PROGRAMMAZIONE DI RETI

Relazione elaborato di fine corso

Traccia 1: Progetto Droni

Studenti:

Anny Bevilacqua - anny.bevilacqua@studio.unibo.it - 0000971524

Salvatore Zammataro - salvatore.zammataro@studio.unibo.it - 0000873293

OBBIETTIVO

Simulare un a rete di consegne a domicilio tramite l'utilizzo di droni. La rete dovrà essere composta da:

- Un client da cui l'operatore assegna a ciascun drone l'indirizzo di consegna
- Tre droni che effettueranno le consegne
- Un gateway che provvederà ad effettuare il relay dei messaggi verso i droni e provvederà a fare da concentratore per raccogliere i messaggi provenienti dai droni ed inviarli al client

SPECIFICHE TECNICHE RICHIESTE

Il client deve poter inviare l'indirizzo di consegna la drone solo se esso si è presentato al gateway come disponibile.

Dal client deve poter essere inserito l'indirizzo di consegna e l'identificativo, o equivalentemente l'indirizzo IP, del drone cui far effettuare la consegna. Tali informazioni dovranno essere trasmesse al gateway, che provvederà a comunicare al drone incaricato l'indirizzo di destinazione.

Il drone dovrà mostrare l'indirizzo di destinazione ed inviare al gateway un messaggio di avvenuta consegna e rendersi nuovamente disponibile.

Il gateway deve mostrare tutti messaggi in transito con sorgente e destinatario.

La connessione tra client e gateway deve essere di tipo TCP, mentre la connessione tra gateway e droni deve essere di tipo UDP.

Ogni drone ha un suo indirizzo IPv4, e cosi le due interfacce del gateway e il client.

I 3 droni hanno un indirizzamento appartenente ad una rete di Classe C del tipo 192.168.1.0/24.

Il Gateway ha due interfacce di rete: quella verso i droni il cui IP Address appartiene alla stessa network dei dispositivi mentre l'interfaccia verso il client ha indirizzo ip appartenente alla classe 10.10.10.0/24, classe a cui appartiene anche l'IP address del gateway.

SVILUPPO

Linguaggio utilizzato*: python*

Si è deciso di implementare il progetto sviluppando su classi differenti ciascun elemento della rete di consegne: Client, Gateway e Drone.

Ciascuna di queste classi genera un'interfaccia grafica che permette all'utente di simulare il funzionamento della rete interamente attraverso di esse. Tale interfaccia viene generata nel momento in cui la classe corrispondente viene istanziata, tale compito è svolto dal metodo **createWindow** per ciascuna delle classi.

La scelta di dotare il progetto di un'interfaccia grafica è stata presa con l'intento di fornire all'utente un'esperienza esteticamente più gradevole.

