Game of Drones

Area Coverage Simulator

Autori:

Adriana Pia Barbalace

Noemi Calandro

Alessia Castronovo

Damiano Jack Coccia

Contents

1 Introduzione

Il progetto **Game of Drones** è un programma che gestisce varie funzionalità di una formazione di droni (movimento, ricarica ecc.) avente come obiettivo la copertura periodica di una data area da sorvegliare.

2 Descrizione generale

Per soddisfare i requisiti, la formazione di droni deve coprire l'intera area ogni 5 minuti. I droni partono dal centro di controllo e iniziano a muoversi verso le loro posizioni di partenza per la copertura della mappa; nel momento in cui ogni drone ha raggiunto la propria posizione di partenza, il timer dei 5 minuti viene innescato e i droni iniziano a coprire la mappa secondo le indicazioni dell'algoritmo di movimento. I droni rilevano automaticamente le scarsità di autonomia e iniziano a dirigersi verso il centro di controllo (per la ricarica) quando ciò accade. Il numero di droni sufficiente a coprire la mappa per una epoca di 5 minuti è 725, perciò questo è il numero di droni partenti dal centro di controllo all'inizio della simulazione; tuttavia data la necessità di copertura ininterrotta della mappa, e anche l'elevato tempo di ricarica per ogni drone, questo numero non è sufficiente per la copertura perpetua della mappa. Per ovviare a questo problema, quando c'è bisogno che un nuovo drone parta dal centro di controllo ma in quest'ultimo non ci sono droni (oppure i droni presenti sono in ricarica), viene creato un nuovo drone, con la batteria al massimo, e viene subito inviato nella mappa.

3 Requisiti

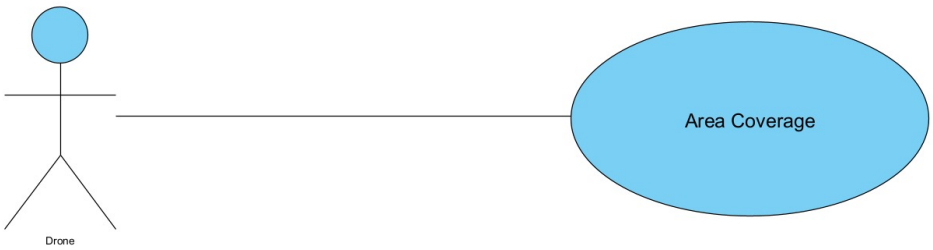
- Area di copertura: 6 Km x 6 Km
- Periodicità della copertura: 5 minuti
- Autonomia dei droni: 30 minuti
- Tempo di ricarica: valore casuale tra 120 e 180 minuti
- Velocità dei droni: 30 Km/h
- Raggio di copertura dei droni: 10 metri

