Componenti gruppo:
- Merengone Simone, 4984409
- Tomasella Simone, 5311626
- Calafiore Marco, 5342305
- Minotti Eleonora, 5293542

Relazione SETI Laboratorio ping-pong

TCP_PING

Le difficoltà riscontrate nella realizzazione di questo laboratorio sono state:

- Inizialmente utilizzavamo la funzione int timespec_get(struct timespec *ts, int base), per memorizzare l'ora corrente in send_time.

Però ha iniziato a darci problemi quando provavamo TCP con il server_pong, allora abbiamo cercato una funzione alternativa per svolgere lo stesso compito.
Così l'abbiamo sostituita con clock_gettime(CLOCK_TYPE, &send_time) che ci ha risolto i problemi dati da timespec_get.

- Successivamente ci siamo accorti che per segnalare i problemi dovevamo utlizzare le funzioni della libreria fail ed allora abbiamo inserito e modificato tutti i controlli sulle system call e le funzioni, come ad esempio la stessa clock_gettime e le varie chiamate necessarie ad implementare tcp, come ad esempio: socket e connect.

UDP_PING

Le difficoltà riscontrate nella realizzazione di questo laboratorio sono state:

- La principale difficoltà è stata un'incomprensione con la seguente istruzione:

```
/*** Receive answer through the socket (non blocking mode, with timeout) ***/
```

Avevamo capito che dovevamo creare noi un timer di lunghezza time-out entro la quale la funzione receive doveva ricevere dei dati e non doveva dare errore, altrimenti terminavamo il programma.
Poi ci siamo accorti che era inutile perché ciò veniva svolto nel while successivo controllando i tempi registrati precedentemente con:

```
while ( recv_bytes < 0 && (recv_errno == EAGAIN || recv_errno == EWOULDBLOCK)
        && roundtrip_time_ms < timeout) {
```

Nonostante ciò, non siamo riusciti a replicare il comportamento dei retry che fa a livello utente (circa una pausa di un secondo tra un try e l'altro)

PONG_SERVER

Le difficoltà riscontrate nella realizzazione di questo laboratorio sono state:

- Riuscire a comunicare con UDP_ping a causa di una malcomprensione del timeout di UDP che ci causava problemi(citata precedentemente).

- un’errore all’interno della funzione open_udp_socket che dopo la bind non tornavamo l’udp e di seguito non veniva aggiornata pong_port

SCRIPT BASH

Le difficoltà riscontrate nella realizzazione di questo laboratorio sono state:

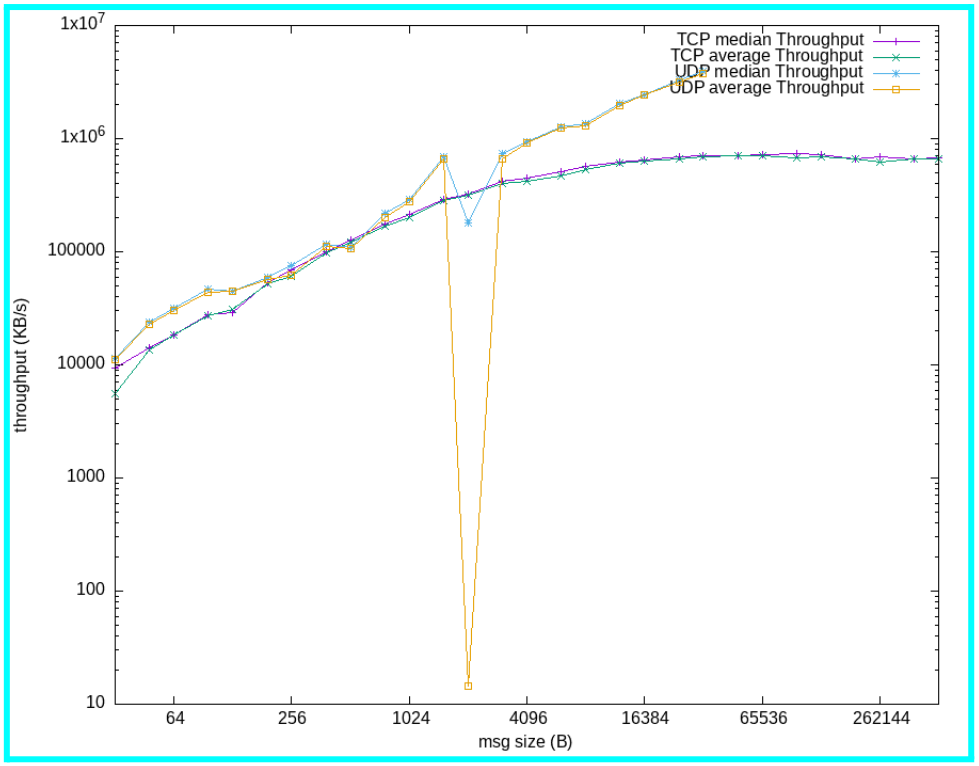
- La conoscenza molto limitata dell’argomento, che ci ha causato inizialmente un tale rallentamento nella scrittura del codice.
- Per qualche motivo la libreria gnuplot non veniva installata sul laptop con virtual machine, non permettendoci quindi di poter creare i grafici su quella macchina.
- Inizialmente è nata della confusione quando abbiamo generato la cartella “data” perchè i valori calcolati da due laptop differenti (usando lo stesso codice) erano notevolmente differenti.