- CLIENT

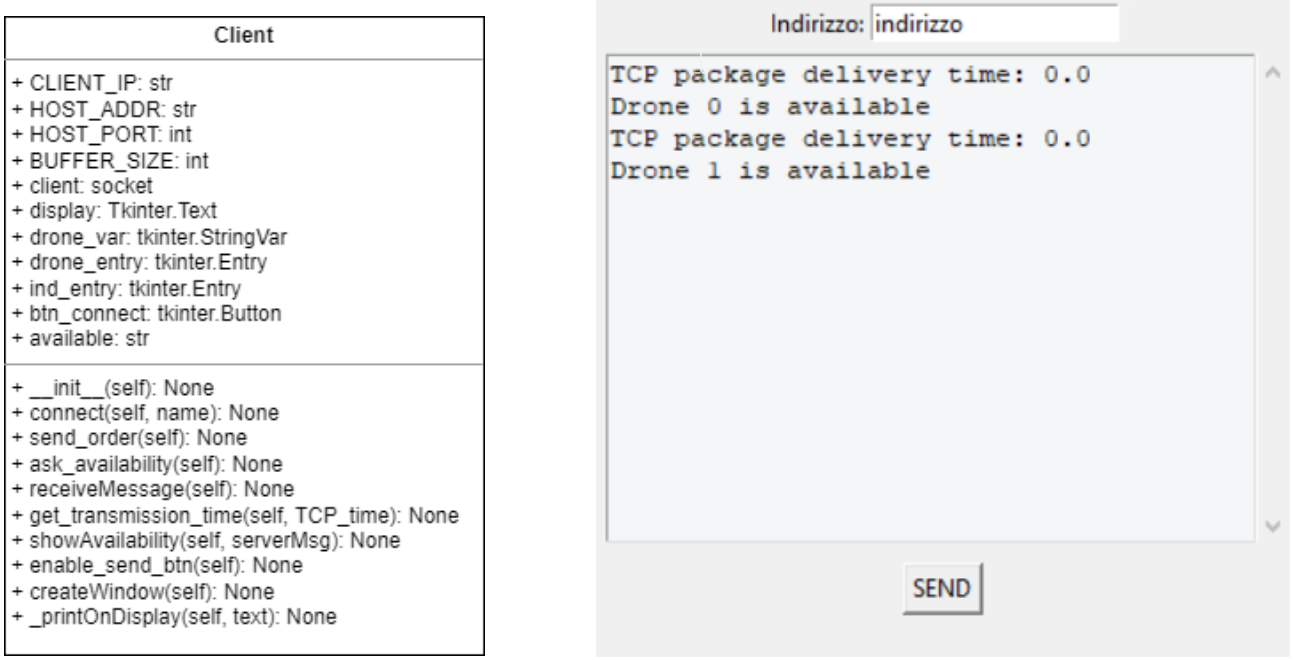


Diagramma UML classe Client (DX) e interfacciaClient (SX)

La classe Client, rappresentata nel diagramma UML della figura precedente, sviluppa il concetto di client della Rete di Consegne.

All'avvio viene creata l'interfaccia grafica grazie al metodo **createWindow, come precedentemente accennato,** e stabilita la connessione al gateway chiamando il metodo **connect.**

Tale metodo **connect** avvia la connessione TCP col gateway ed un messaggio di saluto ad esso. Un volta stabilita la connessione viene avviato un thread legato al metodo **receiveMessage** per rimanere in ascolto dell'host.

I messaggi che il client riceve dal gateway sono in risposta alle richieste di disponibilità dei droni. Tali messaggi, se correttamente formattati vengono elaborati attraverso il metodo **showAvailability**, il quale salva nella proprietà **available** il risultato dell'interrogazione e richiama il metodo **enable_send_btn** che gestisce l'abilitazione del **btn_send** ed il metodo **_printOnDisplay**, che stampa sul display del client un messaggio che comunica l'esito dell'interrogazione.

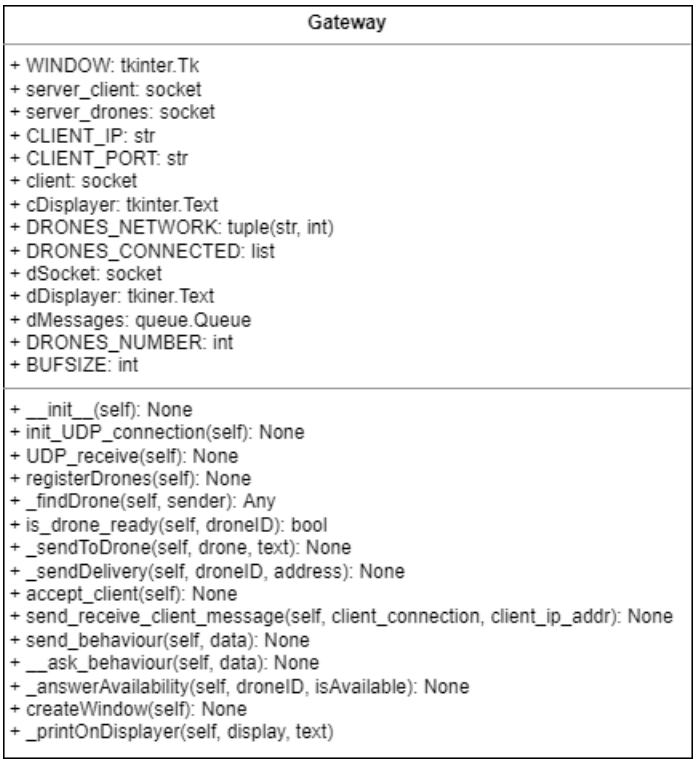
Tali interrogazioni al gateway vengono effettuate in automatico ogni volta che il campo ID/IP DRONE viene editato, sollevando l'utente dalla necessità di richiedere manualmente la disponibilità dei droni.

Tale comportamento è stato ottenuto impostando un ascoltatore sul campo ID/ IP DRONE, che ogni volta che accade l'evento di scrittura chiama il metodo **askAvailability.**

L'evento di click del **btn_send** è legato al metodo **send_order**, il quale si occupa di trasmettere al gateway l'identificativo del drone scelto per la consegna e l'indirizzo della stessa, dopodiché viene reimpostata la proprietà **available** a false e richiamato il metodo **enable_send_btn** che disattivera il **btn_send.**

Assieme al risultato delle interrogazioni al gateway, viene mostrato sul display anche il tempo impiegato per la trasmissione dei messaggi in entrata.

