# Sistema P2P persistente e bilanciato

*Emanuele Spinella - 0000171281*
*Reti di Calcolatori LS - Prof. A.Corradi - A.A. 2003-2004*

26 settembre 2004

**Abstract**

*Il progetto consiste nella realizzazione di un sistema P2P che possa garantire persistenza e bilanciamento nel carico associato ad ogni peer e ogni server. Il contesto applicativo in cui collocare tale sistema deve poter valorizzare le caratteristiche di sicurezza e ottimizzazione, che rappresentano la peculiaritá distintiva rispetto a tutti gli altri software appartenenti alla stessa classe. Si pensi, ad esempio, ad una rete di terminali che colleghi tutte le stazioni di polizia, le associazioni investigative e le organizzazioni per la sicurezza presenti nel mondo: per mezzo di un sistema P2P sarebbe possibile condividere facilmente documenti riservati, rapporti, files di vario genere e quant'altro. E' evidente che, data l'importanza delle informazioni scambiate, la persistenza dell' architettura dovrebbe essere garantita prima di tutto. Inoltre l'espansione geografica della rete, da cui deriva la necessitá di utilizzo di internet, dovrebbe comportare l'impiego di sofisticati sistemi crittografici di cui non si tratterá in questo articolo.*

## 1 Reti Peer To Peer

Una rete Peer To Peer (P2P) è un sistema distribuito che permette agli utenti che possiedono un certo tipo di applicazione di connettersi fra loro e accedere direttamente ai files l'uno dell'altro. La logica dell'architettura riprende ed espande il classico paradigma Client/Server, in cui peró ogni terminale puó fungere alternativamente o simultaneamente da client e da server, a seconda che stia effettuando rispettivamente il download e/o l'upload di un file. La struttura delle reti P2P puó seguire diversi approcci, raggruppabili in tre macro categorie: reti centralizzate, decentralizzate e ibride.

### 1.1 Reti P2P Centralizzate

Questa categoria di reti P2P, di cui faceva parte l'ormai defunto Napster, è caratterizzata da un unico sistema centrale che gestisce il traffico degli utenti. Tale sistema centrale è costituito da un insieme di server che operano in parallelo e mantengono delle directories in cui sono presenti le indicazioni relative ai files condivisi dagli utenti. Quando un client effettua la richiesta di un file, viene eseguito l'accesso al database del sistema centrale, il quale fornisce le informazioni necessarie per stabilire una connessione diretta fra richiedente e proprietario, per mezzo della quale avverrá il trasferimento.

E'importante sottolineare il fatto che i server del sistema centrale non contengono i dati da trasferire, la loro

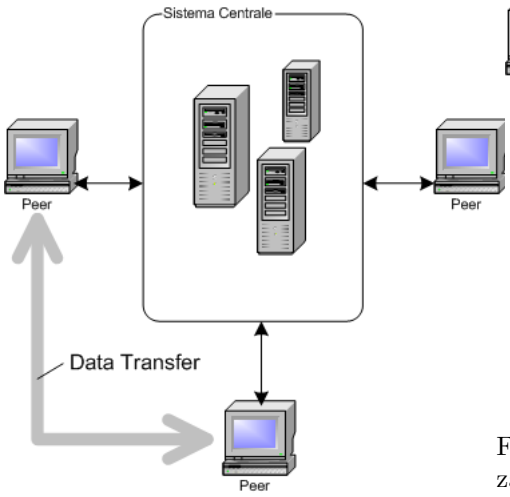


Figura 1: Sistema P2P centralizzato

unica funzione è quella di far comunicare i nodi e realizzare la ricerca dei files. I vantaggi di questo tipo di soluzione sono riconducilbili a quelli di un comune servizio di naming: la presenza di un indice centrale rende la ricerca dei files rapida ed efficiente. D'altra parte le directories di cui si avvalgono i server del sistema centrale potrebbero non avere una frequenza di aggiornamento sufficientemente elevata per evitare che vengano temporaneamente ignorate le risorse di utenti appena registrati, o che si continuino a rilevare quelle di utenti giá usciti dal sistema. Tuttavia la vera e propria debolezza che ha causato l'abbandono di questa architettura nei software attuali è determinata dalla presenza di un unico sistema centrale di server, che rappresenta il punto vitale della rete e dal cui corretto funzionamento dipende direttamente la disponibilitá del servizio.

