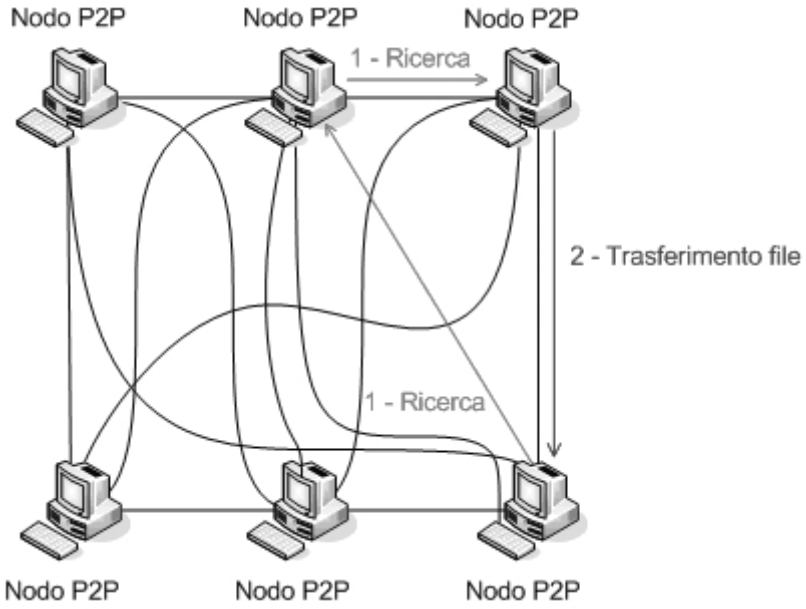


FIGURA 16 – Esempio di Architettura decentralizzata

Per questo tipo di strutture osserviamo quindi i seguenti vantaggi:

- Sono presenti moltissimi punti d'accesso, pertanto reti di questo tipo non soffrono della presenza di colli di bottiglia;
- Le reti decentralizzate sono decisamente robuste sia dal punto di vista "burocratico" che da quello topologico.

Gli svantaggi presentati sono invece:

- La mancanza di una registrazione fa sì che non sia sempre garantito un livello soddisfacente di QoS (elevata qualità del servizio);
- La struttura presenta il difetto di non rendere accessibile l'intera rete a ciascun nodo. Di conseguenza non sarà sempre possibile soddisfare una richiesta nonostante la risorsa sia presente in rete.

2.1.3 Sistemi ad architettura ibrida

L'architettura ibrida, detta anche "semi-centralizzata", è stata pensata per sfruttare i vantaggi dei due modelli visti prima: piccole reti centralizzate all'interno di una più grande rete decentralizzata. Freenet, BitTorrent, WinMX e KaZaA sono un esempio di sistemi p2p con lookup ibrido.

Come si vede in figura, in tale architettura sono presenti uno o più nodi di controllo che assumono un ruolo principale all'interno della rete, mentre tutti gli altri nodi si comportano come pari. Il piano di controllo viene dunque suddiviso su due livelli: quello dei peer normali, connessi ai nodi di controllo in modo client/server, e quello dei nodi di controllo, connessi tra loro secondo un'architettura p2p pura. La ricerca in tale caso viene effettuata in questo modo: il peer fa la richiesta al nodo centrale, il quale controlla se qualcuno tra i peer ad esso collegati offre la risorsa cercata. Se non la trova, inoltra la richiesta in modo p2p puro ai nodi ai quali è connesso, i quali cercheranno nei loro archivi di risorse condivise.

Una volta trovato il peer che offre la risorsa, viene instaurata una connessione diretta tra il richiedente e l'offerente e si effettua il trasferimento.

Questo tipo di architettura porta i seguenti vantaggi:

- aumento di scalabilità;
- efficienza dei protocolli di routine;
- ricerca del primo nodo a cui collegarsi.

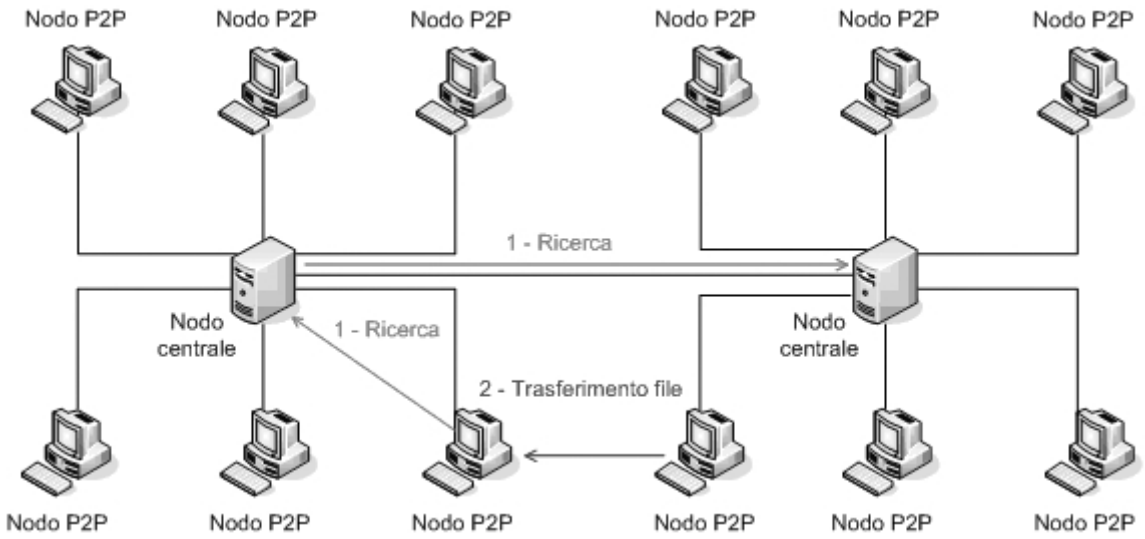


FIGURA 17 – Esempio di Architettura ibrida

Una struttura simile viene utilizzata per lo scambio di e-mail attraverso Internet: i client di posta elettronica hanno una relazione di tipo centralizzata con uno specifico mail server, mentre i mail server stessi si scambiano le e-mail in maniera decentralizzata.

2.2 Classificazione generazionale delle reti p2p

Alcuni descrivono le reti peer-to-peer per il file sharing per generazione. In questo contesto, si riferiscono solo alle famose reti p2p per lo scambio file basate su Internet, non a ricerche precedenti né a sistemi p2p commerciali.

2.2.1 Prima generazione

Le reti peer-to-peer per il file sharing di prima generazione avevano un elenco di file centralizzato, come Napster. I tribunali negli Stati Uniti stabilirono che chiunque controllava questo elenco di file centralizzato che conteneva opere protette da copyright era responsabile per ogni violazione dei diritti d'autore. Alla fine Napster fu ritenuto responsabile e dovette chiudere.

In un modello peer-to-peer centralizzato, un utente invierebbe una richiesta di ricerca al server centralizzato di quello che sta cercando ad esempio una canzone, un video, un film. Il server invia indietro un elenco di quali peer hanno i dati e permette la connessione ed il download.

2.2.2 Seconda generazione

Dopo che Napster incontrò problemi legali, Justin Frankel della Nullsoft cercò di creare una rete senza un server indice centrale e il risultato fu Gnutella. Sfortunatamente, il modello di Gnutella in cui tutti i nodi sono uguali soffrì gravemente di colli di bottiglia non appena la rete crebbe con i contributi dei rifugiati del defunto Napster. FastTrack risolse il problema rendendo "alcuni nodi più uguali di altri".

Rendendo alcuni nodi, quelli con maggiori capacità, nodi indice e facendo in modo che i nodi con minori capacità dipendessero dai primi, si creò una rete che aveva grandi capacità di scalabilità. Gnutella adottò velocemente questo modello, e la maggior parte delle attuali reti p2p lo seguono poiché permette di realizzare grandi reti decentralizzate.

Nella seconda generazioni di programmi sono comprese le Tabelle Hash distribuite (DHT), che risolvono il problema della scalabilità eleggendo vari nodi ad indicizzare alcuni hash (che vengono usati per identificare i file), permettendo una ricerca veloce ed efficiente di ogni instanza di un file sulla rete.

2.2.3 Terza generazione

Reti peer-to-peer di terza generazione sono quelle che hanno integrato caratteristiche di anonimato. Esempio di reti anonime sono Freenet, I2P, GNUnet, Entropy, MUTE, ANTs P2P, WASTE.

Le reti di terza generazione tuttavia non hanno raggiunto una diffusione di massa per lo scambio file a causa dell'alto overhead che le caratteristiche di anonimato richiedono, presentando prestazioni notevolmente più scarse rispetto ai programmi p2p tradizionali (di seconda generazione).

2.3 Potenzialità e svantaggi

Il modello p2p presenta numerose differenze rispetto al classico sistema client/server, differenze dovute principalmente al tipo di architettura di rete e alle diverse applicazioni per cui viene utilizzato.

Infatti le reti p2p sfruttano la banda disponibile attraverso la rete intera usando una grande varietà di canali di comunicazione, contrariamente alle tradizionali comunicazioni client/server, nelle quali specifici cammini verso destinazioni popolari possono sovraccaricarsi.

L'architettura p2p permette la comunicazione attraverso una grande quantità di route, riducendo così la congestione della rete. In particolare l'eterogeneità di queste reti permette la comunicazione tra tipi di connessioni fisicamente differenti garantendo trasmissione di dati ad uguale priorità. In questo modo le risorse vengono offerte con una grossa disponibilità ad un costo molto più basso, minimizzandone l'uso da parte di ogni singolo peer connesso alla rete. Mentre le soluzioni client/server contano sull'aggiunta di banda ed apparecchiature per mantenere una soluzione robusta, il modello p2p può offrire un livello di robustezza simile diffondendo le richieste di risorse attraverso la rete.

Il p2p prevede l'implementazione di diversi servizi, di cui ricordiamo le applicazioni a sfondo di ricerca scientifica, che, unendo le capacità normalmente inutilizzate di centinaia di computer in tutto il mondo, creano l'equivalente di un singolo super computer (distributed computing).

Ma il modello p2p comporta anche dei svantaggi: le richieste inviate attraverso una rete p2p potrebbero non ottenere una risposta immediata e, in alcuni casi, potrebbero non ottenere affatto una risposta. Infatti le risorse di tale rete possono scomparire in qualunque momento, a causa della disconnessione dell'utente che sta offrendo il servizio in quel momento: questa situazione è profondamente diversa da quello che accade nel Web, dove i servizi vengono sempre offerti poiché le risorse sono continuamente disponibili.

In ogni caso, l'architettura p2p può superare tutte queste limitazioni: sebbene le risorse possano scomparire in qualunque momento, un'applicazione p2p può implementare la funzionalità di replicare le risorse più richieste su più peer, offrendo in questo modo un accesso ridondante alle risorse stesse. In questo modo è possibile che un gran numero di peer interconnessi tra loro riduca la probabilità che una richiesta di servizio rimanga insoddisfatta. In pratica, la struttura della rete p2p, causa dei suoi problemi, può essere essa stessa utilizzata per risolverli.

Capitolo 3

La sicurezza nelle applicazioni Peer-to-Peer

L'elevata dinamicità dell'ambiente e la parziale conoscenza del contesto in cui gli host si trovano ad operare rendono la garanzia della sicurezza una delle sfide maggiori per chi progetta protocolli e applicazioni peer-to-peer. Fino a poco tempo fa gli aspetti legati alla sicurezza sono stati trascurati in questo contesto, preferendo piuttosto venire incontro ad altre esigenze, quali la definizione dei protocolli e l'efficienza.
Attualmente sono in corso vari progetti che si propongono di affrontare i diversi aspetti legati alla sicurezza, ma manca una visione organica di quali siano le problematiche di sicurezza relative alle applicazioni peer-to-peer e le tecniche utilizzate per risolverle.
In questo capitolo verranno elencate le esigenze di sicurezza che si riscontrano nelle applicazioni p2p di tipo File Sharing e Instant Messaging, con lo scopo di sottolineare come queste siano diverse a seconda dell'applicazione considerata. Successivamente verranno individuate le problematiche di sicurezza più generiche relative alle applicazioni p2p.

3.1 Esigenze di sicurezza

Lo sviluppo di sistemi peer-to-peer su larga scala pone, tra le altre cose, nuovi problemi dal punto di vista della sicurezza. I sistemi p2p, infatti, permettono collegamenti diretti tra utenti che in genere non si conoscono. Mentre quando si collega al sito di un'azienda o di un'istituzione nota l'utente sa con un discreto grado di sicurezza chi è il soggetto con cui ha instaurato il rapporto (per lo meno se è presente qualche forma di autenticazione); in una rete peer-to-peer è spesso impossibile ottenere un grado di fiducia comparabile perché in genere l'utente non conosce il proprio interlocutore. La protezione della *privacy* (e quindi anche dell'*anonimato*) degli utenti su Internet è per altro un diritto fondamentale che deve essere protetto, per questioni di riservatezza personale e anche per garantire la libertà di espressione e la libera circolazione dell'informazione.

È evidente, tuttavia, che le forme più forti di anonimato rendono problematica la creazione di un rapporto di *fiducia* tra i nodi. Inoltre, anche in casi in cui i nodi non sono strettamente anonimi, rimane il problema che, soprattutto in reti in cui si trovano migliaia di utenti, è sempre difficile per il singolo utente fidarsi di interlocutori di cui sa comunque poco o nulla,

un po' come avviene ad una persona quando si trova in una città nuova in cui non conosce nessuno. Si vede dunque un potenziale contrasto tra due esigenze che hanno entrambe a che fare con la sicurezza: da un lato l'esigenza di proteggere la privacy degli utenti, e dall'altro la necessità di avere elementi in base ai quali decidere con quali altri utenti si vuole avere a che fare e con quali no.

L'obiettivo principale delle reti peer-to-peer consiste nel fornire a chiunque ne faccia parte la possibilità di condividere informazioni, siano esse file, messaggi o risorse generiche.
Se in linea di principio questa filosofia sembra la più adatta in un'epoca in cui le informazioni viaggiano rapidamente da un capo all'altro del globo e non sono più prerogativa di un gruppo ristretto di persone, nella pratica essa lascia libero il campo a tanti comportamenti scorretti: eventuali utenti malintenzionati possono sfruttare i meccanismi e la struttura della rete p2p per gli scopi più vari, come violare la privacy degli utenti o alterare le risorse della rete diffondendo virus e troian.

Si possono individuare esigenze di sicurezza comuni alle varie applicazioni e altre comuni nell'ambito di una specifica tipologia applicativa.

In particolare, per quanto riguarda il File Sharing, queste sono:

- *Identificazione e Autorizzazione degli utenti e dei dati*: l'accesso e la manipolazione dei file possono essere effettuati solo se gli utenti ne possiedono i diritti;
- *Anonimato*: è necessario tutelare la privacy sia di chi offre la risorsa che di chi la richiede;
- *Riservatezza*: le comunicazioni relative allo scambio dei file non devono essere ascoltate da parti non interessate;
- *Integrità dei file scambiati*: i file scambiati devono essere trasportati in modo sicuro in modo tale che non vengano alterati da utenti malintenzionati o a causa di disservizi della rete;
- *Disponibilità*: le risorse devono essere continuamente disponibili in rete, è necessario quindi evitare l'interruzione del servizio, sia essa dovuta ad attacchi di tipo Denial of Service (DoS) piuttosto che a problemi intrinseci nella rete.