- GATEWAY



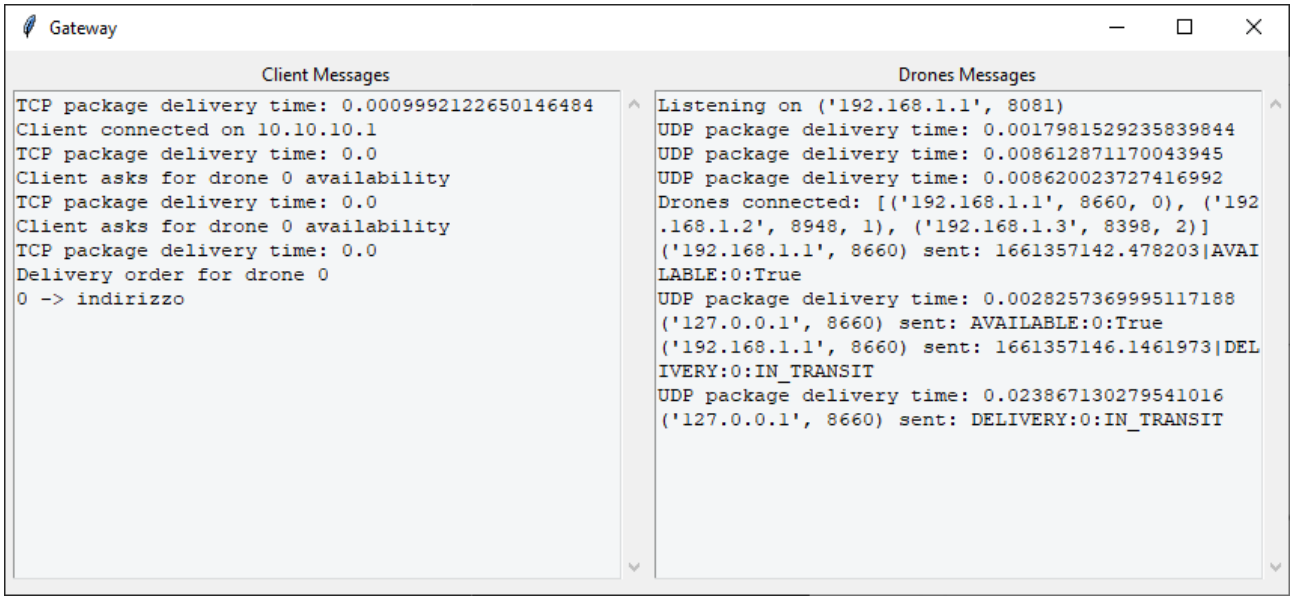


Diagramma UML classe Gateway (SOPRA), interfaccia Gateway (SOTTO)

La classe Gateway, rappresentata nel diagramma UML della figura precedente, sviluppa il concetto di gateway della Rete di Consegne.

All'avvio viene creata l'interfaccia grafica grazie al metodo **createWindow,** come precedentemente accennato ed avviate le connessioni TCP e UDP su due nuovi threads, al fine di non sovraccaricare il thread della GUI.

Una volta avviata la connessione UDP il gateway rimane in ascolto sinché tutti i droni non si sono presentati, tale comportamento viene implementato all'interno del metodo **registerDrones,** il quale memorizza inoltre indirizzo IP, porta ed identificativo di ciascun drone all'interno della lista DRONES_CONNECTED. Tale lista verrà poi mostrata sul display del gateway dedicato ai droni.

La comunicazione degli indirizzi di consegna si droni vengono gestiti dal metodo **_sendDelivery,** il quale grazie al metodo **_getDrone** ottiene l'IP del drone scelto dall'utente, poi costruisce il messaggio da inviare e lo passa al metodo **_sendToDrone** che si occupa della trasmissione di tutti i messaggi diretti ai droni.

La connessione col client viene invece gestita dal metodo **send_receive_client_message**, il quale a seconda del messaggio ricevuto, che può essere di tipo ASK o SEND, ossia richiesta di disponibilità o invio di pacchetto, richiama i metodi opportuni per rispondere al client o comunicare al drone.