3.1 Requisiti funzionali

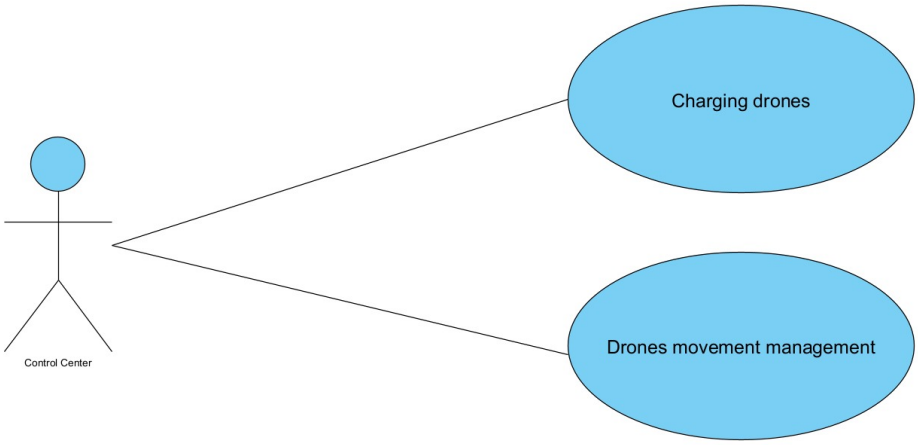
3.1.1 User requirements

1. U-Req-Coverage:
    - Ogni punto dell'area deve essere coperto almeno una volta ogni 5 minuti.
2. U-Req-RaggioCopertura:
    - Perché un punto sia coperto da un drone, il drone deve trovarsi, in un dato istante dell'intervallo di 5 minuti, a massimo 10 metri dal punto.
3. U-Req-Ricarica:
    3.1. Il sistema deve gestire automaticamente i cicli di ricarica dei droni.
    3.2. Il tempo di ricarica è scelto ad ogni ricarica casualmente nell'intervallo [2h, 3h].

3.1.1.1 Use Case UML



1. Use Case Drone

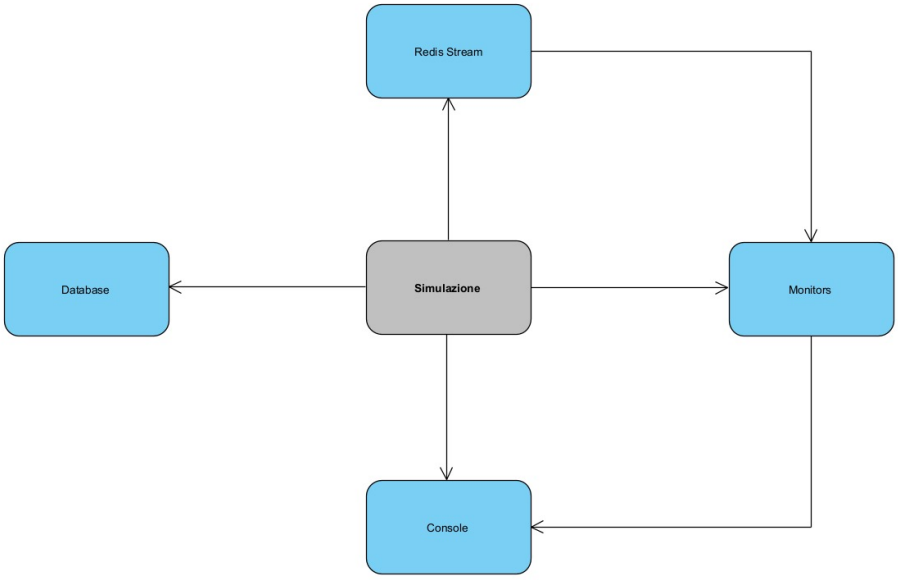


2. Use Case Control Center

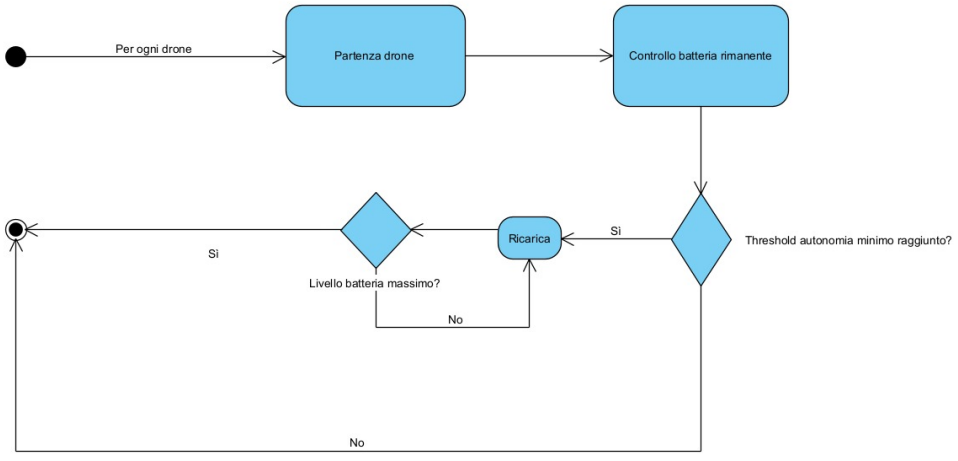
3.1.2 System requirements

1. S-Req-Ricarica:
    - Il sistema deve calcolare automaticamente i tempi di ricarica e di volo residui per ogni drone.
2. S-Req-PianificazionePercorso:
    - Il sistema deve pianificare il percorso di ogni drone per massimizzare l'efficienza della copertura dell'area.

