Homework 2

Metriche di rete con comando `ping`

Giacomo Calabria - 2007964

June 10, 2023

1 Introduzione

In questo Homework si vogliono utilizzare alcuni strumenti messi a disposizione dal sistema operativo per stimare, tramite modelli matematici, il throughput di una connessione.

1.1 Ambiente utilizzato

Per eseguire il codice, è necessario avere accesso a un sistema operativo che supporti il comando `ping`. Nel nostro caso, abbiamo utilizzato il sistema operativo Linux per eseguire il codice, ma il codice può essere adattato a qualsiasi sistema operativo compatibile con il comando.

Il codice per l'acquisizione dei dati del comando è stato scritto in bash, un linguaggio di scripting ampiamente utilizzato in ambiente Linux. Per eseguire il codice è necessario eseguire il comando: `sudo bash ./ping_ttl.sh`. Si è scelto di utilizzare la shell di Linux in quanto l'applicativo `ping` è nativamente più completo su Linux.

Per l'analisi dei dati raccolti con gli script, abbiamo invece utilizzato l'ambiente di MATLAB ® in quanto fornisce metodi semplici per creare i grafici e i metodi numerici per stimare il coefficiente a a partire dagli RTT minimi.

NB: tutto il codice è contenuto nel file "Codice.zip" ed è stato inserito anche un file con i dati raccolti nell' esperimento

1.2 Il comando `ping`

L'applicazione `ping` utilizza il protocollo *Internet Control Message Protocol* (ICMP), che si appoggia direttamente sull' *Internet Protocol* (IP), senza coinvolgere alcun protocollo di livello di trasporto. Il suo scopo principale è quello di fornire uno strumento di diagnostica di rete che consente di capire se un certo host è raggiungibile o meno.

In breve, l'applicazione `ping` invia una serie di pacchetti a una destinazione specificata, la quale risponde con pacchetti della stessa dimensione. L'applicazione misura e visualizza il cosiddetto *Round-Trip Time* (RTT) per ogni pacchetto, ovvero il tempo che intercorre tra l'invio della richiesta e la ricezione della risposta.

Il RTT è influenzato da diversi fattori di ritardo su ogni singolo link del percorso da sorgente a destinazione e da destinazione a sorgente. I valori osservati di RTT dipendono dalla lunghezza L del pacchetto, ma a parità di L, ogni esecuzione della misura può dar luogo a valori osservati diversi a causa dei ritardi di accodamento nei vari link. Il valore del RTT può variare notevolmente a seconda delle condizioni di rete, come la congestione, la qualità del collegamento e la distanza fisica tra i nodi.

Dal manuale linux `ping(8)` abbiamo:

```
ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP
ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (''pings'') have
an IP and ICMP header, followed by a struct timeval and then an arbitrary
number of ''pad'' bytes used to fill out the packet.
```

Il comando `ping` è molto completo nell' ambiente Linux e permette di avere numerse opzioni/-parametri; noi abbiamo utilizzato le seguenti:

```
ping -A -q [ -c count] [ -s packetsize] [ -t ttl] destination
```

- `-A` permette di avere il cosiddetto "ping adattivo". Ovvero l'intervallo tra i pacchetti si adatta al tempo di andata e ritorno, in modo che effettivamente non più di una richiesta senza risposta sia presente nella rete.
- `-q` output silenzioso. Non viene visualizzato nulla tranne le righe di riepilogo all'avvio e al termine.
- `[ -c count]` interrompe il comando dopo aver inviato #`count` pacchetti ECHO_REQUEST
- `[ -s packetsize]` specifica il numero di byte di dati da inviare.
- `[ -t ttl]` imposta il Time to Live del protocollo IP

Tra i server disponibili, abbiamo scelto un server a Londra per eseguire i test dell' esperimento; raggiungibile all'indirizzo `lon.speedtest.clouvider.net`

2 Stima del numero di link attraversati

Nell'header del protocollo IP, il *Time-To-Live* (TTL) è un campo che viene impostato dal mittente e decrementato di una unità ogni qual volta il pacchetto viene inoltrato da un nodo intermedio. Questo campo ha lo scopo di evitare che i pacchetti rimangano intrappolati indefinitamente nella rete. Quando un nodo riceve un pacchetto con un TTL di 1, se non è il destinatario finale, scarta il pacchetto e invia un messaggio di errore all'applicazione mittente. Pertanto, il TTL determina il numero massimo di link che il pacchetto può attraversare prima di raggiungere la sua destinazione.