Analogamente, per l'Instant Messaging:

- *Identificazione e Autenticazione tra peer*: per creare i contatti e per effettuare la comunicazione, gli utenti devono essere identificati univocamente all'interno della rete e devono autenticarsi;
- *Identificazione e Autenticazione dei peer con il server*: nel caso in cui ci sia un server centrale presso il quale avvengono le registrazioni o tramite il quale si creano i contatti, è necessario che gli utenti si identifichino e si autentichino con esso;
- *Riservatezza della comunicazione*: lo scambio di messaggi e di file non deve essere intercettato da chi non è coinvolto nella comunicazione;
- *Riservatezza della lista dei contatti*: se la lista dei contatti non è locale al peer, ma risiede su un server, deve essere conservata e trasmessa in modo sicuro, affinché un malintenzionato non possa ricavarla;
- *Integrità dei dati*: i messaggi, ed eventualmente i file, devono viaggiare in modo sicuro all'interno della rete affinché non vengano alterati.

Da diversi anni, la comunità scientifica ha individuato delle problematiche di sicurezza relative alle reti p2p che, pur essendo inerenti a quelle finora elencate, si pongono su un piano più elevato rispetto ad esse. Queste problematiche, che per essere affrontate richiedono dei meccanismi molto complessi, sono:

- Distribuzione della fiducia tra peer;
- Gestione della reputazione dei peer;
- Tutela dell'anonimato dei peer.

Il problema della distribuzione della fiducia è strettamente correlato a quello della gestione della reputazione, perché i concetti di "fiducia" e di "reputazione" sono molto legati. Questo lo si può notare già a partire dalle loro definizioni riportate di seguito:

- La *fiducia* (e, simmetricamente, la sfiducia) è un livello soggettivo particolare della probabilità con la quale un individuo effettuerà una particolare azione, prima che sia verificabile e in un contesto in cui possa influire sulle nostre azioni;
- La *reputazione* è l'aspettativa sul comportamento di un individuo sulla base delle informazioni raccolte sul suo precedente comportamento.

La garanzia dell'anonimato è invece un problema separato, che può essere affrontato indipendentemente. Le tre problematiche sopra elencate vengono analizzate in dettaglio nel resto del capitolo.

3.2 Fiducia e reputazione

In un contesto completamente anonimo, il problema della fiducia appare irrisolvibile, in quanto l'utente non ha nessuna informazione su chi sia il suo interlocutore. Tutela dell'anonimato e distribuzione della fiducia possono quindi sembrare inconciliabili, ma il contrasto diviene molto più debole se dall'anonimato puro si passa a forme di pseudonimato. Con questo termine si intende un contesto in cui ad ogni utente è associato un identificativo (lo pseudonimo): tale identificativo non ha alcuna relazione con l'identità reale dell'utente, che rimane dunque sostanzialmente anonimo (in genere inoltre l'utente può in qualunque momento decidere di cambiare pseudonimo, oppure utilizzare diversi pseudonimi), tuttavia permette di collegare diverse azioni compiute dallo stesso utente, che in base al suo comportamento può quindi costruirsi una reputazione. Si può allora pensare di raccogliere in qualche modo le opinioni di utenti che hanno avuto rapporti con la persona o il servizio in questione, valutarle in base a una certa metrica, e poi decidere se fidarsi o no. Sistemi di reputazione sono presenti in diversi contesti, quali ad esempio il sito di aste on-line eBay (per valutare l'affidabilità degli utenti), il motore di ricerca Google (per il ranking delle pagine) o le community specifiche ad un determinato argomento. Si tratta però di soluzioni di tipo centralizzato e quindi non direttamente applicabili a contesti tendenzialmente privi di qualunque "autorità" come le reti peer-to-peer.

La costruzione di un sistema peer-to-peer che tuteli la privacy degli utenti ma che fornisca meccanismi di distribuzione della fiducia è quindi allo stato attuale in larga parte ancora una sfida, ma se fosse vinta avrebbe probabilmente potenzialità enormi, permettendo agli utenti di sviluppare vari tipi di rapporti e transazioni in rete senza dover ogni volta rivelare la propria identità e tutte le varie informazioni connesse, e potendo contare su una certa probabilità che tutto vada a buon fine.

3.3 Anonimato e riservatezza

È indubbio che un grosso problema relativamente alle comunicazioni in rete è quello della privacy. Un problema dato dalla relativa semplicità con cui è possibile ottenere informazioni personali o addirittura tracciare il comportamento di un singolo utente, con il rischio che si

possano dedurre dati sensibili come gli orientamenti politici, religiosi, sessuali, o le malattie da cui un individuo è affetto. Si pensi solo alle conclusioni che si potrebbero trarre su una persona conoscendo la lista dei libri che ha comprato o consultato negli ultimi mesi, o i file che ha richiesto ad una community o ha messo in condivisione con gli altri.

Rendere anonimi gli utenti di una rete non è un problema semplice da risolvere e purtroppo non è forse nemmeno molto sentito; vediamo brevemente se e come hanno affrontato questo punto critico alcuni noti sistemi peer-to-peer.

Napster, per cominciare, non prendeva in considerazione il problema: tutti gli utenti comunicavano ad un server centrale quali erano i file che volevano condividere, e le ricerche avvenivano chiedendo direttamente al server centrale, che disponeva di tutte le informazioni; l'effettivo trasferimento del file era poi effettuato tramite connessione diretta tra i due peer interessati.

In Gnutella un certo grado di anonimato è invece in parte garantito: le ricerche sono infatti effettuate in maniera sostanzialmente anonima, in quanto le query vengono propagate in flooding e non contengono l'indirizzo del mittente. La stessa cosa non avviene per le risposte, che vengono retro propagate verso il nodo che aveva fatto la richiesta, ma contengono l'indirizzo del nodo che possiede la risorsa, il quale quindi si "scopre" e perde l'anonimato. Quando poi il nodo richiedente decide di scaricare la risorsa da un certo nodo che la possiede apre con lui una connessione diretta, perdendo quindi anche il suo anonimato.

Il più noto e importante progetto peer-to-peer per quel che riguarda la condivisione anonima di risorse è Freenet. Questo più che un sistema di file sharing (come Napster e Gnutella) è un sistema di pubblicazione contenuti, progettato per essere anonimo e resistente alla censura. In Freenet infatti le risorse non rimangono sul nodo originario, ma quando vengono richieste si propagano e si replicano all'interno della rete, in modo che non sia possibile stabilirne l'origine. Un altro progetto "nato" da poco che riguarda la condivisione anonima di risorse è Mute

3.4 Attacchi alle reti p2p

L'utilizzo di architetture p2p comporta inevitabilmente una serie di problemi di sicurezza, a vari livelli, che è possibile suddividere in due aree.

La prima, che chiameremo "p2p-related", include tutti quegli aspetti di sicurezza che è necessario prendere in considerazione nella normale progettazione di applicazioni distribuite (l'autenticazione dei partecipanti, la confidenzialità delle trasmissioni, la resistenza ad attacchi di tipo DoS) che vengono però amplificati dalla particolare natura p2p. Ad esempio,

l'assenza di uno o più server "centrali" non permette di mantenere facilmente un elenco di partecipanti su cui effettuare procedure di autenticazione.
La seconda, che chiameremo "IT-related", include tutti gli aspetti di sicurezza delle infrastrutture di rete (necessità di modificare la configurazione dei firewall per permettere la comunicazione tra peer, esposizione dei client/peer ad attacchi provenienti da peer "maliziosi", utilizzo eccessivo della banda di comunicazione per la trasmissione/ricezione di file multimediali) che sono dovuti alla presenza, all'interno di una certa infrastruttura di rete, di partecipanti ad una comunità p2p. Ad esempio, notevoli difficoltà nascono dall'utilizzo di tecniche, come il tunneling nel protocollo http, che rendono estremamente difficile tenere sotto controllo attività di tipo p2p all'interno di una infrastruttura di rete.

Il paradigma p2p implica uno spostamento dei punti di vulnerabilità di una rete dai sistemi server a quelli client/peer. Non solo i peer si espongono direttamente ad attacchi, come accade per i server nel modello client-server, ma il numero dei peer è molto elevato ed è quindi più probabile trovare almeno un sistema "debole". Ciò causa notevoli difficoltà nella definizione di una politica di protezione dell'infrastruttura di rete, anche perché i punti di vulnerabilità (client/peer) sono spesso gestiti dall'utente finale che raramente ha sensibilità per i problemi di sicurezza e/o competenze tali da poterli affrontare al meglio. Questo problema diventa ancora più rilevante se si tiene conto che delle comunità p2p possono far parte anche dispositivi portabili che spesso sono semplicemente non gestiti.

Esempio di attacchi comprendono:

- attacchi da avvelenamento (distribuire file il cui contenuto è diverso dalla descrizione);
- attacchi DoS - attacchi che possono portare la rete a funzionare molto lentamente oppure a non funzionare più;
- attacchi da diserzione (utenti o software che usano la rete in maniera passiva senza contribuire ad essa con risorse);
- inserimento di virus nei contenuti disponibili (i file scaricati o trasportati sulla rete possono essere infetti da virus o altre forme di malware);
- malware nel software p2p stesso (ad esempio il software p2p può contenere spyware);
- filtri (gli operatori di rete possono cercare di impedire il traffico dati sulle reti p2p imponendo dei filtri);

- attacchi di identità (ad esempio identificando gli utenti della rete e minacciandoli e/o attaccandoli legalmente);
- spam (invio di informazioni indesiderate sulla rete - non necessariamente come attacco DoS).

La maggior parte degli attacchi possono essere sconfitti o controllati progettando attentamente la rete p2p oppure attraverso l'uso della crittografia.

3.5 Malware

Si definisce malware un qualsiasi software creato con il solo scopo di creare danni più o meno gravi al computer su cui viene eseguito. Il termine deriva dalla contrazione delle parole inglesi "malicious" e "software" e ha dunque il significato letterale di "programma malizioso".
Si distinguono molte categorie di malware, anche se spesso questi programmi sono composti di più parti interdipendenti e rientrano pertanto in più di una classe. Vista inoltre la rapida evoluzione in questo campo, la classificazione presentata di seguito non è da ritenersi esaustiva.

- *Virus*: sono parti di codice che si diffondono copiandosi all'interno di altri programmi, o in una particolare sezione del disco fisso, in modo da essere eseguiti ogni volta che il file infetto viene aperto. Si trasmettono da un computer a un altro tramite lo spostamento di file infetti ad opera degli utenti;
- *Worm*: questi malware non hanno bisogno di infettare altri file per diffondersi, perché modificano il sistema operativo della macchina ospite in modo da essere eseguiti automaticamente e tentare di replicarsi sfruttando per lo più Internet. Per indurre gli utenti ad eseguirli utilizzano tecniche di social engineering, oppure sfruttano dei difetti (bug) di alcuni programmi per diffondersi automaticamente;
- *Trojan horse*: software che oltre ad avere delle funzionalità "lecite", utili per indurre l'utente ad utilizzarli, contengono istruzioni dannose che vengono eseguite all'insaputa dell'utilizzatore. Non possiedono funzioni di auto-replicazione, quindi per diffondersi devono essere consapevolmente inviati alla vittima. Il nome deriva dal famoso cavallo di Troia;
- *Backdoor*: letteralmente "porta sul retro". Sono dei programmi che consentono un accesso non autorizzato al sistema su cui sono in esecuzione. Tipicamente si diffondono in abbinamento ad un trojan o ad un worm, oppure costituiscono una

forma di accesso di emergenza ad un sistema, inserita per permettere ad esempio il recupero di una password dimenticata;

- *Spyware*: software che vengono usati per raccogliere informazioni dal sistema su cui sono installati e per trasmetterle ad un destinatario interessato. Le informazioni carpite possono andare dalle abitudini di navigazione fino alle password e alle chiavi crittografiche di un utente. Divenne famoso quando nel programma per file sharing KaZaA si scoprì che durante l'installazione, oltre al client, venivano aggiunti programmi nascosti atti alla raccolta di informazioni per uso commerciale. Questo episodio così eclatante rese noto che non erano solo gli utenti a compiere azioni maliziose;
- *Dialer*: questi programmi si occupano di gestire la connessione ad Internet tramite la normale linea telefonica. Sono malware quando vengono utilizzati in modo truffaldino, modificando il numero telefonico chiamato dalla connessione predefinita con uno a tariffazione speciale, allo scopo di trarne illecito profitto all'insaputa dell'utente.

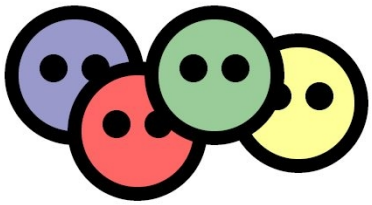
Nell'uso comune il termine virus viene utilizzato come sinonimo di malware e l'equivoco viene alimentato dal fatto che gli antivirus permettono di rilevare e rimuovere anche altre categorie di software maligno oltre ai virus propriamente detti.
Si noti che un malware è caratterizzato dall'intento doloso del suo creatore, dunque non rientrano nella definizione data i programmi contenenti bug, che costituiscono la normalità anche quando si sia osservata la massima diligenza nello sviluppo di un software.

Capitolo 4

Il sistema FOAF

FOAF (Friend Of A Friend) è un progetto Open Source nato nel 1999, ma ha guadagnato un interesse significativo solo negli ultimi anni dovuto all'incremento degli studi riguardo il Semantic Web e permette il raggruppamento di persone per affinità su Internet. Il principio è semplice: creare uno schedario di metadati contenente la descrizione di ogni persona, dei suoi gusti, del suo profilo e delle persone che conosce ed apprezza. Lo schedario può essere inserito sul proprio sito o su un blog.

4.1 Descrizione



Il progetto FOAF [36] è un'applicazione del Semantic Web che può essere utilizzata per fornire informazioni personali in una forma adatta per l'elaborazione automatizzata.
Da una pagina HTML di un sito personale l'autore può fornire le informazioni personali, quali i particolari per essere contattato (e-mail, telefono, ecc.), gli interessi, ecc. Tali informazioni sono utili per la lettura, ma non possono essere facilmente recuperati da processi automatizzati. FOAF è stato progettato per permettere a tali informazioni di essere fornite in una forma tale da poter essere visualizzate dalle convenzionali pagine Web, ma di poter essere utilizzate anche da processi automatizzati.
Fondato da Dan Brickley e Libby Miller, il progetto FOAF mira a definire un vocabolario standard, denominato "FOAF Vocabulary Specification" [37], basato sul RDF per esprimere metadati riguardo le persone, i loro interessi, i rapporti e le attività.

Le definizioni del vocabolario FOAF sono state scritte utilizzando un linguaggio di programmazione, denominato RDF/OWL (Resource Description Framework/Ontology Web Language); attraverso i termini presenti nel vocabolario è possibile creare facilmente documenti FOAF.
Un documento FOAF, che tuttavia è diverso da una pagina Web tradizionale, può essere combinato con altri documenti FOAF per generare un'unica base di dati di informazioni. Ciò consente ad un software di elaborare queste descrizioni, magari all'interno di un motore di

ricerca, allo scopo di trovare informazioni su una persona e sulle comunità delle quali fa parte. Inoltre FOAF può portare a nuove interessanti possibilità di sviluppo per le comunità on-line.

Il punto iniziale di FOAF è stata la descrizione della gente, poiché la gente è l'argomento principale che collega insieme la maggior parte degli oggetti e delle attività: ad esempio un persona può essere "presente" in documenti, foto, partecipare alle riunioni e così via.

