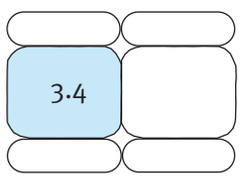
PEER-TO-PEER
OLTRE IL FILE-SHARING

Luca Caviglione
Mauro Migliardi
Roberto Podestà

Il paradigma peer-to-peer (p2p) è da sempre associato ai servizi orientati al *file-sharing*. Nel tempo, applicazioni sempre più complesse hanno sfruttato le teorie, le tecniche e gli strumenti sviluppati dalla comunità p2p, aprendo nuove opportunità commerciali e campi di sperimentazione. La telefonia su Internet, la diffusione di contenuti in *streaming*, le *Distributed Hash Table* e il *Grid Computing* su p2p ne sono alcuni esempi. Questo articolo descrive l'evoluzione del p2p oltre il *file-sharing*, evidenziandone quali siano le derivazioni più interessanti in campo industriale e scientifico.



1. INTRODUZIONE

Il *peer-to-peer* (p2p) è un paradigma per sistemi software distribuiti basato sull'assunzione che ciascun componente abbia la stessa "importanza". In pratica, ogni elemento che costituisce il sistema agisce ora da *client* e ora da *server* in una topologia che consente a tutti i partecipanti di comunicare direttamente su iniziativa bi-direzionale. Il termine tradotto letteralmente è auto-esplicativo e si riferisce, appunto, a un'interazione "da pari a pari". Concettualmente è quindi un paradigma profondamente diverso dal *client/server*, dove si ha un componente (il *server*) che esegue una certa operazione a seguito di una richiesta da parte di un altro componente di natura diversa (il *client*). Le due entità hanno ruoli precisi e invariabili: il *client* chiede, il *server* esegue. Nel p2p non esiste distinzione e ciascun componente è in grado di chiedere ed eseguire allo stesso tempo. Il concetto del p2p si diffonde quasi un decennio più tardi dall'avvento del Web, l'architettura *client/server* per eccellenza su Internet. L'architettura ideata da Tim Berners Lee ha portato a veicolare i contenuti, negli anni sempre più evoluti, esposti dai server Web verso i computer degli utenti di Internet che ne fanno richiesta. Le prime applicazioni p2p nascono con un obiettivo ancora più ambizioso: la condivisione di informazioni provenienti da tutti i computer connessi a Internet superando quindi il concetto di un insieme limitato e noto a priori di fornitori di informazione. Il concetto è evidentemente un passo avanti rispetto al Web e a tutti i livelli applicativi facenti più o meno riferimento al paradigma *client/server*. Fa ormai parte della storia il fatto che la spinta iniziale allo sviluppo di applicazioni p2p sia stata la necessità di vaste comunità di utenti di scambiare file senza dover passare per server centralizzati. Sebbene tale scambio potesse parzialmente avere lo scopo di aggirare le leggi sul *copyright*, la motivazione alla base dell'avvento dei primi sistemi p2p risiedeva in una carenza chiaramente riscontrabile in tutte le applicazioni esistenti su Internet intorno alla fine degli anni '90. Infatti, la pubblicazione[1] di contenuti era allora esclu-

[1] Nel contesto di questo articolo con il termine "pubblicazione" si intende l'atto di rendere fruibile su Internet una qualunque risorsa.

sivo appannaggio degli amministratori di *server* (Web, ftp ecc.) mentre era preclusa alle altre tipologie di utenti. Quale modo migliore, quindi, di superare tale carenza se non quello di rendere possibile la pubblicazione immediata di contenuti direttamente selezionati dal proprio *hard-disk*? Le prime applicazioni p2p essenzialmente consentivano questo e gli utenti che vi partecipavano avevano la possibilità di scambiare reciprocamente i propri file. La necessità di aggirare i menzionati problemi legali ha stimolato lo sviluppo di sistemi autonomi sempre più decentralizzati. Ulteriori fattori, come la diffusione di "barriere" tra le sottoreti di Internet come *firewall*, filtri sui *router* e le reti private nascoste dai dispositivi NAT (*Network Address Translation*), hanno portato a soluzioni tecniche sempre più innovative. Le sfide affrontate toccavano temi d'interesse per tutti coloro che operavano nei sistemi distribuiti e non lasciavano indifferenti la comunità scientifica e l'industria. Sia l'una che l'altra, infatti, dopo pochi anni, hanno iniziato a contribuire fortemente allo sviluppo di teorie e tecniche p2p. Si arriva quindi ad uno stadio di avanzamento del mondo p2p ove questo paradigma diventa uno strumento per fornire applicazioni e servizi che vanno oltre il *file-sharing*. Il presente articolo vuole illustrare l'evoluzione del paradigma p2p. Oltre a fornire un inquadramento storico e tecnico, vengono descritti alcuni dei segmenti applicativi dove il p2p ha creato successi industriali basati su nuovi modelli di business nella *e-economy*, per esempio il *media streaming* e la telefonia su IP, e in ambiti più vicini alla ricerca come le strutture dati distribuite e il *Grid Computing*.

2. P2P E *FILE-SHARING*: STORIA ED EVOLUZIONE

Come detto, per molti la nascita delle architetture p2p è temporalmente associata all'avvento dei servizi di *file-sharing*. Probabilmente, proprio per questo motivo, il termine p2p è considerato, quasi all'unanimità un sinonimo di "scambio/condivisione di file".
In realtà, il paradigma di comunicazione da pari a pari è sempre stato intimamente connesso con la struttura stessa di Internet. Infatti, il Dr. Vint Cerf, uno degli ideatori di Internet, ha scritto: "*People who think peer-to-peer is a new idea should understand that when we were designing the Internet, that was a critical and core part of the design. The TCP/IP protocols are precisely designed so that all elements in the Net are essentially of equal standing*" [1].
È però vero che questa tecnologia raggiunge la popolarità sostanzialmente con il rilascio dei primi programmi per la condivisione di file. Shawn Fanning nel Giugno del 1999, con l'invenzione di Napster, archetipo di tali programmi, ha consegnato "alle masse" tale paradigma di comunicazione. La sua vera intuizione è stata dunque quella di sfruttare le potenzialità di tale paradigma. Infatti, i trasferimenti dei file avvengono mediante un'interazione diretta tra i due utenti interessati (i *peer*) e non tramite un *server* dedicato; a supporto, Napster offre una *directory* centrale per la ricerca dei contenuti, e, di conseguenza, degli utenti che pubblicano tali contenuti. Questa peculiarità lo pone quindi nella categoria definita come "ibrida", ovvero, basata su una componente p2p e su una componente centralizzata. La presenza di un'entità centralizzata, se da un lato è stata una buona scelta ingegneristica, dall'altro ha decretato la fragilità legale del sistema. Infatti, l'utilizzo di una *directory* centrale ha consentito di ridurre la complessità del sistema a livelli facilmente gestibili risolvendo in modo tradizionale il problema delle ricerche dei file. Tuttavia, per lo stesso motivo, quando sono iniziati i guai legali per Napster, la sua scomparsa è stata altrettanto semplice e immediata: rendere inaccessibile la *directory* centrale rende totalmente inutilizzabile il servizio.
La storia di Napster ha un forte impatto sulla comunità di Internet e la porta a concentrare l'attenzione di una sua larga parte sul superamento del problema delle *directory* centralizzate al fine di un irrobustimento delle strutture p2p. Questa ricerca diventa un cavallo di battaglia di molte organizzazioni volte a difendere la libertà di espressione e assume quindi uno scopo culturale e scientifico.
Contemporaneamente, la comunità più *underground* (anche se, per esempio, il programma di *file-sharing* Gnutella e il relativo protocollo sono stati originariamente sviluppati da Justin Frankel per America On-Line nel Marzo 2000) ha cercato di rendere il più possi-

