Relazione Lab Ping Pong

Aionei Giorgia - 4942137

Caviglia Lisa - 4993924

Gastaldi Sebastiano - 2916642

Grosso Francesco - 4114387

Peres Andrea - 4817585

Introduzione

Il laboratorio ping-pong si basa sullo sviluppo di una comunicazione client-server tramite l'utilizzo dei socket, che saranno di tipo STREAM o DGRAM a seconda della scelta del protocollo da utilizzare (TCP o UDP).

- TCP: protocollo di tipo connesso, utilizziamo il socket STREAM
- UDP: protocollo di tipo connectionless, utilizziamo il socket DGRAM

In questo laboratorio chiameremo "ping" i due client (TCP o UDP) e "pong il server".

TCP_PING.C

Descrizione generale:

Il programma invia un messaggio al server "pong" e attende una risposta, misurando il tempo che intercorre tra l'invio del messaggio e la ricezione della risposta.

Il tcp_ping è strutturato in due parti principali: le funzioni `do_ping` e `main`.

La funzione `main` si occupa di configurare il socket TCP, connettersi al server, inviare la richiesta al server e gestire la risposta. Inoltre, chiama la funzione `do_ping` per un numero specificato di volte e registra gli RTT.

La funzione `do_ping` invia un messaggio al server e attende una risposta. Misura il tempo prima dell'invio del messaggio e dopo la ricezione della risposta, calcolando così

il Round Trip Time (RTT), ovvero il tempo che intercorre tra l'invio del messaggio e la ricezione della risposta.

Problematiche

Non è stata riscontrata nessuna particolare problematica.

UDP_PING.C

Descrizione generale:

Il programma svolge la stessa funzione del tcp_ping ma utilizzando un socket UDP.

Il programma è strutturato in tre parti principali: le funzioni `do_ping`, `prepare_udp_socket` e `main`.

La funzione `main` si occupa di gestire i parametri di input, configura un socket TCP per comunicare con il server, inviare la richiesta al server e gestire la risposta. Inoltre, chiama la funzione `do_ping` per un numero specificato di volte e registra i tempi di RTT.

La funzione `do_ping` invia un messaggio al server e attende una risposta. Misura il tempo prima dell'invio del messaggio e dopo la ricezione della risposta, calcolando così il Round Trip Time (RTT), ovvero il tempo che intercorre tra l'invio del messaggio e la ricezione della risposta.

La funzione `prepare_udp_socket` si occupa di configurare il socket UDP, connettersi al server e gestire eventuali errori.

Problematiche

Inizialmente abbiamo riscontrato problemi con la risposta del server, ci aspettavamo un “OK”, non consapevoli del fatto che che la risposta fosse più lunga, abbiamo risolto utilizzando la funzione strncmp() per limitare la lettura del buffer answer solamente ai primi due caratteri, risolto questo il programma funzionava senza problemi.

L’eseguibile di esempio andava in segmentation fault, quindi non avevamo un riferimento per sapere cosa aspettarci.

Abbiamo capito solo in seguito che l’utilizzo del time-out non dovesse essere gestito da noi.

Per la readwrite non c’erano istruzioni su come usarla.

Già da questo file abbiamo riscontrato problematiche con la connessione al server seti.dibris.unige.it

SERVER_PONG.C

Descrizione generale:

Il server pong ascolta le connessioni in arrivo su una porta specificata dall'utente.

Il `main` inizializza il socket e lo mette in ascolto per le connessioni in arrivo, poi chiama la funzione server_loop.

La funzione `server_loop` gestisce le connessioni in arrivo sul socket del server accettandole e svolgendo delle fork in modo da delegare al processo figlio il compito di occuparsi del singolo client e lasciando al padre la possibilità di rimanere in ascolto di eventuali connessioni da parte dei client in arrivo.

Problematiche

Abbiamo avuto un problema nella funzione `server_loop`, perchè quando eseguivamo il file `pong_server.c` ci generava un errore di mancata connessione del server pong, questo perchè il `request_socket` ci restituiva un valore di -1. Abbiamo risolto introducendo una condizione in cui se l’errore era di tipo `EINTR` (si verifica quando una system call viene interrotta da un segnale prima che possa completarsi), allora il programma doveva rientrare nel loop per ritentare la system call.

Confronto throughput con modello Banda-Latenza

In questa sezione della relazione, analizzeremo i dati raccolti durante il laboratorio per valutare la banda e la latenza delle comunicazioni tra i client "ping" e il server "pong" tramite i protocolli TCP e UDP e metterli a confronto con il throughput effettivo.

Abbiamo generato dei grafici, mediante uno script che utilizza l'applicazione GNUPLOT, utilizzando i dati raccolti nei file tcp.throughput.dat e udp.throughput.dat per valutare se i risultati ottenuti corrispondono alle nostre aspettative.