3.1.2.1 Architettura del Sistema

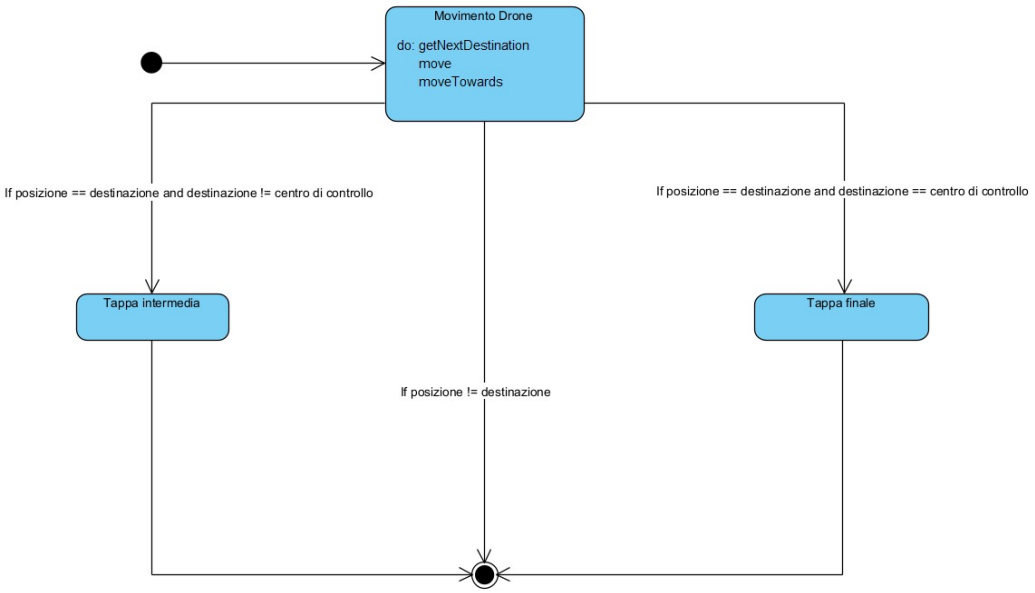


3.1.2.2 Activity Diagram UML



AD-UML relativo all'operazione di Ricarica

3.1.2.3 State Diagram UML



SD-UML relativo all'operazione di Ricarica

3.2 Requisiti non funzionali

- **Affidabilità:** i droni sanno quando devono tornare per non scaricarsi al ritorno.
- **Efficienza:** viene generato un nuovo drone solamente quando c'e la necessità che un drone parta dalla base per sostituirne un altro, ma non ci sono droni disponibili.

4 Implementazione

4.1 Componenti del sistema

4.1.1 Modello per la mappa e il movimento dei droni

Per semplificare l'implementazione del movimento dei droni, l'area da coprire è stata suddivisa in caselle quadrate con lato di 10 metri. La posizione di un

drone in qualsiasi momento della simulazione è sempre su uno dei vertici di una casella, mentre i movimenti possibili sono: il movimento laterale o verticale di un'unità, per il quale occorrono 1,2 secondi, e il movimento diagonale di un'unità, per il quale occorrono 1,7 secondi.

4.1.2 Setup della formazione droni

All'inizio della simulazione, la chiamata della funzione setDronesPathings() popola, per ogni drone, il suo attributo "path": un vettore contenente ogni tappa che il drone dovrà effettuare (l'ultimo elemento del vettore è sempre il centro di ricarica droni). Successivamente, la funzione moveToStartingPositions() dà inizio al movimento dei droni, facendoli spostare ciascuno verso la sua prima tappa. In questo frangente, i droni non stanno ancora coprendo punti della mappa; stanno raggiungendo, partendo dal centro di ricarica, le posizioni da cui inizieranno la copertura dell'intera area. Il percorso di ogni drone è designato tenendo conto delle aree da coprire, dell'autonomia e dei percorsi degli altri droni.

