Peer to Peer: una vecchia novità

Parlare oggi di Peer to Peer (o P2P in breve) fa correre alla memoria un solo nome: Napster. Perché, bisogna ammetterlo, se oggi si parla così tanto di peer to peer è perché Napster ne è l'implementazione più popolare. Esamineremo i concetti alla base di questa tecnologia, le principali differenze dall'architettura client/server e i possibili sviluppi futuri.

di Marco Marongiu

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 WWW).

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 server sono decisamente più sfumati, se non completamente assenti. Ma è solo nell'ultimo decennio che si è assistito all'esplosione di questo tipo di servizi.

Concetti generali

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;
- tutti possono essere contemporaneamente client e 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.

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

Il talk

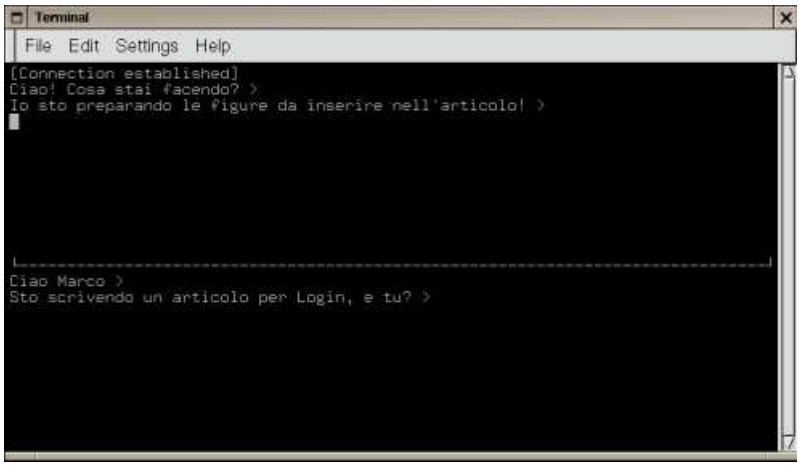
Il talk è una delle tante utility che UNIX deve a BSD. La prima versione di talk fece infatti la sua comparsa nello UNIX 4.2BSD sviluppato dall'Università della California di Berkeley (UCB), pubblicato nel 1983.
Per inciso, in 4.2BSD esordivano quelle che oggi sono pietre miliari del networking: fu infatti 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 voglio parlare con un utente connesso ad un altro host, eseguo 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 (**Figura 1**). Gli utenti possono scrivere contemporaneamente, il contenuto del terminale è aggiornato in sincrono e la comunicazione viene chiusa quando uno dei due utenti interrompe il programma con la pressione di CTRL-C (o della sequenza di interruzione definita sul sistema).
Talk è un programma forse un po' arcaico, ma

FIGURA 1

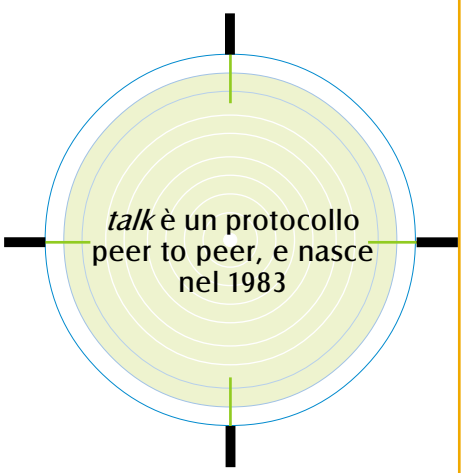
Una sessione di talk



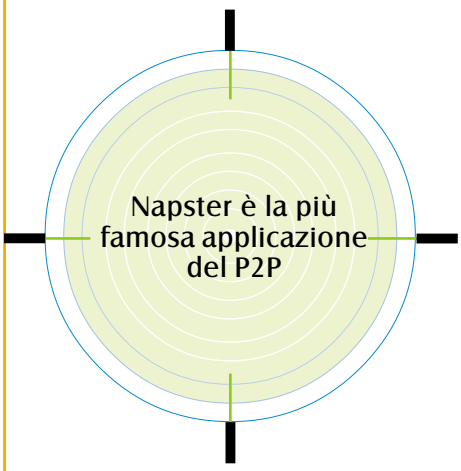
ci dà l'idea di un protocollo in cui, una volta che la connessione è stabilita, nessuno dei partecipanti può dirsi server o client: i due host sono gerarchicamente alla pari.