Nell'analisi dei grafici, ci siamo concentrati su due aspetti principali: il throughput e il rapporto tra banda e latenza.

1. Il throughput rappresenta la quantità di dati trasmessi tra il client e il server in un'unità di tempo. Abbiamo calcolato il throughput per le comunicazioni TCP e UDP, e mediante uno script fornitoci, vengono prodotti i due file di cui abbiamo accennato precedentemente, e tramite essi viene creato il grafico che mostra come varia il throughput.
2. Il modello Banda Latenza considera due aspetti principali:
    - La Banda rappresenta la capacità della rete di trasmettere dati in una certa unità di tempo: una Banda più elevata significa una maggiore capacità di trasferimento dati.
    - La Latenza rappresenta il ritardo o il tempo necessario per un messaggio per percorrere la rete. Una latenza più bassa indica una comunicazione più rapida e responsiva.

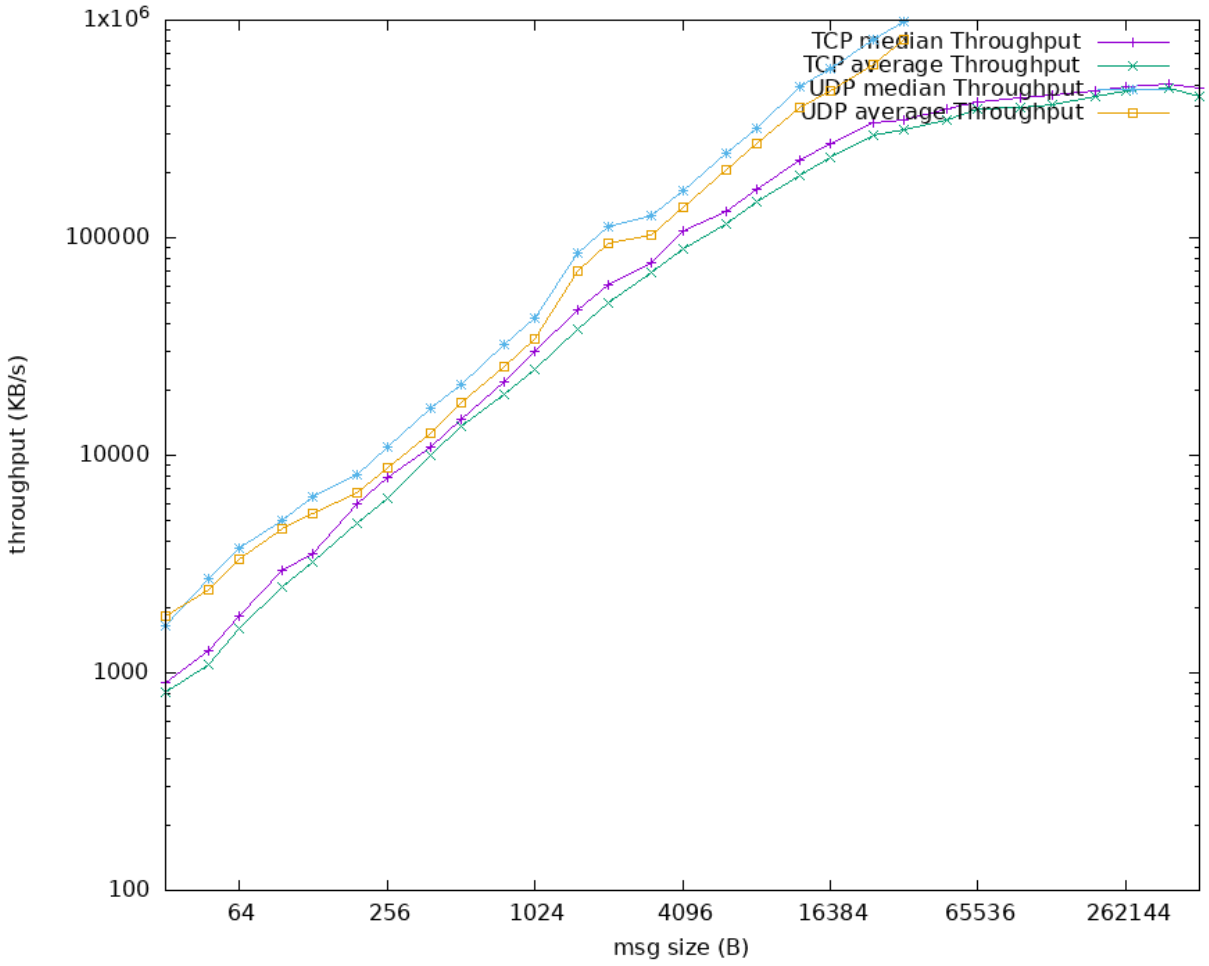
Riassumendo:

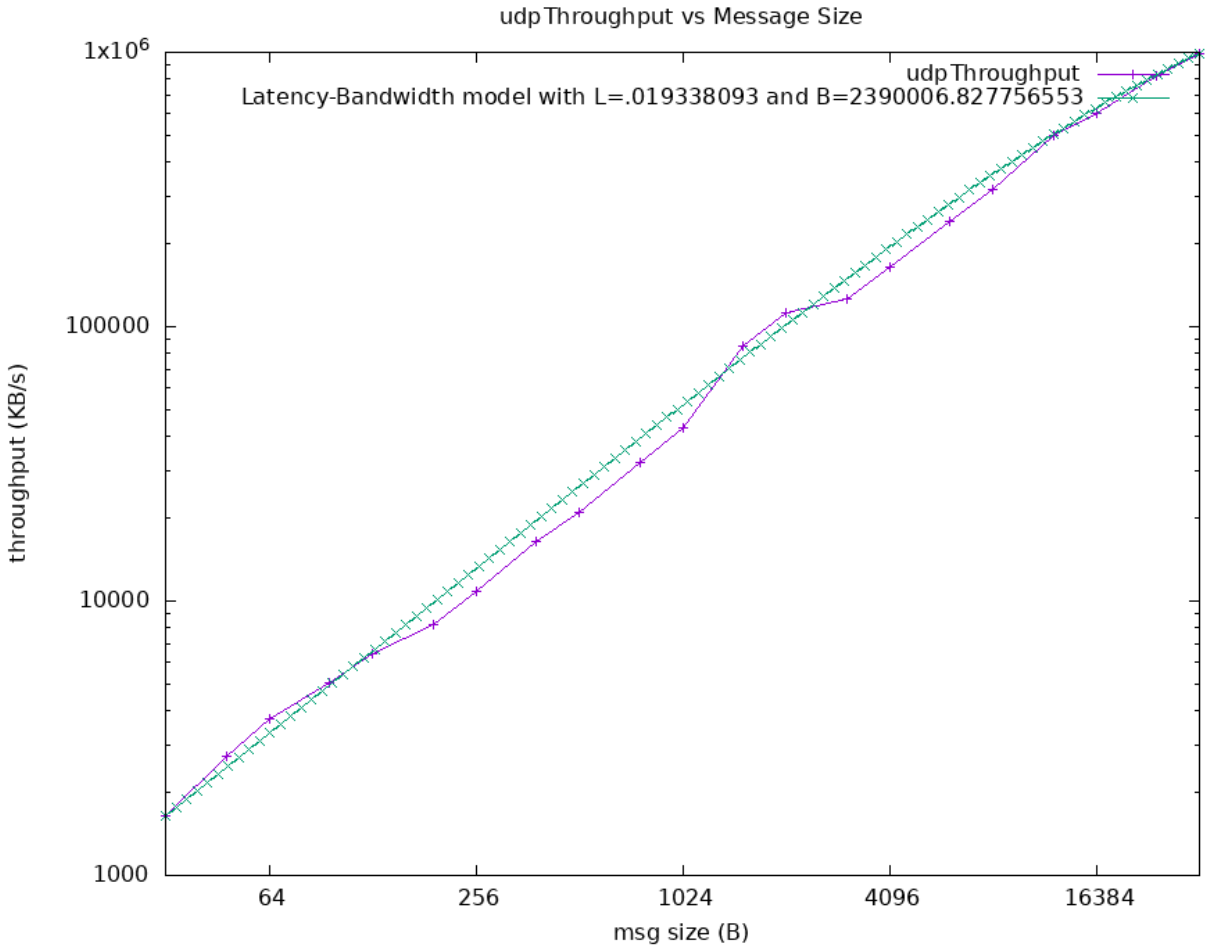
La latenza è particolarmente importante nel contesto del modello Banda/Latenza poiché influisce sulla reattività delle comunicazioni.

