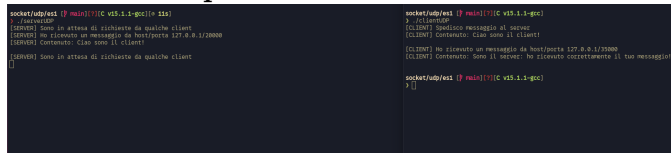


# Esercizi 1–7 - Client/Server UDP

## Esercizi basati su clientUDP e serverUDP

1. Lanciare prima il server e poi il client. Cosa si osserva? Invertire la sequenza di lancio. Cosa si osserva?

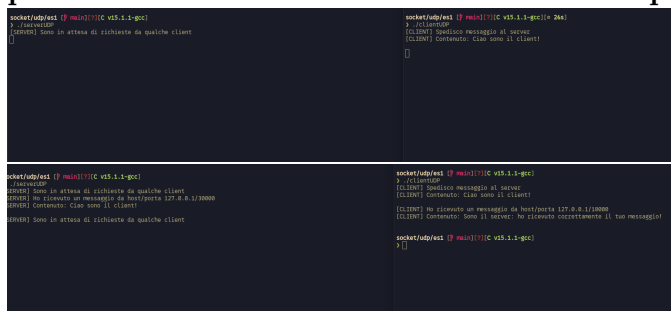


```
socket/udp/est [?] main() [?] [C v5.1.1-gcc]
1. createUDPInterface
[SERVER] Sono in attesa di richieste da qualche client
[CLIENT] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!
[CLIENT] Sono in attesa di richieste da qualche client

socket/udp/est [?] main() [?] [C v5.1.1-gcc]
1. createUDPInterface
[CLIENT] Spedisco messaggio al server
[SERVER] Contenuto: Ciao sono il client!
[CLIENT] Ho ricevuto un messaggio da host/porta 127.0.0.1/10000
[SERVER] Contenuto: Sono il server! Ho ricevuto correttamente il tuo messaggio!
socket/udp/est [?] main() [?] [C v5.1.1-gcc]
> |
```

Il server, appena avviato, resta in ascolto. Quando si avvia il client, questo invia un messaggio al server e rimane in attesa di una risposta; dopo averla ricevuta, il client termina mentre il server rimane attivo per ulteriori richieste. Invertendo l'ordine, il client invia un messaggio a un server non in ascolto (messaggio perso) e poi resta in attesa senza ricevere nulla, mentre il server avviato successivamente rimane in ascolto senza aver ricevuto richieste.

2. Modificare i sorgenti per mettere il server che riceve sulla porta 10000 e il client che trasmette dalla propria porta 30000.



```
socket/udp/est [?] main() [?] [C v5.1.1-gcc]
1. createUDPInterface
[SERVER] Sono in attesa di richieste da qualche client
[CLIENT] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!
[CLIENT] Sono in attesa di richieste da qualche client

socket/udp/est [?] main() [?] [C v5.1.1-gcc]
1. createUDPInterface
[CLIENT] Spedisco messaggio al server
[SERVER] Contenuto: Ciao sono il client!
[CLIENT] Ho ricevuto un messaggio da host/porta 127.0.0.1/10000
[SERVER] Contenuto: Sono il server! Ho ricevuto correttamente il tuo messaggio!
socket/udp/est [?] main() [?] [C v5.1.1-gcc]
> |
```

Bisogna cambiare il parametro della porta nella chiamata `createUDPInterface(10000)` sul server e `UDPSend(..., 30000)` sul client. Nella versione errata (prima immagine) la porta non è stata aggiornata; nella versione corretta (seconda immagine) la comunicazione avviene regolarmente.

3. Mettere il server in ascolto sulla porta 100 e osservare cosa succede.

```
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
[CLINT] Sono in attesa di richieste da qualche client
[SERVER] No ricevuto un messaggio da host/porta 127.0.0.1/7000
[SERVER] Contenuto: Ciao sono il client
[SERVER] Sono in attesa di richieste da qualche client
}
```

Dopo aver modificato la funzione di inizializzazione per usare la porta 100, il server non si avvia se eseguito senza privilegi. Le porte inferiori a 1024 sono riservate e richiedono permessi di root (sudo).

4. Sostituire “127.0.0.1” prima con “localhost” e poi con “pippo” e osservare cosa succede.

```
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
[CLINT] Sono in attesa di richieste da qualche client
[SERVER] No ricevuto un messaggio da host/porta 127.0.0.1/7000
[SERVER] Contenuto: Ciao sono il client
[SERVER] Sono in attesa di richieste da qualche client
}
```

Con “localhost” la risoluzione DNS restituisce 127.0.0.1 e la comunicazione funziona. Con “pippo” non esiste alcun nome host corrispondente, quindi la chiamata fallisce.

5. Accordarsi per lavorare su coppie di macchine in modo che server e client siano su macchine diverse. Come bisogna modificare i sorgenti?

Bisogna trovare l’indirizzo IP della macchina che ospita il server e sostituire “127.0.0.1” nella chiamata UDPSend(..., serverIP, ...) con tale indirizzo.

6. Lanciare due volte il server usando due terminali. Cosa si osserva? Funzionano entrambi?

```
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
socket/udp/est { main() { IC v45.1.1-gcc } > wdsb; }
[CLINT] Sono in attesa di richieste da qualche client
[SERVER] No ricevuto un messaggio da host/porta 127.0.0.1/7000
[SERVER] Contenuto: Ciao sono il client
[SERVER] Sono in attesa di richieste da qualche client
}
```

Entrambe le istanze del server si avviano, ma solo una riceve il messaggio dal client. Le due istanze competono per la stessa porta UDP e il sistema operativo consegna il pacchetto a una sola di esse.

7. Modificare il server in maniera che soddisfi 5 richieste prima di terminare. E se volessi che non terminasse mai?

```
socket/udp/es1 [?] main[?][C v15.1.1-gcc]
> sudo ./serverUDP
[SERVER] Sono in attesa di richieste da qualche client
[SERVER] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!

[SERVER] Sono in attesa di richieste da qualche client
[SERVER] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!

[SERVER] Sono in attesa di richieste da qualche client
[SERVER] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!

[SERVER] Sono in attesa di richieste da qualche client
[SERVER] Ho ricevuto un messaggio da host/porta 127.0.0.1/30000
[SERVER] Contenuto: Ciao sono il client!
```

Nel sorgente fornito il server è già scritto con un ciclo infinito e quindi non termina mai. Per limitarlo a 5 richieste si potrebbe introdurre un contatore e uscire dal loop dopo 5 iterazioni.