4.1.3 Copertura della mappa

Quando ogni drone ha raggiunto la sua prima tappa, inizia la fase di copertura della mappa e, ogni 5 minuti a partire da questo momento, l'intera mappa dovrà essere stata coperta nel rispettivo intervallo di 5 minuti precedente. In questo momento vengono quindi aggiornate le due mappature dronesToMove e dronesToReplace: alla prima vengono aggiunti, in corrispondenza delle chiavi relative al timestamp attuale, ognuno dei 725 droni iniziali, poiché ognuno inizierà a muoversi da questo timestamp; alla seconda vengono aggiunti, in corrispondenza delle chiavi relative a un determinato timestamp futuro, tutti i droni il cui drone rimpiazzatore dovrà lasciare il centro di controllo in quel determinato timestamp.

A questo punto, un ciclo while gestisce il movimento dei droni, l'aggiornamento dei timestamp e delle strutture dati ausiliarie, e le chiamate dei monitor tramite Redis Streams. La prima parte del corpo di questo ciclo consiste in controlli sulle strutture dati ausiliarie (mappature, vettori e set), seguiti da eventuali modifiche sullo stato attuale dei droni, dei punti della mappa o delle strutture dati stesse:

- **Controllo sulla ricarica dei droni:** questo controllo verifica che il set mappato dalla chiave timeSinceStart (il timestamp attuale) nella map-

patura dronesDoneCharging sia vuoto. In caso contrario, ogni drone che ha finito di ricaricarsi viene aggiunto al set idleDrones e il suo stato passa da "charging" a "ready".

- **Controllo sul rimpiazzamento dei droni:** questo controllo verifica che il set mappato dalla chiave timeSinceStart (il timestamp attuale) nella mappatura dronesToReplace sia vuoto. In caso contrario, per ogni drone che ha bisogno di essere rimpiazzato, verrebbe creato un nuovo drone oppure stanziato un drone inoccupato; a questo drone verrebbe quindi assegnato lo stesso "path" del drone originale e verrebbe aggiunto al set mappato dalla chiave timeSinceStart nella mappatura droneToMove.

A questo punto l'algoritmo istanzia nextReplacingTimestamp, il timestamp in cui questo nuovo drone dovrebbe essere sostituito, e modifica dronesToReplace come ha fatto in precedenza; aggiunge anche questo timestamp al vettore checkTimestamps qualora non vi fosse già presente.

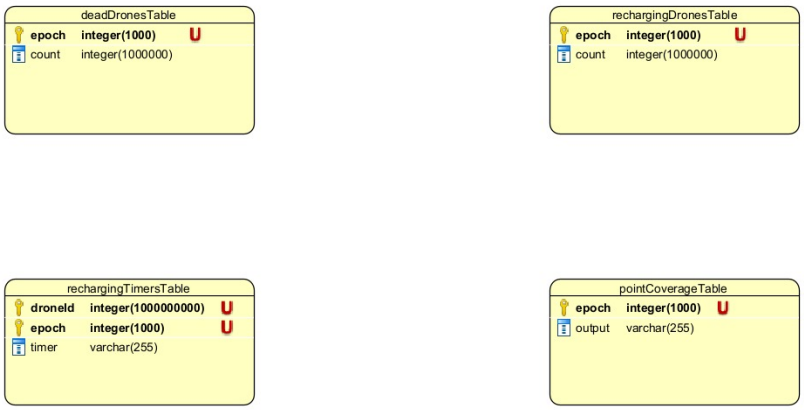
4.1.4 Movimento, ricarica e gestione dei timestamp