Per stimare il numero di link attraversati da un pacchetto, abbiamo sfruttando l'opzione del comando `ping` che permette di variare il campo TTL `[ -t ttl]`. Specificando il valore iniziale del TTL da utilizzare durante l'invio dei pacchetti, abbiamo gradualmente incrementato il valore del TTL finchè non abbiamo ottenuto una connessione senza errore, indicando che il pacchetto è riuscito a raggiungere la destinazione.

Abbiamo creato uno script bash "ping_ttl.sh" che automatizza questo processo e ci permette di determinare il valore minimo di TTL. Per eseguirlo è necessario inserire il comando da terminale: `sudo bash ./ping_ttl.sh`

Lo script ha prodotto il seguente output:

```
sudo ping -c 2 -t 4 lon.speedtest.clouvider.net
sudo ping -c 2 -t 5 lon.speedtest.clouvider.net
[...]
sudo ping -c 2 -t 10 lon.speedtest.clouvider.net
sudo ping -c 2 -t 11 lon.speedtest.clouvider.net

E' necessario attraversare: n = 11 link per raggiungere la destinazione

traceroute to lon.speedtest.clouvider.net (5.180.211.133), 30 hops max, 60 byte...
 1  _gateway (192.168.5.253)  0.585 ms  0.559 ms  0.556 ms
 2  *.*.*.* (*.*.*.*)  0.727 ms  0.812 ms  0.960 ms
 3  *.*.*.* (*.*.*.*)  8.553 ms  8.550 ms  8.547 ms
 4  *.*.*.* (*.*.*.*)  3.926 ms  3.924 ms  4.062 ms
 5  *.*.*.* (*.*.*.*)  8.818 ms  8.813 ms  8.811 ms
 6  151.7.112.69 (151.7.112.69)  15.662 ms  12.502 ms  12.494 ms
 7  linx-lon1.eq-ld8.peering.clouvider.net (195.66.225.184)  32.714 ms  44.599 ...
 8  10.1.10.65 (10.1.10.65)  32.670 ms  32.667 ms  32.831 ms
 9  185.245.80.44 (185.245.80.44)  32.607 ms  32.605 ms  32.602 ms
10  185.245.80.45 (185.245.80.45)  37.011 ms  37.008 ms  37.007 ms
11  5.180.211.133 (5.180.211.133)  31.788 ms  31.785 ms  31.497 ms
```

Abbiamo avviato i test con un valore iniziale di TTL pari a 4, poiché i primi nodi della rete sono dedicati all'uscita dal modem e quindi non ha senso testarli. Successivamente, abbiamo confrontato i risultati ottenuti utilizzando lo strumento di tracciamento del percorso `traceroute`. In entrambi i test, abbiamo osservato che il numero di link attraversati era di 11.

Tuttavia, è importante notare che il numero di link attraversati ottenuto tramite il decremento del TTL o mediante l'utilizzo di `traceroute` non include il percorso di ritorno. Pertanto, per ottenere il numero totale di link n attraversati durante il percorso di andata e ritorno, dobbiamo considerare il doppio del valore ottenuto tramite TTL o `traceroute`.

3 Andamento di RTT

Per valutare l'andamento dell'RTT minimo, medio, massimo e della deviazione standard in funzione della dimensione del pacchetto abbiamo creato uno script bash "ping_rtt.sh". Questo script automatizza il processo di esecuzione del comando `ping` con diverse dimensioni del pacchetto e raccoglie i valori di RTT corrispondenti.

Lo script utilizza un ciclo `for` per iterare attraverso una serie di dimensioni del pacchetto. Per ogni dimensione del pacchetto, esegue il comando ping specificando il numero di pacchetti da inviare e la dimensione del payload del pacchetto.
Successivamente, estrae l'ultima linea di output del comando ping, che contiene i valori di RTT minimo, medio, massimo e deviazione standard. Questi valori vengono salvati in un file di testo in un formato conveniente per l'acquisizione dei dati da parte di MATLAB ®.

Nel test, abbiamo scelto di inviare per ogni dimensione del pacchetto $K = 100$ pacchetti per avere una valutazione migliore delle statistiche dell' RTT. La dimensione dei pacchetti è variabile da 10 a 1472 byte con passi di 1 byte. Questa scelta ci consente di ottenere una rappresentazione dettagliata dell'andamento dell'RTT in base alla dimensione del pacchetto.

