PROGETTO DI **RETI INFORMATICHE**
A.A. 23-24 (615567 – Andrea Migliore)

[1] Introduzione

Il progetto si focalizza sulla realizzazione di un'applicazione distribuita che simula il gioco dell'*escape room*. In particolare era richiesta la progettazione del server e del client, insieme alla realizzazione di una funzionalità aggiuntiva che coinvolgesse l'interazione tra due client.

Come funzionalità a scelta è stata implementata la figura del *supervisore*, un utente con il compito di monitorare le sessioni di gioco in corso, fornire assistenza ai giocatori attraverso messaggi di aiuto e regolare il tempo rimanente.

Il protocollo di trasporto adottato è il TCP, in quanto è principalmente richiesta l'affidabilità della comunicazione piuttosto che la velocità. Nello specifico lo scambio di dati avviene attraverso *socket bloccanti* e si ipotizza che non si verificano invii o ricezioni parziali. In questo modo basta una singola chiamata a send() o recv() per inviare o ricevere completamente un messaggio.

I messaggi sono identificati da un tipo e non hanno una lunghezza prefissata. Durante la fase di trasmissione, il tipo e la lunghezza del payload vengono inviati in formato binario, seguiti dal contenuto del messaggio in formato testuale.
Ogni messaggio è strutturato nel seguente modo:

```c
typedef struct {
    msg_type type;
    char payload[MAX_PAYLOAD_DIM];
} desc_msg;
```

Nonostante l'utilizzo di un protocollo testuale possa richiedere una maggiore larghezza di banda, questa scelta è stata preferita in quanto la maggior parte delle informazioni è di natura testuale.

Il tipo del messaggio consente al destinatario di comprendere la natura e la quantità dei dati trasportati nel payload.
Per una spiegazione dettagliata dei tipi di messaggio utilizzati durante la comunicazione, si può consultare il file 'shared.h'.

Per quanto riguarda gli enigmi, sono principalmente di due tipologie:

1. Utilizzo combinato di oggetti, come ad esempio l'utilizzo di una chiave su un lucchetto.
2. Enigmi testuali, che includono un testo generico e possono consistere in una domanda a risposta multipla, il completamento di una sequenza, eccetera. Questi enigmi richiedono una risposta testuale da parte del giocatore.

[2] Server

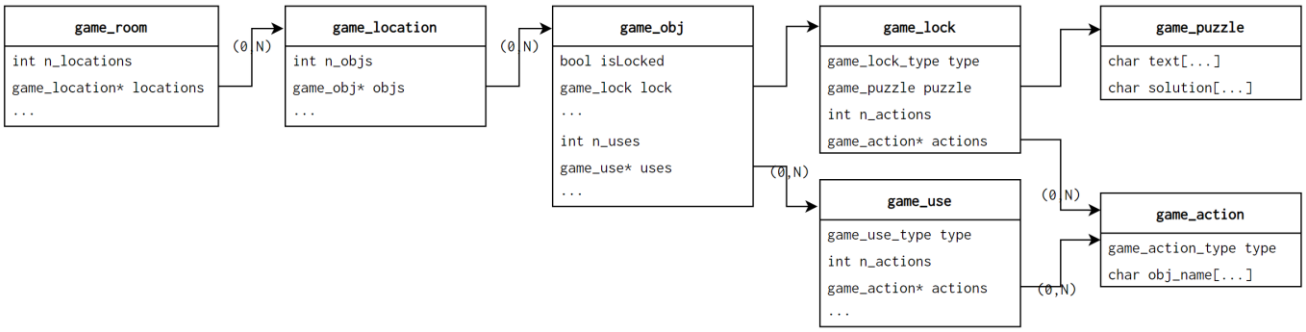
Il server è implementato utilizzando l'*I/O multiplexing*, una scelta motivata dalla natura I/O-bound dell'applicazione. Poiché i giocatori trascorrono la maggior parte del tempo a pensare alle mosse e a risolvere gli enigmi, l'I/O multiplexing si adatta bene alle esigenze dell'applicazione, consentendo al server di gestire efficacemente molte connessioni simultaneamente mentre attende le risposte degli utenti e ottimizzando le risorse del sistema.
Le informazioni sulle sessioni di gioco in corso sono conservate in memoria tramite una lista.
Una sessione è rappresentata dalla struttura *game_session*:

```c
typedef struct game_session
{
    int sd;
    char username[MAX_USR_DIM];
    int room;

    int help_msg_id;
    char help_msg[MAX_HELP_DIM];

    int n_special_objects;
    time_t start_time;
    time_t seconds;
    int token;

    int dim_bag;
    int n_bag_objs;
    game_obj** bag_objs;

    int n_locked_objs;
    game_obj** unlocked_objs;

    int n_hidden_objs;
    game_obj** revealed_objs;

    int n_consumable_objs;
    game_obj** consumed_objs;

    struct game_session* next;
}
game_session;
```