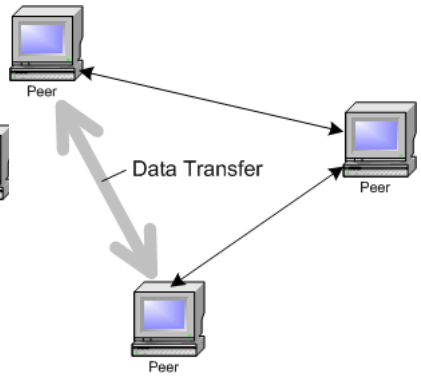


Figura 2: Sistema P2P decentralizzato

## 1.2 Reti P2P Decentralizzate

A differenza del caso precedente, in questo tipo di reti non esiste alcun sistema centrale che si occupa della ricerca dei files e delle comunicazioni fra i peers. Ogni nodo si fa carico di tutte le operazioni che, nel modello centralizzato, erano demandate al sistema centrale: ricerca del peer proprietario del file desiderato, richiesta di connessione ed eventuale trasferimento.

Una rete P2P cosí strutturata, di cui Gnutella rappresenta un tipico esemplare, non soffre dei problemi di persistenza di quella precedente e, al contrario, risulta particolarmete flessibile in quanto non esistono punti di rottura in grado di inibire il servizio. Il crash di un peer provoca, come unica conseguenza, l'indisponibilitá dei files da esso condivisi, ma non è pregiudicata in alcun modo l'attivitá degli altri. Inoltre, mentre le reti centralizzate prevedono solitamente la registrazione degli utenti all'interno del sistema centrale, questo secondo modello è caratterizzato da una partecipazione all'attivitá di sharing completamente anonima, con tutti i vantaggi che questo comporta, special-

mente sul fronte legale. Purtroppo proprio la mancanza di registrazione, e quindi di controllo sugli utenti della 'comunitá', fa sí che non sia sempre garantito un livello soddisfacente di QoS. Inoltre non è detto che tutta la rete sia accessibile a ciascun nodo, quindi potrebbero non essere soddisfatte richieste di files nonostante la loro presenza nella rete.

### 1.3 Reti P2P Ibride

L'ultima tipologia di reti P2P, il cui modello viene oggi adottato da quasi tutti i principali programmi, come ad esempio WinMX, Kazaa, eMule e molti altri, prende il nome di 'ibrida', in quanto costituisce il risultato dell'unione delle due precedenti soluzioni, proponendosi di cogliere e sfruttare i vantaggi derivanti da entrambe e di limitare i punti deboli. Il sistema prevede l'esistenza di numerosi server (o supernodi), presso ognuno dei quali sono registrati un certo numero di peers.

Quando un utente vuole ricercare un file, inoltra la richiesta al proprio supernodo il quale, a sua volta, la diffonde agli altri supernodi. Dal risultato di questa operazione si individuano uno o piú utenti che possiedono la risorsa desiderata, dopodiché si stabilisce una connessione diretta per mezzo della quale avviene il trasferimento.

I peers da cui un utente scarica i files possono essere registrati sia sul suo stesso server, ma anche su server differenti. Tale struttura conferisce al sistema un adeguato livello di decentralizzazione pur preservando, in buona parte, l'efficienza di ricerca derivante dalla presenza dei supernodi. E' evidente che, in ogni caso, i server rappresentano un punto di rottura pericoloso, poiché il crash di uno di essi comporta l'esclusione dalla rete di tutti gli utenti ad esso collegati.

