Servizi di Content Delivery per comunitá di utenti

Vittorio Manetti, Consorzio CRIAI
vittorio.manetti@criai.it

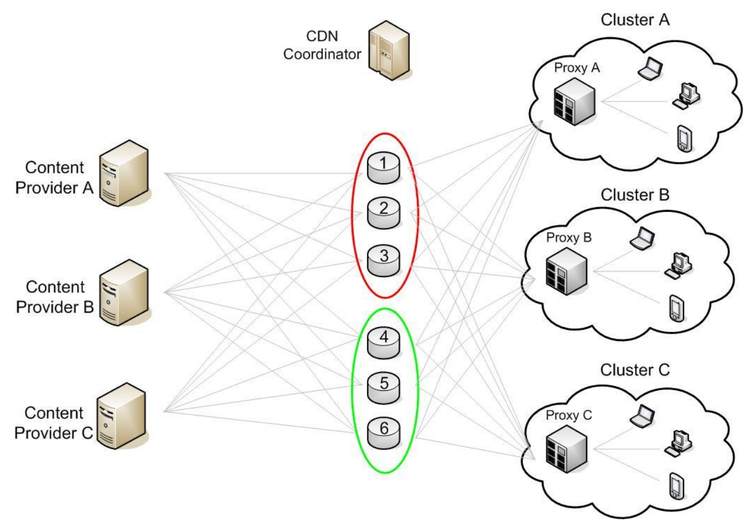
Roberto Canonico, Walter de Donato, Giorgio Ventre,
Dipartimento di Informatica e Sistemistica
Universitá di Napoli Federico II
{roberto.canonico, walter.dedonato, giorgio.ventre}@unina.it

Introduzione

L'eterogeneitá dei contenuti offerti in Internet é estremamente elevata, é possibile tuttavia individuare delle comunitá di utenti definite come tali sia per quanto riguarda l'infrastruttura utilizzata per l'accesso alla rete, sia per quanto riguarda gli interessi dai quali sono accomunati. Molto spesso le comunitá di utenti sono organizzate in sottogruppi, e molto spesso si rivolgono a terze parti per ricevere servizi che migliorino le prestazioni relative alla fruizione di contenuti.
Obiettivo del nostro lavoro é quello di realizzare una infrastruttura di rete che implementi servizi per il caching e la distribuzione di contenuti Web a comunitá di utenti. Entrambi i servizi vengono eseguiti sfruttando delle cache messe a disposizione da terze parti, quindi uno o piú *Internet Service Provider* (*ISP*), ed offerti solo per determinati domini, quelli a cui gli utenti fanno accesso con maggior frequenza.
L'obiettivo é quello di incrementare la qualitá percepita dal generico utente implementando meccanismi di caching che riducano la latenza relativa al meccanismo di accesso ai contenuti richiesti. Il modello proposto é organizzato in maniera tale che gli utenti della comunitá abbiano la possibilitá di sottoscrivere il servizio presso il Service Provider in maniera dinamica.

Un sistema centralizzato per il Content Delivery

L'architettura che proponiamo é composta da un certo numero di entitá alle quali é assegnato il compito di gestire alcune repliche di contenuti Web all´interno di un sistema di cache, al fine di ottimizzare la qualitá percepita dal generico utente della comunitá. Le cache che compongono il sistema sono fornite da terze parti, e dislocate all'interno della rete. Ogni cache é utilizzata per implementare le funzionalitá tipiche, ed alcuni moduli per il monitoraggio e la configurazione automatica.



Introduciamo le entitá *Proxy* e *CDN Coordinator*.

Il nodo Proxy

Obiettivo del modello é quello di mettere a disposizione degli utenti alcune politiche di *request routing* di tipo *content-aware*. La soluzione che adottiamo é quella di introdurre un nodo Proxy per ogni cluster di utenti. Compito del Proxy é quello di instradare le richieste sulla base di informazioni relative alla localizzazione dei contenuti nel sistema di cache. I Proxy vengono periodicamente istruiti da una ulteriore componente riguardo le politiche di routing da adottare per redirigere la specifica richiesta verso la cache opportuna.
Ogni nodo Proxy gestisce una *Content Routing Table*; la tabella realizza la corrispondenza tra i contenuti e le relative cache. Ogni riga della tabella é formata dalla coppia *chiave-valore*; il primo parametro costituisce la chiave di accesso alle informazioni, quindi l'URL della risorsa, mentre il secondo costituisce l'identificativo della cache verso cui la richiesta deve essere rediretta.

