

Specifiche del Progetto per l'esame
Laboratorio di Programmazione di Rete
a.a 2001/02
Prof. Panzieri

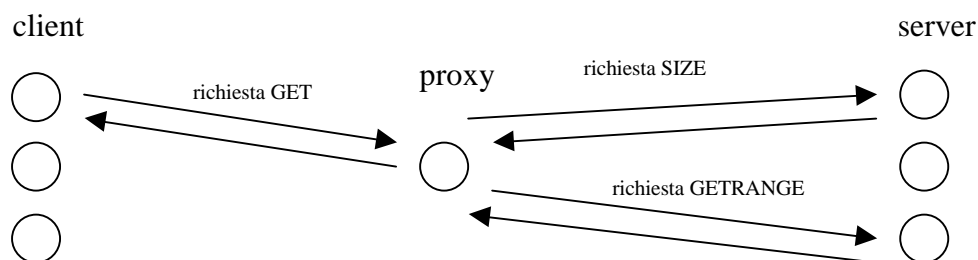
Si consideri un'architettura costituita da 3 tipi di entità: Client, Proxy e Server. I server operano su host che contengono esattamente gli stessi file, tutti di piccole dimensioni (meno di 100 kbyte). Lo scopo di queste architetture è di trasportare file contenuti negli host in cui eseguono i server, fino agli host in cui si trovano i client.

I client possono effettuare richieste (di tipo GET) per ottenere l'intero file richiesto.

I server rispondono solo a richieste (di tipo GETRANGE) che chiedono porzioni (**di dimensioni minori o uguali a 100 byte**) del file richiesto, e a richieste (di tipo SIZE) in cui viene chiesta la dimensione di un file.

I proxy quindi debbono ricevere dai client le richieste di tipo GET per un dato file, e mediante una sequenza di richieste, chiedere ai server la dimensione del file richiesto e successivamente chiedere porzioni (di dimensioni minori o uguali a 100 byte) di tali file ai diversi server, per ricomporre il file richiesto e restituire al client il file richiesto intero.

Si vuole velocizzare la trasmissione dei file verso il client, quindi man mano che il file viene ricomposto sul proxy, sarebbe una buona idea cominciare a trasferirlo verso il client. Ovviamente il client deve ricevere i byte che costituiscono il file richiesto nell'ordine in cui tali byte stanno nel file.



Tutte le comunicazioni tra le diverse entità avvengono mediante connessioni TCP. Per ogni richiesta, il richiedente apre una connessione TCP col server, trasmette la richiesta nel formato specificato, ottiene la risposta sulla stessa connessione, a questo punto il server ed il richiedente chiudono la connessione. In particolare, i server possono simulare un errore ed interrompere la comunicazione in un qualsiasi momento. Nel caso che il server riceva una richiesta non corretta, o per un file che non esiste, ecc., il server spedisce al proxy una risposta di tipo ERRORE, senza specificare il motivo dell'errore.

Ogni proxy può ricevere richieste da diversi client. Per servire ciascuna risposta, il proxy può comunicare contemporaneamente con tutti i server, ma al massimo può mantenere con ciascun server una connessione per ciascuna richiesta che sta servendo. Quando la connessione con un dato server è stata chiusa, il proxy può aprirne una nuova, per effettuare una nuova richiesta.

Il progetto prevede di implementare un client ed un proxy che svolgano i compiti descritti, mentre i server sono già stati implementati.

I server devono essere almeno 3, e possono essere eseguiti sullo stesso host. Anche client e proxy possono essere eseguiti sullo stesso host dei server. Il proxy ottiene gli indirizzi IP ed i numeri di porta dei server mediante parametri passati a riga di comando. Analogamente, i client ottengono l'indirizzo IP e la porta del proxy, ed il nome del file da chiedere mediante parametri passati a riga di comando. Inoltre i client prendono dalla riga di comando il nome del file in cui viene salvato il file ricevuto.

