ELABORATO PER IL CORSO DI PROGRAMMAZIONE DI RETI

A.A 2021/2022

Tipologia 1.

Presepi Alex - 0000976898
alex.presepi@studio.unibo.it
Lugaresi Simone - 0000970392
simone.lugaresi@studio.unibo.it

Indice

Obbiettivo

Realizzare un'applicazione che simuli la spedizione di pacchi mediante dei droni in una zona circoscritta, in cui è presente un client che decide da remoto le destinazioni dei droni. Le varie comunicazioni IP tra client e droni vengono simulate utilizzando dei socket con protocolli TCP e UDP nella rete di loopback.

Descrizione delle scelte di progetto effettuate

Costanti

In ogni sorgente, basandosi anche sugli esempi visti in classe, nella parte iniziale vengono definite le costanti, myIP, impostato a “0.0.0.0” se ancora non lo si conosce, e myMAC che fanno riferimento all’elemento selezionato e sono univoci all'interno della nostra rete simulata, poi ogni elemento ha anche l’ IP e il MAC del gateway per sapere con chi deve comunicare. Si definisce la porta su cui è in ascolto la determinata interfaccia del gateway e in fine viene anche definita una variabile BUFFER stanziata a 1024 byte (con una discreta ridondanza per sicurezza), perché basandosi sul peso complessivo dei pacchetti, che possiamo stimarlo attorno ad una media di 250 byte, e presumendo il caso peggiore in cui ci siano tutti e tre i droni collegati che rispondono in simultanea al Client si potrebbe arrivare ad occupare fino a 800 byte. Nel client e nei droni c'è anche una variabile chiamata printPacket che di default è impostata a False, essa se impostata a True permette di vedere un log dei pacchetti inviati e ricevuti molto più dettagliato.

```
17  printPacket = False
18
19  myIP = "0.0.0.0"
20  myMAC = "CC-CC-CC-00-00-01"
21
22  gatewayIP = "10.10.10.1"
23  gatewayMac = "AA-AA-AA-00-00-00"
24
25  buffer = 1024
26  gatewayPort = "8080"
```

Gateway

Il gateway rappresenta l'elemento della rete principale, il suo scopo è quello di mettere in comunicazione due o più sottoreti, con la possibilità di instradare i pacchetti con anche protocolli di comunicazione differenti.

- ➔ **Avvio**: in questa fase ci si aspetta un collegamento che può essere sia di tipo UDP da parte dei droni, sia di tipo TCP dal client. Se quest'ultimo non si è ancora collegato i droni possono solo presentarsi come disponibili per poi mettersi in attesa. Appena sarà stabilita la connessione TCP del client saranno disponibili tutte le funzioni. Durante questa prima richiesta che è di tipo “IP request” per il Client abbiamo deciso di definire un IP statico all'interno del gateway, mentre per le prime connessioni dei droni viene assegnato un IP dinamico sulla base di quelli disponibili all'interno della subnet, quest'ultimo viene poi salvato nell arp table del gateway, in modo che se in futuro ci saranno altre connessioni da parte dello stesso drone (finché il gateway è attivo) sarà riassegnato lo stesso IP
- ➔ **Esecuzione**: per gestire le richieste ci sono due funzioni (ReciveTCP e ReciveUDP) sempre in ascolto, una per la comunicazione con il client → TCP e una per la comunicazione con i droni → UDP. A seconda del tipo di messaggio ricevuto il gateway deciderà con quale tipo di protocollo instradare il messaggio al destinatario. Per l'inoltro dei pacchetti al destinatario sono presenti due funzioni una per tipologia di comunicazione. La funzione SendUDP verifica l'IP del destinatario che può esistere o meno, oppure può essere l'IP di broadcast, in questo caso il messaggio viene inoltrato a tutti i droni collegati in quel momento mentre in caso di IP non trovato si risponde al client con un errore. La funzione SendTCP, controlla che il client sia connesso e in caso affermativo gli invia il messaggio.
- ➔ **Chiusura**: si verifica quando il client decide di chiudere la connessione.

Client

É la console di un ipotetico “operatore” che deve gestire l’indirizzamento dei droni, esso comunica con il gateway con il protocollo TCP.

