Relazione Progetto LabReti a.a 2022-2023

`Studente: Flavio Romano, Matricola: 614801`

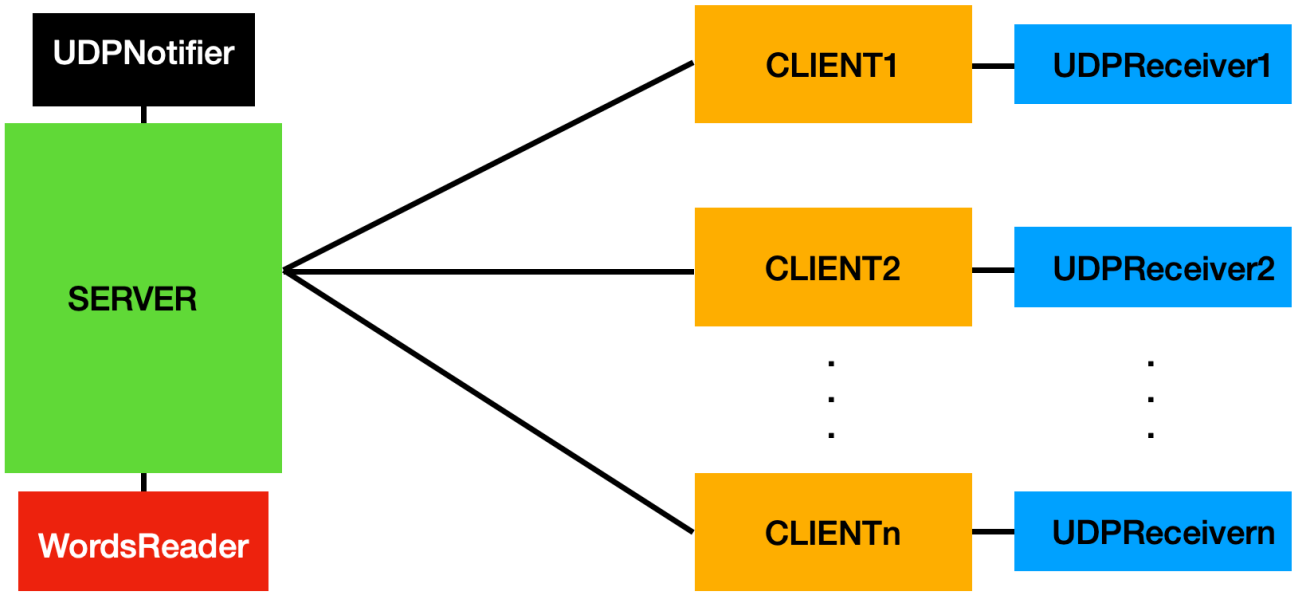
Istruzioni per l'avvio

- Per compilare il file ServerMain.java: `javac -cp :./Server/gson-2.10.jar -d Out ./Server/*.java`
  - Per eseguire il ServerMain.class: `java -cp :./Server/gson-2.10.jar:Out Server.ServerMain`
- Per compilare il file ClientMain.java: `javac -d Out ./Client/*.java`
  - Per eseguire il ClientMain.class: `java -cp Out Client.ClientMain`

Note per il client: il logout dell'utente viene eseguito correttamente anche a seguito dell'interruzione del processo con un `SIGINT` (^C sulla tastiera).