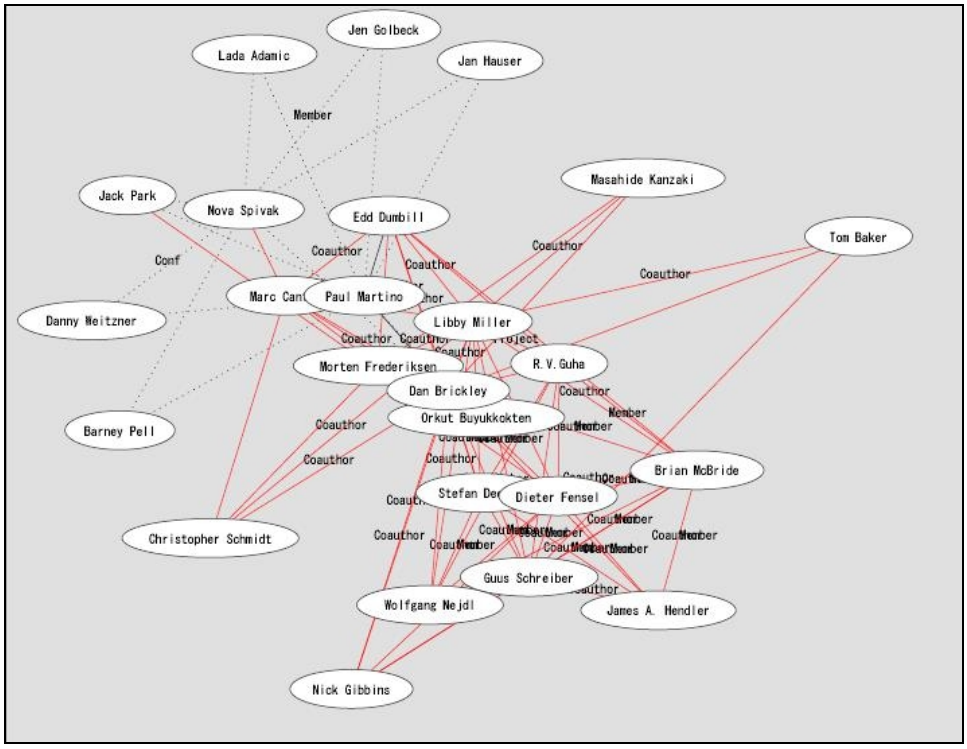


FIGURA 18 – La Social Network dei membri di FOAF

L'evoluzione di FOAF avviene migliorando la stabilità dei termini individuali che formano il vocabolario. I termini utilizzabili progrediscono attraverso le categorie "unstable", "testing" e "stable".

Oltre al vocabolario FOAF, una delle caratteristiche più interessanti di un file FOAF è che può contenere indicatori "see also" che "puntano" ad altri file FOAF. Ciò fornisce una base per strumenti di raccolta automatici per "attraversare" un Web di file collegati tra loro, e venire a conoscenza di nuove persone, documenti, servizi, dati, ecc.

4.2 L'idea di base

L'idea di base è abbastanza semplice: se la gente pubblica le proprie informazioni nel formato dei documenti FOAF, si possono utilizzare apposite applicazioni che, tramite tali informazioni, eseguano ricerche mirate. Se questi file contengono le voci "see also" (vedi inoltre), cioè riferimenti ad altri documenti, varie applicazioni potranno scrutare intorno al Web per trovare documenti che soddisfano tali requisiti, escludendo gli esseri umani da tale lavoro, memorizzando le informazioni trovate, mantenere una lista di "see also" che puntano ad altri documenti, controllare le firme digitali (per la sicurezza) e costruire pagine web e servizi di question-answering, simili alle F.A.Q. (Frequently Asked Questions, ovvero domande poste di frequente), basati sui documenti raccolti.

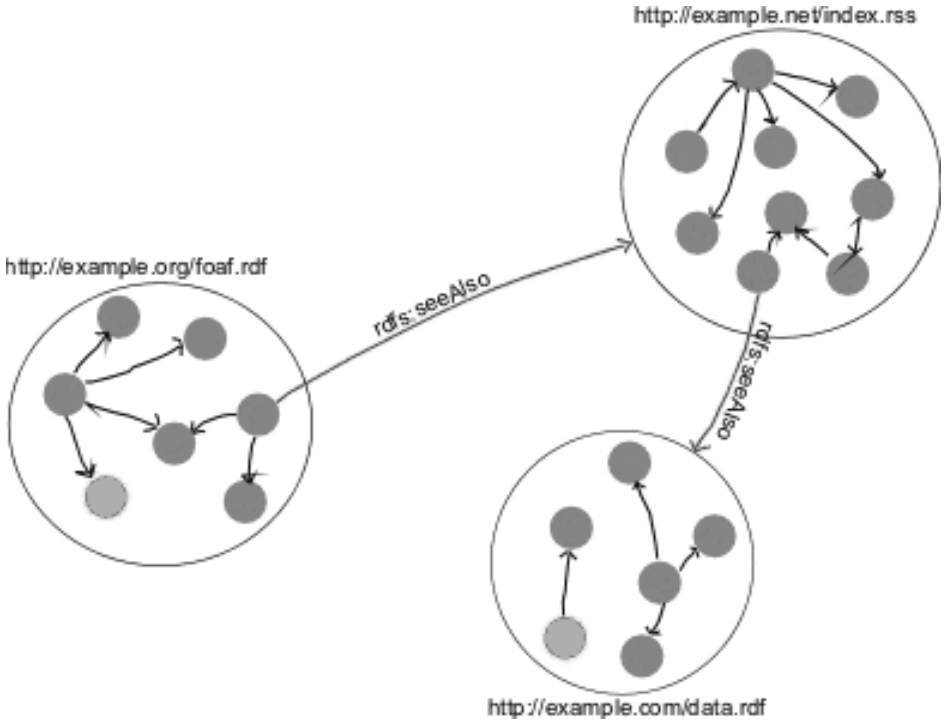


FIGURA 19 – File RDF connessi tramite rdfs:seeAlso

Ma che cosa è il formato di un documento FOAF? I file FOAF sono solo dei documenti di testo (documenti in Unicode): sono scritti attraverso la sintassi del Extensible Markup Language (XML), adottano le convenzioni del Resource Description Framework (RDF) e l'ontologia viene descritta attraverso la Ontology Web Language (OWL).

In più, il vocabolario FOAF definisce alcune utili costruzioni che possono comparire nei file FOAF, vicino ad altri vocabolari RDF definiti altrove. Per esempio, FOAF definisce le categorie (classi) come `foaf:Person`, `foaf:Document` e `foaf:Image` accanto ad alcune

proprietà funzionali, come `foaf:name`, `foaf:mbox` (cioè l'internet mailbox), `foaf:homepage` ecc., così come alcuni generi di rapporto che legano fra loro i membri di queste categorie. Per esempio, un tipo interessante di rapporto è `foaf:depiction` che collega qualcosa (per esempio `foaf:Person`) a `foaf:Image`.

FIGURA 20 – Le categorie dei gruppi e dei termini di FOAF

4.3 RDF Schema Specification

Le ontologie sono una forma di conoscenza attraverso cui le persone e le applicazioni possono dare una interpretazione comune ad un dominio. Rivestono quindi un ruolo decisivo nel Semantic Web.

L'importanza delle ontologie nello sviluppo del Semantic Web ha creato l'esigenza di introdurre un linguaggio che permetta di specificarle. Il WWW Consortium (W3C) ha definito l'RDF Schema Specification [38] attraverso il quale è possibile rappresentare un'ontologia in formato standard come schema RDF.

Uno schema RDF può essere visto come una sorta di dizionario comprensibile alle macchine. Con esso si definiscono infatti i vocaboli e il significato corrispondente, espresso attraverso la descrizione delle relazioni esistenti tra i termini.
Uno schema RDF raccoglie la dichiarazione delle categorie alle quali possono appartenere le entità che si vogliono descrivere e le relazioni che le legano (queste ultime possono anche essere viste come proprietà/attributi delle categorie).
Il linguaggio di specifica RDFS è un linguaggio dichiarativo poco espressivo ma molto semplice da implementare. Si esprime attraverso la sintassi di serializzazione RDF/XML e si avvale del meccanismo dei namespace XML per la formulazione delle URI (Uniform Resource Identificator) che identificano in modo univoco le risorse (definite nello schema stesso). Le agevolazioni introdotte dai namespace XML giocano un ruolo cruciale nello sviluppo degli schemi RDF poiché consentono il riutilizzo di termini definiti in altri schemi. Concetti e proprietà già dichiarati per un dominio possono essere impiegati di nuovo o precisati per incontrare le esigenze di una particolare comunità di utenti.

4.4 Come creare le descrizioni

Per entrare nel mondo FOAF, un utente deve generare un profilo, con il quale descrive se stesso rispettando alcune regole ed indicazioni, e lo deve pubblicare nel Web.
Per poter realizzare una descrizione FOAF, si può procedere in vari modi: si può crearne una nuova utilizzando apposite applicazioni, oppure si può modificare una già esistente.
Tra le applicazioni che permettono di generare definizioni FOAF citiamo FOAF-a-Matic e FOAF-a-Matic Mark 2.

4.4.1 Modificare un modello

Il profilo deve attenersi alla ontologia; può essere generato utilizzando un'apposita applicazione oppure copiare e modificare un profilo di un utente esistente, già all'interno del mondo FOAF.
La tabella mostra le proprietà generali per poter descrivere una persona:

Proprietà	**Valore**
foaf:name	Il nome di qualcosa (della persona, dell'azienda, ecc.)
foaf:title	In base alla persona (Mr, Mrs,Ms, Dr, ecc.)
foaf:gender	Il sesso della persona
foaf:firstname	Nome della persona

foaf:surname	Cognome della persona
foaf:birthday	Giorno e mese di nascita
foaf:nick	Stringa utilizzata per identificare la persona sul sistema
foaf:mbox	Mailbox personale
foaf:logo	Un logo che rappresenta qualcosa
foaf:homepage	Link alla home page personale
foaf:workplacehomepage	Link dove sono posizionati i lavori personali
foaf:publications	Link alle pubblicazioni personali
foaf:depiction	Link di un'immagine in cui è raffigurato qualcosa
foaf:phone	Numero di telefono
foaf:weblog	Link al blog personale, o alla azienda, ecc.
foaf:interest	Interessi personali

Ecco un esempio di descrizione FOAF (riguardante i dati riferiti alla persona):

```
<foaf:Person>
  <foaf:name>Luciano Ligabue</foaf:name>
  <foaf:title>Mr</foaf:title>
  <foaf:gender>Male</foaf:gender>
  <foaf:firstname>Luciano</foaf:givenname>
  <foaf:surname>Ligabue</foaf:family_name>
  <foaf:birthday>13-03</foaf:birthday>
  <foaf:nick>Liga</foaf:nick>
  <foaf:mbox rdf:resource="mailto:luciano@ligabue.com"/>
  <foaf:homepage rdf:resource="www.ligabue.com"/>
  <foaf:workplaceHomepage
        rdf:resource="http://www.ligabue.com/lavori.html"/>
  <foaf:depiction rdf:resource="http://www.ligabue.com/foto.jpg"/>
  <foaf:weblog rdf:resource="http://www.ligabue.com/blog/"/>
  <foaf:interest>
        <rdf:Description rdf:about="http://www.mtv.com/">
              <dc:title>MTV - MusicTeleVision</dc:title>
         </rdf:Description>
  </foaf:interest>
</foaf:Person>
```

È molto importante proteggere gli indirizzi e-mail per evitare che vengano "raccolti" e utilizzati per poterli inviare messaggi e/o pubblicità non voluta dall'utente, in poche parole spam. Così nel sistema FOAF si è adoperato il metodo della protezione degli indirizzi e-mail attraverso la "SHA1-encyrpted" come alternativa alla visualizzazione per intero degli stessi.

Così il campo contenente l'e-mail della persona

```
<foaf:mbox rdf:resource="mailto:luciano@ligabue.com"/>
```

può essere protetto dagli spammer attraverso il campo `foaf:mbox_sha1sum`

```
<foaf:mbox_sha1sum> f320551ba7e1f00599b78952e3320d022f78a320
</foaf:mbox_sha1sum>
```

Generalmente le descrizioni FOAF sono molto più lunghe poiché comprendono anche altri campi, ad esempio:

- i nomi ed i contatti degli amici

```
<foaf:knows>
  <foaf:Person>
    <foaf:firstname>Vasco</foaf:firstname>
    <foaf:surname>Rossi</foaf:surname>
    <foaf:nick>Vasco</foaf:nick>
     <foaf:mbox_sha1sum>29dbf1d738057be4a6336c3b773d3b248e8152eb
     </foaf:mbox_sha1sum>
    <rdfs:seeAlso rdf:resource="http://www.vascorossi.net/foaf.rdf"/>
  </foaf:Person>
</foaf:knows>
```

- le informazioni scolastiche

```
<foaf:schoolHomepage rdf:resource="http://www.ing.unimo.it/"/>
```

- la locazione geografica

```
<foaf:based_near>
  <geo:Point>
    <geo:lat>51.389</geo:lat>
    <geo:long>-2.4014</geo:long>
  </geo:Point>
</foaf:based_near>
```

- eventuali progetti correnti e/o passati

```
<foaf:currentProject>
  <rdf:Description rdf:about="http://www.ukoln.ac.uk/qa-focus/">
    <dc:title>QA Focus</dc:title>
  </rdf:Description>
</foaf:currentProject>

<foaf:pastProject>
  <rdf:Description rdf:about="http://www.cultivate-int.org/">
    <dc:title>Cultivate Interactive</dc:title>
  </rdf:Description>
</foaf:pastProject>
```

Alcuni aspetti importanti di FOAF:

- La proprietà `foaf:knows` punta ad altre persone conosciute dall'utente stesso, creando, così, una comunità in rete;

- Nel mondo FOAF, gli utenti non hanno bisogno di un URI, perché sono identificati attraverso la loro `foaf:mbox` (oppure `foaf:mbox_sha1sum`), ad esempio, l'indirizzo e-mail;
- Le proprietà `foaf:knows` non assumono il valore dell'URI di altre persone, ma puntano ad un anonimo nodo RDF del tipo `foaf:Person`, che contiene la `foaf:mbox` dell'altra persona;
- `foaf:mbox_sha1sum` viene utilizzata per nascondere l'indirizzo e-mail per motivi di privacy;
- Altri file di FOAF sono collegati attraverso `rdfs:seeAlso`.

4.4.2 FOAF-a-Matic

FOAF-a-Matic [39] è un'applicazione creata da Leigh Doll e realizzata in JavaScript con il compito di semplificare e velocizzare la creazione di descrizioni FOAF.

L'applicazione in questione richiede almeno l'immissione del proprio nome e dell'indirizzo di posta elettronica per poter effettuare la descrizione; senza tali dati la descrizione non ha nessun valore. Inoltre permette di poter inserire il nome e l'indirizzo di posta di amici che si vuole aggiungere alla descrizione: è utile aggiungere i riferimenti ai propri amici, perché in questo modo si permette agli spider, che indicizzano ed analizzano le descrizioni, di collegare tra loro gruppi di persone.