- ➔ **Avvio**: si invia al gateway un messaggio di tipo “IP request”, aspettando la risposta contenente il suo l’IP. Poi si mette in attesa di un comando dall’operatore, soltanto se la richiesta dell’IP è andata a buon fine.
  (FOTO menu)
- ➔ **Esecuzione**: è possibile inserire o un comando o un IP di un drone. I comandi disponibili sono 2, LIST che invia un messaggio di broadcast sulla rete dei droni, per verificare quali sono disponibili aspettando una risposta da ognuno di essi, CLOSE, procede alla chiusura. Inserendo invece un IP e successivamente una destinazione si cercherà di far partire il drone. Durante questa fase è attiva una funzione in background di ascolto che stampa sulla console i messaggi ricevuti, nella realizzazione abbiamo notato che se si stanno per ricevere 2 più messaggi quasi contemporaneamente (come le risposte al comando LIST) i messaggi vengono uniti, per questo motivo alla lettura dividiamo il messaggio.
- ➔ **Chiusura**: all’inserimento del messaggio close invia un messaggio di broadcast con l’obiettivo di non ricevere una risposta ma di “spegnere” i droni e subito dopo chiude la propria connessione anche con il gateway.

Drone

esso comunica con il gateway con protocollo UDP e conosce l'IP del client in quanto statico.

- ➔ **Avvio**: si invia al gateway un messaggio di tipo "IP request", aspettando la risposta contenente il suo l'IP. Subito dopo si presenta al gateway come "disponibile", per poi aspettare ulteriori istruzioni, soltanto se la richiesta dell'IP è andata a buon fine.
- ➔ **Esecuzione**: rimane in attesa di ricevere un messaggio, a seconda del quale procede in modo differente. Se riceve LIST invierá al Client un messaggio con la propria disponibilitá, CLOSE procede alla chiusura, in ultimo caso sta ricevendo un indirizzo di consegna. In questo caso procederá ad inviare al Client un messaggio di partenza, poi aspetta un tempo random per simulare il tempo necessario per effettuare una consegna. Successivamente invia al Client il messaggio di avvenuta consegna per poi aspettare di nuovo lo stesso tempo per ritornare in base e presentarsi come disponibile. La fase di consegna è gestita su una funzione separata dal thread principale, il quale continuerá ad ricevere messaggi ai quali risponderá sempre che il drone non è disponibile, nel caso particolare del messaggio di chiusura il drone completerá comunque la consegna, per poi spegnersi una volta tornato in base.
- ➔ **Chiusura**: chiude la comunicazione e termina il processo.

Struttura dati utilizzata

Pacchetto

Ad ogni invio di messaggi tra client drone o viceversa ci siamo avvalsi di una struttura dati precisa “costruita da noi”, la quale viene trasmessa in formato JSON.

```
packet = {
        "sourceMAC": MAC mittente,
        "destinationMAC": MAC destinatario,
        "sourceIP": IP mittente,
        "destinationIP": IP destinazione,
        "message": message,
        "time": momento esatto
        }
```

- ➔ **Header**: Possiamo definire la prima parte del pacchetto come header del pacchetto, contiene tutti i dati del destinatario e del mittente, come DestinationIP-DestinationMAC, SourceIP-SourceMAC.
- ➔ **Message**: É la sezione del pacchetto dedicata all'informazione da inviare.
- ➔ **Time**: É dove viene scritto l’istante macchina al momento della creazione del pacchetto, questa scelta è stata scaturita dalla necessità di indicare il tempo necessario per l’invio del pacchetto, di fatto alla ricezione del pacchetto si fa la sottrazione con l'istante macchina corrente e si controlla quanto tempo è stato impiegato, nel progetto utilizzando per simulare il tutto l’interfaccia di loopback i tempi che vengono fuori sono estremamenti bassi quasi nulli, ma è ovvio che in situazioni reali dove magari i droni sono distanti decine di km dalla base potremmo misurare un ritardo più consistente.

```
SourceMac : ""
DestinationMAC : ""
SourceIP : ""
DestinationIP : ""
Message : ""
```

Schema generale thread

Gateway

Nome Funzione	Protocollo utilizzato	Descrizione	Terminazione
Main Thread	None	Provvede alla chiusura di tutte le connessioni al termine dell'applicazione.	Quando termina il thread ReceiveTCP.
receiveUDP	UDP	É una funzione che rimane sempre in ascolto di un messaggio proveniente dai droni, per poi rispondere sempre al drone in caso di “IP request” oppure lo instrada al Client.	Quando il socket usato per la comunicazione UDP viene chiuso scatta un eccezione che termina il thread.
receiveTCP	TCP	É una funzione che rimane sempre in ascolto di un messaggio proveniente del Client, a seconda del tipo di messaggio risponde sempre al Client (“IP request”) oppure lo instrada al/ai drone/i.	Quando il socket usato per la comunicazione TCP viene chiuso scatta un eccezione che termina il thread.