bile distribuito il servizio di *directory*, più per questioni legali e di responsabilità, piuttosto che per una mera ricerca di prestazioni. Avviene dunque in questo periodo e, principalmente, per questo scopo, la nascita di sistemi come il già citato Gnutella [2], WinMX e Kazaa. Queste applicazioni implementano, con tecniche simili, il concetto di un registro decentralizzato per la ricerca dei file. Anziché su un singolo *server* come in Napster, tale registro è ospitato da un insieme di *peer* speciali coordinati tra loro, i cosiddetti *super-peer*, in modo che il servizio risulti utilizzabile anche in caso di inacessibilità di uno di questi.
Parallelamente, la comunità scientifica comincia ad analizzare le proprietà di queste applicazioni [3], a caratterizzarne il traffico e a quantificarne l'impatto sulla rete. Già in questa prima fase di analisi, appare immediatamente ovvio che i servizi sviluppati con il paradigma p2p si qualificano per avere caratteristiche di scalabilità mai osservate prima.
Questi risultati iniettano nella comunità ICT la consapevolezza che sia possibile sfruttare queste metodologie per realizzare applicazioni commerciali e per creare nuovi modelli di business che superino la semplice produzione di sempre più ingegnosi sistemi per la condivisione di file. Tra i primi a compiere questo passo, portando il p2p oltre il *file-sharing*, ci sono Niklas Zennström e Janus Friis. Questi, ricchi dell'esperienza acquisita con lo sviluppo di Kazaa, sviluppano il primo esempio di successo di telefonia su Internet basata completamente su una rete p2p: Skype. L'esperienza accumulata durante lo sviluppo di Kazaa ha sicuramente rappresentato un vantaggio competitivo per gli autori di Skype. Infatti, l'utilizzo della popolare piattaforma di *file-sharing* ha consentito di testare, in maniera praticamente gratuita, l'infrastruttura p2p su una base d'utenza considerevole. Inoltre, l'innovazione confluita in Skype non è esclusivamente tecnologica, ma riguarda anche il modello di business. L'infrastruttura necessaria per l'erogazione del servizio è distribuita tra gli utenti e sfrutta così tutte le potenzialità del paradigma cooperativo. Gli utenti stessi divengono così una sorta di "*server farm*" gratuita, riducendo, quindi, la necessità di avere infrastrutture dedicate da parte del fornitore del servizio. Risulta dunque evidente come il p2p fornisca un'opportunità per ridurre i costi, in particolare nella fase di *start-up*, potendo così offrire servizi a prezzi molto contenuti, o addirittura gratuiti.
Proprio in quest'ottica viene sviluppato BitTorrent. Esso, infatti, ha come scopo principale l'implementazione di un servizio semplice ed efficiente per la distribuzione scalabile di contenuti, e pone invece in secondo piano la realizzazione di sofisticati meccanismi per l'elusione dei controlli legali.
Per questo motivo, si ritorna a un'architettura di tipo ibrido, composta da una componente centralizzata, denominata *tracker*, e da una parte prettamente p2p, composta dagli utenti, denominata *swarm*. L'utilizzo di un'entità ben definita per la gestione della rete virtuale che si sovrappone a Internet come insieme di connessioni tra utenti del sistema p2p (*overlay network*) consente quindi di semplificare sensibilmente il sistema e i requisiti delle interfacce *client* adottate. Il *tracker* è responsabile di garantire la creazione dello *swarm* permettendo agli utenti di trovarsi tra loro. Questi ultimi poi utilizzeranno euristiche distribuite basate sulla teoria dei giochi (citiamo, per esempio, il Tit-for-Tat [4]) al fine di massimizzare l'uso delle risorse e premiare gli utenti più virtuosi.

3. ARCHITETTURE, PROPRIETÀ E MOTIVAZIONI

Le architetture p2p disponibili sono molteplici, ognuna con le sue proprie peculiarità. Come si è già visto nella sezione precedente, molto spesso si parla di *overlay network* o, semplicemente, *overlay* come "sinonimo" di sistema p2p. In realtà questo termine indica la struttura della rete del sistema p2p che si sovrappone (*overlay* in inglese) alla rete reale cioè Internet. L'*overlay* è quindi l'elemento più caratterizzante di un'architettura p2p. La letteratura non offre una tassonomia univoca, ma la più diffusa è quella che le raggruppa in tre categorie principali [5]: **pure**, **ibride** e **strutturate**.
Le architetture **pure** sono caratterizzate dalla mancanza di una struttura ben definita dell'*overlay* realizzato dai *peer*. I meccanismi per la creazione di una rete di questo tipo sono semplici e sostanzialmente basati su uno scambio di messaggi di tipo ping-pong. La mancanza di struttura rende difficile la ricerca dei contenuti,

la quale viene sostanzialmente effettuata mediante algoritmi di *flooding* (inondazione), che prevedono la semplice propagazione di una richiesta attraverso tutti i nodi dell'*overlay*. In virtù delle funzionalità pressoché identiche dei *peer*, un'organizzazione di questo tipo è estremamente resistente ai malfunzionamenti (siano essi dovuti a problemi di connettività o a problemi interni ai *peer* stessi). Una variante di questa categoria di *overlay* è data dall'architettura p2p pura con supernodi. I supernodi sono *peer* con funzionalità aggiuntive specifiche destinate a garantire la presenza di un'organizzazione minimale alla rete e capaci quindi di facilitare le operazioni di ricerca e di instradamento delle informazioni. A differenza della tipologia precedente, le reti dotate di supernodi resistono bene ai malfunzionamenti casuali ma vengono messe in crisi da attacchi organizzati, quali, per esempio, quelli mirati a colpire i supernodi. Per evitare attacchi di questo tipo e dare maggior resilienza alla struttura rendendo più difficoltosa l'identificazione dei supernodi, la comunicazione fra i supernodi viene spesso cifrata. Un esempio di infrastruttura con comunicazione cifrata fra supernodi è dato da Kazaa.

