*Advanced Topics in Computer Systems and Networks*

Multiple http-ping



RELAZIONE

Professore L. Rizzo Giulio Scrima  
  
 Fabio Sofia

1. Introduzione

1.1 Repository http\_ping

Oggetto del seguente progetto è la modifica della repository per Freebsd http\_ping. Tale programma appartiene al dominio dei progetti di rete ed è ampiamente sfruttato per diversi fini, primi tra tutti test di rete, permettendo di calcolare la velocità di connessione tra due computer o server che comunicano online. In particolare, sono state ampliate le funzionalità di base, per far si che si potesse inviare il ping non solo a singolo indirizzo, come previsto di base dal programma, ma a destinazioni multiple.

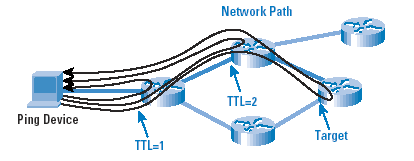


Figura 1: Ping-http, Network Path

Il progetto è stato sviluppato in linguaggio di programmazione C.

La repository originale è presente al seguente indirizzo: <http://www.acme.com/software/http_ping/>.  
Il port FreeBSD è presente al seguente indirizzo: <https://svnweb.freebsd.org/ports/head/net/http_ping/>.

1.1 Http ping   
  
Come già anticipato, il ping permette di stabilire alcuni parametri in una comunicazione di rete. Esso comporta l'invio di una richiesta http e misura il tempo trascorso tra l’invio di una richiesta da un computer fino al raggiungimento di questa ad un destinatario, mediante l'indicazione esplicita dell'indirizzo di rete da parte dell'utente. In particolare, vengono inviate un numero non stabilito in precedenza di richieste http (sarà l'utente a decidere quando fermare le richieste) e sono tre le statistiche di interesse, che vengono misurate e restituite nella stampa di output: il minimo, la media e il massimo di tempo di connessione, tempo di risposta e tempo di trasferimento dati.

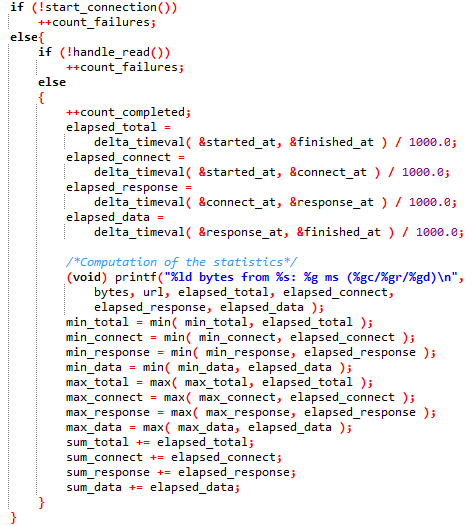


Figura 2: Calcolo delle statistiche all’interno del codice di http\_ping

Dunque, l'utente effettua la richiesta inserendo su riga di comando il nome del programma seguito dall'URL di interesse. Per ogni richiesta, il programma mostra come output le suddette statistiche, al completamento di un numero di fetch http che il programma prevede di base che non sia stabilito a priori. Inoltre è possibile inserire differenti opzioni che comportano un risultato di output diverso. Tra queste troviamo l'inserimento di un numero massimo di ping che verranno effettuate, l'inserimento di un intervallo di tempo tra una richiesta ed un'altra, un timeout massimo per le richieste e differenti altre.   
 ./http\_ping [-option] URL

Un esempio di esecuzione del programma è il seguente:

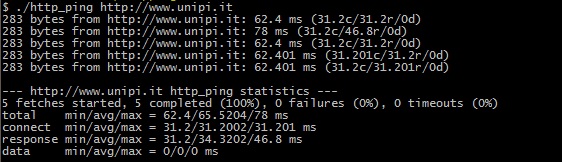


Figura 3: Esecuzione di http\_ping

2. Multiple http ping

2.1 Struttura del programma

Sono state apportate delle modifiche al codice per ottenere l'obiettivo principale del progetto, ossia permettere l'invio del ping a destinazioni multiple. Innanzitutto sono state rimosse le opzioni precedentemente discusse, in quanto non in linea con gli obiettivi del progetto. E' stato solamente imposto un numero di fetch fissato a priori, che è stato fissato a cinque. Al termine di questi verrà pertanto prodotto il risultato.   
E' stato considerato il tempo di risposta come unico elemento da considerare del ping http per classificare i siti inseriti dall'utente. Per la gestione di più richieste http, è stato scelto di avere un processo master il quale crea dei processi figli e assegna a ciascuno di questi una singola richiesta http. Il numero di richieste non è limitato ma viene stabilito a tempo di esecuzione, a seconda del numero indirizzi inseriti dall'utente su riga di comando. Pertanto si avranno un numero di processi figli pari al numero di indirizzi inseriti dall'utente.   
Insieme al nome del programma e ai siti, come secondo parametro si può inserire una tra le seguenti opzioni, ognuna riferita a una delle tre statistiche restituite dalla http-ping:

-min, nel caso si voglia un ordinamento secondo il minimo tempo di risposta alla richiesta http;

-avg, nel caso si voglia un ordinamento in base al tempo medio di risposta (delle cinque fetch effettuate) alla richiesta http;

-max, nel caso si voglia un ordinamento in base al tempo massimo di risposta alla risposta http.