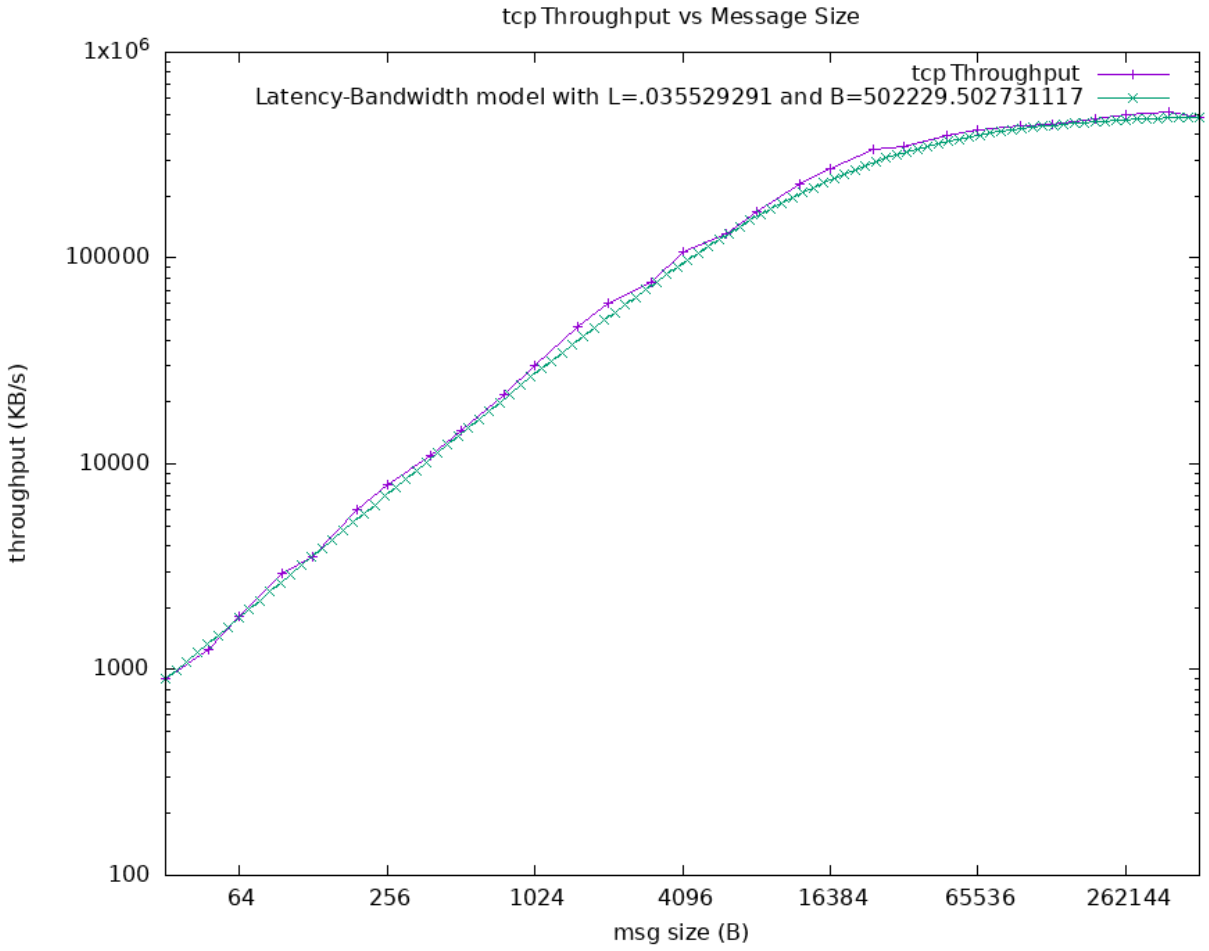
La sua riduzione contribuisce a migliorare la qualità e l'efficienza delle comunicazioni, mentre la banda influisce sulla quantità di dati che possono essere trasmessi in un determinato periodo di tempo.

Ora, analizzando i dati raccolti ci aspettiamo che le comunicazioni TCP abbiano una latenza più costante rispetto alle comunicazioni UDP, mentre le comunicazioni UDP potrebbero mostrare una maggiore variabilità nella latenza a causa della mancanza di connessione stabile.

Se ci sono discordanze significative, potremmo dover esaminare ulteriormente i nostri codici o considerare altre variabili che potrebbero influire sulla banda e sulla latenza.


1x10^6
100000
10000
1000
100
throughput (KB/s)
64
256
1024
4096
16384
65536
262144
msg size (B)
TCP median Throughput
TCP average Throughput
UDP median Throughput
UDP average Throughput


udpThroughput vs Message Size
udpThroughput
Latency-Bandwidth model with L=.019338093 and B=2390006.827756553
1x10^6
100000
10000
1000
throughput (KB/s)
64
256
1024
4096
16384
msg size (B)



Visto che i grafici ottenuti corrispondono alle aspettative, possiamo concludere che le nostre implementazioni dei client "ping" e del server "pong" funzionano come previsto.

Problematiche

È importante notare che, a causa delle problematiche incontrate con la connessione al server seti.dibris.unige.it, tutti i test sono stati svolti in locale, il che potrebbe influenzare i risultati e le aspettative.

Infatti per quanto riguarda la curva del modello banda-latenza per il protocollo udp assume un andamento quasi lineare, avendo la latenza praticamente costante.

Pertanto, i risultati ottenuti in questa relazione dovrebbero essere considerati in un contesto di laboratorio locale.