

# Relazione PingPong

## Componenti:

- Samuele Barrago
- Daniele Sacco
- Lorenzo Migliore

Il **Pong Server** è un'applicazione che risponde ai messaggi inviati dai client **Ping**, sia tramite protocollo **TCP** che **UDP**.

Il server è progettato per accettare **connessioni multiple** da parte dei client Ping e rispondere alle richieste in modo efficiente.

## Errori riscontrati e soluzioni adottate

Durante lo sviluppo del Pong Server, abbiamo riscontrato e risolto diversi problemi. Di seguito riportiamo i principali con le relative soluzioni:

- **Invio dei pacchetti**
  - **Problema:** L'invio dei messaggi attraverso il socket in modalità bloccante non riusciva a inviare tutti i dati.
  - **Soluzione:** Abbiamo sostituito la funzione send con una funzione personalizzata chiamata blocking\_write\_all.
- **Scrittura del numero del messaggio nel buffer**
  - **Problema:** La scrittura del numero del messaggio (msg\_no) all'inizio del buffer non funzionava correttamente.
  - **Soluzione:** Abbiamo utilizzato la funzione sprintf per formattare correttamente la stringa.
- **Ricezione pacchetti dal socket**

- **Problema:** La ricezione delle risposte in modalità non bloccante con timeout presentava problemi.
- **Soluzione:** Abbiamo implementato un ciclo for per ricevere i dati in più passaggi, anziché in un'unica chiamata.
- **Segmentation fault alla ricezione di tutti i pacchetti**
  - **Problema:** Dopo aver ricevuto tutte le risposte, il programma sollevava un segmentation fault.
  - **Soluzione:** L'errore era legato alla gestione degli argomenti nella funzione timespec\_delta2milliseconds.

## Debug

Durante lo sviluppo del server Pong, abbiamo adottato un approccio graduale al **debug**.

Inizialmente abbiamo testato il server con il tool `gc_*ping`, successivamente con i nostri programmi ping.

Abbiamo inoltre utilizzato la modalità **debug del Makefile** per individuare e correggere gli errori. Questo approccio ci ha permesso di migliorare il server in modo efficace e incrementale.

## Utilizzo

Il Pong Server deve essere eseguito specificando la **porta** su cui deve rimanere in ascolto.

Esempio di esecuzione:

```
./pong_server 1491
```

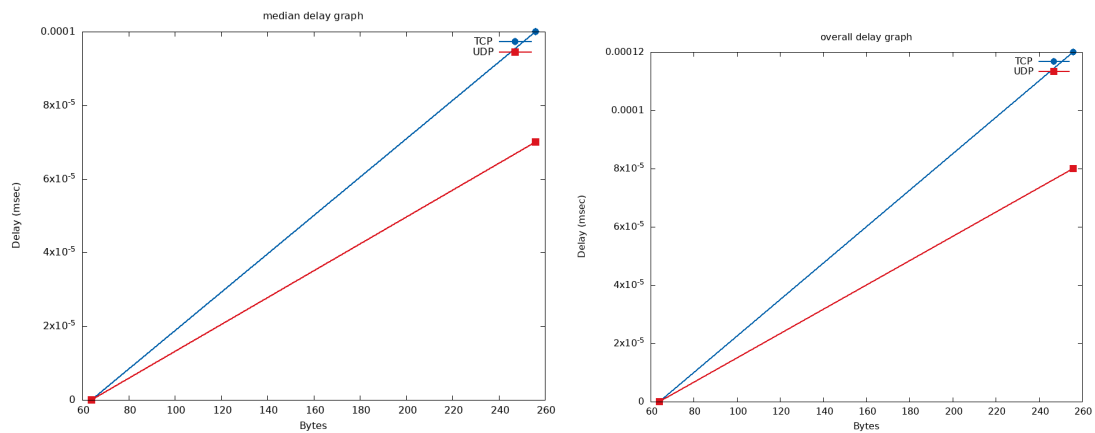
Il server rimarrà in ascolto sulla porta indicata, accetterà le connessioni dei client Ping e risponderà alle loro richieste.

## Grafici: delay in funzione dei byte ricevuti

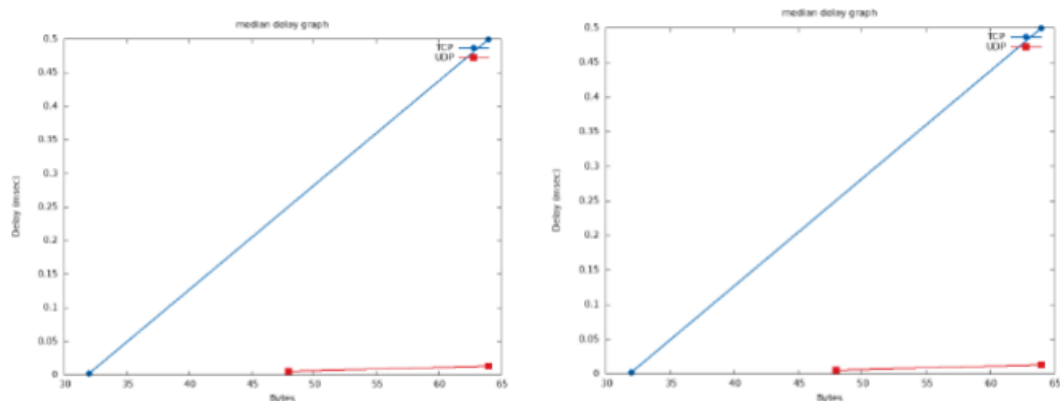
I grafici sono stati generati usando **Gnuplot**, tramite uno script bash che calcola il delay dei pacchetti ricevuti.

- I **primi due grafici** mostrano i risultati su localhost con pacchetti da **32 e 64 byte**.

Si nota che il **delay TCP** è leggermente maggiore, ma in generale il tempo è molto basso.



- I **grafici successivi** sono stati ottenuti pingando il server **seti.dibris.unige.it**.  
Si nota un delay molto più elevato, dovuto alla distanza e latenza di rete.



Un problema che **non siamo riusciti a risolvere** riguarda l'invio di pacchetti **UDP da 32 byte**, che venivano rifiutati dal server.

Gli **ultimi grafici** rappresentano pacchetti da **64 e 256 byte** in localhost.

Anche in questo caso, si conferma che **TCP ha un delay maggiore** rispetto a UDP.