Le architetture **ibride** sono invece composte dalla sovrapposizione di due paradigmi di comunicazione, ovvero quello *client/server* e quello p2p. Il primo viene per esempio impiegato per le operazioni di ricerca, rendendole così più veloci e semplici. Il secondo è impiegato per le operazioni di scambio dati, sfruttando così le risorse messe a disposizione dai nodi terminali e riducendo la presenza di eventuali colli di bottiglia. Sebbene questo tipo di architettura coniughi semplicità e prestazioni, lo svantaggio principale è dato dalla componente centralizzata che può comunque avere problemi di prestazioni, resistenza ai guasti e sicurezza.

Da ultimo troviamo le architetture **strutturate**. Come intuibile, i *peer* si organizzano nell'*overlay* seguendo uno schema topologico definito quale, ad esempio, un anello. Degli algoritmi di instradamento regolano la struttura dei nodi e conseguentemente l'inserzione e la ricerca dei dati all'interno dell'*overlay*. Tra queste architetture, una delle più diffuse e interessanti è quella delle *Distributed Hash Table* (DHT) [15] che consentono di indirizzare i *peer* nell'*overlay* mediante identificativi calcolati con opportune funzioni di hash. L'*overlay* permette di effettuare ricerche di tipo *chiave-valore* in maniera efficiente in modo del tutto simile a una normale *hash-table*. Questa maggiore efficienza delle *overlay* strutturate si paga però in termini di una maggiore complessità dei protocolli, di maggiori requisiti di memoria nei nodi (a causa delle tabelle di instradamento), e di una maggior banda richiesta per mantenerne consistente la struttura. Visti peraltro gli svariati campi di utilizzo delle DHT, il loro funzionamento verrà illustrato in dettaglio nel paragrafo successivo.

4. APPLICAZIONI

In questa sezione sono presentati alcuni esempi paradigmatici riguardanti l'utilizzo delle tecnologie p2p per applicazioni "oltre" il *file-sharing*.

4.1. p2p per lo *streaming*

La crescente disponibilità di banda nella rete di accesso impiegata dagli utenti *Small Office Home Office* (SOHO) ha favorito l'impiego delle architetture p2p anche per l'erogazione di servizi di streaming multimediale. A differenza di quanto necessario per lo scambio di file, la fruizione di contenuti multimediali on-line necessita di garanzie sul rispetto dei vincoli temporali nel trasporto dei dati e di capacità trasmissive non sempre facilmente ottenibili.

Ma è davvero necessario l'utilizzo del p2p per la distribuzione di contenuti multimediali?

Tecnologicamente parlando, la risposta è negativa. Infatti, il protocollo IP ha già al suo interno un'elegante soluzione tecnologica, denominata *multicast* [6]. Purtroppo però, tale meccanismo non è supportato da tutti i *router* di Internet: al contrario, anche per motivi di sicurezza, è scarsamente utilizzato e gran parte dei *router* limitano scientemente la propagazione dei pacchetti *multicast*. Per questo motivo, le tecnologie basate su *multicast* tendono ad "allontanarsi" dai livelli bassi dello *stack* ISO/OSI, fino a raggiungere il livello di applicazione. Non si parla più, quindi, di IP *multicast*, ma di *Application Layer Multicast* (ALM) [7]. In questo modo è possibile creare un *overlay* addizionale (infatti, tali meccanismi a volte sono anche denominati *overlay multicast*) per rimediare al mancato supporto di tali meccanismi a

livello di rete. In molte delle soluzioni disponibili, l'utilizzo delle tecniche di ALM è stato coadiuvato dall'inserimento in punti strategici della rete di opportuni *proxy* o *server* che aiutano nel superare ulteriori ostacoli posti dalla rete come *firewall* e reti private. Un'ulteriore evoluzione consiste nell'integrazione dell'ALM con infrastrutture dedicate alla distribuzione di contenuti, ad esempio le *Content Delivery Network* (CDN)[2]. Tuttavia, in tempi recenti, la già citata disponibilità di banda degli utenti, unita alla messa a punto di architetture p2p efficienti, ha spostato nei nodi utente l'implementazione dell'infrastruttura stessa. Proprio come un sistema di *file-sharing*, gli utenti ricevono e si scambiano porzioni dello *stream*, sfruttando così le risorse disponibili ai bordi della rete e creando una sorta di cache distribuita virtuale. Questo tipo di funzionamento si discosta anche da quello fornito da IP *multicast* in quanto, mentre nel caso di IP *multicast*, la trasmissione proviene ed è generata sempre da una sola sorgente centrale (e quindi poco resistente ai guasti), nel caso di scambio di porzioni di *stream* i dati sono replicati e trasmessi da molteplici sorgenti distribuite.
Per ottenere i risultati sopra descritti, occorre però apportare alcuni correttivi ai protocolli standard utilizzati per i servizi di *file-sharing* in quanto essi tipicamente non garantiscono alcun vincolo sulla tempistica di consegna dei dati. Esistono quindi meccanismi semplici ma capaci di servire solo una piccola popolazione di utenti (ne esistono, per esempio, alcuni basati sul BitTorrent, precedentemente descritto) dove lo *stream* è suddiviso in blocchi da 1 minuto e trattato come se fosse un file da trasferire. Questo meccanismo, però, è un adattamento di un'architettura nata per scopi diversi e presenta diversi limiti, per esempio nella garanzia di consegna *real-time*. Esistono invece soluzioni tecnologiche *ad-hoc* per lo streaming multimediale e architetture più complesse quali, per esempio, Zattoo, PPStream e GridMedia. Questa seconda tipologia di soluzioni si basa su una famiglia di protocolli denominata *pull-based streaming protocol* [8]. In questa modalità, i singoli *peer* selezionano in maniera indipendente altri *peer*, in modo da creare una comunità locale (si parla di *neighborhood*, vicinato) per lo scambio dello *stream*, organizzandosi così in opportuni *overlay* non strutturati. Tali meccanismi sono ancora piuttosto semplici [9], ma esistono metodologie ancora più raffinate: ovviamente il costo da pagare è un maggiore *overhead* nella trasmissione dei dati, a causa del passaggio delle necessarie informazioni di controllo, e una maggiore complessità dovuta alle peculiarità degli algoritmi. Per concludere la carrellata sulle applicazioni per il *media streaming*, si possono citare i meccanismi basati su *spanning tree* multipli [10], dove per ogni gruppo di flussi è realizzato un *overlay* ad albero ottimizzato in base ad opportune metriche.