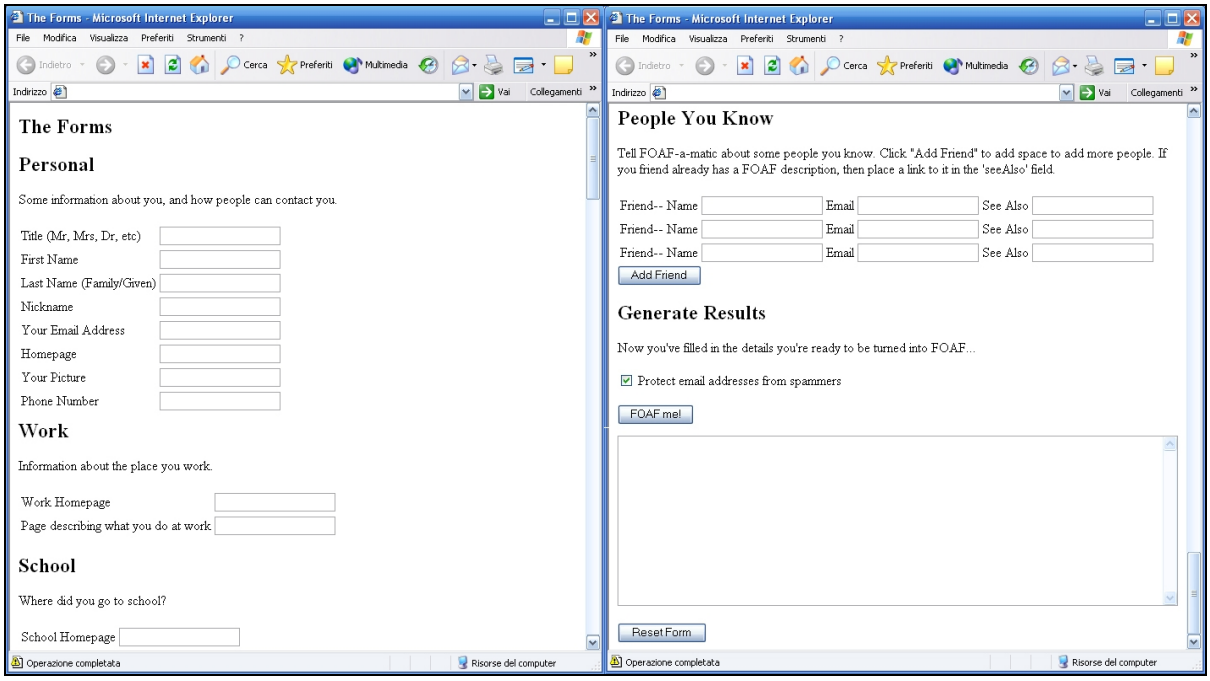


FIGURA 21 – FOAF-a-Matic

4.4.3 FOAF-a-Matic Mark 2

FOAF-a-Matic Mark 2 [40] è un'applicazione desktop, creata da Leigh Doll, per creare e gestire i dati FOAF. Essa è una ripresa dell'applicazione precedentemente menzionata, ma si presenta come applicazione GUI, Open Source ed è distribuita utilizzando la Attribution License dalla Creative Commons [41]. È realizzata in Java ed utilizza il Thinlet framework per lo sviluppo dell'interfaccia utente.

L'applicazione controlla i dati inseriti ed avvisa se la convalida non è andata a buon fine, come ad esempio la mancanza dell'indirizzo e-mail.

La descrizione FOAF viene visualizzata a video, con la possibilità di salvarla su file. Inoltre permette di gestire una "essenziale" rubrica con i dati nominativi, l'indirizzo e-mail e, se esiste, un'eventuale link alla relativa descrizione FOAF. Ovviamente tali informazioni sono inserite nel file FOAF.

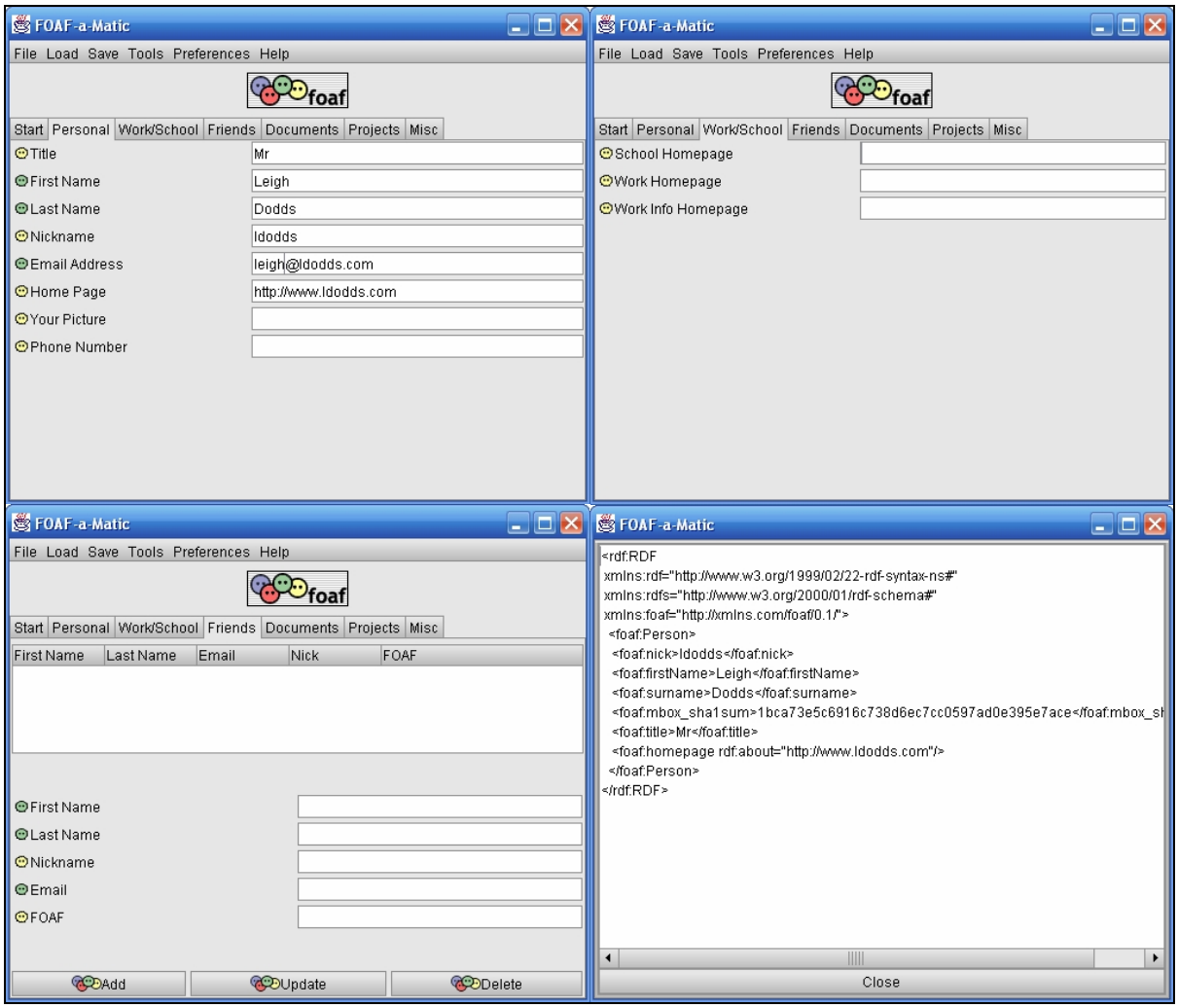


FIGURA 22 – FOAF-a-Matic Mark 2

4.5 Controllare la descrizione

Se si vuole controllare l'esattezza della propria descrizione FOAF, esiste un'applicazione Web che permette ciò e si chiama RDF Validation Service [42].

Questo servizio di convalida RDF è un'applicazione basata su Another RDF Parser (ARP) [43] creata e sostenuta da Jeremy Carroll per la HP-Labs di Bristol.

Ma tale servizio, oltre alla convalida, fornisce un grafico dettagliato seguendo tutti i campi `foaf:knows` presenti riguardo gli "amici" che si trovano nella descrizione.

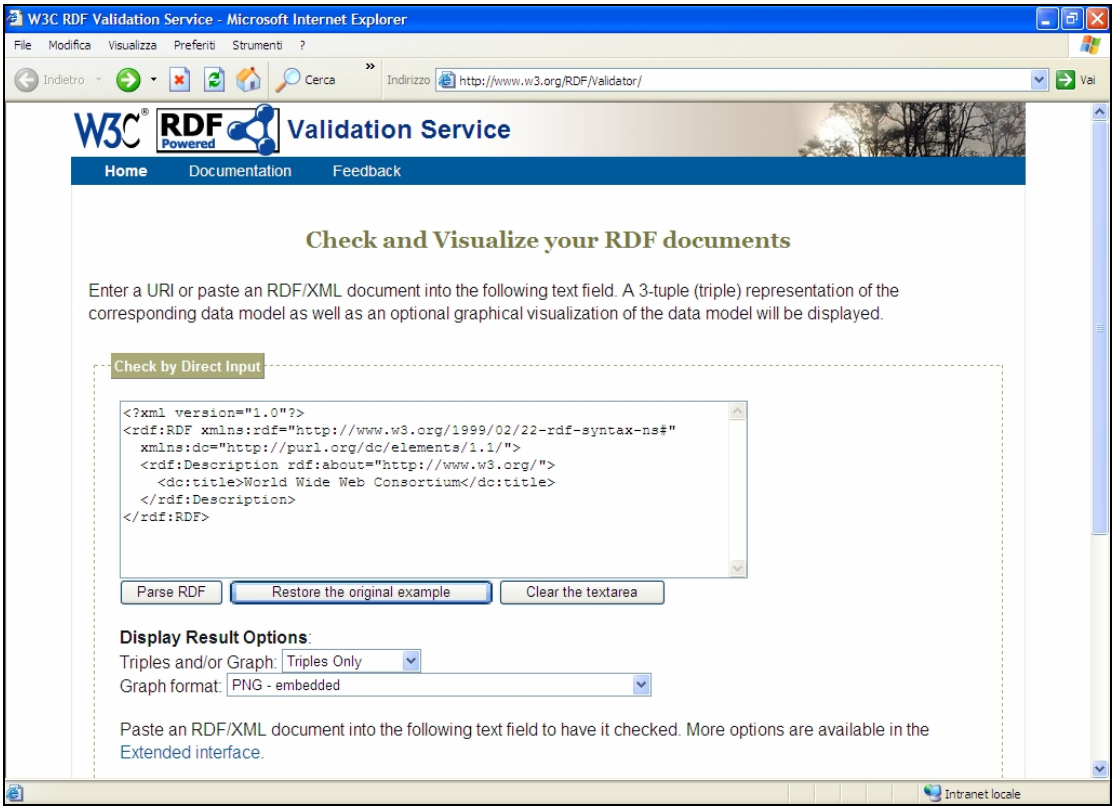


FIGURA 23 – RDF Validation Service

4.6 Pubblicare la descrizione

Dopo aver generato la propria descrizione FOAF, basta semplicemente inserirla in un file. Si può pubblicare tale file sul sito web personale: è una buona idea chiamare questo file "`foaf.rdf`" in modo da permettere l'uso attraverso un motore di ricerca, ad esempio con Google, per trovare i file FOAF presenti nel Web.

Adesso che la descrizione FOAF è pronta, è il momento di unirsi alla comunità permettendo alle altre persone partecipanti di trovarla e di leggerla.

Gli aspetti di "scoperta" di FOAF (ad esempio il modo in cui le applicazioni compatibili trovano le descrizione) sono ancora oggetto di discussione. Al momento vi sono varie possibilità da provare:

- *foaf:knows*: tale proprietà punta ad altre persone conosciute dall'utente stesso creando, così, una comunità in rete. Lo si può fare attraverso il campo `rdfs:seeAlso`;
- *Bulletin Board*: vengono memorizzati in un'apposita pagina web, denominata Wiki page, i link a molti file FOAF;
- *Auto-Discovery*: si genera una pagina HTML contenente il link al documento FOAF.

4.6.1 foaf:knows

Un modo per avere il proprio file FOAF indicizzato è di avere altre persone (ad esempio amici) che puntano ad esso nel loro FOAF. Uno spider FOAF può quindi trovare nuovi file seguendo i link presenti nei file già indicizzati. Lo si può fare apportando le seguenti modifiche alla descrizione FOAF:

1. Modificare l'elemento `rdf:RDF` per aggiungere il namespace RDF-Schema, come segue:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

2. Aggiungere i link ad altre descrizioni FOAF aggiungendo un elemento `rdfs:seeAlso` per ciascun file aggiuntivo, come segue:

```
<foaf:knows>
  <foaf:Person>
   ...
    <rdfs:seeAlso rdf:resource="http://www.example.com/friends.xrdf"/>
    <rdfs:seeAlso rdf:resource="http://www.ldodds.com/webwho.xrdf"/>
   ...
  </foaf:Person>
</foaf:knows>
```

4.6.2 Bulletin Board

Un modo molto semplice per trovare descrizioni FOAF è quello di possedere un elenco di persone, simile ad un elenco telefonico. Esiste un modo semplice per farlo, chiamato FOAF Bulletin Board.

FOAF Bulletin Board [44] è un forum FOAF che può essere utilizzato da qualunque persona e, dopo aver creato la propria descrizione FOAF, la si può aggiungere alla lista.

Semplicemente si visita il FOAFWiki [45] e, dopo una breve e piccola registrazione, si modifica la pagina aggiungendo il proprio nome ed il link alla propria descrizione FOAF.

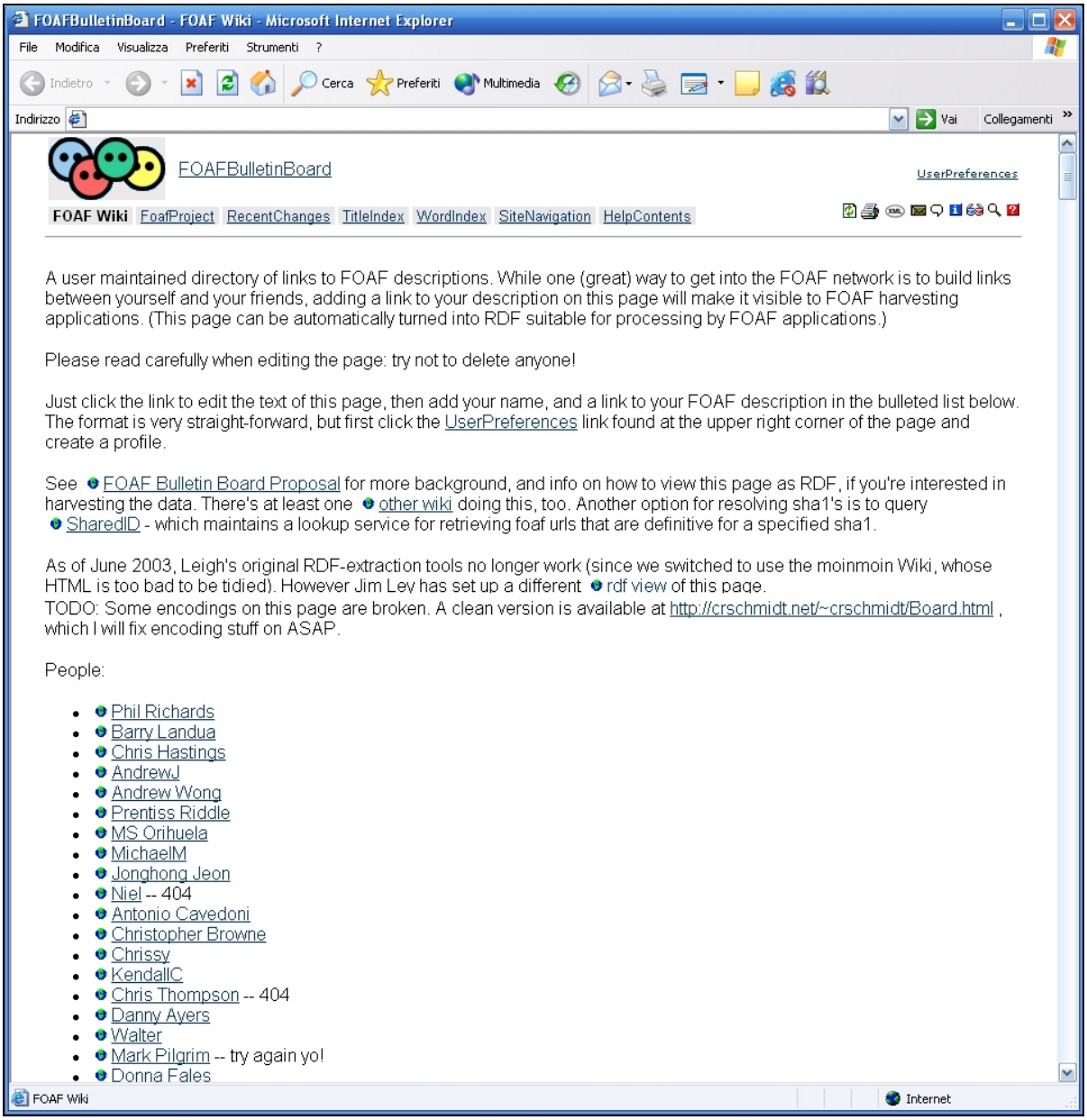


FIGURA 24 – FOAF Bulletin Board

4.6.3 Auto-Discovery

Un altro modo ed è quello di usare il tag HTML Link per puntare alla descrizione FOAF, allo stesso modo in cui molti blogger puntano ai loro feed RSS. Il nome consigliato da assegnare al file FOAF è `foaf.rdf` [46].