Per l'ordinamento è stata sviluppata una funzione qsort, a cui è associata una apposita funzione compare().

Lanciato in esecuzione il programma, vengono mostrate a schermo le richieste effettuate e vengono visualizzati i singoli risultati (per singole fetch). Dopodiché il processo padre, alla chiusura di ogni processo figlio precedentemente lanciato, aggiornerà la struttura contenente i risultati, ordinati secondo la statistica scelta in precedenza.

Per la scrittura e successiva lettura delle statistiche da parte dei singoli processi figli si fa uso di scrittura/lettura su/da file, non bufferizzata e su cui viene garantita la mutua esclusione (in fase di scrittura). In particolare, i processi figli, che eseguiranno il ping sugli indirizzi indicati dall'utente, andranno a scrivere su file il nome dell'indirizzo seguito dalle tre statistiche di min, max e avg sulle cinque fetch http effettuate. Una volta terminata l'esecuzione, i processi figli restituiranno i risultati, che saranno raccolti dal processo padre che chiamerà la qsort e andrà a ordinare i dati aggiornati alla esecuzione di http ping del figlio appena terminato. Ciò vuol dire che i dati saranno aggiornati in tempo reale man mano che i processi eseguono la loro funzione, ossia inviare il ping all'indirizzo che è passato a loro dal processo master, ricevere risposta dal sito e, infine, aggiornare la "classifica", che verrà ordinata secondo l'opzione selezionata mediante riga di comando, come descritto precedentemente.

2.2 Analisi del funzionamento del codice

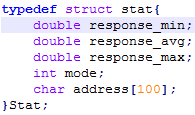
Per quanto riguarda la struttura del codice, esso contiene una struttura Stat, che verrà utilizzata per memorizzare le statistiche misurate dai processi figli, durante l'esecuzione del codice. Tale struttura conterrà le informazioni sul singolo indirizzo ed è formata da cinque campi:

Figura 4: Struttura Stat

response\_min, ossia un double che conterrà il tempo di risposta minimo dell'indirizzo;

response\_avg, ossia un double che conterrà il tempo di risposta minimo dell'indirizzo;

response\_max, ossia un double che conterrà il tempo di risposta minimo dell'indirizzo;

mode, intero che potrà valere uno dei valori tra 0,1 e 2 a seconda della scelta della statistica effettuata dall'utente;

address, array di caratteri che conterrà l'indirizzo che ha prodotto le statistiche sopra.

Sono state implementate tre differenti funzioni che operano sulle statistiche passate in input a queste: “alloca\_stat”, “compare” e “best\_response”.   
La funzione alloca\_stat alloca semplicemente la struttura per le statistiche a seconda del numero di richieste ricevute tramite riga di comando.   
La funzione compare rappresenta la callback presente nella funzione qsort e serve per l'ordinamento delle statistiche che verrà effettuato al fine di determinare la classificazione finale degli indirizzi passati al programma.   
La funzione best\_response è la funzione che, ricevute come input le statistiche, andrà a chiamare la funzione qsort e stamperà i risultati come output.

2.3 Analisi del funzionamento del programma

Il programma chiamato "master.exe" riceve in ingresso un valore fra "-avg", "-min", "-max" e una serie di indirizzi internet.   
I primi comandi servono a indicare con quale criterio stampare i risultati delle statistiche che vengono estrapolate. Particolare attenzione viene posta al caso di errore nell’inserimento di un comando diverso da quelli sopra elencato; in tal caso si rimanda all’utente capire dove sta l’errore nella digitazione del comando.  
Per memorizzare il comando scelto viene eseguita la lettura della seconda stringa da riga di comando e inserito una catena di if-else che effettuano il confronto con i tre possibili comandi, e una variabile intera viene modificata appositamente per tenere memoria della scelta effettuata; successivamente questa variabile verrà usata nell’inserimento del campo “mode” della struttura “Stat”.

Viene creato dal processo master un processo figlio, uno per ogni indirizzo inserito da riga di comando; pertanto ad ogni processo figlio verrà assegnato un singolo indirizzo. Ciò viene effettuato mediante l'uso della primitiva fork e della funzione "execl".   
La fork crea un processo figlio che userà la funzione “execl” per l’esecuzione di “http.exe”. Ogni processo figlio usa come input per l’esecuzione della propria versione del programma “http.exe” uno degli indirizzi inseriti da riga di comando.