- DRONE

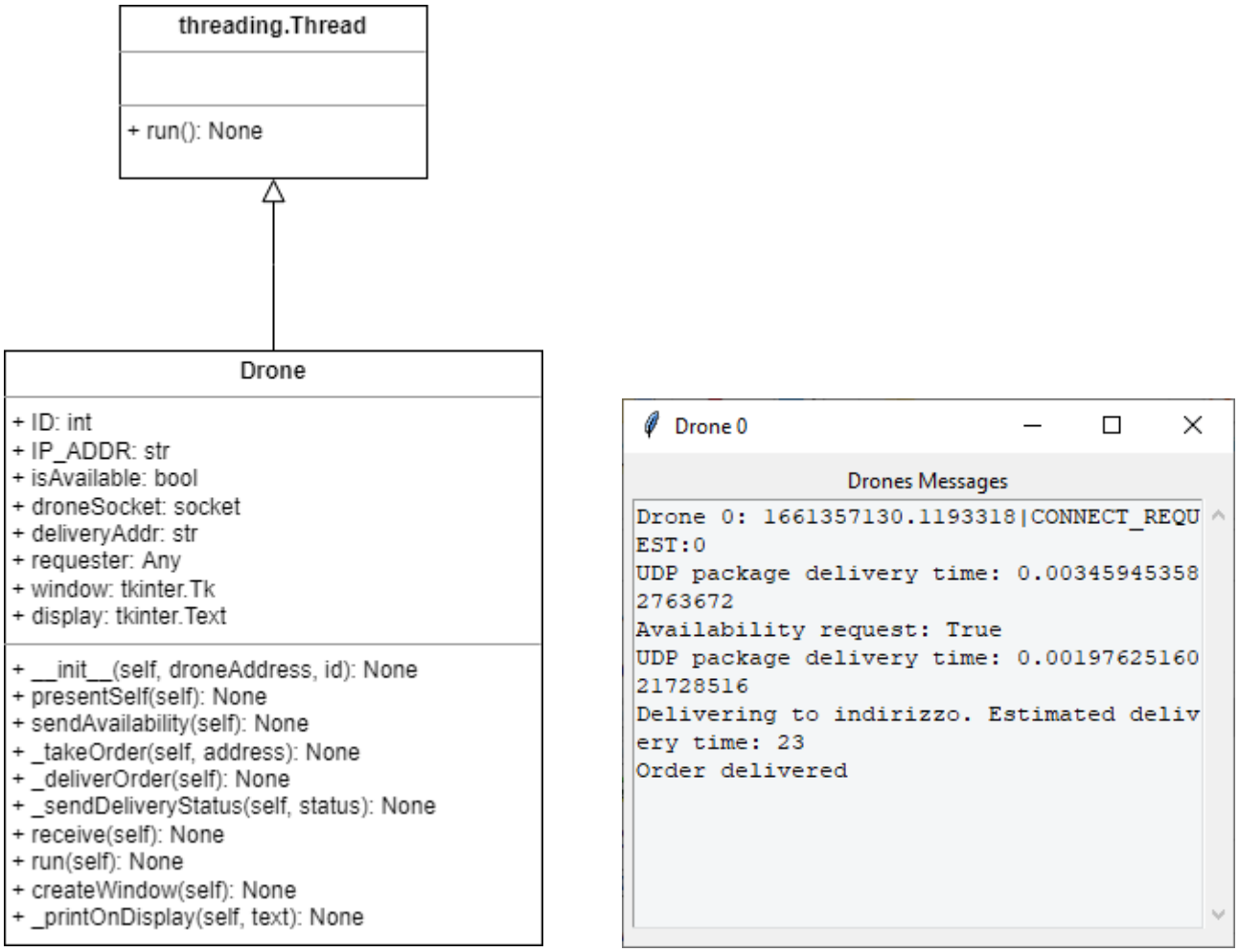


Diagramma UML classe Drone (DX) e interfacciaClient (SX)

La classe Drone, rappresentata nel diagramma UML della figura precedente, sviluppa il concetto di drone della Rete di Consegne.

All'avvio viene creata l'interfaccia grafica grazie al metodo **createWindow,** come precedentemente accennato ed avviate le connessioni TCP.