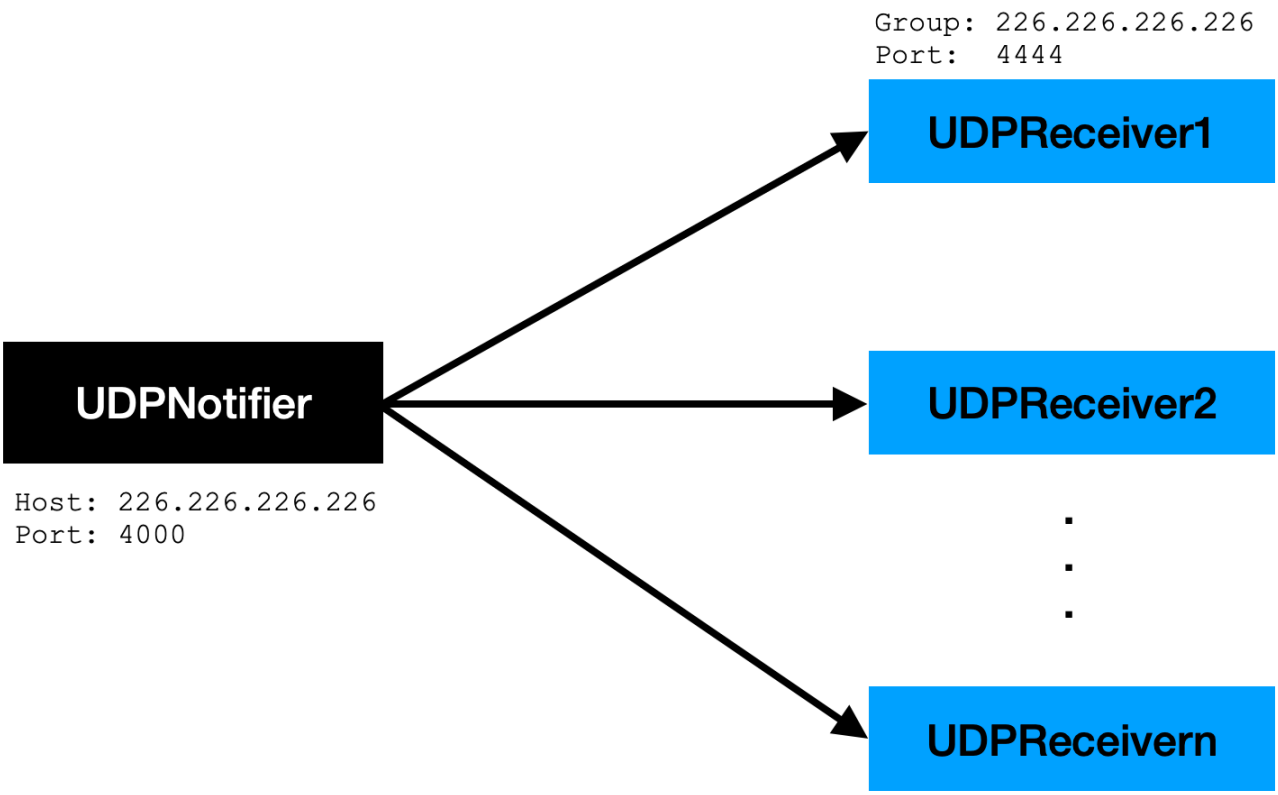
Scelte implementative

La connessione client-server è gestita tramite Stream Sockets attraverso cui i messaggi vengono scambiati.



Server

- Memorizza gli oggetti User come lista all'interno di un json chiamato `WordleUsers.json`, che viene aggiornato mediante il metodo della classe `Gson` chiamato `fromJson` (che prende come parametri un `FileReader` e il tipo per la serializzazione tramite reflaction).
- Memorizza i post condivisi dagli utenti in un file txt chiamato `Social.txt`.
- I parametri di configurazione sono memorizzati in un apposito file chiamato `configServer.txt` letto dalla classe `ServerSetup` tramite `BufferedReader`.
- Viene estratta una nuova parola segreta ogni 30 secondi (parametro `wordTTL` nel file di configurazione del server), è un tempo non realistico ma utilizzato per facilitare il debugging.
- Un pool di thread gestisce le connessioni con i client tramite `ServerSocket`, ad ogni thread viene assegnato un solo client.
- I client che hanno effettuato l'accesso si uniscono ad un gruppo di multicast a cui l'`UDPNotifier` del server (unico per tutto il programma) invia una notifica nel momento in cui un utente condivide una partita.
- Ogni thread utilizza l'inputStream del socket per ricevere i messaggi del client e invia messaggi sull'outputStream del socket.



Client

- I parametri di configurazione sono memorizzati in un apposito file chiamato `configClient.txt` letto dalla classe `ClientSetup` tramite `BufferedReader`.
- Ogni client ha un `UDPReceiver` che si unisce al gruppo Multicast `226.226.226.226` dal login dell'utente fino alla disconnessione. Esso sta perennemente in ascolto, in attesa di ricevere notifiche dal server riguardo la condivisione di una partita da parte di un giocatore.
- Ogni client utilizza l'inputStream del socket per ricevere i messaggi dal server e invia messaggi sull'outputStream del socket.

Varie ed eventuali

- Come statistiche ho scelto di tenere:

```
- Distribuzione dei tentativi necessari per indovinare una parola.
- numero di partite.
- Percentuale di vittoria.
- Serie di vittorie recente.
- Serie di vittorie assoluta.
```

- Un utente non può giocare due volte alla stessa parola, per questo motivo nel Json ogni utente ha un campo `playedWords` che memorizza le parole già giocate.
- Quando un utente condivide una partita sul social, viene condivisa una versione senza spoiler per mostrare agli altri come è arrivato alla soluzione e gli hint che ha ottenuto.
- Alla riga 73 del file `ServerMain.java` ho inserito un `println(secretWord)` per facilitare il test del programma, basta commentarla per non fare stampare la parola segreta corrente al server.
- All'inizio del metodo `main` di `ServerMain.java` e `ClientMain.java` ho inserito il comando `System.setProperty("java.net.preferIPv4Stack", "true");` perché utilizzo Mac M1 e ho avuto problemi nell'utilizzare la classe `MulticastSocket`. Per quanto ho letto online è dovuto all'interfaccia di rete del mio computer. Un'istruzione solleva un'eccezione come se ricevesse un indirizzo ipv6 piuttosto che ipv4.