Client

Nome Funzione	Protocollo utilizzato	Descrizione	Terminazione
Main Thread	TCP	Gestisce i comandi inseriti dall'operatore direttamente dalla console.	Alla ricezione del comando CLOSE.
receiveMessage	TCP	É una funzione che rimane sempre in ascolto di un messaggio proveniente dal gateway provenienti dai droni	Quando il socket di comunicazione viene chiuso scatta un eccezione nel thread e si arresta.

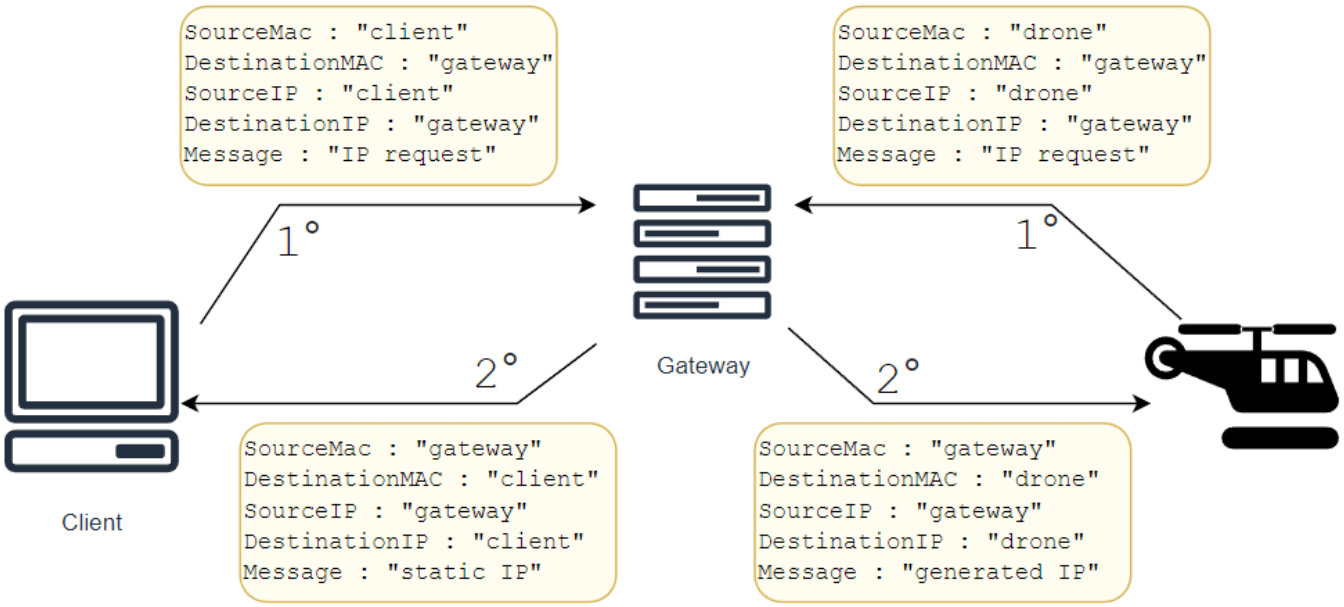
Drone

Nome Funzione	Protocollo utilizzato	Descrizione	Terminazione
Main Thread	UDP	Gestisce tutti i messaggi ricevuti dal Client e procede in base al tipo di messaggio.	Si arresta alla ricezione del comando CLOSE, aspettando però il thread shipment se presente.
shipment	UDP	É una funzione che parte ogni volta che si decide di mettere il drone in volo, effettua le operazioni di consegna aspettando dei tempi random.	Questo thread termina quando il drone torna alla base.