Ecco come dovrebbe apparire il codice della pagina HTML:

```
<html>
 <head>
 <link rel="meta" type="application/rdf+xml" title="FOAF"
                              href="http://www.sitodiesempio.it/foaf.rdf"/>
 </head>
 <body>
 ...
 </body>
</html>
```

4.7 Visualizzare le descrizioni

Una volta generata la descrizione FOAF e "collegata" al Web attraverso uno dei metodi precedentemente menzionati, le informazioni sono integrate nel Semantic Web e possono essere visualizzate attraverso apposite applicazioni, tra cui:

- *FOAF Explorer*;
- *Plink*;
- *FOAFNaut*;
- *TheyRule*.

4.7.1 FOAF Explorer

Se si conosce l'autore di una descrizione FOAF, la si può leggere utilizzando un visualizzatore FOAF. FOAF Explorer [47] permette di esplorare non soltanto lo spazio FOAF, ma può essere utilizzato anche per aggiornare la propria descrizione FOAF con le informazioni sulle persone appena trovate e conosciute.

Per esempio, come appare nella figura sottostante, se esplorate gli amici di Brian Kelly (il link è riportato in figura), potete scoprire che nella sua lista di amici è presente Tom Heath. Cliccando sull'icona appropriata (il simbolo rosso) aggiornerà la copia del vostro file FOAF con i relativi dati.

Un'altra applicazione molto simile è Advanced FOAF Explorer [48].

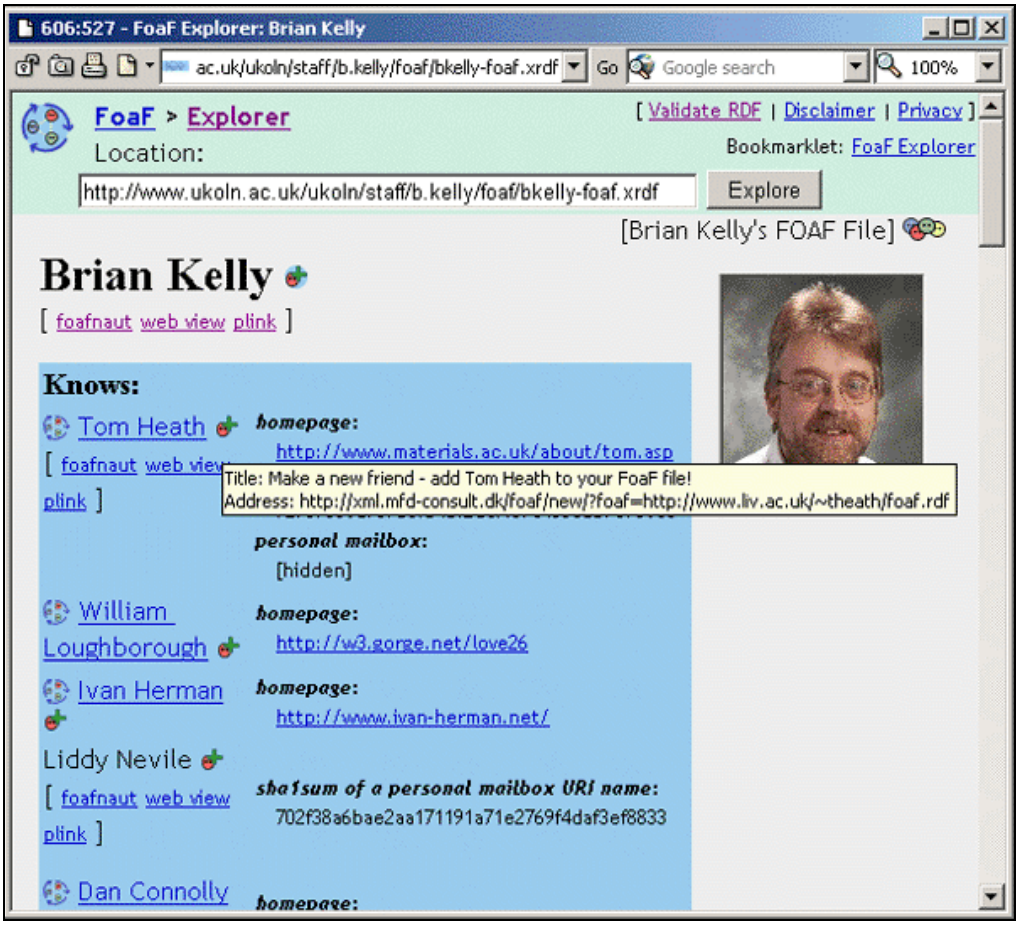


FIGURA 25 – FOAF Explorer

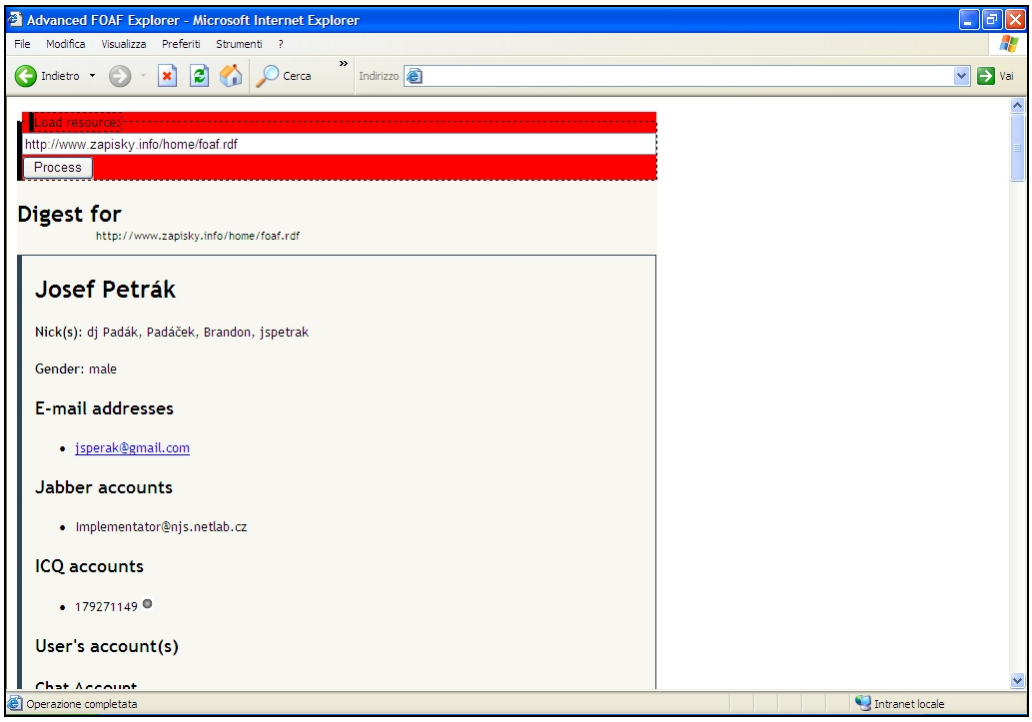


FIGURA 26 – Advanced FOAF Explorer

4.7.2 Plink

Plink [49] è un'applicazione che non fornisce soltanto i collegamenti delle persone conosciute da una data persona e viceversa, ma anche le persone che si trovano geograficamente vicino ad essa.

Infatti, nella descrizione FOAF può essere presente la voce riguardo alla posizione geografica: Plink elabora tali informazioni e le confronta con altri dati FOAF raccolti precedentemente; il motore esplora gli schedari FOAF e ne deduce "automaticamente" se sono presenti "somiglianze". Così, attraverso Plink, si possono visualizzare non solo gli amici di una data persona, ma anche le persone che lavorano vicino ad essa.

Ora tale progetto è stato abbandonato ed il sito è stato chiuso, poiché il gruppo di lavoro ha deciso di svilupparne un altro riguardante le foto su blog.

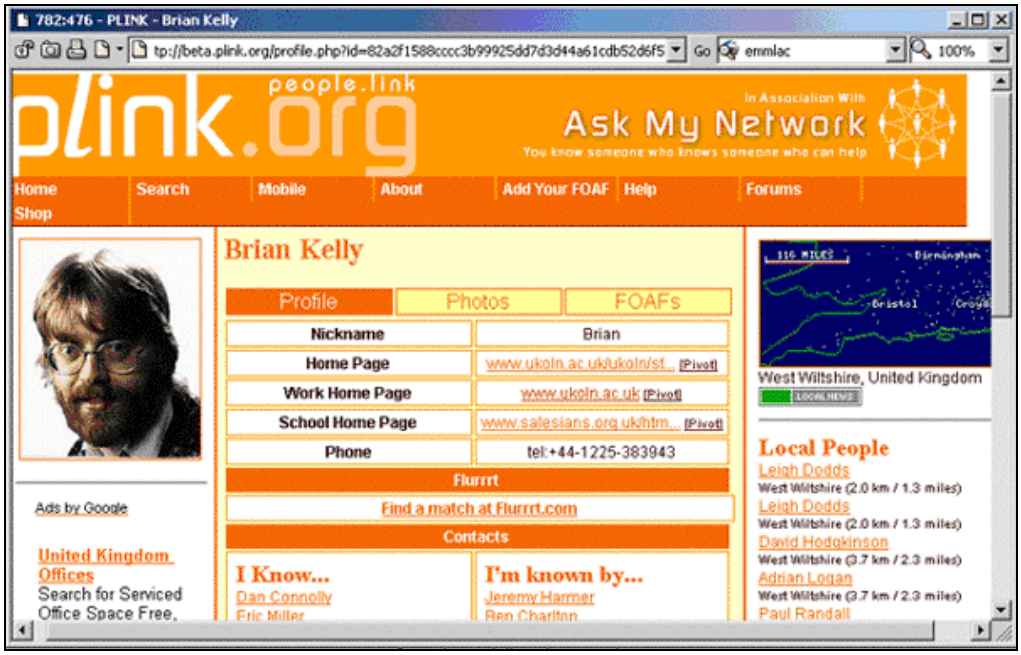


FIGURA 27 – Plink

4.7.3 FOAFNaut

Un modo elegante per poter visualizzare le relazioni che si possono trovare sulla rete FOAF è quello di utilizzare FOAFNaut [50], un'applicazione SVG che fornisce una chiara visualizzazione delle relazioni tra varie persone, legate tra loro attraverso `foaf:knows`.

FOAFNaut è un'applicazione scritta in SVG/Javascript ed stata realizzata da Jim Ley, con l'aiuto di Dean Jackson, Liz Turner e dalle persone appartenenti alla comunità RDFWeb IRC [51].

Per utilizzare tale applicazione basta un comunissimo browser, ma in aggiunta si deve installare il SVG plug-in. L'SVG (Scalable Vector Graphics) [52] è il modello di imaging standard W3C basato su XML che consente agli sviluppatori, ai designer e agli utenti Web di superare i limiti del formato HTML e creare contenuti grafici affidabili e interattivi utilizzando un semplice modello di programmazione dichiarativa. Con questa efficiente tecnologia, gli sviluppatori XML possono creare applicazioni Web basate su grafici personalizzati, interattivi e guidati dai dati provenienti in tempo reale da diverse sorgenti, quali sistemi di e-commerce e database aziendali.

La ricerca in FOAFNaut si effettua attraverso l'indirizzo e-mail della persona che si cerca/conosce, poiché non esiste un modo migliore e più semplice per identificare le persone su Internet. Per poter inserire nuovi link, si deve utilizzare la pagina Wiki dedicata a FOAFNaut [53].

Ad esempio, se effettuiamo una ricerca su Libby Miller (libby.miller@bristol.ac.uk), avremo tali risultati:

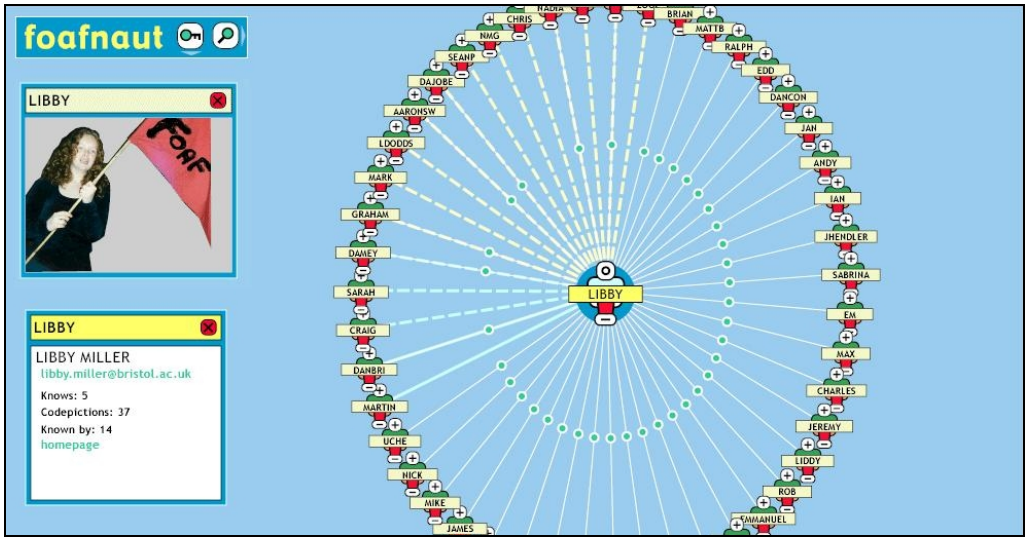
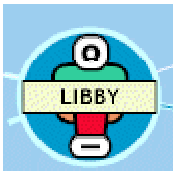


FIGURA 28 – Ricerca su FOAFNaut

Per ogni persona si avranno i riferimenti di tutte le persone che conosce e di quelle che la conoscono. Tali blub si possono a loro volta "espandere" e vedere le altre relazioni.



FOAFNaut rappresenta graficamente le persone attraverso i blub, ognuno contenente il nickname o una parte dell'indirizzo e-mail. I blub sono collegati tra loro attraverso apposite linee, ognuna con appropriate proprietà. Inoltre, in due appositi quadri, appaiono la relativa descrizione della persona ed un'eventuale foto/immagine.

FOAFNaut permette di interlacciare i risultati con altre persone: se dalla ricerca precedente ne effettuiamo un'altra su Edd Dumbill (edd@xml.com), possiamo notare che le due persone poste in relazione hanno degli amici in comune.

Oltre alle persone, FOAFNaut permette di collegare tra loro anche eventi.