Le reti Microsoft

Le reti Microsoft, essendo largamente diffuse e utilizzate, per i lettori sono forse l'esempio più immediato di una architettura in cui i servizi vengono erogati in peer to peer.



Ogni host facente parte di una rete Microsoft ha la possibilità di condividere delle risorse, dove con risorsa si intende un qualsiasi device fisico connesso al computer (una stampante, uno scanner, un'unità CDROM o DVD, un disco...) o una risorsa "logica" (una partizione di un disco, una cartella...), e di utilizzare le risorse che gli altri mettono a disposizione. In un ufficio ove siano presenti alcuni sistemi Windows in rete 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.



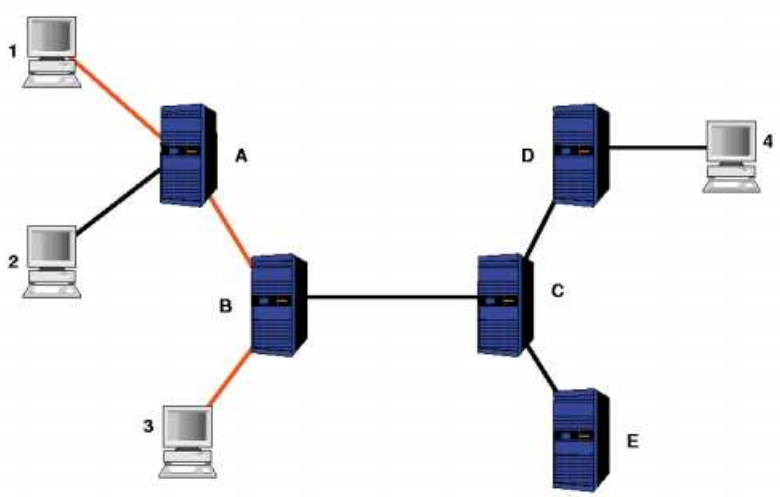
Si noti come questo caso sia nettamente diverso dal precedente. Mentre nel caso del talk le definizioni di client e server non hanno nessun senso, qui un host può essere, anche contemporaneamente, client quando utilizza risorse condivise e server quando le mette a disposizione.

Gnutella

Un altro caso di protocollo peer to peer, uno dei più famosi, dove ciascun nodo può funzionare contemporaneamente da client e server è Gnutella; la sua sto-

FIGURA 2

Una rete IRC. Si osservi che la comunicazione fra i client 1 e 3 è mediata dai server A e B.

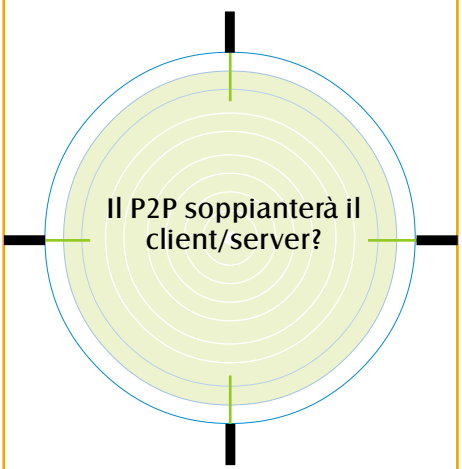


RIQUADRO 1

IRC, acronimo di Internet Relay Chat, è un sistema di teleconferenza basata su testo –ciò che tutti conosciamo appunto come "chat". È un protocollo maturo: la prima edizione delle sue specifiche è stata pubblicata come RFC 1459 nel 1993 [1], ma il suo sviluppo era in corso da ben quattro anni. IRC ha quindi più di dieci anni, e il suo sviluppo non si sia mai fermato; infatti nell'Aprile del 2000 sono stati pubblicati tre nuovi RFC [2], [3], [4] che aggiornano le specifiche del RFC 1459.
Una rete IRC è formata da una serie di server collegati fra loro in catena. I server si sincronizzano continuamente fra loro scambiandosi informazioni di stato (client e utenti collegati, canali attivati, proprietà degli utenti e dei canali…), in modo che due client ovunque connessi "vedano" sempre lo stesso stato della rete (ad esempio gli stessi utenti su un determinato canale, gli stessi operatori…).
IRC è un protocollo pesantemente client/server; infatti ogni comunicazione fra i client, sia essa uno a molti (per esempio i canali) sia uno a uno (da utente a utente, per esempio), è mediata (in inglese: *relayed*, da cui appunto *Relay Chat*) dal server, come è illustrato in Figura 2. I server quindi non si limitano a fornire una interfaccia univoca verso la rete di cui fanno parte, ma sono parte attiva nello scambio di informazioni fra i client.