Si è scritto un programma MATLAB ® che legge i dati dal file e genera un semplice grafico dei valori di RTT in funzione della dimensione del pacchetto. Abbiamo riportato in Figura 1 i risultati.

4 Stima di *R* e *R-bottleneck*

Sappiamo che generalmente l'RTT aumenta con la lunghezza dei pacchetti, nella pratica questa regolazione può essere influenzata dalla variabilità del ritardo di accodamento. Considerando il valore minimo dell'RTT possiamo mitigare l'effetto dei ritardi di accodamento, misurato su una serie di trasmissioni con pacchetti di dimensione costante. Infatti si ipotizza che prima o poi il pacchetto possa trovare tutte le code vuote ad ogni nodo. In formule, assumeremo che, eseguendo un numero K sufficientemente grande di volte il `ping` con L costante, si abbia:

$$RTT_{\min}(L) \approx aL + T \tag{1}$$

In effetti, come osservato dalla Figura 2 è evidente che l'andamento del minimo dell'RTT per ogni valore di L è stato ben approssimato linearmente da una retta. La linearità dell'andamento del minimo dell'RTT implica che all'aumentare della dimensione dei pacchetti, il tempo di trasmissione minimo aumenta proporzionalmente.

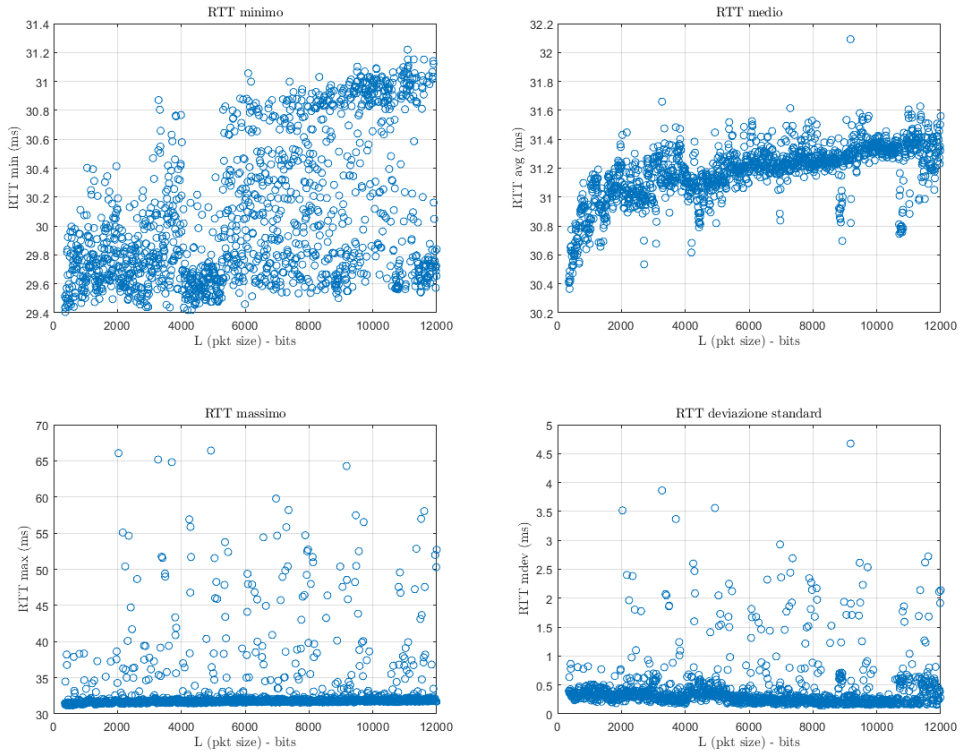

Figure 1: Statistiche dell'RTT in funzione della dimensione del pacchetto

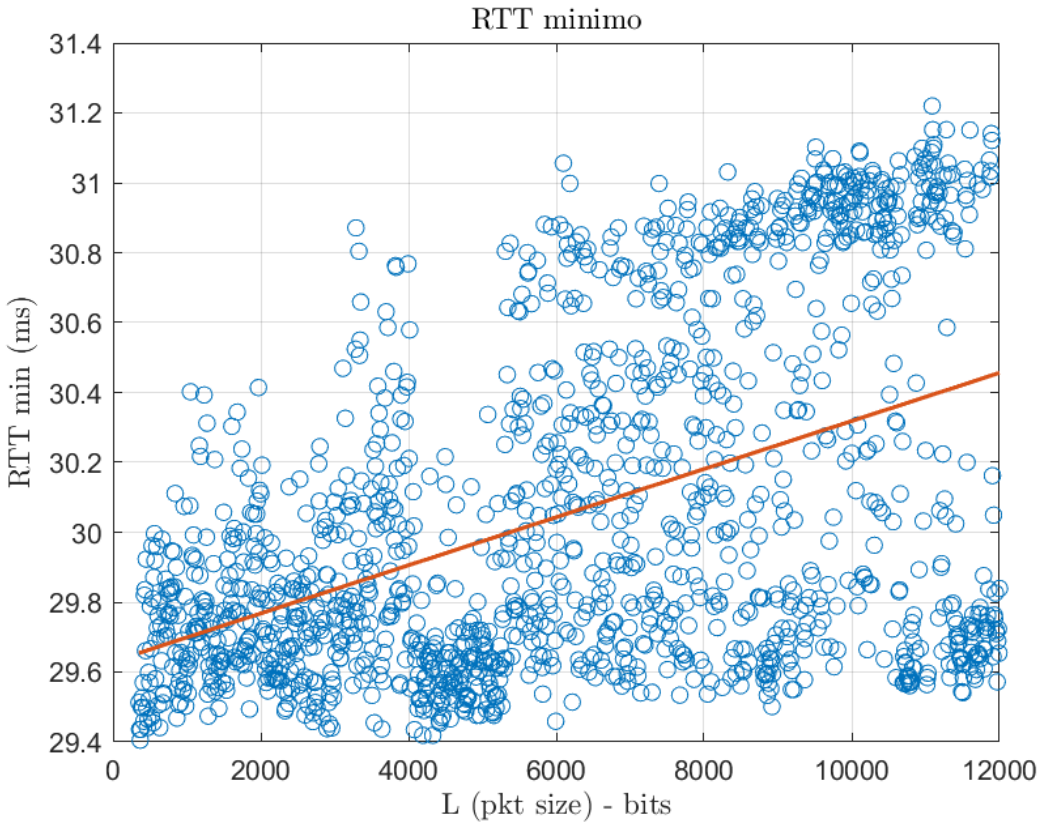

Figure 2: Statistiche dell'RTT minimo in funzione della dimensione del pacchetto con curva di andamento

Stimando la pendenza della retta che approssima l'andamento del minimo dell'RTT in funzione delle dimensioni dei pacchetti, possiamo ottenere una stima del coefficiente a. Questo coefficiente ci fornisce informazioni utili per valutare il throughput della rete in due diverse ipotesi:

- Se tutti i link (andata e ritorno) hanno throughput uguali ad un certo valore R, si ottiene

$$R = \frac{n}{a} \tag{2}$$

- Se invece esiste un link con un throughput molto minore di tutti gli altri (detto *bottleneck*), e supponendo che tale throughput sia lo stesso all'andata ed al ritorno si ha:

$$R_{\text{bottleneck}} \approx \frac{2}{a} \tag{3}$$

È importante sottolineare che queste stime sono basate sulle ipotesi semplificate sull'omogeneità dei link e sulla simmetria del throughput nel caso del bottleneck. Sappiamo dalle stime fatte al punto 2 che il server `"lon.speedtest.clouvider.net"` ha $n = 11 * 2 = 22$ pertanto otteniamo:

$$a = 6.8818 \cdot 10^{-5} \quad R = 319.6849 \text{ kbps} \quad R_{\text{bottleneck}} = 29.0623 \text{ kbps}$$

5 Conclusioni

Attraverso l'utilizzo dell' applicazione `ping` e l'analisi dei valori di RTT siamo stati in grado di apprezzare l'andamento dell'RTT in relazione alla dimensione dei pacchetti, riusciendo a mitigare l'impatto dei ritardi di accodamento sulla misurazione.

Abbiamo osservato che, sebbene l'RTT tendenzialmente cresca con la lunghezza dei pacchetti, questa relazione può essere influenzata dalla variabilità del ritardo di accodamento. Tuttavia, abbiamo dimostrato che considerando il valore minimo di RTT misurato su una serie di trasmissioni con pacchetti di dimensione costante, possiamo mitigare l'impatto dei ritardi di accodamento.

Abbiamo ripetuto l'esperimento anche con altri server: ottenendo prestazioni peggiori con server più lontani geograficamente.