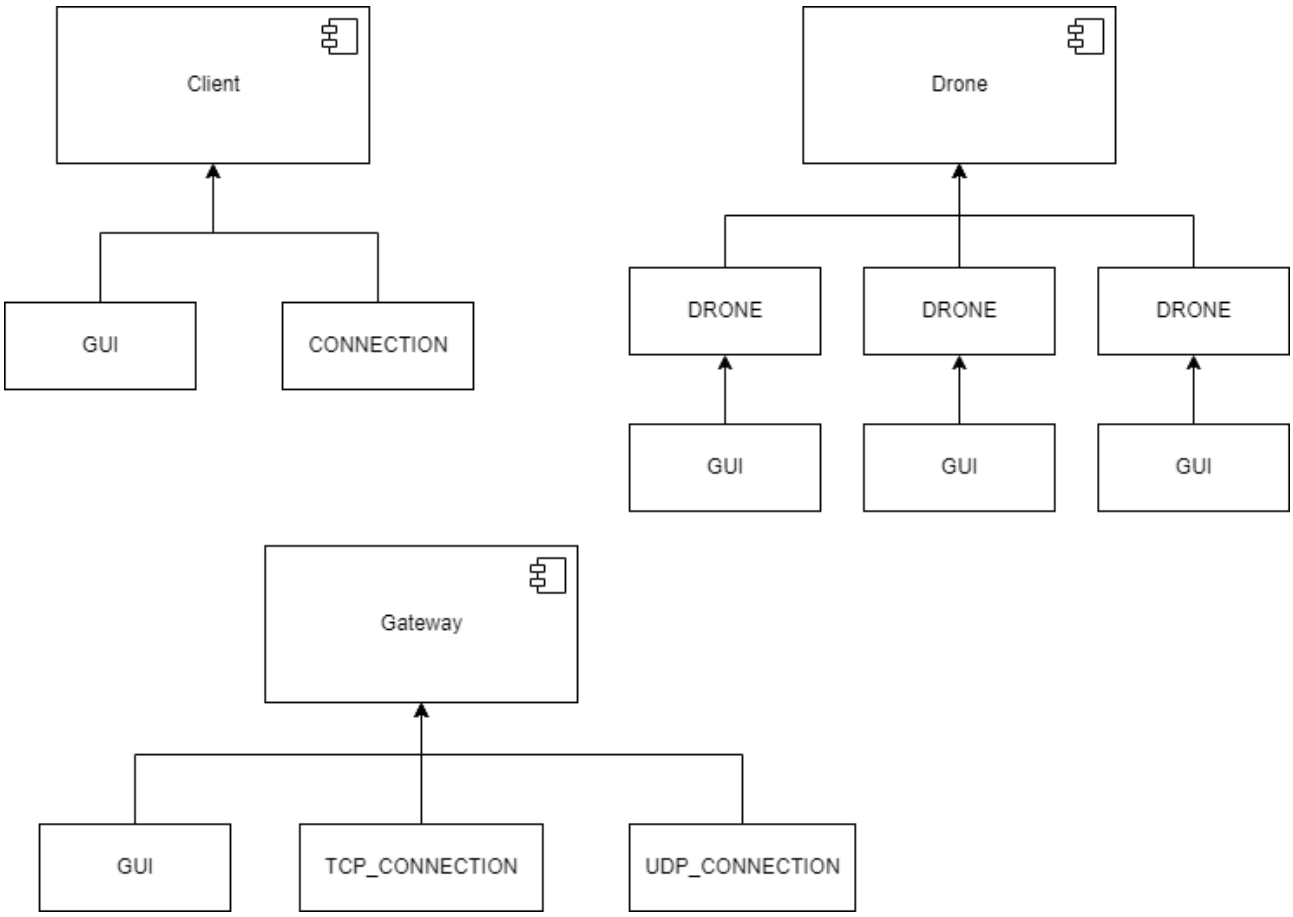
La classe eredita dalla classe Thread e ne sovrascrive il metodo **run**.

Una volta avviato il thread il drone si presenta al gateway comunicando il proprio indirizzo IP, la porta su cui è in ascolto, ed il proprio identificativo, attraverso il metodo **preseentSelf.**

Il metodo **receive** gestisce le richieste provenienti dal gateway. Queste possono essere di due tipi: ASK e DELIVERY, rispettivamente per richiederne la disponibilità o per assegnare una consegna.

Tali richieste vengono gestite dai metodi **sendAvailability**, che comunica al gateway la disponibilità del drone non appena termina una consegna o se è già libero e dal metodo **_takeOrder** che invece prende in carico una consegna e stabilito un tempo random rende il drone non disponibile in tale periodo.

SCHEMA DEI THREADS



Schema dei threads attivi

Lo schema soprastante mostra una sintesi dei thread attivati da ogni classe del progetto.
Si noti come per la classe gateway, essendo quella sottoposta al maggior carico, si è deciso di separare su threads differenti la le connessioni e la gestione della GUI, mentre per la classe drone ogni thread ha a sua volta un solo thread figlio che getsisce tutto.