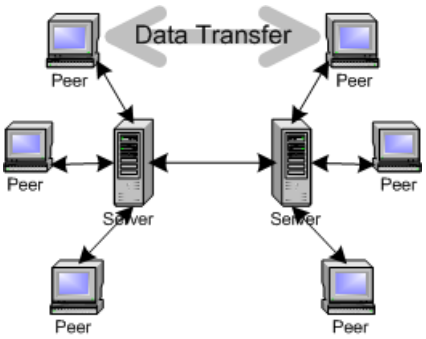


Figura 3: Sistema P2P ibrido

Rispetto al modello centralizzato, l'incidenza di un simile evento sulla QoS risulta essere tanto inferiore quanto piú numerosi sono i supernodi e, conseguentemente, quanto meno numerosi sono gli utenti connessi a ogni server (maggiore decentralizzazione, ma anche maggiori costi di gestione e maggiore difficoltá di realizzazione).

## 2 Architettura

Dall'analisi dei diversi modelli di reti P2P ci si è proposti di ottenere un sistema ulteriormente migliorato che possieda quelle caratteristiche di persistenza e QoS necessarie per un eventuale impiego in ambienti in cui la loro importanza risulta primaria. La struttura architetturale rispecchia fondamentalmente il modello ibrido, ed è quindi costituita da diversi server, presso ognuno dei quali sono registrati un certo numero di peers. Oltre ai due attori principali (server e peer), viene adottato un sistema di replicazione che introduce un terzo attore: la copia.

**LookUpServer**

Le funzioni che deve svolgere questa unitá sono quelle precedentemente descritte per i supernodi del sistema centrale: si tratta sostanzialmente di

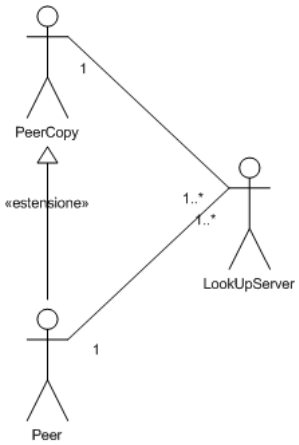


Figura 4: Attori

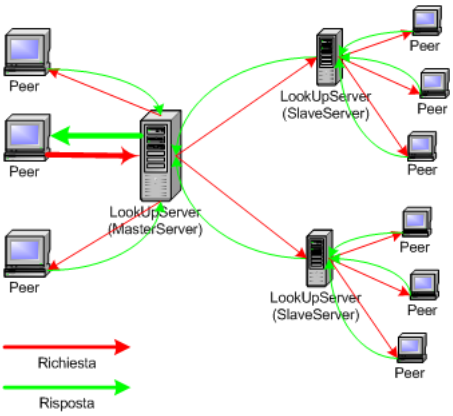


Figura 5: Ricerca di files

un servizio di naming che rende il processo di ricerca dei files efficiente e rapido. La novitá che si è voluta introdurre riguarda le modalitá con cui viene individuato l'utente proprietario di una certa risorsa. Nei modelli precedenti si faceva uso di directories residenti lato server, che potevano presentare problemi qualora la frequenza di aggiornamento non fosse sufficientemente elevata. In questo sistema, invece, si è pensato di gestire la ricerca nel seguente modo: il peer inoltra la richiesta al LookUpServer presso il quale è registrato (MasterServer) il quale, a sua volta, la inoltra sia agli altri LookUpServer (Slave Servers), sia agli altri peers ad esso collegati. Ogni SlaveServer deve quindi richiedere la risorsa a tutti i propri peers e rispondere al MasterServer il quale, dopo aver unito tutti i risultati ricevuti, invierá al peer richiedente l'esito finale della ricerca.

Questa soluzione rende il sistema costantemente aggiornato sulle risorse disponibili, in quanto ogni ricerca coinvolge i soli peers effettivamente connessi alla rete. Ció di cui invece deve tenere traccia ogni LookUpServer è la lista dei peers ad esso connessi: nella fase di ingresso nella rete ogni nodo deve registrarsi presso uno dei servers disponibili. Le modalitá di scelta automatica del LookUpServer verranno trattate nel dettaglio in seguito. Anche la copia deve eseguire la registrazione ma, per non aumentare notevolmente e inutilmente la complessitá del sistema, essa viene associata allo stesso server cui è collegato il peer originale.

