Progetto Reti Di Calcolatori

Ambrosio Ferdinando 0124002668
Adamo Adriano 0124002669

15 February 2024

1 Descrizione del progetto

L'obbiettivo del progetto è simulare un settore spaziale di meteoriti (server) che invia n pacchetti UDP e il client,che interpretra la navicella, deve evitarli. Il client prima dell'impatto viene avvisato cambiando il colore della navicella: da bianco diventa rosso, in modo tale che il client può evitare il detrito. Abbiamo implementato una parte grafica tramite libreria SDL, la direzione del detrito, che dipende dalla posizione della navicella(client) e se non riceve la poszione del client i detriti si generano casualmente.

2 Descrizione e schema dell'architettura

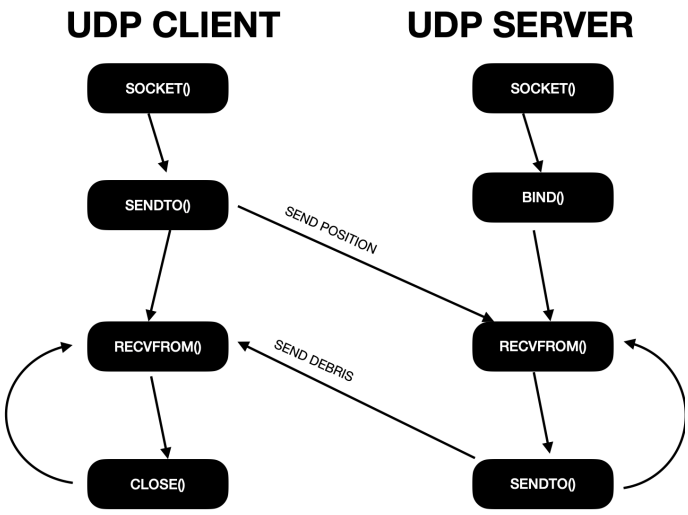

Figure 1: Architettura di rete

3 Dettagli implementativi dei client/server

3.1 Server

All'interno del server troviamo le seguenti istruzioni: una struct di detriti (*debris*),una struct di array di struct per gestire piu detriti contemporaneamente, una **structDebrisPacket** che riceve i detriti attivi e li manda al client, poi **SpaceshipPosition** che sarebbe la struct della posizione ricevuta dalla navicella. Abbiamo implmementato poi una funzione **initDebris()**, che inizializza i detriti in modo casuale, un'altra funzione **updateDebrisPositions()** che fa "cadere" i detriti verso il basso e infine la funzione **generateDebrisBasedOnSpaceship** che genera i detriti a seconda della posizone della navicella. Abbiamo poi **sendDebrisPacket()** che manda, tramite **sendto()**, i pacchetti(detriti) al client. All'interno del Main c'è la configurazione della socket e il binding. Nel ciclo while si controlla se arriva la posizione della navicella; in caso positivo si generano i pacchetti verso la posizione x della navicella e in caso contrario si generano in modo casuale. Successivamente si calcola il tempo richiesto (2 sec.) per generare i detriti e si richiamano le funzioni sopra citate.

3.2 Client

Nel client, le implementazioni sono le seguenti: troviamo le stesse strutture dati più una struttura per l'astronave(x,y,altezza,lunghezza). Qui utilizziamo una libreria grafica SDL e definiamo la funzione `drawCircle()` che opera nel seguente modo: riceve le coordinate dal server tramite `recvfrom` ed effettua il rendering dei detriti. Definiamo le variabili *centerY* e *centerX*, mentre il raggio viene definito autonomamente quando chiamiamo la funzione.
Nel blocco principale configuriamo la socket e definiamo la struttura dell'astronave (`starship`). SDL funziona tramite la gestione di eventi, nel quale definiamo il movimento dell'astronave con le frecce della tastiera. Ogni qualvolta viene premuta una freccia salviamo la posizione per poi mandarla al server. Per gestire l'alert, impostiamo la variabile *alert* come flag e verifichiamo se il detrito si trova entro il margine verticale sopra l'astronave. In caso affermativo, impostiamo l'alert a 1, che successivamente controlliamo per cambiare il colore.
Il ciclo di gioco è gestito tramite un ciclo `while !quit`, e premendo il tasto *esc* sulla tastiera impostiamo `quit=1`, consentendo di uscire dal ciclo e quindi dal gioco. Infine, puliamo lo schermo e chiudiamo sia SDL che la socket.