Il protocollo IRC

ria, così come quella di altri programmi simili, è piuttosto travagliata. Gnutella fu rilasciato al pubblico su Slashdot **[13]** il 14 Marzo 2000 da Justin Frankel e Tom Pepper, i fondatori della Nullsoft (avete presente WinAMP?). Nello stesso giorno la pagina venne oscurata ma ormai troppe copie del programma erano state scaricate, e anche se il progetto della Nullsoft si interruppe qui (almeno ufficialmente), una comunità di programmatori prosegue oggi lo sviluppo del protocollo e di vari programmi client.
Gnutella fa parte di quella categoria di programmi che vengono classificati come "*sistemi distribuiti per la condivisione di file*" (Distributed systems for file sharing). 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 (si veda l'articolo di Mauro Iannucci su questo stesso speciale per maggiori informazioni).



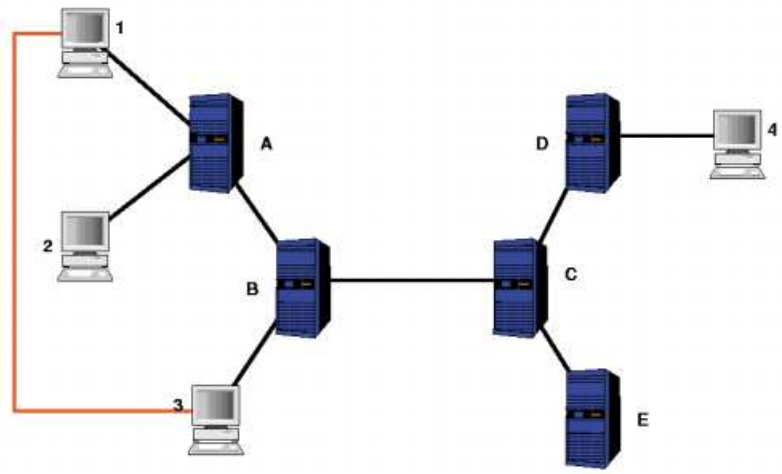
Se vi state chiedendo a questo punto come faccia un nodo isolato a prendere contatto con una rete esistente senza conoscere l'indirizzo di almeno un altro gnode la risposta è… non può – e in questo è del tutto simile al talk. Per ovviare a questo problema esistono dei *cache server*, che hanno il solo compito di tenere contatti con il più alto numero di client possibile e propagare le informazioni che raccolgono; alcuni indirizzi di cache server li potete trovare a partire da **[14]**.

Napster

Parlare di Napster come protocollo non è semplice. La Napster, Inc. non ha, almeno finora, pubblicato delle specifiche, e tutto quello che sappiamo è dovuto al lavoro di reverse engineering compiuto da hacker e appassionati. Alcune informazioni si trovano su SourceForge **[7]**, dove un gran numero di client e server pseudo-Napster viene sviluppato in Open Source per la maggior parte dei sistemi operativi. Si tenga quindi presente che quanto scriviamo di seguito non origina dalla lettura di specifiche pubbliche, ma da documenti che potrebbero essere incompleti o inesatti.
Napster appare strutturalmente molto simile a IRC, di cui una breve descrizione è data nel **Riquadro 1**, 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 giun-

FIGURA 3

La stessa situazione della Figura 2 in una rete Peer to Peer: i client, ottenute le informazioni necessarie, si connettono autonomamente e comunicano direttamente.



ge a termine per qualunque motivo. Si noti dunque che un server Napster non partecipa allo scambio dei file, che avviene direttamente fra i client.