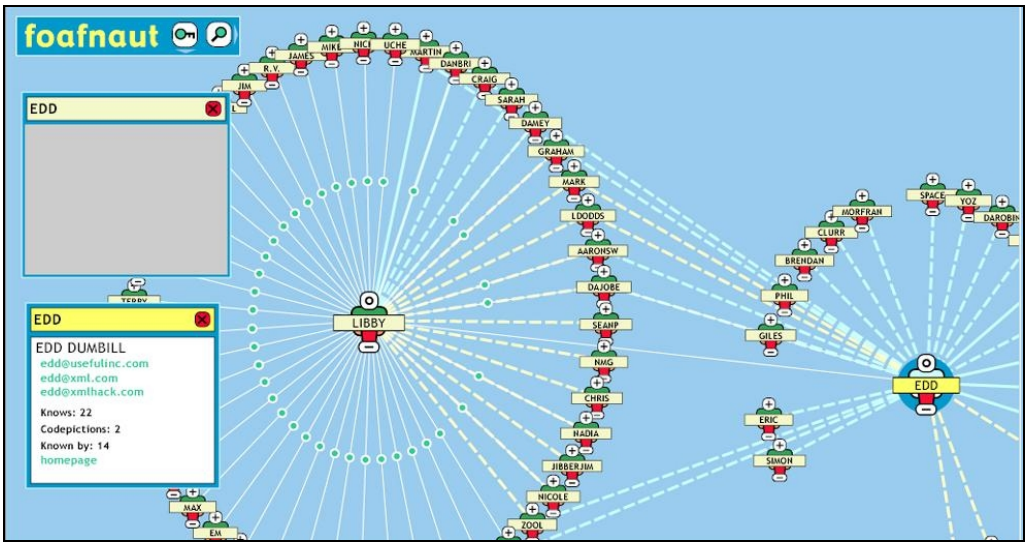


FIGURA 29 – Ricerca interlacciata su FOAFNaut

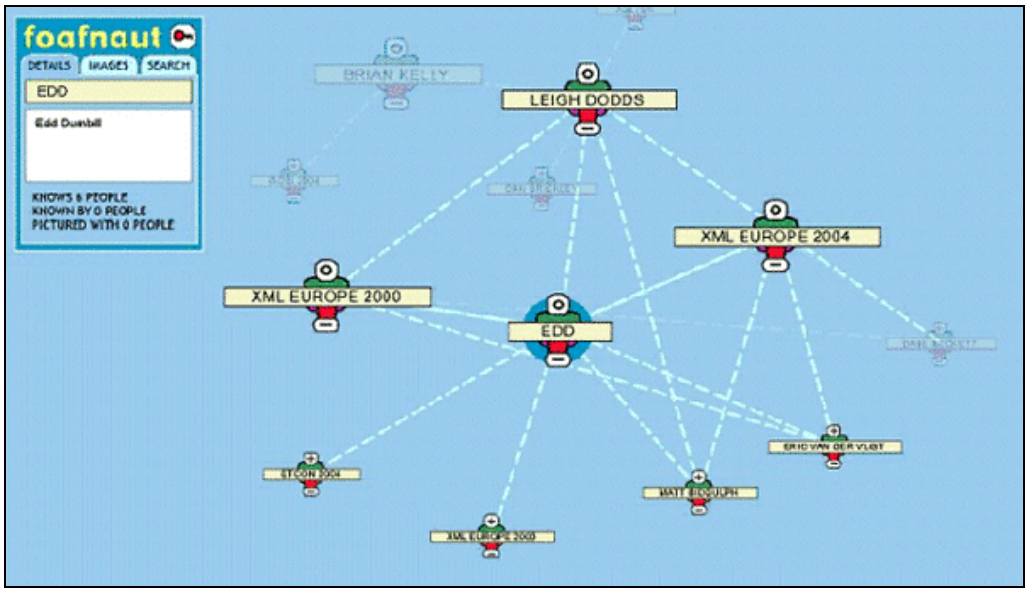


FIGURA 30 – Ricerca di eventi su FOAFNaut

4.7.4 TheyRule

TheyRule [54] è un'interessante applicazione del Semantic Web che riguarda l'intercomunicabilità delle principali 100 compagnie Americane, ossia le relazioni delle persone membri a tali società.

Questa applicazione è stata realizzata tramite l'SVG da un lavoro di Josh On, con l'aiuto di Amy Balkin, Amy Franceschini ed il supporto della FutureFarmers.com. Le informazioni e i relativi link sono aggiornati annualmente.

Tale progetto da un'idea abbastanza pratica di quello che si può realizzare a partire da FOAF.

In TheyRule la rappresentazione della persona viene effettuata attraverso un'apposita icona a forma di uomo o di donna, in base al sesso della stessa, mentre la sua importanza nella società è rappresentata attraverso la sua corposità: più è "grassa" la persona, maggiore è il ruolo che ricopre nell'azienda. Tale rappresentazione non ha alcuna relazione con le caratteristiche fisiche della persona.

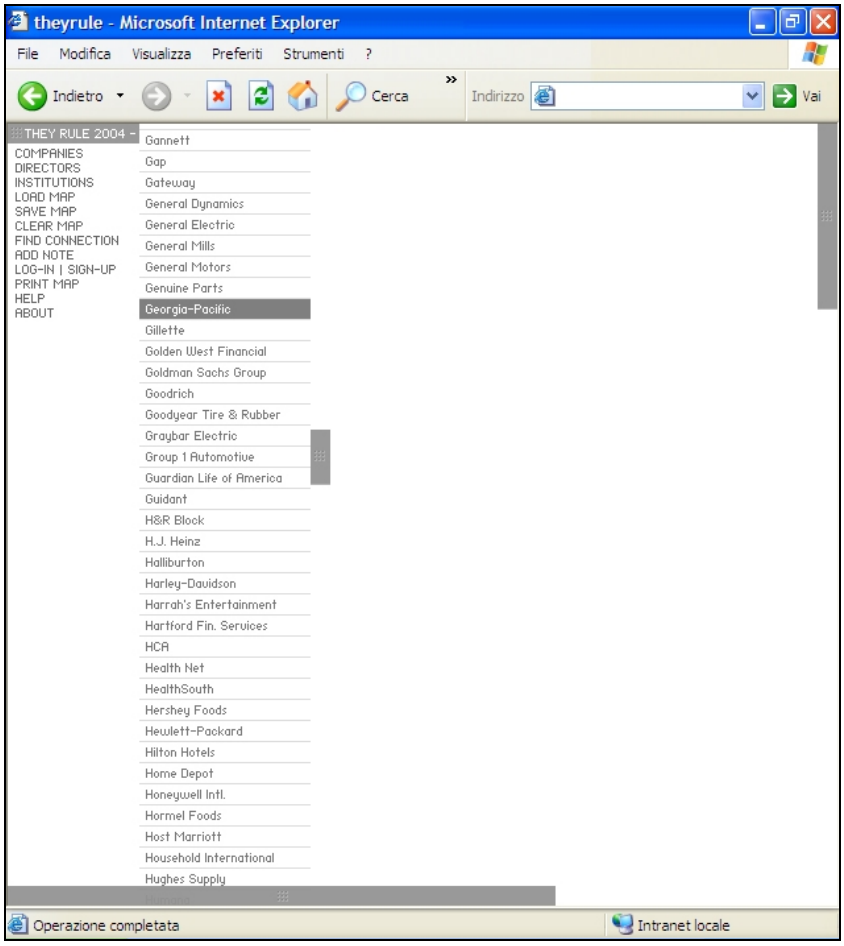


FIGURA 31 – Le compagnie di TheyRule

Per ogni azienda sono memorizzate le informazioni come il sito web ed eventuali contatti; inoltre TheyRule include un pulsante che permette di effettuare ricerche sulla azienda selezionata attraverso un motore di ricerca (ad esempio Google).
Come nell'applicazione FOAFNaut, anche qui si possono mettere in relazione tra loro più compagnie e vederne le varie correlazioni.

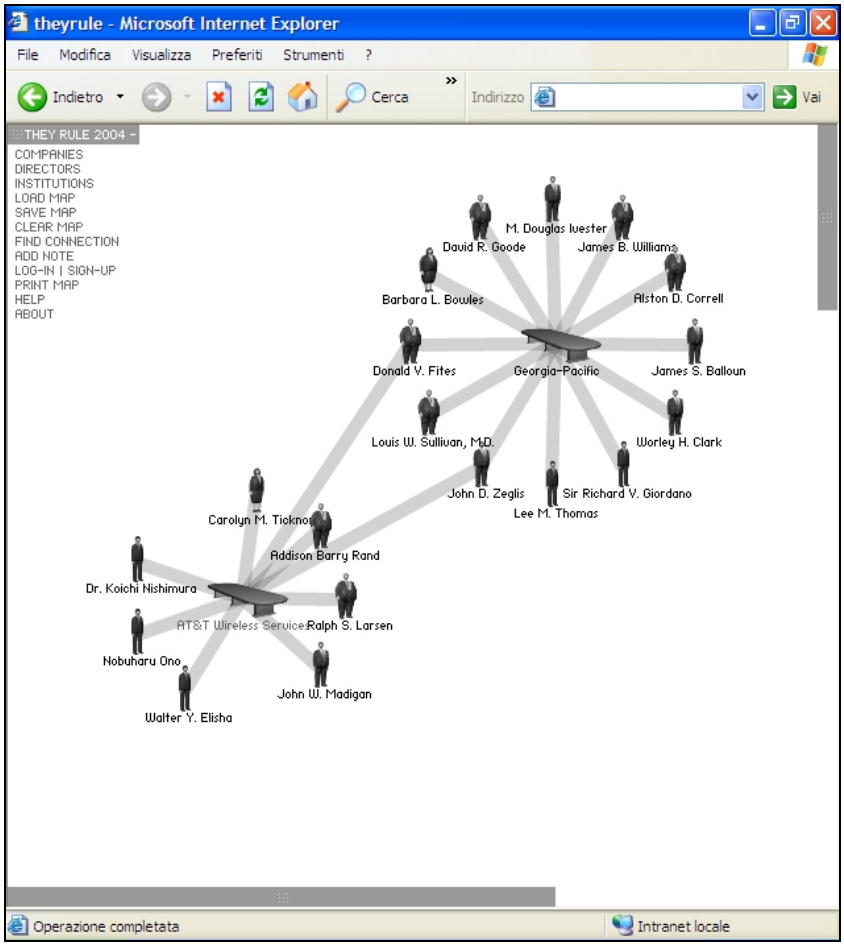


FIGURA 32 – Relazioni interlacciate su TheyRule

Da questo lavoro è stata sviluppata un'altra applicazione, molto simile, che prende il nome di FOAFCorp [55], denominata anche Fat Cats.
FOAFCorp è un'applicazione di Dean Jackson, dove le persone sono rappresentate attraverso dei gatti. Anche qui sono presenti le 100 principali compagnie americane e la relazione tra corposità - importanza.

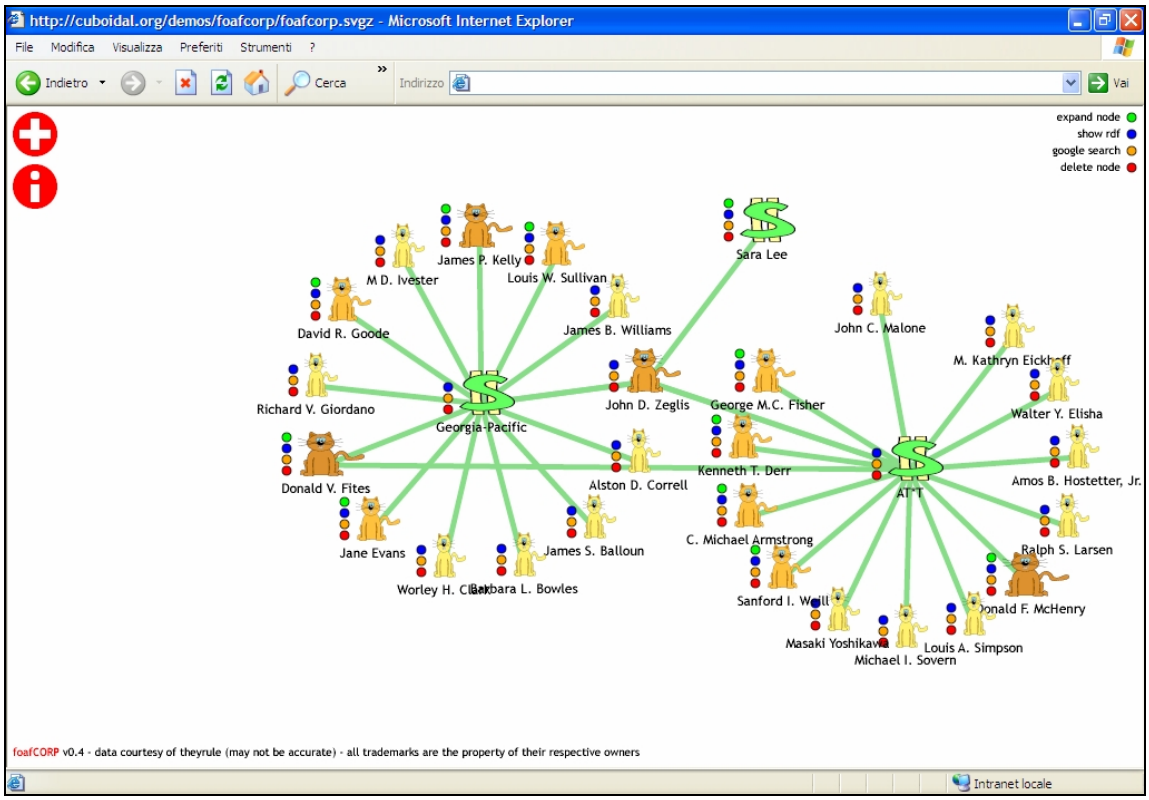


FIGURA 33 – Relazioni interlacciate su FOAFCorp

Entrambi i progetti possono essere modificati e migliorati da chiunque, come ad esempio:

- nella User Interface (interfaccia utente);
- nella rappresentazione delle persone;
- nell'aggiungere più dati (all'interno dell'RDF), come ad esempio: qualcuno è sposato? con chi?;
- ecc.

Tale idea si può utilizzare per poter realizzare anche altri tipi di relazioni, non soltanto quelle relative alle aziende, ma anche in altri contesti, come ad esempio scolastici/universitari, i componenti di un dato progetto, ecc.

4.8 Crawler RDF

Il RDF Crawler [56] [57] è un'applicazione che ha come obiettivo la raccolta di dati RDF/XML presenti nel Semantic Web, collegati tra loro attraverso i link contenuti nei campi `rdfs:seeAlso` delle descrizioni RDF.
È un programma realizzato in Java, multithread e memorizza localmente i dati raccolti codificati nel loro formato originale (Unicode UTF-8).

L'idea originale di realizzare un RDF Crawler non è stata presa dall'autore dell'applicazione in esame, poiché essa era stata presa precedentemente in vari altri progetti.
La versione più vecchia è stata realizzata dalla SECO System [58]; successivamente ne è stata generata un'altra dalla Institute AIFB [59], creata da Kalvis Apsitis, ma successivamente tale progetto è stato abbandonato.
Il crawler risultante dalla Institute AIFB è stato ripreso e revisionato da Aidan Hogan, mentre la versione corrente è stata completamente riscritta da Andreas Harth; entrambi lavorano al DERI (Digital Enterprise Research Institute) [60].

Tale applicazione in esame:

1. si lancia tramite linea di comando e, in aggiunta come parametro, vi è la URI della/e risorsa/e RDF da dove iniziare la ricerca;
2. analizza da tale/i file i link disponibili;
3. aggiunge i link in coda e continua nuovamente a cercare altre risorse RDF da tali link.