Il prototipo della funzione “execl” è il seguente:

int execl(const char \*path, const char \*arg, ...);

La execl è una funzione della libreria “unistd.h”, che appartiene alla famiglia di funzioni exec() e che sostituisce l’immagine di processo corrente con una nuova immagine di processo. Come parametri passati alla funzione abbiamo il parametro “path”, ossia il percorso contenente l’eseguibile del programma, ossia “http.exe”, versione modificata dell’http\_ping, e una serie di argomenti che possono essere passati al processo creato. Tra questi l’eseguibile stesso, come primo argomento, e il successivo è l’i-esimo di n indirizzi di cui si vuole effettuare il ping, ossia all’i-esimo figlio viene passato l’i-esimo indirizzo.

Successivamente alla esecuzione di “http.exe” sull'indirizzo, il processo figlio scrive i risultati ottenuti su file, andando a indicare l'indirizzo a cui è stato inviato il ping e le statistiche, ottenute per response time. Ultimata questa operazione, il processo figlio termina.

Tutte le volte che un processo figlio termina la sua esecuzione, il processo padre prende il suo risultato, andando a estrarre da file le statistiche inserite fino al momento della sua esecuzione, ne esegue l'ordinamento e stampa i siti ordinati secondo la statistica indicata per mezzo della riga di comando.

In dettaglio, per memorizzare le statistiche viene utilizzato un vettore di strutture “Stat” preallocato, utilizzando “argc” come informazione sulla dimensione.

Ogni qualvolta che un processo figlio termina la propria esecuzione di “http.exe”, viene aperto dal processo padre il file dove sono registrati i risultati e questi dati sono inseriti nei campi della struttura “Stat”. L’ordinamento prevede l’utilizzo della funzione “qsort”, il cui prototipo è:

void qsort (void\* base, size\_t num, size\_t size, int (\*compar)(const void\*,const void\*));

Nella funzione “qsort” viene inserito un vettore di strutture, la dimensione del vettore e un puntatore alla funzione “compare” che è stata costruita in modo da essere eseguita con la struttura dati “Stat” utilizzata.

Alla funzione “compare” vengono passati come input due valori da confrontare; in questo caso i valori utilizzati verranno selezionati all’interno della struttura dati a seconda del parametro “mode”, scelto mediante riga di comando, usando uno switch.

A questo punto, il processo padre stamperà a schermo il vettore della struttura “Stat” ordinato nella suddetta modalità. La stampa prevede una formattazione con i dati di ogni indirizzo su ogni riga, comprendendo tempo minimo, massimo e medio, avendo cosi un’ampia visuale sui tempi di risposta di ogni indirizzo.



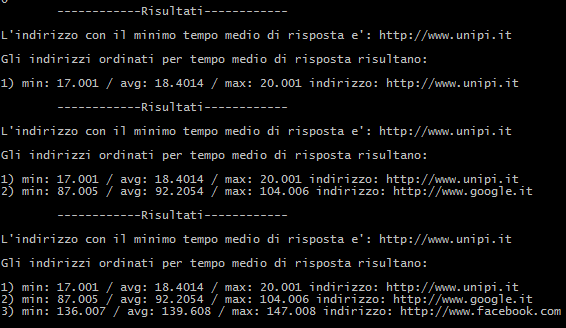


Figura 5: Esecuzione del programma

In particolare, il processo padre cicla grazie ad un while. Nel processo padre si fa uso di “waitpid”, il cui prototipo è:

pid\_t waitpid(pid\_t pid, int \*status, int options);

In particolare viene fatto uso dell’opzione “WNOHANG” che rende la suddetta wait di tipo non bloccante, cosicchè tutti i processi figli possano continuare la loro esecuzione.   
La waitpid assumerà un valore maggiore di 0 quando un processo figlio terminerà e, in particolare, sarà restituito il PID del suddetto figlio. Solo quando la “waitpid” ritorna il valore maggiore di 0 allora il processo padre elaborerà le statistiche aggiornate al momento in cui il figlio ha terminato la sua esecuzione.  
Nel caso in cui ci siano presenti processi figli ancora in esecuzione, il processo padre esegue nuovamente il ciclo fintanto che saranno terminati tutti i processi figli. In tale momento il processo padre deallocherà lo spazio di memoria della struttura Stat, andrà a rimuovere il file e terminerà la sua esecuzione uscendo dal ciclo mediante un break.

3. Patch

Terminata l’implementazione del progetto, è stato realizzato il patch-file al fine di analizzare e riportare le modifiche effettuate alla repository FreeBSD. Ciò è stato effettuato mediante il seguente comando:

diff –ubwrN ./work-orig/port\_name ./work/port\_name >> patch\_file

Le opzioni usati per l’esecuzione del comando “diff” sono:

u : imposta il file di output

r : applica il comando in modo iterativo su tutta la directory selezionata

b, w: ignora spazi e tabulazioni

N(New) : inserisce nel patchfile anche i nuovi file

Il patch file è presente nell’archivio del progetto.