4.2. Il p2p-SIP

Il *Session Initiation Protocol* (SIP) [11] è uno dei meccanismi di segnalazione più versatili e popolari disponibili su Internet. Grazie alla sua flessibilità, seppure sia stato originalmente progettato per il supporto di comunicazioni multimediali in generale e più nello specifico per la telefonia su Internet, si è guadagnato rapidamente lo status di protocollo di riferimento per l'attivazione di sessioni dedicate allo scambio di dati qualsiasi. Inoltre, nel 2000 è stato scelto dal *3rd Generation Partnership Project* (3GPP) come meccanismo di segnalazione a supporto del suo *IP Multimedia Subsystem* (IMS). Nel corso degli anni, SIP è stato anche utilizzato per applicazioni di *Instant Messaging* (IM) e location *awareness*. Focalizzando l'attenzione sul suo campo di utilizzo originale, va rilevato che SIP è alla base dei servizi *Voice over IP* (VoIP) più popolari, ed è supportato da molti apparati commerciali (si parla spesso di *SIP-phones*). Tradizionalmente, le infrastrutture per il VoIP basate su SIP, sono di tipo *client-server*. Vi sono uno (o più) elementi centralizzati, denominati, appunto SIP *server*, responsabili di gestire tutte le richieste inviate dai SIP *client*. Sebbene questo paradigma di funzionamento sia ampiamente collaudato e perfettamente funzionale, in tempi recenti l'*Internet Engineering Task Force* (IETF) ha creato un Gruppo di Lavoro denominato *peer-to-peer* SIP (P2PSIP). Per inciso, la comparsa di questo gruppo sembra quasi un tentativo da parte di IETF di non tra-

[2] Informalmente, con il termine CDN si definisce un tipo di rete attraverso la quale i nodi collaborano tra di loro in maniera trasparente per trasmettere dei contenuti verso una vasta popolazione di utenti.

scurare, come invece fece per i sistemi di *file-sharing* il fenomeno delle applicazioni VoIP basate su architetture p2p di cui l'esempio più noto e utilizzato è Skype. Va però detto che, oggi, il ramo orientato alla ricerca di IETF, cioè l'*Internet Research Task Force* (IRTF) ha comunque colmato la "storica" lacuna relativa all'uso di architetture p2p creando interessanti gruppi di lavoro. Si possono citare, ad esempio, il *Peer-to-Peer Research Group* (P2PRG) e, anche se non strettamente correlato con la tecnologia SIP, il gruppo *Application-Layer Traffic Optimization* (ALTO), nato da pochi anni nella stessa IETF, che studia tematiche inerenti il traffico generato dalle applicazioni p2p per il *file-sharing*.
Sebbene SIP nasca come un'architettura *client-server*, esso si è ulteriormente evoluto grazie a P2PSIP. Lo scopo fondamentale di P2PSIP è quello di dotare l'architettura funzionale SIP anche di un modello di interazione di tipo p2p. Parallelamente, esistono anche diversi studi mirati ad analizzare la possibilità di utilizzo del protocollo SIP stesso come meccanismo di segnalazione per la creazione di *overlay* [12].
In quest'ottica, uno dei maggiori risultati ottenuti dal P2PSIP, è stata la definizione del protocollo *REsource LOcation And Discovery* (RELOAD), il quale permette di creare un'infrastruttura p2p dotata di tutte le funzionalità necessarie all'erogazione dei servizi tipicamente implementati via SIP, quali per esempio, il supporto delle comunicazioni multimediali.

4.3. *Distributed Hash Table* (DHT)

Come accennato in precedenza, le DHT sono la più conosciuta realizzazione di *overlay* strutturata. Queste strutture dati possono essere anche considerate come un vero e proprio elemento nella tassonomia dei *database* distribuiti [13]. Una DHT offre fondamentalmente la funzionalità di una classica *hash-table* tramite una struttura dove, però, le coppie chiave-valore sono distribuite in modo trasparente sull'insieme di nodi che compongono l'*overlay*.
Questo tipo di struttura dati distribuita è stato concepito in ambito scientifico da differenti gruppi di ricerca intorno al 2001 [15, 26]. Tali gruppi, pur lavorando autonomamente, produssero risultati equivalenti che differivano solamente in alcune scelte d'implementazione dei rispettivi prototipi.
Una DHT è autonoma, dinamica e resistente ai guasti. Gli algoritmi che governano la struttura dati consentono l'associazione automatica di un nodo dell'*overlay* a una certa coppia chiave-valore, ossia dove la coppia viene effettivamente memorizzata. Allo stesso modo le richieste per la ricerca di una certa chiave vengono correttamente instradate verso il nodo che ne è responsabile. I nodi possono unirsi oppure lasciare la DHT in qualsiasi momento, e, in caso di caduta improvvisa di un nodo, il resto dell'infrastruttura è in grado di continuare a funzionare seppure, in mancanza di opportuni meccanismi di replicazione, vengano a mancare i dati immagazzinati nel nodo perduto.
Il cuore di una DHT è l'algoritmo di instradamento. Tale algoritmo, oltre a definire la topologia dell'*overlay* e a permettere di instradare correttamente le richieste, ha anche il compito di distribuire in modo uniforme le chiavi sui nodi. Utilizzando opportune funzioni di *hash* (e.g. SHA-1) si ottengono spazi di nomi molto grandi e uniformi. Chiavi e nodi sono mappati con la stessa funzione nel medesimo spazio. Questa affermazione, che in apparenza sembra generare confusione, è la caratteristica distintiva delle DHT e consente di basare l'instradamento sulle chiavi (infatti si parla di *key-based routing*). Le chiavi sono cioè trattate come punti su un anello circolare. Su tale anello, che corrisponde a tutto lo spazio raggiungibile tramite la funzione *hash*, gli stessi nodi dell'*overlay* non sono altro che delle chiavi, con la particolarità di rappresentare un punto "fisico" dell'*overlay* (e.g. per un nodo si usa come chiave il suo indirizzo IP). Le chiavi generiche, che, invece, non hanno tale corrispondenza fisica, sono assegnate a un punto "fisico" (o nodo) dell'anello secondo una regola predefinita. Nell'esempio di figura 1, una chiave appartiene al punto "fisico" successore che è il primo che incontra seguendo il senso orario. Quindi: se i_1 e i_2 sono due punti "fisici", i_1 viene prima di i_2 seguendo il senso orario e k_1, ..., k_n sono chiavi tra i_1 e i_2, allora k_1, ..., k_n appartengono al punto "fisico" i_2. Seguendo sempre figura 1, si vede che le chiavi comprese tra i punti "fisici" (nodi) *1* e *6* sono assegnate, cioè vi sono memorizzate, al nodo 6. Si noti che l'esempio, per ovvie esigenze di spazio e comprensibilità, riguarda uno spazio a 4 bit molto ridotto dove una distribuzione uniforme delle chiavi ai punti "fisici" non è chiaramente evidenziabile. I ca-