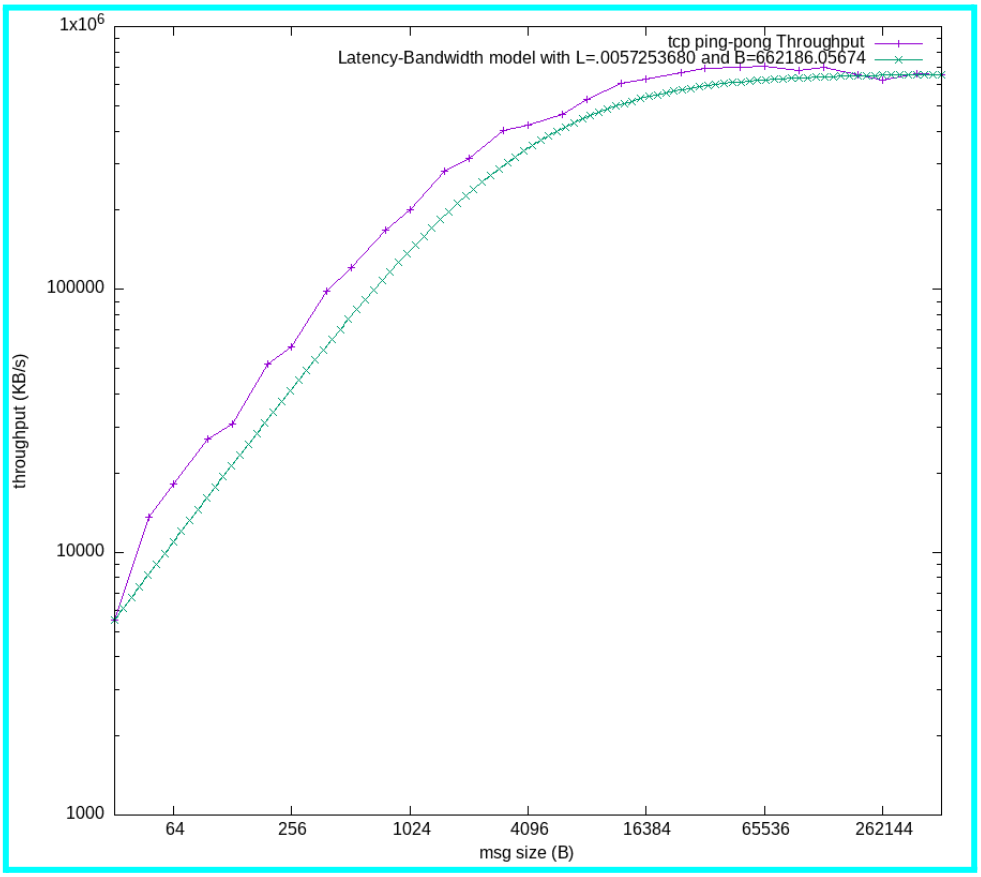
Poi si è scoperta la risposta: uno utilizzava una virtual Machine e l’altro utilizzava Manjaro Linux nativo (basato su Arch Linux), quest’ultimo ovviamente più prestante rispetto ad una macchina virtuale.
A causa della mancanza di gnuplot sulla macchina con virtual machine, abbiamo potuto generare i grafici solo sulla macchina con Manjaro Linux.