Diagrammi UML di sequenza

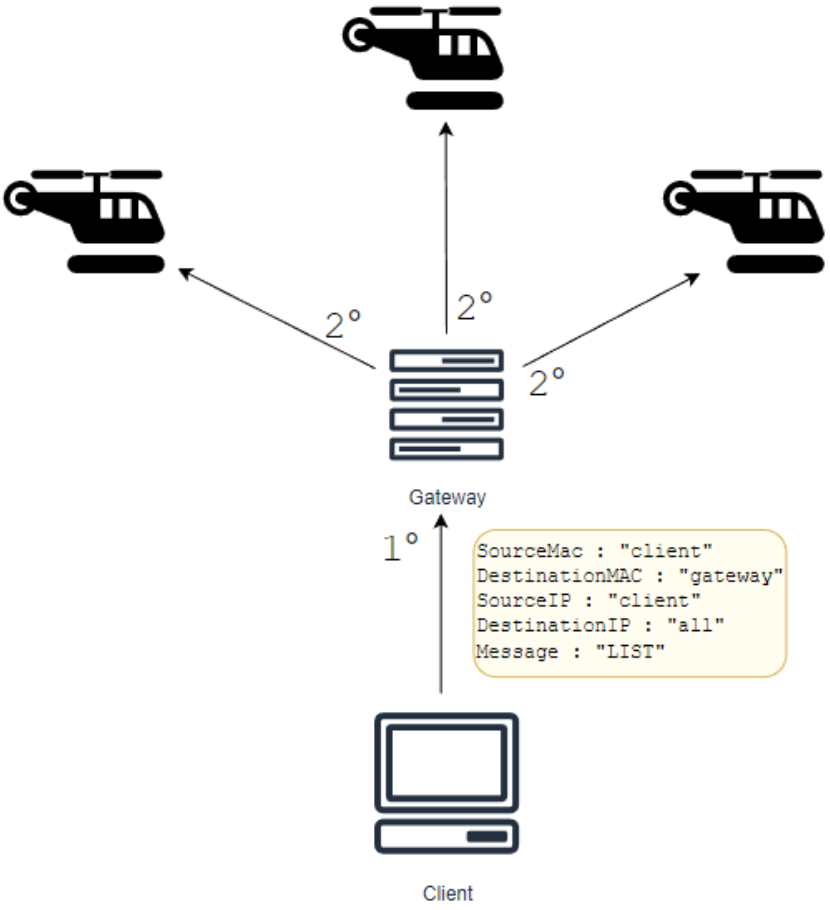
Fase 1

Prima connessione da parte del client, e da parte dei droni, inviano una “IP request" al gateway per farsi assegnare un ip.

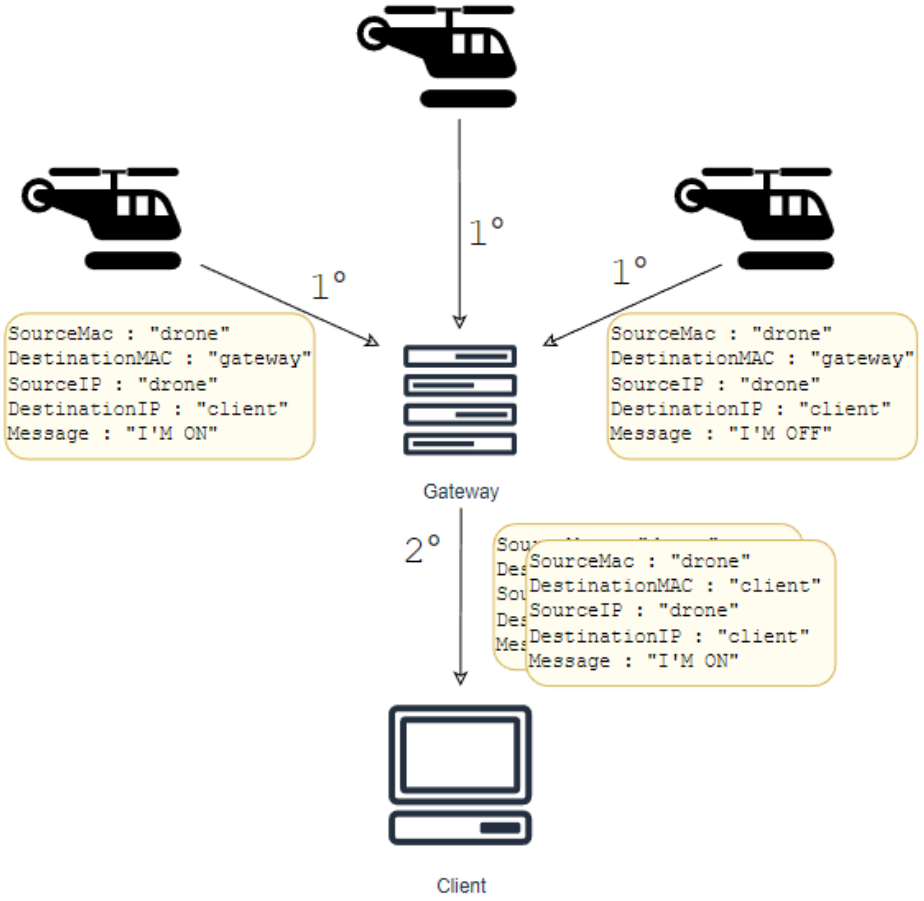


Fase 2

Il Client invia un messaggio in broadcast sulla rete dei droni per verificare chi é disponibile.