**Peer**

Ogni utente connesso alla rete rappresenta un nodo che puó effettuare richieste o riceverne. Il procedimento di richiesta è giá stato analizzato nella sezione precedente dedicata al LookUpServer, ma è necessario considerare separatamente il punto di vista del peer. La richiesta dell'utente non è necessariamente costituita dal nome preciso di un file: spesso viene semplicemente digitata una stringa che rappresenta una sorta di radice, di concetto base, e il sistema deve essere sufficientemente elastico da individuare tutti i documenti che contengono tale stringa. Per questo motivo, quando un peer riceve una richiesta di ricerca da un LookUpServer, potrebbe rilevare nella propria di-

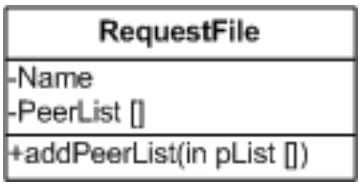


Figura 6: Oggetto RequestFile

rectory di sharing i nomi di piú files che fanno match con la query. Inoltre è possibile che differenti utenti possiedano lo stesso file, ed è quindi compito di ogni LookUpServer elaborare i risultati delle ricerche ricevuti raggruppando le occorrenze uguali. A tal proposito si è pensato di introdurre un oggetto RequestFile, che rappresenta una generica risorsa caratterizzata da un nome e dalla lista dei peers che la possiedono.

Ogni LookUpServer riceverá quindi un certo numero di RequestFile e dovrá unire quelli omonimi in un unico oggetto aggregando le liste dei peers proprietari. L'utente che ha inizialmente fatto la richiesta, alla fine del procedimento, potrá scegliere fra un insieme di files diversi senza sapere a chi appartengono. Il download di una risorsa appartenente a piú nodi viene gestita dal sistema scegliendo il peer 'migliore' da cui scaricare, in modo da bilanciare il piú possibile il carico sulla rete (vedi dopo).

**PeerCopy**

Ogni Peer è dotato di una copia, fisicamente rappresentata da un diverso terminale, che possiede gli stessi files condivisi dall'originale. Il modello di replicazione utilizzato è quello a copie calde e passive. Calde in quanto vengono create insieme all'originale e si mantengono aggiornate sullo stato del sistema; passive perché non eseguono fino al crash del peer associato. Le modalitá di aggiornamento e di rimpiazzo a seguito di un peer failure verranno trattate piú approfonditamente nelle sezioni successive. In figura 4 si nota come il peer estenda il concetto di copia: dal punto di vista logico, infatti, la copia è un peer a tutti gli effetti, a parte il fatto che non puó inoltrare richieste, ma solo riceverne ed eventualmente eseguire l'upload dei files.

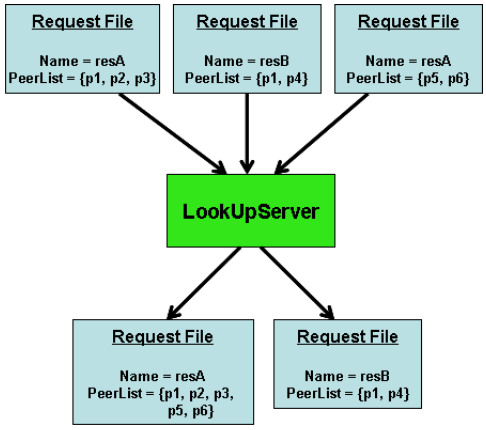


Figura 7: Filtraggio dei RequestFile

# 3 Affidabilitá e QoS

L'obiettivo principale che si è cercato di raggiungere con questo progetto consiste in un elevato livello di affidabilitá e QoS. Inutile specificare che la prima implica parzialmente la seconda, ma in questa trattazione si preferisce considerare separatamente i due casi, associando al concetto di affidabilitá quello di persistenza e al concetto di QoS tutte le restanti caratteristiche che rendono un sistema qualitativamente migliore.

## 3.1 Persistenza

Pensando agli ambienti applicativi in cui potrebbe essere impiegato il sistema, contesti in cui viene richiesta

un'elevata disponibilitá di servizio, si è ritenuto che la persistenza a fronte di crash improvvisi dei nodi connessi alla rete fosse la caratteristica di maggiore rilievo, cui dedicare gran parte degli sforzi. Tutto si basa sostanzialmente su un sistema di heartbeats fra le entitá coinvolte, mediante il quale è possibile rilevare lo stato di attivitá (dead or alive) dei nodi e dei servers. Per comprendere come tale sistema di heartbeats debba essere organizzato si considerino innanzitutto i possibili punti di rottura della rete P2P: il Peer e il LookUpServer.