Attraverso il parametro –h si possono visualizzare informazioni aggiuntive:

```
java -jar rdfcrawler.jar -h
```

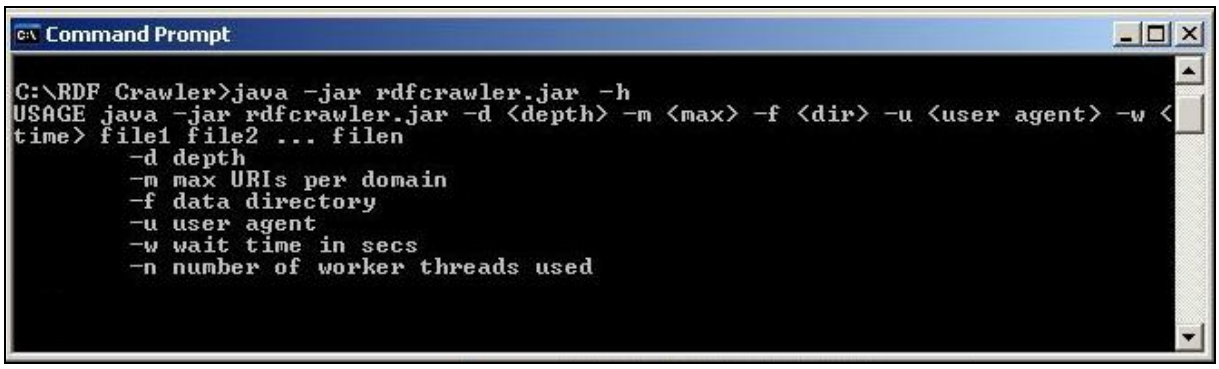


FIGURA 34 – RDF Crawler help

Parametro	Valore
-d	Profondità
-m	Numero massimo di URI per dominio
-f	Specifica la directory dove memorizzare i dati
-u	User Agent
-w	Periodo di attesa in secondi
-n	Numero di worker thread da utilizzare

Ad esempio, se inseriamo da linea il seguente comando:

```
java -jar rdfcrawler.jar -d 3 http://sw.deri.org/~aharth/foaf.rdf
```

dove il link punta alla descrizione FOAF di Andreas Harth. In tale descrizione sono presenti molti campi `rdfs:seeAlso` riguardanti informazioni personali, amicizie, progetti, foto e blog.

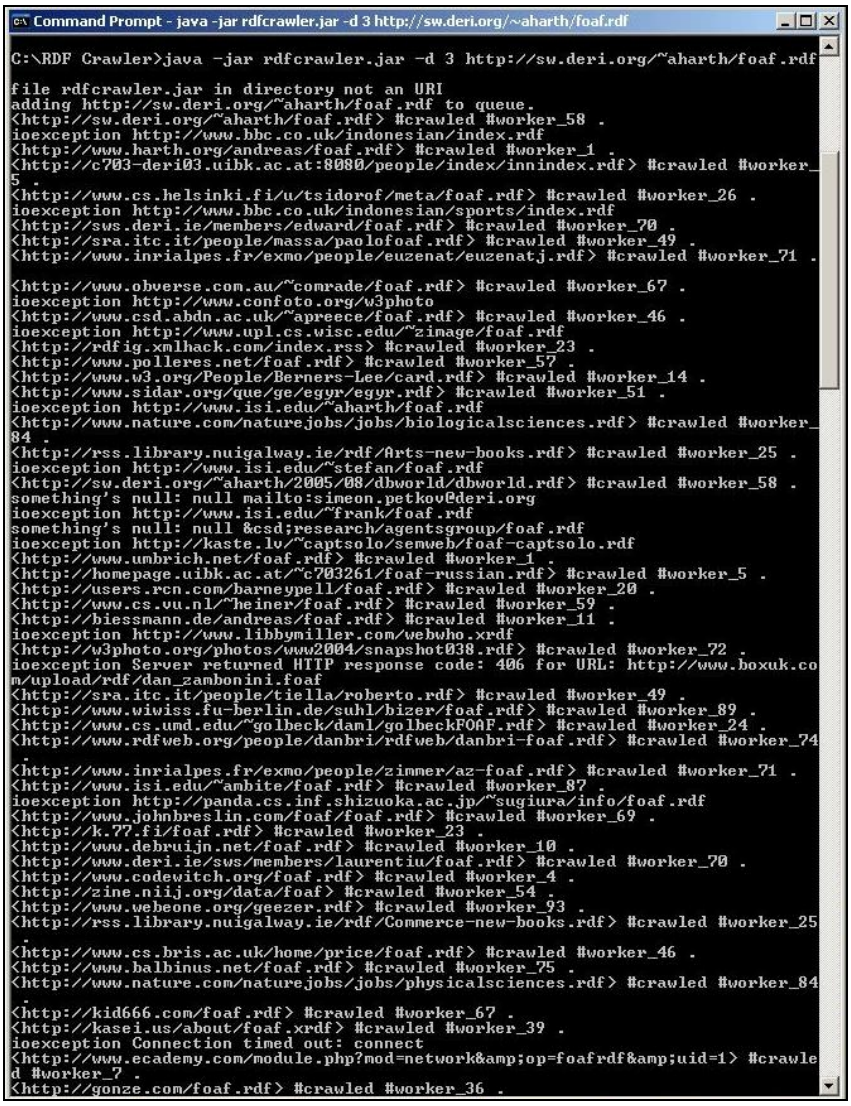


FIGURA 35 – Esecuzione dell'RDF Crawler

Poiché la profondità che abbiamo impostato nella ricerca è stata di 3, il crawler, dopo aver salvato localmente la descrizione FOAF di partenza, inizierà a processare i link contenuti nei vari campi `rdfs:seeAlso` e a salvare tali descrizioni; il crawler finirà la sua ricerca quando sarà arrivato ad una profondità uguale a 3 oppure quando non saranno più disponibili link per poter arrivare alla profondità impostata nella ricerca.

I risultati che si ottengono dalle ricerche, anche partendo dalla stessa descrizione, sono variabili poiché ogni giorno possono essere aggiunti, ma anche rimossi, link ad altrettante descrizioni riguardanti amicizie e/o interessi.

Dalla ricerca impostata, con profondità uguale a 3, sono state trovate le seguenti descrizioni.

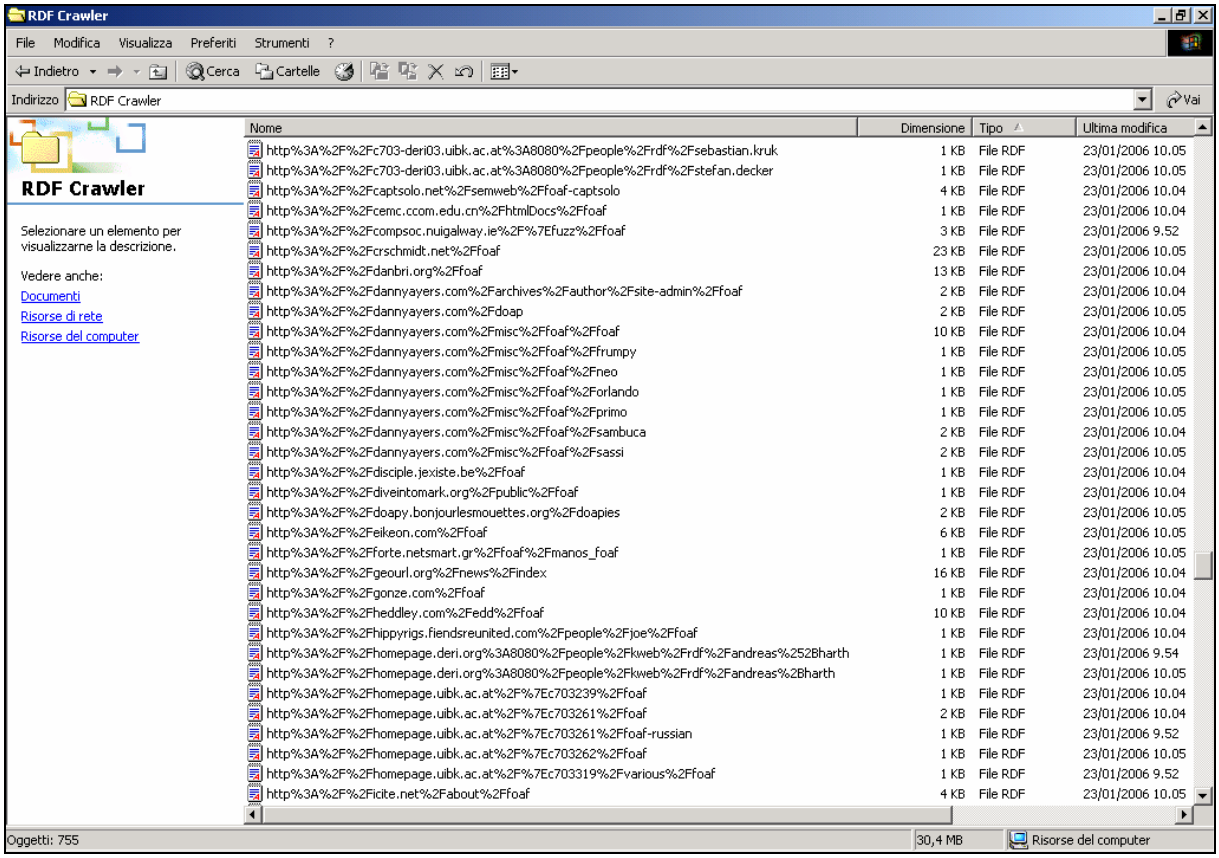


FIGURA 36 – Risultati dell'RDF Crawler

4.9 FOAF-Realm

FOAF è un progetto che fa uso dell'RDF per descrivere le relazioni di amicizia fra le persone. Si possono utilizzare, quindi, queste informazioni anche, ad esempio, per controllare l'accesso alle risorse. L'unica cosa che interessa è conoscere la "distanza" fra una persona ed un'altra. Si può, a questo punto, descrivere come una persona

può guardare, ad esempio, le nostre foto specificando la lunghezza massima del percorso, che separa me da lui o lei.
È molto comune, nel mondo reale, dire che uno dei nostri amici è più intimo degli altri. In molti casi valutiamo le nostre amicizie in base agli eventi accaduti finora. Per questo motivo, esistono delle estensioni di FOAF che forniscono proprietà addizionali utili per dire, ad esempio, che qualcuno è un nostro "grande amico" oppure "non l'ho mai incontrato".

L'intuizione di FOAF-Realm [61] è di trattare le situazioni del mondo reale in maniera simile. Ad esempio, in alcuni casi, noi vorremmo condividere alcune risorse con amici dei nostri amici piuttosto che con i nostri stessi amici. Se abbiamo un grande amico, ciò significa che i suoi grandi amici sono conosciuti meglio da noi rispetto ad alcuni nostri amici che conosciamo appena o che non abbiamo mai incontrato. Questo è il motivo perché la valutazione delle amicizia è considerata una soluzione.
Assumendo che `foaf:knows` rappresenti l'amicizia media, c'è comunque un intero intervallo di amicizia, dal molto stretto (un grande amico) a molto distante (persona che non ho mai conosciuto). Per questo motivo, si è pensato di valutare ogni amicizia da 0% (molto distante) a 100% (molto stretto), definendo con il 50% l'amicizia media.
FOAF-Realm è stato realizzato per permettere tutto questo. L'utente, infatti, può descrivere il ruolo che deve avere colui che può vedere le risorse dell'utente stesso.

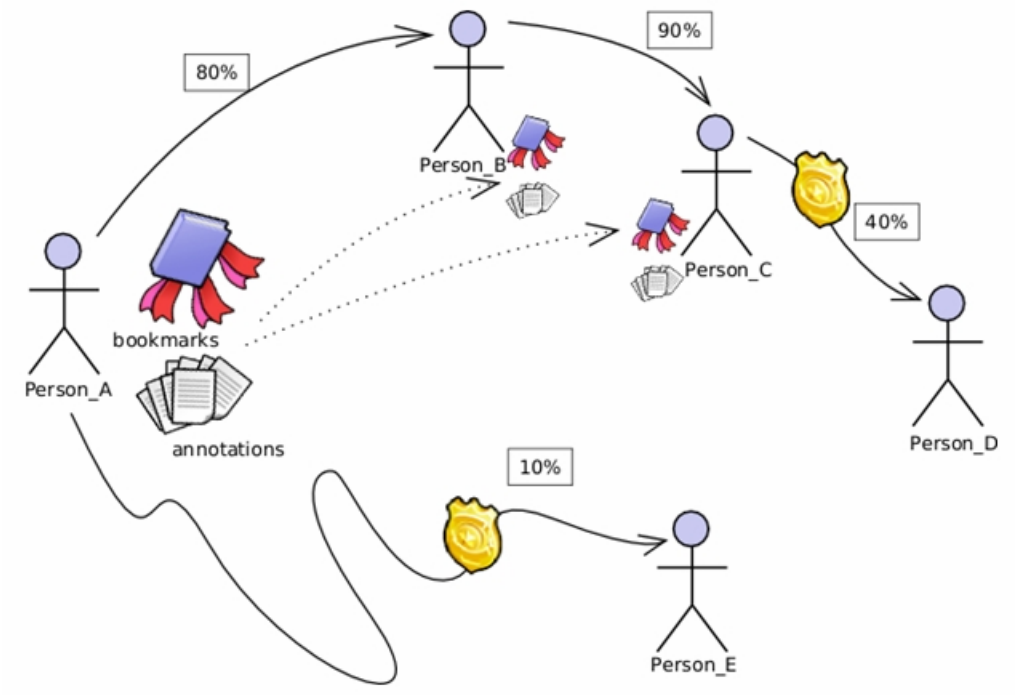


FIGURA 37 – La condivisione con gli amici più stretti in FOAF-Realm

Attraverso le espressioni di ruolo definite in FOAF-Realm, ognuno può specificare molto precisamente chi può vedere o modificare qualche dato. La figura mostra come può la Person_A permettere alla Person_B e alla Person_C di ottenere l'accesso alle sue risorse, mantenendo i suoi stessi dati protetti dagli utenti non autorizzati, come la Person_D e la Person_E.

4.10 Semantic Campus

Altri vocabolari, come ad esempio il Dublin Core, RSS 1.0, ecc., possono essere combinati con i termini FOAF e considerati come estensioni locali; infatti FOAF è stato progettato con la possibilità di essere esteso. Nella pagina Web di FOAFVocab [62], presente nel sito di FOAF Wiki, sono elencate varie estensioni di vocabolari che sono particolarmente idonei ad essere utilizzati con sistema FOAF. Tra tali vocabolari ha un particolare interesse, in ambito universitario, il progetto Semantic Campus.

Il progetto Semantic Campus [63] punta a creare ed a definire un vocabolario RDF per la descrizione dell'ambiente universitario, relativo alle risorse riguardo le università, i dipartimenti, i professori e gli studenti. Tale progetto è stato ideato da Benjamin Nowack.

Come FOAF utilizza un vocabolario comune per le applicazioni RDF relative alla descrizione delle persone, il progetto Semantic Campus prova a costruire, mantenere e documentare un insieme di termini che possono contribuire a descrivere una specializzata ontologia in ambito universitario. Tale vocabolario può essere considerato ed utilizzato come estensione del sistema FOAF poiché rispetta le stesse specifiche.

Tra le varie figure di utenti che possono essere "inserite" in tale progetto, sono presenti: studenti, ex studenti, professori, ricercatori, dipendenti universitari ed organizzazioni esterne, ad esempio aziende.