si reali, tuttavia, implicano spazi molto grandi (nelle implementazioni correnti si usano, per esempio, 160 bit).
L'instradamento dei messaggi è basato su tabelle che contengono i nodi adiacenti sull'anello e, a seconda, dell'algoritmo, uno o più nodi "lontani". Altre differenze tra le varie DHT emergono nelle tecniche per gestire l'ingresso e l'uscita di un nodo e la conseguente ridistribuzione automatica delle chiavi.
Alcune DHT e i relativi algoritmi sono stati implementati in librerie *open-source* che offrono interfacce di programmazione con le quali è possibile sviluppare applicazioni distribuite totalmente decentralizzate. Il riquadro approfondisce questo aspetto mostrando le principali risorse attualmente disponibili per i programmatori. Le DHT offrono un eccellente strumento per supportare applicazioni distribuite, scalabili e resistenti ai guasti. Per questo motivo sono state impiegate in moltissimi progetti con obiettivi diversi tra loro e ben oltre il p2p "tradizionale". Alcuni esempi sono i numerosi file system distribuiti basati su DHT oppure applicazioni per *digital library* [3].

4.4. Grid Computing

Come teorizzato nel 2003 da Ian Foster [18], la convergenza tra *Grid Computing* e p2p è ineluttabile. La stessa filosofia originaria del *Grid* sembra addirittura suggerire un'architettura p2p. Secondo gli obiettivi iniziali definiti più di dieci anni fa dallo stesso Foster e da Carl Kesselmann, la "*Grid*" avrebbe dovuto essere un'infrastruttura computazionale in grado di far leva sulla potenza di calcolo inutilizzata delle macchine genericamente connesse a Internet. Nell'accezione originaria, con la famosa analogia tra la rete elettrica e le risorse di calcolo, si avrebbe un'Organizzazione Virtuale (*Virtual Organization* - VO) che, attraverso il *middleware Grid*, fornirebbe una visione ad alto livello di un unico supercomputer virtuale composto in realtà da migliaia di computer connessi a Internet. L'utente, come per l'energia elettrica, non sa dove e come siano le risorse computazionali ma si limita ad utilizzarle in modo del tutto trasparente.

[3] Tra i numerosi esempi, citiamo il file system Magma (http://www.magmafs.net/) e OverCite (http://www.overcite.org/) come digital library.

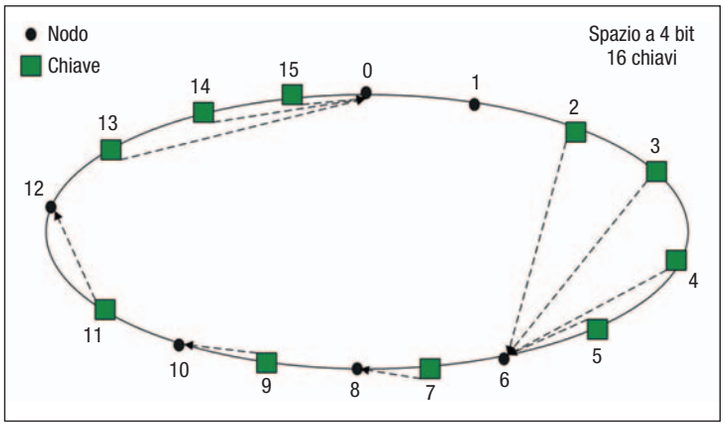


FIGURA 1
Assegnazione delle chiavi in una DHT

Le principali librerie per utilizzare gli *overlay* basati su DHT

Programmare con le DHT
Le API migliori attualmente disponibili per scrivere un'applicazione basata su una DHT sono basate sugli algoritmi Chord [15], Pastry [http://www.freepastry.org/] e Chimera [http://current.cs.ucsb.edu/projects/chimera/] pubblicati quasi contemporaneamente nel 2001. Lo sviluppatore ha a disposizione funzioni trasparenti per inserire coppie chiave-valore e trovare una chiave ottenendone il valore corrispondente. Dal punto di vista strutturale si hanno API per gestire l'entrata e l'uscita di nodo dall'"anello" e per il *monitoring* delle tabelle di *routing*.

Chord (http://pdos.csail.mit.edu/chord/): è probabilmente la DHT più nota ed è "storicamente" una delle librerie di riferimento, in quanto è stata la prima a raggiungere una buona stabilità. Frutto di un progetto del MIT, le API disponibili sono in C per Linux.

Pastry (http://www.freepastry.org/): è implementato in Java e fornisce, oltre alle funzionalità comuni alle altre DHT, delle API per la comunicazione in *multicast* tra i nodi. È il risultato di una cooperazione tra Rice University, Microsoft Research e Max Plank Institut.

Chimera (http://current.cs.ucsb.edu/projects/chimera/): sviluppata presso l'Università della California, è l'implementazione di un algoritmo inizialmente chiamato Tapestry sviluppato sempre all'interno dello stesso gruppo di ricerca. Svariate ottimizzazioni del codice, avvenute negli anni, ne fanno la scelta migliore dal punto di vista prestazionale. È scritto in C sia per Linux che per Windows.

In realtà l'evoluzione avuta dal *Grid Computing* ha però considerevolmente ristretto la tipologia di risorse facenti parte delle VO. Infatti, il principale filone di ricerca sul *Grid Computing* si è orientato verso l'aggregazione di centri di supercalcolo eventualmente appartenenti a domini amministrativi differenti. La motivazione è abbastanza ovvia se si considera che il *Grid* nasce dalla comunità dell'*High Performance Computing* (HPC) che opera su *cluster* computazionali che, aggregati, possono fornire enormi capacità di calcolo. Alcuni sotto-segmenti di ricerca in questo ambito hanno tuttavia mantenuto parte dell'idea originaria: un caso esem-