**Crash di un LookUpServer**

Se si verifica il crash di un LookUp Server, la diretta conseguenza consiste nell'isolamento dalla rete di tutti i nodi registrati presso di esso. L' azione che deve essere intrapresa, poiché non è possibile adottare misure preventive, è quella di registrare i peers rimasti isolati presso un altro server. Ognuno di essi deve perció eseguire un nuovo processo di registrazione, esattamente uguale a quello iniziale. Per consentire la rilevazione di un proprio crash, ogni LookUpServer invia periodicamente degli heartbeats a tutti i peers ad esso connessi. In seguito a un eventuale guasto i peers non ricevono piú alcun segnale e, allo scadere del timeout, possono iniziare la procedura di redirezione su un altro LookUpServer.

**Crash di un Peer**

In caso di crash di un peer è necessario che la copia ad esso associata passi allo stato di esecuzione: per questo motivo la copia riceve periodicamente dei segnali di heartbeat dal peer. Se quest'ultimo va in crash, non è piú in grado di inviare alcun segnale e la copia, scaduto il timeout, puó prendere l'iniziativa ed entrare in esecuzione.

Il processo consiste nell'innescare sul LookUpServer una procedura che si occupi di disattivare il peer uscito dalla rete e attivare la copia. I parametri identificativi dell'originale tuttavia non vengono rimossi, ma persistono in modo da essere utilizzati in seguito a una riattivazione del peer. Il binomio peer-copia rimane sempre presente all'interno del LookUpServer e solo un opportuno flag di switching stabilisce quale delle due entitá è attiva. Le richieste di ricerca dei files effettuate dai LookUpServer vengono inviate ai peers o alle copie, a seconda dello stato dei flag di switching. La procedura eseguita dal server deve anche redirigere gli heartbeats di quest'ultimo sulla copia, la quale si occuperá, fino all'eventuale riattivazione del proprio peer originale, di rilevare lo stato di attivitá del server stesso.

Quando una copia viene attivata inizia a ricevere, al posto dell'originale andato in crash, tutte le richieste di ricerca dei files e potrebbe essere scelta dal sistema per effettuare l'upload di una risorsa, qual'ora fosse giudicata come 'best peer to download' (vedi Data Transfer Balancing).

Il crash di un peer potrebbe avvenire anche durante il trasferimento di un file: per far fronte a questa eventualitá, nel momento in cui viene stabilita la connessione diretta fra il destinatario e la sorgente, quest'ultima comunica al primo l'indirizzo della propria copia. In caso di guasto il peer richiedente, senza dover effettuare nuove ricerche, si collega automaticamente alla copia e riinizia il trasferimento. Nel caso di crash della copia si rivela necessaria una nuova ricerca e il conseguente collegamento a un'altra sorgente.

**Crash di Peer e LookUpServer (Registrazione Virtuale)**

In caso di guasto del LookUpServer, se il peer originale è attivo, esso si deve occupare della redirezione propria e di quella della copia, ma se il server si disattiva quando anche il peer originale è down spetta alla copia effettuare tale operazione. Il fatto che la copia sia attiva, fa sí che il concetto di nodo ad essa associato sia ancora presente sulla rete, anche se il peer originale non c'é piú. Dal punto di vista degli altri nodi e del servizio che deve essere loro offerto è tale concetto a risultare importante; perció quando una copia, attivata in seguito al crash dell'originale, vede guastarsi anche il LookUpServer a cui è connessa, registra presso un altro server sia se stessa, sia il peer disattivo settando opportunamente il flag di switching. Quando l'originale verrá riattivato si troverá connesso a un server diverso da quello precedente. La procedura descritta, poiché prevede la registrazione di un'entitá fisicamente non presente sulla rete, è stata denominata 'registrazione virtuale'.