Attraverso il campo *sd* (descrittore del socket) o *username* è possibile identificare univocamente una sessione di gioco, mentre gli altri campi forniscono informazioni sullo stato del giocatore. Ciò implica che non possono esistere più sessioni con lo stesso *username* o *sd*.

Ulteriori dettagli sui campi sono disponibili nel file 'server.h'.

Oltre alla gestione delle sessioni, il server definisce varie strutture dati che consentono di modellare le stanze e gli elementi del gioco in modo flessibile. Le strutture includono *game_room*, *game_location*, *game_obj*, *game_use*, *game_lock*, *game_puzzle* e *game_action*.



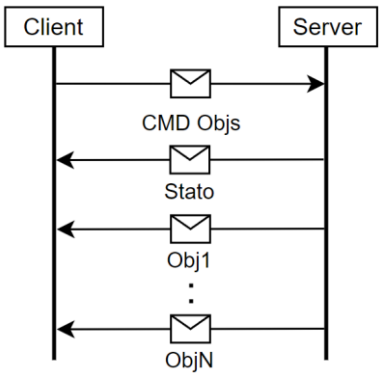
Per ulteriori dettagli su ciascuna di queste strutture, si può fare riferimento al file ‘server.h’.

Dopo l'avvio, il server rimane in attesa che un socket sia pronto in lettura. Una volta ricevuto un messaggio, il server lo interpreta in base al suo tipo, eseguendo le azioni corrispondenti. Infine, invia il risultato.

La gestione degli utenti avviene tramite il file ‘users.txt’. Per accedere alla lista delle stanze, il client deve effettuare il login o, se non ha un account, registrarsi. Durante la registrazione, vengono salvate nel file le coppie [username, password]. Durante il login, il server verifica l'autenticità delle credenziali confrontandole con quelle presenti nel file.

[3] Client

Dato che la complessità del gioco è principalmente gestita dal server, il client risulta essere abbastanza semplice. All'avvio, permette all'utente di autenticarsi e selezionare una stanza per iniziare il gioco. I dati fondamentali della sessione di gioco sono gestiti attraverso la struttura *game_state*, definita nel file ‘client.h’.



Poiché il client non dispone di informazioni sulle location, gli oggetti o gli enigmi presenti nella stanza, ogni comando eseguito dall'utente richiede una comunicazione con il server. Quest'ultimo elabora la richiesta in base allo stato della sessione e invia uno o più messaggi di risposta.

In particolare, per ogni richiesta, il client si aspetta di ricevere un primo messaggio contenente lo stato aggiornato della sessione di gioco (ad esempio, il tempo rimanente, i token ottenuti, ecc.) seguito da uno o più messaggi di dati.

In aggiunta ai comandi standard, sono stati implementati il comando 'drop', per rilasciare un oggetto, e il comando 'help', per consultare il messaggio di aiuto più recente.

[4] Supervisore

Il supervisore ha una struttura simile a quella del client, ma con funzionalità aggiuntive per monitorare le sessioni di gioco in corso e interagire con esse.

Ad esempio, tramite il comando "look" è possibile iniziare ad osservare una sessione di gioco, mentre con "objs" si ottiene la lista degli *oggetti speciali* presenti nella stanza con i relativi stati. Mediante il comando "bag" è possibile visualizzare gli oggetti nello zaino del giocatore, mentre "time" consente di gestire il tempo rimanente aggiungendo, sottraendo o impostando i secondi. Il comando "help" permette invece di inviare un messaggio di assistenza al giocatore.

Si considerano *oggetti speciali* quelli che sono bloccati, nascosti o consumabili (utilizzabili una sola volta).