plare di questo sono le *Desktop Grid*. In questo caso, applicazioni costruite secondo il modello *master-worker* aggregano in una gerarchia ad albero migliaia di normali computer connessi a Internet per servire applicazioni *computation-intensive* che non hanno grossi requisiti di banda. Il famoso *seti@home* è storicamente il primo esempio di questo tipo di *Grid* seguito negli anni da ulteriori progetti di infrastrutture vere e proprie in grado di ospitare molteplici applicazioni come BOINC, XtreamWeb e il progetto europeo EDGeS. Il modello gerarchico delle *Desktop Grid*, sebbene aggreghi risorse che appartengono a normali utenti che volontariamente mettono a disposizione cicli di calcolo quando, per esempio, il computer è sotto-utilizzato, implica l'esistenza di nodi che agiscano come *server* per i vari *worker* o *sotto-server*. Questi nodi sono di solito dedicati e gestiti direttamente dai fornitori del software. Inoltre, l'utente non ha margini di libertà e risulta essere un passivo esecutore di *job* inviati dai *master*. L'introduzione di tecniche p2p mira invece a ottenere infrastrutture autonome che coinvolgono *peer* connessi ad Internet in grado di consentire a un utente Internet il lancio e la gestione della propria applicazione distribuita. La tipologia di risorse aggregate è la stessa delle *Desktop Grid* ma il modello di funzionamento è completamente diverso. Infatti, è orientato a fornire un servizio all'utente che a sua volta "condivide" la risorsa da dove opera. Queste infrastrutture di *Grid over p2p* sono solitamente identificate come *Virtual Cluster* e offrono le funzionalità di un *cluster* computazionale virtualizzando generiche risorse connesse a Internet. Questo segmento di ricerca affronta problematiche non banali molto spesso tralasciate dai segmenti principali, dove buona parte, se non tutta, l'infrastruttura di calcolo è dedicata e amministrata da una sola entità. Una *Grid over p2p*, per definizione, non può né essere dedicata né, soprattutto, essere soggetta a una gestione centralizzata. I suoi componenti devono essere autonomi e in grado di adattarsi alle svariate configurazioni di rete proprio come le applicazioni VoIP-SIP o il "vecchio" *file-sharing*.

Le *Grid over p2p* sfruttano il concetto di *overlay* già visto in altre applicazioni. Attraverso la virtualizzazione della rete fisica, cioè Internet, il *middleware* fornisce un ambiente per l'esecuzione di applicazioni distribuite, tendenzialmente parallele, senza che queste debbano essere ricompilate e senza richiedere interventi sull'amministrazione di rete. Il *parallel computing* diventa un servizio offerto tramite p2p come il *media-streaming* o la telefonia su IP. Questa idea è fondamentalmente trainata dalla diffusione delle tecnologie di accesso a Internet a banda-larga che consente velocità di connessione simili a quelle dei primi *cluster* Beowulf di metà anni novanta. Ovviamente le prestazioni ottenibili sono nettamente inferiori a quelle dei *cluster* attuali e delle Grid dedicate ma, in ogni caso, accettabili per diverse applicazioni e, soprattutto, provenienti da un'infrastruttura a costo nullo. I progetti che hanno maggiormente contribuito a formare l'idea del *Grid over p2p* sono: Harness [19], VioCluster [20], Wide-area Overlay of virtual Workstations (WOW) [21], Cluster-On-Demand [23], Private Virtual Cluster (PVC) [22] e ViNe [24].

Le problematiche chiave affrontate da questi progetti sono essenzialmente tre:

1. la re-direzione verso la rete di sovrapposizione delle applicazioni distribuite supportate;

2. il mantenimento di una visione omogenea delle risorse che appartengono all'infrastruttura di calcolo virtuale;

3. il superamento automatico degli ostacoli posti dalle varie sottoreti (e.g. *firewall* e NAT).

Con dei livelli software che forniscano un'efficace soluzione per queste tre problematiche, si può demandare a livelli superiori, non dipendenti dal *middleware* sottostante, la gestione di aspetti quali il tipo di applicazioni supportate, l'eterogeneità delle risorse e la sicurezza. Per quanto riguarda il livello applicativo, visto il forte interesse di quasi tutti i progetti verso il *parallel computing*, si presuppone che le macchine coinvolte siano equipaggiate con le necessarie librerie come MPI o PVM[4]. Anche la sicurezza è demandata al livello applicativo a cui spetta verificare se un certo *peer* sia affidabile o meno. Nella figura 2 è rappresentata un'architettura di riferimento a livelli alla quale si riconducono tutti i principali progetti. A parte il livello applicazione e il sottostante livello opzionale, i tre livelli marcati in verde rappresentano propriamente gli strati software che

[4] *Message Passing Interface* (MPI) e *Parallel Virtual Machine* (PVM) sono librerie per il calcolo parallelo disponibili in linguaggio C/C++ e Fortran.

devono essere implementati da un *middleware* per *Virtual Cluster*. La pila di tutti e cinque i livelli rappresenta un *peer* e la figura descrive il percorso virtuale (in nero) e reale (in rosso) della comunicazione tra due *peer* (A e B).
Dal punto di vista dell'applicazione (*Application*) si possono avere applicazioni parallele e, in generale, distribuite eseguite su uno strato di virtualizzazione del sistema operativo. Non sempre questa scelta è esplicitamente adottata ma, in generale, va ritenuta applicabile e preferita alle meno recenti *sand box* per garantire un adeguato livello d'isolamento dell'applicazione. In ogni caso, come detto, questo livello va considerato opzionale in quanto, per ragioni, per esempio, di prestazioni, può essere preferibile eseguire direttamente il codice binario. Il livello d'interposizione (*Interposition*) è nello *stack* architetturale il primo vero elemento proprio di un middleware *Virtual Cluster*. Inoltre, il suo modello di funzionamento non cambia se al livello superiore ci sia l'applicazione oppure una macchina virtuale. Questo strato si occupa di intercettare le chiamate alla rete dell'applicazione *target* e re-dirigerle verso la rete del *cluster* virtuale, ossia l'*overlay*. In tale rete vengono adottate svariate tipologie di indirizzamento per l'identificazione dei *peer*. Vengono utilizzati classi di indirizzi IP privati, altri indirizzi non pubblici[5], il MAC[6] oppure indirizzi all'interno di uno spazio di nomi specifico del *virtual cluster*. Al di là del sistema di identificazione delle risorse che dipende da come è organizzato il livello sottostante, cioè il *Virtual Network*, le tecniche utilizzate per re-dirigere un'applicazione dipendono fortemente dalle possibilità offerte dal sistema operativo per evitare cambiamenti e la ricompilazione dell'applicazione *target*. L'orientamento verso un sistema operativo *open-source* come Linux consente un margine di manovra decisamente superiore rispetto ai sistemi operativi proprietari. Le tecniche adottatate sono solitamente: il TUN/TAP, un *driver* di rete virtuale; il modulo Kernel Netfilter, che filtra le chiamate verso l'*overlay*; e, infine, intercettori che, con l'*overload* di alcune API del SO, sfruttano particolari indirizzamenti non riconosciuti dalle funzioni native (e.g. *namespace ad-hoc*).