## 3.2 Load Balancing

Un altro aspetto, relativo alla QoS, su cui si è lavorato è il bilanciamento del carico sia a livello di Peer/Copia, per ció che riguarda il trasferimento dati, sia a livello di LookUpServer, per quanto concerne invece il numero di registrazioni.

**Registration Balancing**

Ogni utente che intende entrare nella rete, come giá spiegato, deve effettuare una registrazione presso un LookUpServer. Innanzitutto il peer deve disporre di una lista di servers memorizzata in locale, in seguito alla lettura della quale contattare il primo disponibile di essi. Questo dovrá aggiornare la server list del peer ed eseguire un controllo sul numero di connessioni di tutti i LookUpServer presenti sulla rete, al fine di determinare quello meno carico e registrare presso di esso il nuovo utente. La server list memorizzata in locale, ovvero quella utilizzata in fase di avvio per effettuare il primo collegameto a un LookUpServer, non deve necessariamente essere aggiornata allo stato attuale della rete, ma è sufficiente che contenga l'indirizzo di almeno un server attivo, dal quale ricevere poi gli IP di tutti gli altri.

**Data Transfer Balancing**

Al termine del processo di ricerca dei files che soddisfano una data query, l'utente puó scegliere di effettuare il download di una risorsa posseduta da piú peers sorgenti. Il meccanismo su cui si basa la scelta del 'miglior' peer da cui scaricare (best peer to download) è simile a quello adottato per il Registration Balancing: viene stabilita la connessione con il nodo che, allo stato attuale, sta eseguendo il minor numero di trasferimenti. Il peer richiedente contatta tutti i nodi presenti nella peer list dell'oggetto RequestFile associato alla risorsa desiderata, per conoscere quello meno carico e connettersi ad esso.

# 4 Implementazione

La fase realizzativa del progetto ha visto l'impiego della piattaforma Java e, in particolare, del supporto RMI, come sistema di invocazione remota di metodi, del MultiThreading, per la gestione della persistenza e delle Sockets per i trasferimenti dati.

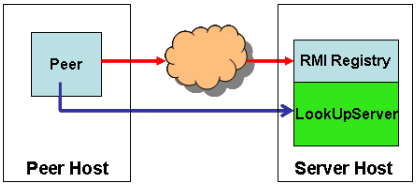


Figura 8: Ottenimento del riferimento remoto

## 4.1 RMI

Il concetto su cui si basa il meccanismo consiste nel binding di ogni oggetto (servizio), al quale è necessario poter accedere dalla rete, con un RMI registry, che ha il compito di fornire, a chiunque lo contatti, il riferimento remoto all'oggetto registrato presso di esso. Tale riferimento puó essere utilizzato, dal nodo che ne ha fatto richiesta, come se l'istanza risiedesse in locale. Il modello adottato si configura collocando l'RMI registry sul sistema nel quale risiede l'oggetto su cui si vuole ottenere un accesso remoto.

La figura 8 mostra come il peer riesce ad ottenere il riferimento remoto al LookUpServer. La server list memorizzata in locale da ogni peer, di cui si è parlato precedentemente e mediante la quale è possibile accedere alla rete nella fase iniziale di avvio, contiene perció gli indirizzi e le porte che identificano gli RMI registry che bisogna contattare per ottenere i riferimenti remoti ai LookUp Server. Anche gli stessi peers e le relative copie, poiché devono essere accedute per la ricerca dei files e le richieste di trasferimento dati, vengono connessi a dei registry.

## 4.2 MultiThreading

L'invio e la ricezione degli heartbeats devono essere attivitá che eseguono in background, mentre i processi principali (peers, copie e servers) si occupano di svolgere le loro operazioni di ricerca, registrazione, trasferimento e quant'altro. Ogni entitá genera quindi dei threads che, autonomamente, gestiscono la persistenza del sistema senza interferire con il flusso di attivitá del processo principale. Nei paragrafi successivi si vedrá come i threads sono stati utilizzati anche sul fronte dei trasferimenti dati.