Tale progetto può essere utilizzato, ad esempio, per:

- annotare e cercare pubblicazioni;
- cercare appunti sulle lezioni mancate;
- cercare persone che possono dare aiuto in un dato argomento;
- cercare persone che preferiscono le stesse letture o che preparano gli stessi esami;
- trovare le aule per un dato corso di studi;
- cercare tra gli studenti iscritti ad un determinato esame, corso, anno, ecc.;
- offrire e cercare argomenti riguardanti le tesi;

- cercare università che focalizzano determinati argomenti;
- piano di studio basato sul Web;
- cercare il numero delle aule e gli orari di ricevimento;
- annunci sui cambiamenti delle aule, date di registrazione, …;
- descrivere/cercare eventi e le relative risorse;
- trovare risposte alle F.A.Q. su un determinato argomento;
- ecc.

In base alle azioni sopra menzionate, si possono definire una lista di termini per poter descrivere in modo specifico determinate persone o cose in ambito universitario (a parte i termini riutilizzabili dalla specifica di FOAF). Da tale termini possono essere ricavate le seguenti classi:

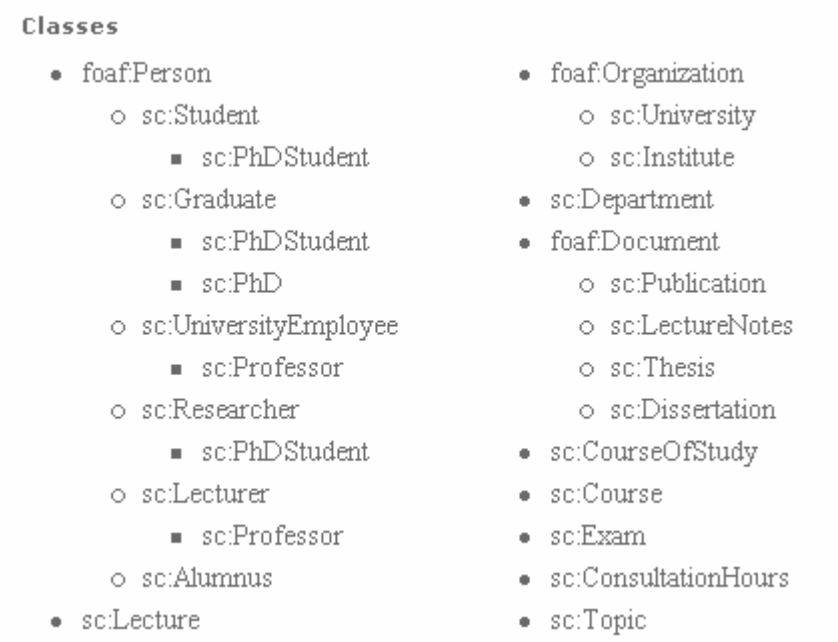


FIGURA 38 – Le classi della Semantic Campus

dove *sc* è utilizzato come "namespace identifier" per l'ontologia in esame, cioè la Campus ontology, mentre i termini appartenenti alla specifica di FOAF sono preceduti da *foaf*.

Conclusioni e lavoro futuro

A partire dall'anno 2000 una nuova architettura software ha focalizzato l'attenzione degli utilizzatori della rete Internet: l'architettura Peer-to-Peer. Fautore di tale esplosione è stata la massiccia diffusione del software Napster e Gnutella per la distribuzione on-line di file digitali come immagini, audio, video e software.

Il Peer-to-Peer (p2p), però, non è solo sinonimo di "pirateria" di risorse digitali, ma è un'architettura che promette di rivoluzionare completamente la rete con sistemi per file sharing, condivisione delle risorse dei PC, lavori collaborativi ed altri servizi non ancora identificabili, eliminando la distinzione tra computer client e computer server.

I PC su cui viene eseguito un software p2p, tendono ad organizzarsi dinamicamente e autonomamente in comunità virtuali costituite da PC paritetici, i Peer appunto, e secondo modalità specifiche per ogni differente applicativo.

Il fatto che aziende del calibro di Microsoft e Sun si stiano interessando a questa architettura, dimostra che il p2p non è utile solo per lo scambio illegale di materiale tutelato da leggi sul diritto d'autore: entrambe stanno realizzando piattaforme di sviluppo (rispettivamente .NET Framework della Microsoft e JXTA della Sun MicroSystem) che facilitino la creazione di nuovi tool aderenti a questa architettura.

Come tutte le nuove architetture, anche questa non è priva di limitazioni e problemi: tra questi quello sicuramente più importante è il problema della sicurezza, legato sia alla mancanza all'interno delle comunità virtuali, che si formano, di un punto di controllo centralizzato che ne permetta una completa gestione e amministrazione, sia al fatto che ogni Peer fornisce, ad altri membri della comunità, l'accesso a risorse presenti sul proprio computer.

Altra fondamentale caratteristica, e allo stesso tempo limite, di un'architettura p2p completamente distribuita, è l'enorme traffico che si genera sulla rete per gestire correttamente una comunità virtuale anche di piccole dimensioni.

In questa tesi è stato affrontata l'analisi di architetture Peer-to-Peer (p2p) utili per la gestione di database distribuiti.

Inizialmente è stato introdotto l'argomento attraverso la descrizione delle diverse tipologie di reti, protocolli, applicazioni, anche quelle anonime, con l'obiettivo di darne una vasta introduzione alla materia (Capitolo 1).

Successivamente sono state descritte le varie architetture e la classifica generazionale (Capitolo 2), sono stati trattati gli aspetti generali riguardo la sicurezza e le problematiche che si riscontrano nelle applicazioni p2p (Capitolo 3).
Infine è stato analizzato FOAF, un sistema che fa uso dell'RDF per descrivere le relazioni di amicizia fra le persone; sono state analizzate anche alcune applicazioni che sono correlate ad essa, come FOAF-a-Matic, FOAF Explorer, FOAFNaut e il RDF Crawler (Capitolo 4).
L'idea di FOAF è quella di definire un grafico diretto e distribuito delle relazioni di amicizia, ma incontra qualche problema che dovrebbe essere risolto.
Il primo è il problema della sicurezza, che, oltre a tutte le informazioni descritte dall'ontologia FOAF, richiede anche che sia fornito un valore SHA1 di password. Come si possono scambiare informazioni lontano dal sistema FOAF se le e-mail sono protette dal sistema SHA1? Gli indirizzi e-mail non si potrebbero proteggere dallo spam in altri modi?
Il secondo è il problema della fiducia e della gestione: siccome l'informazione è distribuita, bisogna controllare che nessun estraneo aggiunga nuove relazioni di amicizie fittizie.
Inoltre FOAF utilizza un formato statico per i dati, ma potrebbe evolversi per essere simile ad una API (Application Program Interface), includendo un meccanismo per l'assegnazione dei permessi e l'autenticazione dei richiedenti?
Attualmente il campo `foaf:knows` può essere utilizzato solamente per puntare ad altri file FOAF, ma potrebbe essere utilizzato anche per altri motivi, come ad esempio rappresentare l'"user's address book".
Quindi la migliore condotta che la comunità FOAF possa fare è quella di considerare con attenzione i problemi legati a questo progetto e di sviluppare soluzioni tecniche riguardo la sicurezza sulle informazioni personali e dare risposte concrete per permettere a FOAF di poter transitare da un progetto di ricerca ad una tecnologia di corrente utilizzo.

Bibliografia

Riferimenti:

[1] **ICQ**, http://www.icq.com

[2] **MSN**, http://www.msn.com

[3] **AIM: AOL Instant Messenger**, http://www.aim.com

[4] **Jabber**, http://www.jabber.org

[5] **GIMPS**, http://www.mersenne.org

[6] **SETI@home**, http://setiathome.berkeley.edu

[7] **Distributed.net**, http://www.distributed.net

[8] **FightAIDS@Home**, http://www.fightaidsathome.org

[9] **Intel/Devices cancer research project**, http://members.ud.com/projects/cancer

[10] **IRC**, http://www.irc.org

[11] **Napster**, http://www.napster.com

[12] **Gnutella**, http://gnutella.wego.com e http://www.gnutella.com

[13] **Slashdot**, http://www.slashdot.org

[14] **Gnutella2**, http://www.gnutella2.com

[15] **Shareaza**, http://www.shareaza.com

[16] **FastTrack**, http://www.fasttrack.nu

[17] **Morpheus**, http://www.morpheus.com

[18] **eDonkey**, http://www.edonkey2000.com

[19] **eMule**, http://www.emule-project.net e http://www.emule.it

[20] **BitTorrent**, http://bittorrent.com

[21] **OpenNap**, http://opennap.sourceforge.net

[22] **WinMX**, http://www.winmx.com e http://www.winmxitalia.it

[23] **Direct Connect**, http://www.neo-modus.com e http://dcplusplus.sourceforge.net

[24] **Freenet**, http://www.freenetproject.org e http://freenet.souceforge.net

[25] **MojoNation**, http://www.mojonation.net/broker

[26] **Publius**, http://www.cs.nyu.edu/~waldman/publius

[27] **Mute**, http://mute-net.sourceforge.net

[28] **Sony Corporation of America. v. Universal City Studios Inc**, http://en.wikipedia.org/wiki/Sony_Corp._v._Universal_City_Studios

[29] **MGM Studios Inc. v. Grokster Ltd**, http://en.wikipedia.org/wiki/MGM_v._Grokster

[30] **Chord lookup service**, http://www.pdos.lcs.mit.edu/chord

[31] **PAST distributed storage utility**, http://www.research.microsoft.com/~antr/PAST

[32] **CoopNet cooperative content distribution system**, http://research.microsoft.com/~padmanab/projects/coopnet

[33] **OceanStore Project**, http://www.oceanstore.org

[34] **JXTA**, http://www.jxta.org

[35] **Sun MicroSystem**, http://www.sun.com

[36] **FOAF: Friend Of A Friend**, http://foaf-project.org e http://beta.foaf-project.org

[37] **FOAF Vocabulary Specification**, http://xmlns.com/foaf/0.1

[38] **WWW Consortium**, http://www.w3.org/Consortium

[39] **FOAF-a-Matic**, http://www.ldodds.com/foaf/foaf-a-matic.html

[40] **FOAF-a-Matic Mark 2**, http://www.ldodds.com/wordtin/Wiki.jsp?page=FOAFaMaticMark2

[41] **Creative Commons**, http://www.creativecommons.org

[42] **RDF Validator Service**, http://www.w3.org/RDF/Validator

[43] **Another RDF Parser**, http://www-uk.hpl.hp.com/people/jjc/arp

[44] **FOAF Bulletin Board**, http://www.ldodds.com/foaf/bulletin-board.html

[45] **FOAFWiki**, http://rdfweb.org/rweb/wiki/wiki.pl?FOAFBulletinBoard

[46] **Auto-Discovery**, http://rdfweb.org/topic/Autodiscovery

[47] **FOAF Explorer**, http://xml.mfd-consult.dk/foaf/explorer

[48] **Advanced FOAF Explorer**, http://foaf-explorer.zapisky.info

[49] **Plink**, http://www.plink.org

[50] **FOAFNaut**, http://www.foafnaut.org

[51] **RDFWeb IRC**, http://rdfweb.org/irc

[52] **SVG: Scalable Vector Graphics**, http://www.adobe.it/enterprise/svg.html e http://www.w3.org/Graphics/SVG/Overview.htm8

[53] **Wiki FOAFNaut**, http://rdfweb.org/rweb/wiki/wiki?FoafNaut

[54] **TheyRule**, http://www.theyrule.net

[55] **FOAFCorp**, http://cuboidal.org/demos/foafcorp e http://www.rdfweb.org/foaf/corp/intro.html

[56] **RDF Crawler**, http://sw.deri.org/svn/sw/2005/06/crawler

[57] **RDF Crawler jar file**, http://sw.deri.org/2005/06/crawler/dist/rdfcrawler.jar

[58] **SECO System**, http://seco.semanticweb.org

[59] **Institute AIFB**, http://ontobroker.semanticweb.org/rdfcrawl

[60] **DERI**, http://www.deri.org

[61] **FOAF-Realm**, http://www.foafrealm.org

[62] **FOAFVocab**, http://rdfweb.org/topic/FoafVocab

[63] **Semantic Campus**, http://www.semanticcampus.org

Fonti:

- **Wikipedia**, http://www.wikipedia.org
- **OpenP2P**, http://www.openp2p.com
- **P2P Italia**, http://www.p2pitalia.com
- **P2P Sicuro**, http://www.p2psicuro.it
- **DataBase Group**, http://www.dbgroup.unimo.it
- **M. Marongiu: *"Peer-to-peer: una vecchia novità"*, Login Internet Export n.28, 2001**, http://online.infomedia.it/riviste/login/28/pdf/articolo05.pdf
- **C. Zunino, A. Sanna: *"A JXTA-based architecture for 3D distributed visualization: D3D"***, 2004, http://sanna.polito.it/Versioni_Postscript/CCCT_2004.pdf
- **T. Buhnik: *"Analisi topologica delle reti peer-to-peer basate sul protocollo Gnutella"***, 2002, http://www.dis.uniroma1.it/~laura/didattica/tesi/supptesi/buhnik.pdf
- **M. Morello: *"Problematiche di sicurezza nelle applicazioni peer-to-peer: un approccio all'anonimato"***, 2003, http://web.cefriel.it/~cerri/doc/tesi_venturi-gdoi.pdf.gz
- **D. Cerri: *"Sicurezza e peer-to-peer: tra anonimato e fiducia"*** , 2003, http://web.cefriel.it/~cerri/doc/p2p_sett03.pdf
- **P. Tagliati, T. Pesante: *"Security in peer-to-peer system"***, 2005, http://xoomer.virgilio.it/pesante/download/relPesTag.pdf
- **D. Brickley, L. Miller: *"FOAF Vocabulary Specification"* Technical report, RDFWeb FOAF Project**, 2005, http://xmlns.com/foaf/0.1
- **L. Dodds: *"An Introduction to FOAF"***, 2004, http://www.xml.com/pub/a/2004/02/04/foaf.html

- **E. Dumbill:** ***"Finding friends with XML and RDF"***, 2001, http://www-128.ibm.com/developerworks/xml/library/x-foaf.html
- **J. Mori, Y. Matsuo, M. Ishizuka, B. Faltings:** ***"Keyword Extraction from the Web for FOAF Metadata"***, 2001, http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/keyword_extraction_from_the_web
- **J. Smarr:** ***"Technical and Privacy Challenges for Integrating FOAF into Existing Applications"***, 2001, http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/technical_and_privacy_challenges
- **A. Harth:** ***"An Integration Site for Semantic Web Metadata"***, 2003, http://www.websemanticsjournal.org/ps/pub/2004-12
- **A. Harth:** ***"SECO: Mediation Services for Semantic Web Data"***, 2004, http://ieeexplore.ieee.org/iel5/9670/29151/01315543.pdf?arnumber=1315543
- **B. Nowack:** ***"Semantic Campus - A FOAF Extension"***, 2004, http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/pp/semantic_campus
- **S. R. Kruk:** ***"FOAF-Realm - control your friends' access to resources"***, 2004, http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/foaf_realm
- **L. Martino:** ***"Annotazione Semantica di Portali Web"***, 2003, http://www.dbgroup.unimo.it/tesi/martino.pdf
- **R. Govoni:** ***"Progetto e sviluppo di procedure per il trasferimento selettivo di dati tra database in formato XML"***, 2003, http://www.dbgroup.unimo.it/tesi/govoni.pdf
- **M. Bonacorsi:** ***"Realizzazione di una interfaccia Web per la progettazione di uno schema ER e la sua traduzione in RDF"***, 2004, http://www.dbgroup.unimo.it/tesi/bonacorsi.pdf
- **A. Galavotti:** ***"Gestione dei database in architetture peer-to-peer"***, 2004, http://www.dbgroup.unimo.it/tesi/galavotti.pdf
- **M. G. Caruso:** ***"Analisi e Confronto di Architetture Peer to Peer per la Gestione di Database Distribuiti"***, 2005