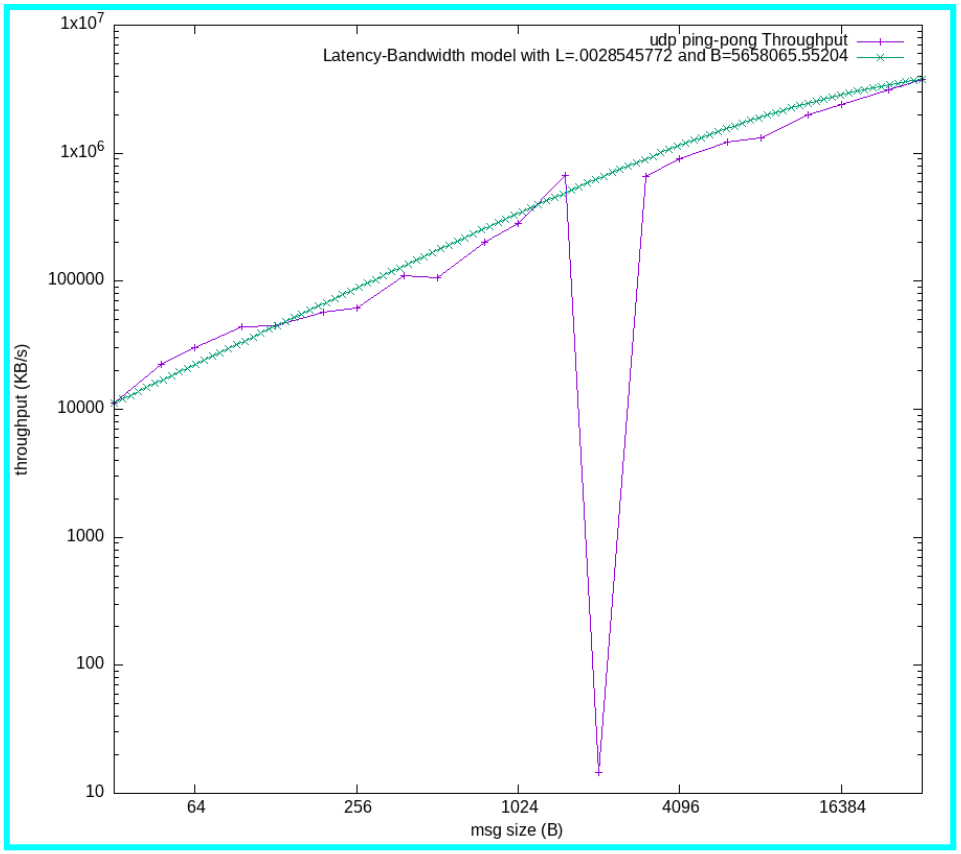
- A causa di valori molto alti dati dalla macchina con Manjaro Linux, abbiamo avuto dei problemi con la conversione dei valori con l’esponenziale per renderli compatibili con il processo “bc”. Abbiamo poi sistemato tale problema andando a calcolare preventivamente il valore con il suo esponenziale (esempio: 3.78998e+06 => 3789980)
- Purtroppo non siamo riusciti a risolvere il problema del picco verticale all’interno dei grafici di Throughput medio e del modello banda latenza di UDP causato dall’MTU (Maximum Transmission Unit), ovvero a 1460 bytes.
- Inoltre non siamo riusciti ad aumentare il range dell’msg_size di UDP.

Nelle pagine successive ci saranno i grafici calcolati con lo script:

1. **Throughput mediano e medio di TCP ed UDP**
2. **Modello Banda Latenza TCP**
3. **Modello Banda Latenza UDP**


TCP median Throughput
TCP average Throughput
UDP median Throughput
UDP average Throughput
throughput (KB/s)
msg size (B)
1x10^7
1x10^6
100000
10000
1000
100
10
64
256
1024
4096
16384
65536
262144


tcp ping-pong Throughput
Latency-Bandwidth model with L=.0057253680 and B=662186.05674
throughput (KB/s)
msg size (B)
1x10^6
100000
10000
1000
64
256
1024
4096
16384
65536
262144


udp ping-pong Throughput
Latency-Bandwidth model with L=.0028545772 and B=5658065.55204
throughput (KB/s)
msg size (B)
1x10^7
1x10^6
100000
10000
1000
100
10
64
256
1024
4096
16384