**Peer**

Ogni peer genera tre threads di tipo deamon, che hanno la caratteristica di avere un tempo di vita coincidente con quello del processo che li ha generati. Il primo (PeerDeamon) deve occuparsi di inviare gli heartbeats alla copia, il secondo (PeerDeamonSrv) deve ricevere gli heartbeats dal LookUpServer presso cui è registrato il peer. Se il peer va in crash anche PeerDeamon viene distrutto perció la copia, non ricevendo piú alcun segnale, allo scadere del timeout inizia il processo di attivazione. Tutti i threads adibiti alla ricezione degli heartbeats sono dotati di un contatore (watch dog) che si decrementa a intervalli di tempo regolari. Quando viene raggiunto lo zero count inizia la procedura di recovery. Se il guasto colpisce il LookUpServer, il contatore watch dog di PeerDeamonSrv riesce ad azzerarsi provocando l'innesco del processo di registrazione presso un altro server. La frequenza di invio degli heartbeats è stata impostata a 5 sec, mentre il contatore watch dog è inizializzato a 10 sec e viene decrementato ogni secondo. Questo margine di tolleranza (5 sec) consente di far fronte ad eventuali ritardi dovuti al sovraccarico della rete o delle macchine. L'ultimo thread generato dal peer (TransferOutThread) ha il compito di occuparsi dei trasferimenti di dati in uscita (uploads). Questo processo leg-

gero, anch'esso di tipo deamon, crea una socket e la mette in ascolto di eventuali richieste. La necessitá di utilizzo dei threads diventa ora evidente; infatti il metodo accept(), essendo bloccante, non potrebbe essere eseguito dal processo principale del peer.

**LookUpServer**

Ogni LookUpServer genera un unico thread di tipo deamon (ServerDeamon) per ogni peer registrato presso di esso. Questi threads inviano gli heartbeats ai PeerDeamonSrv dei peers, in modo che ognuno di essi possa, in seguito ad un eventuale crash del server, eseguire una nuova registrazione.

**PeerCopy**

Ogni copia deve generare un deamon thread CopyDeamon, incaricato di ricevere gli heartbeats dal peer originale, un TransferOutThread per i trasferimenti in uscita e un PeerDeamonSrv, che peró non viene istanziato in fase di costruzione. Solo in seguito a un guasto del peer originale viene creato e avviato tale thread, poiché su di esso devono essere rediretti gli heartbeats inviati dal LookUp Server. Se si verifica che anche quest'ultimo va in crash, è la copia che si occupa del processo di registrazione (virtuale) presso un altro LookUpServer.

## 4.3 Sockets

I trasferimenti dei files fra i peers sono stati affidati alle sockets, per le quali si è deciso di utilizzare la configurazione di collegamento con connessione (TCP), sicura, ordinata e affidabile. La maggiore velocitá e semplicitá di utilizzo di UDP non hanno, in questo contesto, rilevanza tale, rispetto alle peculiaritá di TCP, da