Il nodo CDN Coordinator

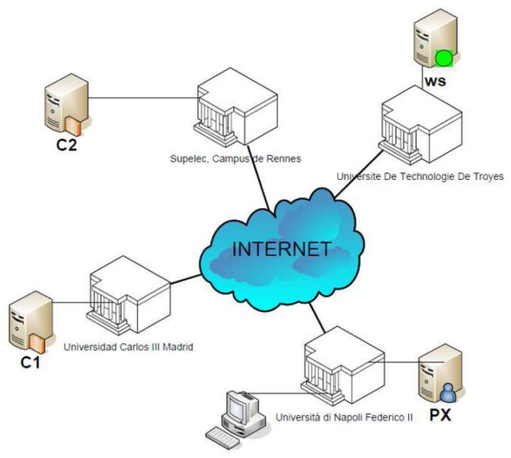
Il CDN Coordinator svolge nell'ordine le seguenti funzioni:

- raccolta di informazioni;
- calcolo dello schema di localizzazione ottimo;
- caricamento dei contenuti nelle cache;
- aggiornamento delle Content Routing Table dei proxy.

Il coordinatore si occupa della raccolta di informazioni relative a: stato della rete, banda a disposizione, traffico generato dagli utenti di ogni cluster, numero di accessi ad ogni risorsa. La misura di tali parametri viene eseguita da specifiche entitá e non direttamente dal CDN Coordinator.
Il principale compito assegnato al CDN Coordinator é quello di calcolare lo schema di localizzazione ottimo dei contenuti Web all'interno del sistema di cache, e a tale scopo implementa un modello di programmazione lineare. Lo schema di localizzazione viene calcolato periodicamente, e a seguito di tale operazione il CDN Coordinator predispone la rete affinché vengano rispettate le politiche di request routing ottenute come risultato della computazione.
Il Proxy viene messo al corrente dello schema di localizzazione dei contenuti appena calcolato e sulle relative politiche di request routing. Interviene nuovamente il CDN Coordinator, il quale é a conoscenza di tali informazioni ed esegue l'aggiornamento delle tabelle di routing dei Proxy. Viene utilizzato a tale scopo un protocollo di comunicazione appositamente realizzato (*Content Routine Management Protocol - CRMP*), grazie al quale il CDN Coordinator accede alla Content Routing Table dei Proxy per inserire, aggiornare, eliminare righe della tabella.

Studi sperimentali

Per valutare le prestazioni dell'architettura proposta in uno scenario realistico, é stata effettuata una serie di esperimenti sfruttando l'infrastruttura Planetlab.