La parte restante del ciclo while si occupa del movimento dei droni, della loro entrata in fase di ricarica, e della gestione dei timestamp. Viene istanziato movingDrones: il set mappato dalla chiave timeSinceStart nella mappatura dronesToMove. I droni all'interno di questo set sono quelli che dovranno muoversi nel timestamp gestito dall'attuale iterazione del ciclo, perciò per ognuno di essi l'algoritmo controlla se la batteria si sia scaricata (in tal caso, chiama il metodo shutDown() sul drone), e in caso contrario chiama il metodo move() sul drone: un metodo che fa muovere il drone verso la sua attuale destinazione e ritorna il tempo impiegato per compiere il movimento (in decimi di secondo: 12 se laterale, 17 se diagonale). Questo metodo gestisce anche la disattivazione del drone, tramite il metodo shutDown(), qualora abbia raggiunto la sua destinazione finale; infatti, dopo aver chiamato move(), l'algoritmo controlla se il drone sia ancora attivo e, in caso contrario, lo fa entrare in ricarica. Questo avviene con la chiamata della funzione randomRechargingTime() che, secondo i requisiti del sistema, assegna al drone un tempo di ricarica casuale tra le 2 e le 3 ore, con l'aggiunta dei dati sulla ricarica a Redis Streams, che verranno usati per la chiamata del monitor, e la modifica delle mappature dronesDoneCharging, chargeCompleteAt e del vettore checkTimestamps.

Infine, l'algoritmo deve creare un nuovo timestamp e aggiungerlo al vettore checkTimestamps, così che il movimento dei droni che si sono mossi

in questa iterazione possa essere innescato dalle seguenti iterazioni del ciclo while. Questo timestamp viene creato aggiungendo il tempo di movimento al timestamp attuale e, solamente nel caso particolare in cui un drone ha bisogno di stazionare per rimanere in sincronizzazione con tutti gli altri, vengono aggiunti anche 552 decimi di secondo (il tempo standard di stallo per i droni che coprono la parte centrale della mappa). Dopo aver aggiornato droneToMove, l'algoritmo elimina da checkTimestamps il timestamp utilizzato dal ciclo while nell'iterazione appena trascorsa, e assegna a timeSinceStart il nuovo elemento con indice 0 di checkTimestamps.

4.2 Database



Di seguito forniamo una descrizione delle tabelle presenti nel database:

- **deadDronesTable:** rappresenta i droni che sono morti e devono essere ricaricati, si conserva l'epoca (epoch) e il numero dei droni morti in quell'epoca (count).
- **rechargingDronesTable:** rappresenta i droni che si stanno ricaricando, si conserva l'epoca (epoch) e il numero dei droni che almeno in un momento dell'epoca si stavano caricando (count).
- **rechargingTimersTable:** rappresenta il tempo di ricarica dei droni, si conserva l'id del drone (droneId), l'epoca (epoch) e il timer (timer).

- **pointCoverageTable:** rappresenta i punti coperti dai droni, si conserva l'epoca (epoch) e l'output, che riporta "pass" se tutti i punti del piano sono stati coperti, "fail" in caso contrario.

4.2.1 Connessione con PostGres

Per comunicare con il server Postgres, utilizzato per il salvataggio dei log di esecuzione, viene istanziata la stringa conninfo contenente il nome del db, l'utente, la password, l'indirizzo dell'host e la porta. In seguito vengono istanziate le 4 tabelle del db che verranno riempite durante l'esecuzione, e le si aggiungono al db nel server tramite la funzione create_table().

4.3 Connessione con Redis

Per comunicare con la stream Redis, utilizzata per la scrittura dei dati relativi ai monitors e la chiamata dei monitors stessi, vengono istanziati nella funzione runSimulation() i valori hostname e port (i valori default per la connessione con un server Redis lanciato in locale).
Viene poi chiamata la funzione connectRedis() che prende in input questi due valori e ritorna un puntatore a oggetto redisContext, durante l'esecuzione sarà questo puntatore ad essere utilizzato per ogni interazione con Redis Streams.
Le funzioni chiamate nel corpo di connectRedis(), la classe redisContext e ognuna delle funzioni utilizzate in seguito per l'interazione con il server fanno tutte parte della libreria importata: hiredis/hiredis.h.