4 Parti rilevanti del codice sviluppato

4.1 *initDebris()*

```
   void initDebris() {
    int count = 0;
    for (int i = 0; i < GRID_SIZE; i++) {
        if (!debris[i].active && count < MAX_DETRITI) {
            debris[i].x = rand() % GRID_SIZE;
            debris[i].y = 0;
            debris[i].active = 1;
            count++;
        }
    }
}
```

Listing 1: funzione che genera i pacchetti in modo casuale

in questa funzione dichiariamo inizializziamo un contatore, all'interno del for controlliamo se il detrito è attivo e se il contatore è minore di MAX DETRITI per non generarne troppi, successivamente si assegna una posizione randomica a x e la posizione 0 a y tale che il detrito parte dall'alto.

4.2 *generateDebrisBasedOnSpaceship()*

```
  void generateDebrisBasedOnSpaceship(int pos)
 {
    int count = 0;
    for (int i = 0; i < GRID_SIZE; i++) {
        if (!debris[i].active && count < MAX_DETRITI) {
            debris[i].x = pos;
            debris[i].y = 0;
            debris[i].active = 1;
            count++;
        }
    }
}
```

Listing 2: funzione che genera i pacchetti verso la posizione della navicella

Qui la funzione opera analogamenta a quella citata prima, solo che la posizione viene generata a seconda della posizione della navicella.

4.3 *select()*

```
retval = select(sockfd + 1, NULL, &writefds, NULL, &tv);
        if (retval == -1) {
            perror("select()");
        } else if (retval) {
        recvfrom(sockfd, &pos, sizeof(pos), 0,
        (struct sockaddr*)&clientAddr, &clientAddrLen);
            generateDebrisBasedOnSpaceship(pos.x);
            printf("ricevo in posizione %d",pos.x);

        } else {
            initDebris();
        }
```

Listing 3: gestione della generazione dei detriti

In questa sezione del codice, utilizziamo la funzione select() per monitorare un insieme di file descriptor per capire se ci sono dati disponibili per la lettura o la scrittura. Il nostro obiettivo è di determinare il momento ottimale per ricevere dati dal client (specificamente, la posizione della navicella) e, in base a questi dati, generare detriti nella stessa posizione o, in assenza di dati, generare detriti in posizioni casuali.La chiamata a select() viene configurata per monitorare solo la disponibilità alla scrittura dei file descriptor, poiché il set di lettura e quello delle eccezioni sono impostati su NULL. La variabile tv specifica un timeout, che limita il tempo di attesa di select() per un evento sui file descriptor.

4.4 *updateDebrisPositions()*

```
void updateDebrisPositions() {
    for (int i = 0; i < GRID_SIZE; i++) {
        if (debris[i].active) {
            debris[i].y += 1;
            if (debris[i].y > WINDOW_HEIGHT) {
                debris[i].y = 0;
                debris[i].x = rand() % GRID_SIZE;
            }
        }
    }
}
```

Listing 4: caduta dei detriti

Questa funzione gestisce la caduta dei detriti attraverso lo schermo. Questa funzione viene chiamata ad ogni ciclo del gioco per aggiornare la posizione di ogni detrito attivo, simulando così il loro movimento verso il basso.Per ogni detrito, la funzione controlla se è attivo. Questo controllo è necessario perché solo i detriti attivi devono essere mossi.

5 Manuale utente con le istruzioni su compilazione ed esecuzione

5.1 compilazione

Avviare prima il server e poi il client.

5.2 esecuzione

Spostare la navicella tramite le freccette della tastiera, premere ESC per uscire dal gioco.