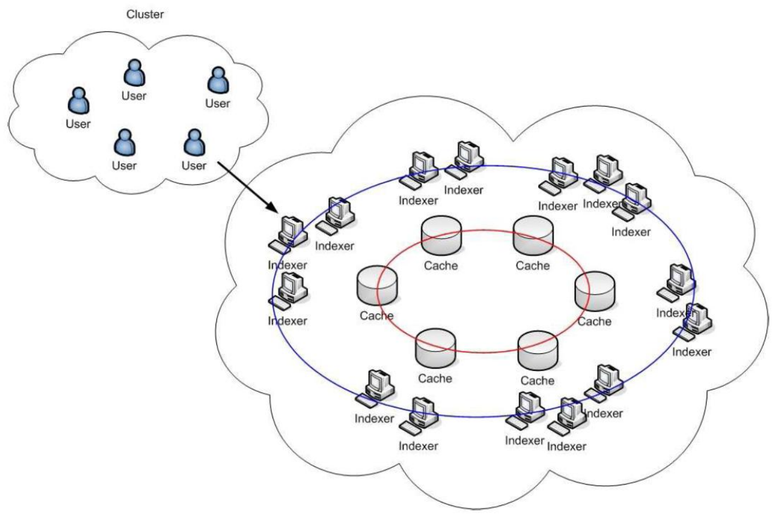


La scelta dei nodi é stata effettuata in maniera tale da evidenziare i benefici introdotti dall'architettura. In particolare, il web server é stato collocato in una rete con accesso ad Internet a banda limitata per rendere evidente la convenienza del caching dei contenuti da esso forniti, e le due cache sono state posizionate in luoghi geograficamente lontani tra loro e con connettivitá di capacitá differente.
Gli esperimenti sono stati condotti valutando i tempi di consegna dei contenuti in diverse situazioni. Per fare ció si é collocato sul web server un oggetto di dimensione pari a 4.162.048 bytes e sono state effettuate 8 richieste per tale contenuto da parte del client in diversi scenari. Di tali richieste é stato valutato il tempo di risposta, cioé quello che intercorre tra l'invio della richiesta e la completa ricezione del contenuto.
I risultati ottenuti confermano che un opportuna configurazione della tabella di content routing del Proxy puó migliorare significativamente le prestazioni percepite dagli utenti.

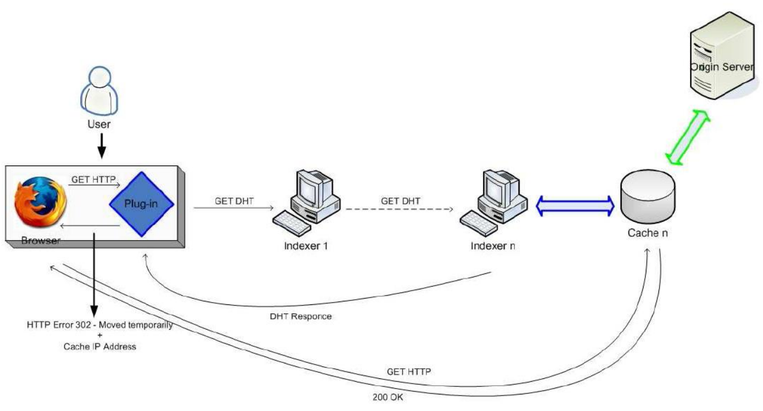
Evoluzione verso una infrastruttura peer-to-peer

L'architettura peer-to-peer che proponiamo é strutturata come una *Stealth DHT*; i nodi che compongono una rete di questo tipo possono essere classificati come:

- *service node*: possiede le medesime caratteristiche del generico peer di una DHT classica, é quindi responsabile dello storage di contenuti, del routing e del forwarding di richieste;
- *stealth node*: non esegue storage di contenuti. Gestisce una routing table con una singola entry ed é in grado di identificare solo il *first hop* verso cui inoltrare una richiesta; non puó ricevere richieste perché non é in grado di gestirle. Durante il *join* ad una DHT esegue solo la fase di *state gathering* e non esegue la fase di *announcement*, per cui rimane trasparente ai service node.



Utilizziamo l'organizzazione delle Stealth DHT perché non soffrono di problemi dovuti al *churning* dei nodi trasparenti, ed il modello che proponiamo é caratterizzato dalla presenza di un elevato numero di nodi non stabili.
Nella nostra architettura il generico terminale utente puó essere identificato come la combinazione di due componenti:

- *browser*: é l'interfaccia utilizzata dall'utente per accedere ai servizi offerti dall'architettura;
- *stealth node*: é il punto di accesso alla DHT.

Il browser possiede un plug-in che, in riferimento allo specifico contenuto richiesto, é in grado di verificare se il servizio di caching e delivery é fornito o meno; in caso negativo, invia una richiesta direttamente verso il server di origine.
Identifichiamo il generico service node come *Indexer*; ad ogni Indexer é associata una Cache secondo una relazione del tipo 1 a 1 oppure n a 1. L'Indexer possiede una interfaccia DHT mediante la quale comunica con altri service node e con gli stealth node, ed una interfaccia HTTP mediante la quale comunica con la Cache. Gli utenti (stealth node) appartenenti al medesimo cluster fanno riferimento al medesimo Indexer (service node) come first hop per le loro richieste. L'Indexer non viene messo a disposizione dall'ISP, ma é una risorsa della comunitá: il generico terminale utente che verifica determinati requisiti di stabilitá, puó essere eletto Indexer.