Peer to peer e protocolli tradizionali: non tutte rose e fiori

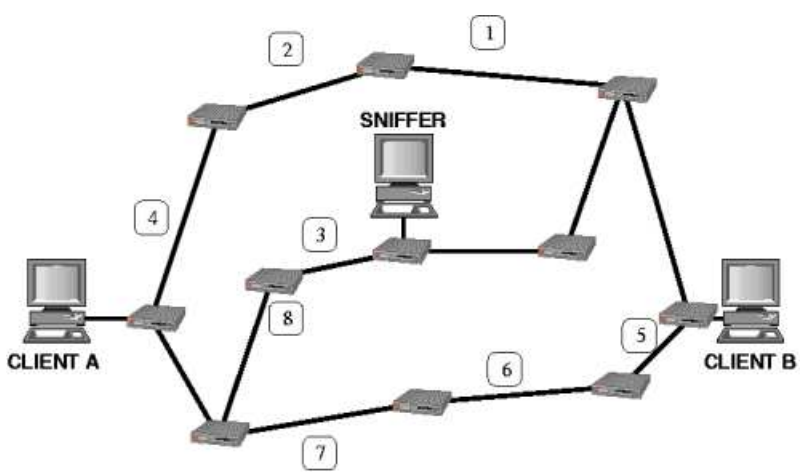
Viene da chiedersi se a questo punto non sia conveniente reimplementare alcuni dei tradizionali protocolli client/server in peer to peer. La risposta non è così scontata, per almeno due motivi. In primo luogo è estremamente difficile riuscire a fare a meno di un server centrale che tenga traccia dei nodi che compongono la rete; si può effettivamente fare a meno di un server quando la rete è basata sul concetto di comunità (un certo numero di persone che, di comune accordo, compongono la "loro" rete peer to peer) o quando per l'individuazione dei client sia sufficiente una richiesta di tipo broadcast. In secondo luogo esistono dei protocolli client/server per i quali l'introduzione del peer to peer non porterebbe vantaggi considerevoli. Consideriamo ad esempio IRC: i problemi congeniti al protocollo che troverebbero soluzione sarebbero ben pochi. Per la sua stessa struttura una rete IRC è tutt'altro che scalabile, e la probabilità che essa entri in crisi cresce molto rapidamente in dipendenza del traffico generato dalla rete stessa. Il traffico, a sua volta, è funzione del numero dei client e dei server presenti, ed una parte cospicua del carico è prodotta non tanto dalla comunicazione fra i client quanto dalla sincronizzazione fra i server. La semplice aggiunta di un server ad una rete IRC, o un server che va fuori linea per qualche minuto e rientra successivamente può bastare, in determinate condizioni, per mandare in crisi tutto il sistema. Una reimplementazione di IRC che comprenda la comunicazione diretta fra i singoli client in peer to peer (come illustrato in **Figura 3** e come avviene nel caso di Napster) alleggerirebbe la rete del carico generato da conversazioni e scambio di file uno a uno, ma non risolverebbe il problema della sincronizzazione dei server. Ci troviamo infatti in una situazione simmetrica rispetto a Napster, dove il carico generato dai client è assolutamente preponderante.

Quando il server non c'è i client ballano...

Sui dettagli dei protocolli p2p e sulle problematiche connesse alla loro implementazione leggerete negli altri articoli dello speciale (per gli sviluppatori segnalo in particolare l'articolo di Stefano Sanna sulla realizzazione di una chat multiutente in Java). Soffermiamoci invece su un aspetto collaterale a questi: il controllo delle informazioni. Come abbiamo visto nel caso di Napster, in un protocollo p2p i server (quando presenti) non intervengono nelle connessioni fra i client, limitandosi a fornire loro le informazioni che gli sono necessarie per instaurare una connessione. Come risultato, l'unica possibilità per controllare i dati scambiati fra due client e verificare, ad esempio, la loro legalità sarebbe quella di "sniffare" la connessione, cioè di esaminare tutti i pacchetti che i due host si scambiano. Ma per la struttura stessa del TCP/IP (IP in particolare) que-

FIGURA 4

Inutilità dello sniffing a grande distanza: sono rappresentati otto pacchetti che il client 1 manda al client 2. Solo i pacchetti 3 e 8 passano sul segmento di rete del host sniffer, rendendo probabilmente vano il suo lavoro.



sto è tanto più difficile quanto più è grande la distanza minima (in termini di hop) del controllore dai client. I pacchetti che compongono il flusso di dati potrebbero infatti prendere strade diverse, aggirando l'host sniffer (**Figura 4**), e le informazioni raccolte sarebbero così poche e frammentate che se ne potrebbe dedurre ben poco.

D'altra parte, se anche ci fosse scambio di dati illegali fra i client (materiale coperto da copyright, pornografia infantile, numeri di carte di credito, chiavi di accesso a sistemi protetti per citare gli esempi più noti) i gestori dei server, almeno in linea teorica, non sono colpevoli, così come i gestori dei vari motori di ricerca non sono responsabili delle informazioni che i loro spider indicizzano, o ancora come non sarebbe colpevole la casa produttrice di un coltello da cucina utilizzato per un omicidio.
Alla Napster, Inc., in buona o in mala fede, la pensavano esattamente come me, ma le grandi case discografiche li hanno citati lo stesso, e sappiamo bene come sono andate le cose. E non è ancora finita...