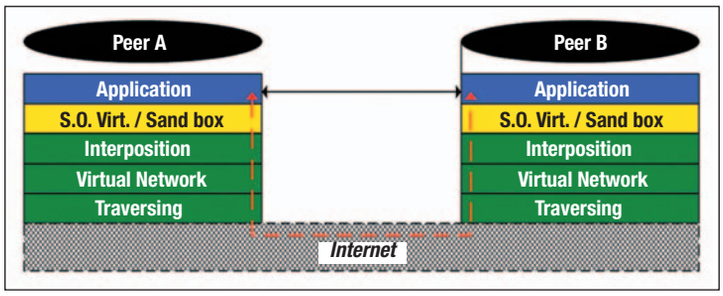


FIGURA 2
Gli strati software dei middleware per Virtual Cluster

Quando la comunicazione di un'applicazione viene intercettata, interviene il livello di virtualizzazione della rete (*Virtual Network*). Il compito fondamentale di questo livello è il mantenimento di una visione omogenea dell'insieme di risorse coinvolte. Questo obiettivo è raggiunto da componenti che offrono ai *peer* servizi di *brokering*. Tipicamente, bisogna tenere conto delle tipologie di entità previste da una particolare *Grid* che possono essere p2p pure oppure suddivise in gerarchie. Indipendentemente da questa organizzazione, i servizi che deve offrire questo livello per ciascuna entità sono: la registrazione, il *lookup*, il riconoscimento e la resistenza dei/ai guasti, la ricerca avanzata di un *peer* (ricerca, cioè, basata su caratteristiche generiche anziché sul suo identificativo come avviene invece per il semplice *lookup*), l'abilitazione alla connessione diretta e, infine, il mantenimento delle informazioni necessarie al *middleware* per la gestione del sistema. Va osservato che, se i primi quattro sono servizi trasversali per tutti i progetti, per gli ultimi tre vi è una forte dipendenza dallo specifico *middleware*. Il livello di virtualizzazione della rete è in pratica l'insieme di servizi offerti dall'*overlay* sulla quale vengono mappati i nodi appartenenti al *Virtual Cluster*. In particolare, un tentativo di standardizzazione [25] di questo livello sfrutta una DHT come registro per memorizzare i dati relativi ai nodi del *Virtual Cluster* e al contempo espone l'insieme dei servizi elencati con dei *plug-in* che implementano le interfacce richieste dai livelli di *Interposition* di *middleware* differenti.
Infine, dopo l'intercettazione e la virtualizzazione della rete, la comunicazione passa per il livello di attraversamento (*Traversing*) che si

[5] È il caso della classe sperimentale E nel range di indirizzi 240.0.0.1 - 255.255.255.254.

[6] Si intende l'identificativo assegnato a livello *Medium Access Control* (MAC).

rende necessario per il superamento delle barriere che possono essere poste dalle varie sottoreti, cioè dai *firewall* e dai meccanismi di NAT. Si tratta del livello che deve fatalmente interagire con le problematiche reali di Internet che, in fin dei conti, sono la motivazione di tutto lo *stack* architetturale. L'attraversamento degli "ostacoli" di Internet è una problematica tipica di ogni sistema p2p e buona parte delle tecniche usate nel *Grid over p2p*, derivano dai risultati ottenuti dal *file-sharing*. Come per il livello di *interposition*, si presentano soprattutto sfide di pura integrazione piuttosto che la ricerca di nuove soluzioni. Nei diversi progetti sono utilizzate tecniche quali il *tunneling* usato dalle VPN, semplici *proxy*, l'UPnP e l'*hole-punching*[7]. Altri progetti specializzano tecniche già note come il NAT traversal di WOW [21] e il Traversing TCP di PVC [22].
Come risulta evidente, il *Grid over p2p* rappresenta un ulteriore modello applicativo costruito per utilizzare nell'ambito del calcolo parallelo/distribuito i concetti sviluppati a partire dal *file-sharing*. Alcuni dei risultati ottenuti da questi progetti sperimentali sono, tra l'altro, stati riutilizzati in chiave commerciale e industriale nei *cluster* virtuali che fanno da *back-end* ai servizi offerti dai sistemi di *Cloud Computing*. Inoltre, sebbene ad oggi possa non essere molto evidente, il *Cloud Computing* stesso potrebbe offrire un modello di business per la diffusione su larga scala dei *Virtual Cluster.* Infatti, l'integrazione dei *broker* che operano sul livello di *Virtual Network* con le *Cloud* potrebbe far sì che le stesse *Cloud* reperiscano risorse esterne messe a disposizione dai *Virtual Cluster* per fronteggiare, per esempio, picchi di carico inattesi. Questo implica tuttavia un modello di business, non ancora sviluppato, per il quale un *Virtual Cluster* e i suoi *peer* siano effettivamente motivati a cooperare con le *Cloud*.

5. CONCLUSIONI

In questo articolo abbiamo presentato quella che definiamo un'evoluzione del p2p oltre il *file-sharing*. Questo tipo di applicazione è stato il principio che ha reso questo paradigma universalmente noto. Sebbene tuttora i due termini possano essere riportati quasi come sinonimi, il p2p è in realtà la base per un vasto insieme di applicazioni che superano con ogni evidenza l'iniziale obiettivo. Teoricamente supportato sin dagli albori di Internet, il vero sviluppo del p2p è stato opera di comunità di programmatori volontari svincolati da particolari interessi scientifici o economici prima ancora che dalla ricerca e dall'industria. Questa peculiarità è stata resa ancor più interessante con l'avvento di un modello di business totalmente nuovo che consente al gestore dell'applicazione p2p di sfruttare un'infrastruttura IT (composta dai *peer*) con costi estremamente ridotti. Il contributo della ricerca e dell'industria ha successivamente consacrato questo paradigma come base per applicazioni con potenziali economici enormi come la telefonia su Internet e il *media-streaming*. Anche nel calcolo distribuito si iniziano a intravedere nuovi interessanti scenari resi possibili dall'adozione del p2p. Tecniche come le DHT, inoltre, sono una base naturale per un insieme molto vasto di campi applicativi.
In futuro, il progressivo miglioramento della banda di accesso a Internet non farà altro che consentire una maggiore "distribuzione" delle applicazioni verso gli utenti finali consolidando quindi sempre di più l'adozione del paradigma p2p. Da ultimo, il paradigma p2p non è più solo relegato all'utilizzo dei nodi utenti come *peer*. Infatti, le moderne architetture per la distribuzione dei contenuti, sono direttamente mutuate dagli *overlay* creati mediante le tecnologie p2p. In questo caso però, i *peer* sono veri e propri nodi installati e mantenuti dal *provider* stesso. Esempi tipici sono quelli dello *streaming video*, che fanno ipotizzare che il p2p, oltre che il *file-sharing*, potrà fortemente contribuire al successo della IPTV e delle future applicazioni di *Video on Demand*.