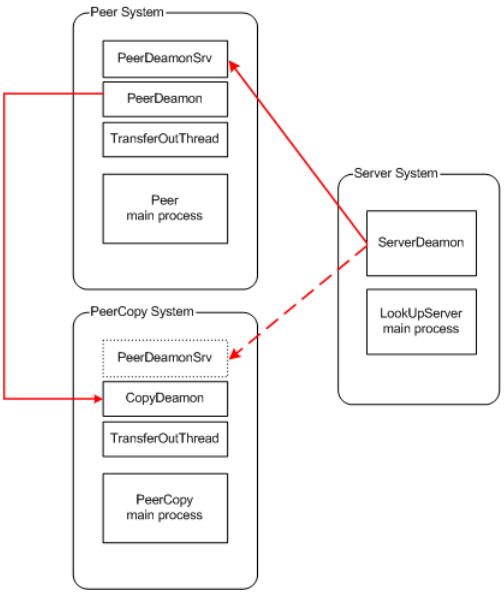


Figura 9: Sistema di Heartbeats

optare per un trasferimento a datagrammi. Quando un utente decide di effettuare il download di una risorsa, in seguito alla determinazione del 'best peer to download' viene generato un thread (TransferInThread) che crea una socket e tenta di collegarla all'end point remoto. Il peer destinatario quindi, mediante il proprio TransferOutThread che sta in ascolto di eventuali richieste di connessione, effettua l'accept di collegamento e viene stabilito il canale di comunicazione vero e proprio. A questo punto, dopo una fase preliminare di negoziazione e scambio di informazioni sulle caratteristiche della risorsa, comincia il trasferimento. Ogni peer svolge la funzione di un server concorrente, in quanto puó eseguire l'upload di piú files contemporaneamente: per permettere ció è necessaria la creazione di un ulteriore thread, in seguito alla fase di accept, che si occupi del trasferimento mentre il TransferOutThread torna in ascolto di altre richieste di connessione.

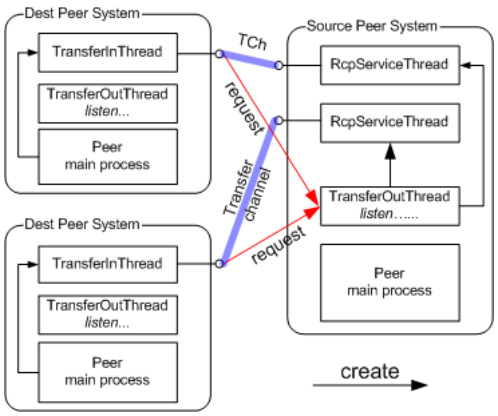


Figura 10: Richiesta, connessione e trasferimento

# 5 Test

La fase di test si è rivelata determinante nell'individuazione di bugs e malfunzionamenti. La sequenza delle prove eseguite è stata qui di seguito elencata.

- Creazione di due LookUpServer e di due peers (con relative copie) per testare il bilanciamento nella registrazione
- Aggiunta di un LookUpServer e verifica dell' effettivo suo ingresso nella rete mediante la creazione di un altro peer (con relativa copia). Si puó riscontrare la registrazione del nuovo peer presso il nuovo server (registration balancing)
- Crash di un peer, per verificare l'attivazione della copia
- Riattivazione del peer, per verificare la disattivazione della copia
- Crash di un LookUpServer, per verificare che tutti i peers (e le copie) ad esso connessi eseguano la procedura di redirezione verso un altro server
- Crash di un Peer e del relativo LookUpServer per testare la procedura di registrazione virtuale presso un altro server eseguita dalla copia
- Riattivazione del peer per verificare che si trovi registrato presso il nuovo LookUpServer su cui la copia ha precedentemente eseguito la redirezione virtuale
- Ricerca di un file posseduto da piú sorgenti effettuata dal peer 'A'
- Ricerca dello stesso file effettuata dal peer 'B'
- Richiesta di download del file effettuata da 'A'
- Richiesta contemporanea di download del file effettuata da 'B', per verificare che la sorgente sia diversa da quella di 'A' (Data Transfer Balancing)
- Ricerca di un file posseduto da una sola sorgente effettuata da due peers differenti
- Richiesta di download del file effettuata contemporanenamente da entrambi i peers, per verificare che la sorgente si comporti come un server concorrente
- Crash di un peer unico proprietario di un file
- Ricerca del file per verificare che esso venga comunque trovato grazie all'attivazione della copia
- Richiesta di download del file per verificare che la copia esegua correttamente il trasferimento

# 6 Conclusioni

La realizzazione del progetto ha permesso di approfondire alcuni degli argomenti principali del corso di Reti di Calcolatori LS, come la disponibilitá di servizio, le tecniche di replicazione e il bilanciamento del carico. E' stato dunque possibile analizzare concretamente tutti i passi, con relative problematiche, che hanno portato al raggiungimento di un soddisfacente livello di QoS. Il sistema realizzato presenta numerose semplificazioni rispetto ai moderni software P2P. D'altra parte l'obiettivo era quello di concentrare gli sforzi maggiori sugli aspetti sopra citati, magari trascurando alcune funzionalitá di contorno o di importanza minore, le quali potrebbero costituire un buon punto di partenza per eventuali sviluppi futuri. I test eseguiti hanno consentito un buon debugging dell'applicazione, cercando di coprire il piú possibile lo spettro dei possibili eventi di guasto.

# Indice