Il futuro del peer to peer

Il paradigma p2p appare oggi come un bellissimo giocattolo, almeno a guardare le applicazioni più diffuse. In realtà anche il mercato sta scoprendo nuove e potenti applicazioni di questa tecnologia. Negli ultimi anni la potenza di calcolo dei PC è cresciuta in maniera spaventosa, insieme alla dimensione delle memorie di massa; tutta questa potenza di calcolo rischia di rimanere inutilizzata o sottoutilizzata in una architettura client/server. Si ha quindi la necessità di decentrare maggiormente il carico, dai server verso i client, e il p2p sembra fatto apposta!
Molte grandi società dell'Information Technology hanno fiutato l'affare e gli investimenti cominciano a crescere anche in questa direzione. Insieme agli investimenti è cominciata la fase di proselitismo in cui tutte le aziende che si sono buttate a capofitto sul p2p cercano di "venderlo" come la più grande novità mai apparsa dai tempi di Mosaic o, perché no?, la panacea di tutti i mali che affliggono Internet. Due esempi emblematici sono l'articolo **[8]** che, pur offrendo ottimi spunti e interessanti osservazioni sui possibili sviluppi del P2P, appare decisamente "partigiano", e l'applicazione Groove della Groove Networks che attraverso il sito Web dedicato **[9]** e i vari white paper pubblicati **[10]**, **[11]** indica il suo prodotto come il profeta di una grande rivoluzione che cambierà il nostro modo di vedere ed usare Internet.
Al di là dei toni trionfalistici e di grandi rivoluzioni annunciate, è chiaro e innegabile che il p2p ha finalmente trovato la sua collocazione, che molte idee e interessi si muoveranno in questa direzione e non tarderanno a produrre buoni frutti.
Cominciano ad essere indette le prime grandi conferenze sull'argomento: la Intel ha dato vita al "Peer to peer Working Group"**[5]** ed organizza delle sessioni dedicate all'interno delle sue "Intel Developer Forum Conference"; la O'Reilly & Associates ha inserito una "Peer to Peer Conference" nel programma della sua Open Source Software Convention 2001 e pubblicherà quest'anno il suo primo titolo sull'argomento. Insomma: c'è un grande fermento che non mancherà di produrre qualcosa. Il primo sforzo sarà quello di ricercare uno standard unico e una comune piattaforma di sviluppo. Ed è uno sforzo destinato a fallire, visti gli interessi in gioco e i nomi dei concorrenti. Ma, come ogni volta, alla fine sopravvivranno quei due o tre standard che permetteranno dei buoni ritmi di sviluppo e di evoluzione. La corsa è cominciata, e c'è da scommettere che per imporre sul mercato il "loro" concetto di peer to peer i concorrenti lotteranno senza esclusione di colpi (bassi).

Bibliografia

[1] J.Oikarinen, D.Reed: Internet Relay Chat Protocol; RFC 1459, May 1993;
[2] C.Kalt, Internet Relay Chat: Architecture, RFC 2810, April 2000;
[3] C.Kalt, Internet Relay Chat: Channel Management, RFC 2811, April 2000;
[4] C.Kalt, Internet Relay Chat: Client Protocol, RFC 2812, April 2000;
[5] http://www.peer-to-peerwg.org
[6] http://conferences.oreilly.com
[7] http://sourceforge.net/
[8] D.Barkai, An introduction to peer to peer computing, in Intel Developer Update Magazine, Febbraio 2000; disponibile presso http://developer.intel.com
[9] http://www.groove.net/
[10] Groove Networks, Introduction to Groove; dal sito [9]
[11] Groove Networks, The Connection Age – How the Internet is transforming the world around us; dal sito [9]
[12] F.R.A.Bordignon, G.H.Tolosa; Gnutella: distributed system for information storage and searching, model description; su http://dss.clip2.com
[13] http://www.slashdot.com
[14] http://www.gnutella.co.uk

Marco Marongiu è Sistemista Programmatore presso il Centro di Ricerca CRS4 in Sardegna. È co-fondatore del Gruppo Utenti Linux di Cagliari (GULCh) e partecipa attivamente all'organizzazione dei seminari promossi dal gruppo. Si occupa sia della manutenzione dei sistemi, sia della realizzazione di applicazioni di utilità e supporto in linguaggio Perl. È contattabile all'indirizzo email marongiu@infomedia.it