[7] L'UPNP (*Universal Plug aNd Play*, http://www.upnp.org/) e l'*hole punching* sono tecniche che consentono l'attraversamento automatico di firewall e NAT.

Bibliografia

[1] Cerf V.G.: *Musing on the Internet*. Educause, Sept. - Oct. 2002, p. 74 - 84.

[2] Portmann M., Sookavatana P., Ardon S., Seneviratne A.: *The cost of peer discovery and searching in the Gnutella peer-to-peer file sharing protocol*. Proc. of the IEEE International Conference on Networks, 2001.

[3] Aa.vv.: Proc. of the first International Conference. In: *Peer-to-peer Computing*, Aug. 2001.

[4] Cohen B.: *Incentives build robustness in BitTorrent*. Proc. Workshop the Economics of Peer-to-Peer Systems, June 2003.

[5] Caviglione L.: *File-sharing Applications Engineering*. Networking Series, Nova Science Publisher, 2009.

[6] Deering S., Estrin D., Farinacci D., Jacobson V., Liu C., Wei L.: *The PIM Architecture for Wide-Area Multicast Routing*. IEEE/ACM Trans. Networking, Dic. 1997, p. 784 - 803.

[7] Jao L.,. Cui J.-H, Gerla M., Maggiorini D.: *A Comparative Study of Multicast Protocols: Top, Bottom or In the Middle*? Proc. of IEEE INFOCOM, March 2005.

[8] Zhang M., Zhang Q., Sun L., Yang S.: Understanding the Power of Pull-Based Streaming Protocols: Can We Do Better? *IEEE Journal on Selected Areas in Communications*, Vol. 25, n. 9, Dec. 2007, p. 1678 - 1694.

[9] Zhang X., Liu J., Li B., Yum T.-S.P.: *Coolstreaming/donet: A data-driven overlay network for efficient media streamin*g. Proc. of IEEE INFOCOM, March 2005.

[10] Li J., Chou P.A., Zhang C.: Mutualcast: *An Efficient Mechanism for one-to-many content distribution*. Proc. of ACM SIGCOMM Asia Workshop, Apr. 2005.

[11] Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M., Schooler E.: *Sip: Session Initiation Protocol*. RFC3261, IETF - RFC Editor.

[12] Zheng X., Oleshchuk V.: *Improving Chord Lookup Protocol for P2PSIP-Based Communication Systems*. Proc. of NISS09, June-July 2009.

[13] Bonifati A., Chrysanthis P., Ouksel A., Sattler K.: *Distributed databases and peer-to-peer databases: past and present*. Proc. of ACM SIGMOD, June 2008.

[14] Jennings C., Rescorla E., Baset S., Schulzrinne H.: *A SIP Usage for RELOAD, draft-ietf-p2psip-sip-03*. IETF, 2010.

[15] Dabek F., Brunskill E., Kaashoek M.F., Karger D., Morris R., Stoica I., Balakrishnan H.: *Building peer-to-peer systems with chord, a distributed lookup service*. Proc. of Workshop in Hot Topics in Operating Systems, May 2001.

[16] Yang X., de Veciana G.: *Service capacity of peer-to-peer networks*. Proc. of IEEE INFOCOM, March 2004.

[17] Ghemawat S., Gobioff H., Leung S.-T.: The Google File System. *ACM SIGOPS Operating Systems Review*, Vol. 37, n. 53, 2003, p. 29 - 4.

[18] Foster I., Iamnitchi A.: On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. *LNCS*, Vol. 2735, Springer, 2003.

[19] Migliardi M., Sunderam V., Geist A., Dongarra J.: Dynamic Reconfiguration and Virtual Machine Management in the Harness Metacomputing System. *LNCS*, Vol. 1505, Springer, 1998.

[20] Ruth P., McGachey P., Xu D.: VioCluster: *Virtualization for Dynamic Computational Domains*. Proc. of IEEE International Conference on Cluster Computing, Sept. 2005.

[21] Ganguly A., Wolinksky D.I., Boykin P.O., Figueiredo R.: *Improving Peer Connectivity in Wide-area Overlays of Virtual Workstations*. Proc. of HPDC, June 2008.

[22] Rezmerita A., Morlier T., Néri V., Cappello F.: Private Virtual Cluster: Infrastructure and Protocol for Instant Grids. *LNCS*, Vol. 4128, Springer, 2006.

[23] *Cluster On Demand* (COD): http://www.cs.duke.edu/nicl/cod/

[24] Tsugawa M., Fortes J.: *A Virtual Network (ViNe) Architecture for Grid Computing*. Proc. IPDPS, April 2006.

[25] Podestà R., Iniesta V., Rezmerita A., Cappello F.: An Information Brokering Service Provider for Virtual Clusters. *LNCS*, Vol. 5870, Springer, 2009.

[26] Rowstron A., Druschel P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *LNCS*, Vol. 2218, Springer, 2001.

Luca Caviglione è nato a Genova nel 1978. È ricercatore presso l'Istituto di Studi sui Sistemi Intelligenti per l'Automazione (ISSIA) del Consiglio Nazionale delle Ricerche. I suoi principali interessi di ricerca riguardano le architetture p2p, i sistemi wireless e l'analisi di traffico. È autore o co-autore di circa 60 lavori scientifici e del libro *File-sharing Applications Engineering*. Inoltre è co-autore di diversi brevetti internazionali sui sistemi p2p, ed è uno dei WG Leader della Task Force Italiana di IPv6.
E-mail: luca.caviglione@ge.issia.cnr.it

Mauro Migliardi è nato a Genova nel 1966. Dopo essere stato uno dei principali ricercatori del progetto HARNESS per il meta e Grid Computing presso la Emory University di Atlanta, è stato Ricercatore Universitario presso l'Università di Genova ed è ora Professore Associato presso l'Università di Padova. Mauro Migliardi ha pubblicato oltre ottanta articoli scientifici soggetti a peer-review e ha tra i suoi principali interessi didattici e di ricerca le tecnologie e le metodologie per la progettazione e lo sviluppo di sistemi software distribuiti complessi.
E-mail: mauro.migliardi@unipd.it

Roberto Podestà ha lavorato per diversi anni come assegnista di ricerca presso l'Università di Genova dove, nel 2007, ha conseguito il Dottorato in Scienze e Tecnologie dell'Informazione. Nello stesso anno si è trasferito in Francia all'INRIA (Institut National de Recherche en Informatique et en Automatique) nella sezione di Parigi-sud. La sua attività di ricerca si è sempre focalizzata su middleware e integrazione di sistemi distribuiti e, in particolare, su Grid, p2p e, più recentemente, Cloud Computing. Dalla fine del 2009 è rientrato in Italia e opera come consulente in ambito IT.
E-mail: ropode@gmail.com