Il codice (C) che implementa i server è messo a disposizione all'indirizzo:

http://www.cs.unibo.it/~ghini/didattica/lab_prog_rete/lpr_index.html

I server prendono in input mediante parametri passati a riga di comando rispettivamente:

- La porta TCP su cui rimanere in attesa.
- Un numero intero che funge da seme per inizializzare il generatore di numeri casuali.
- Un numero floating point che rappresenta la probabilità che accada un errore durante la trasmissione di un byte. Questo parametro è consigliabile sia maggiore o uguale a 0.0002 ed inferiore a 0.01.
- Un numero intero che rappresenta il ritardo in millisecondi tra la trasmissione di due byte successivi. Tale parametro è consigliabile sia settato per i tre diversi server rispettivamente ai valori di 10, 20, 50 o superiori.

Il protocollo tra le diverse entità prevede lo scambio di messaggi in cui il primo byte rappresenta il tipo di richiesta/risposta che viene trasmessa. Tutti gli altri campi dei messaggi sono sequenze di caratteri terminate da un byte di valore 0.

Se ad esempio un campo che deve trasportare la dimensione del file ha lunghezza 10, allora per trasportare la lunghezza sono disponibili 9 caratteri, ed il decimo carattere deve essere **OBBLIGATORIAMENTE** impostato al valore 0, per rappresentare la fine della stringa.

Così, se la dimensione da indicare è 999111888, allora il campo avrà formato:

'9'	'9'	'9'	'1'	'1'	'1'	'8'	'8'	'8'	0
-----	-----	-----	-----	-----	-----	-----	-----	-----	---

Invece, se la dimensione da indicare è 481, allora il campo avrà formato:

'4'	'8'	'1'	0	0	0	0	0	0	0
-----	-----	-----	---	---	---	---	---	---	---

Notare come tutti i caratteri non utilizzati per memorizzare la lunghezza debbano essere posti a zero. In particolare, i messaggi sono così formati:

messaggio richiesta GET, da client a proxy:

'G'	1 byte , comando Get
'n' 'o' 'm' 'e' 0	255 byte, nome del file richiesto

messaggio risposta alla GET, da proxy a client:

'G'	1 byte , risposta Get
'2' '7' '4' '1' 0	10 byte, dimensione del file richiesto
[]	Corpo del messaggio, tanti byte quanti specificati dalla dimensione del file richiesto, contengono esattamente il file richiesto.

messaggio risposta ERRORE, da proxy a client e da server a proxy:

'E'	1 byte , risposta Errore
'n' 'o' 'm' 'e' 0	100 byte, eventuale messaggio, ma i server li mette tutti a zero

messaggio richiesta SIZE, da proxy a server:

'S'	1 byte , comando Get
'n' 'o' 'm' 'e' 0	255 byte, nome del file di cui si vuole la dimensione

messaggio risposta alla SIZE, da server a proxy:

'S'	1 byte , comando Get
'2' '7' '4' '1' 0	10 byte, dimensione del file (es 2741)

messaggio richiesta GETRANGE, da proxy a server:

nell'esempio si richiedono 12 byte, dal 65-esimo al 76-esimo compresi del file "nome", dove il primo byte del file viene considerato avente indice 0.

'R'	1 byte , comando getRange
'n' 'o' 'm' 'e' 0	255 byte, nome del file richiesto
'6' '5' 0	10 byte, primo byte dell'intervallo richiesto
'7' '6' 0	10 byte, ultimo byte dell'intervallo richiesto

messaggio risposta alla GETRANGE, da server a proxy:

nell'esempio si restituiscono 12 byte, dal 65-esimo al 76-esimo compresi, come richiesto dalla precedente GETRANGE.

'R'	1 byte , risposta getRange
'1' '2' 0	10 byte, dimensione dell'intervallo richiesto
[]	Corpo del messaggio, tanti byte quanti specificati dalla dimensione dell'intervallo richiesto, contengono esattamente i byte dell'intervallo richiesto