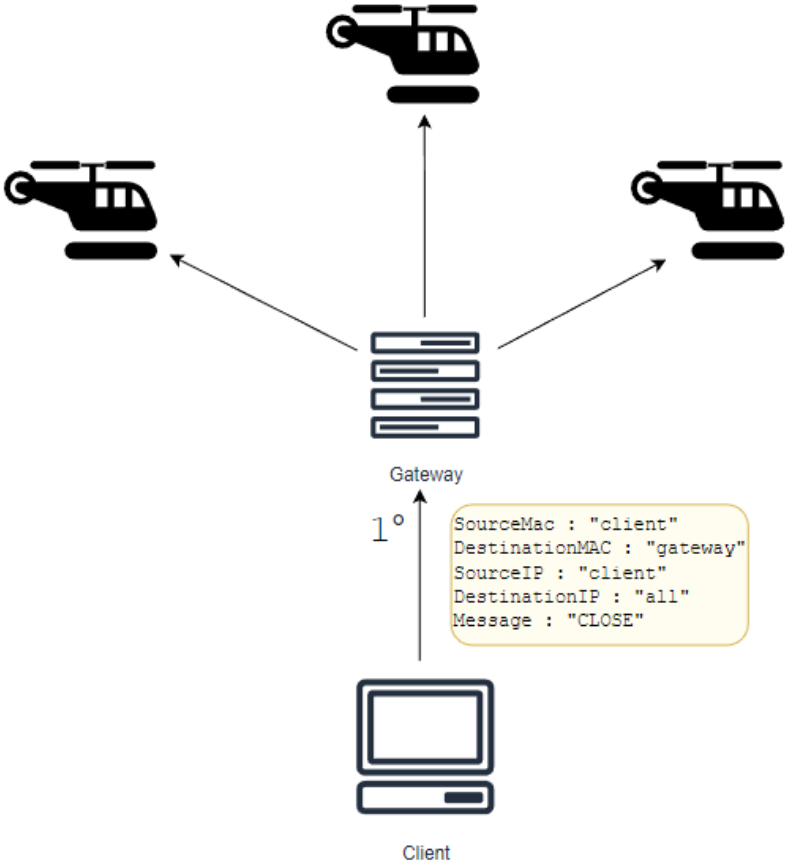


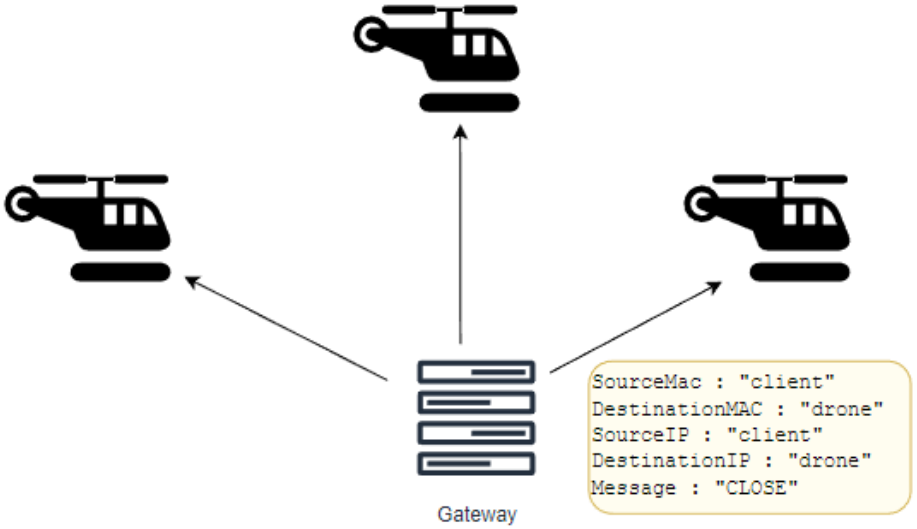
I vari droni inviano un messaggio al Client con cui specificano il loro stato.



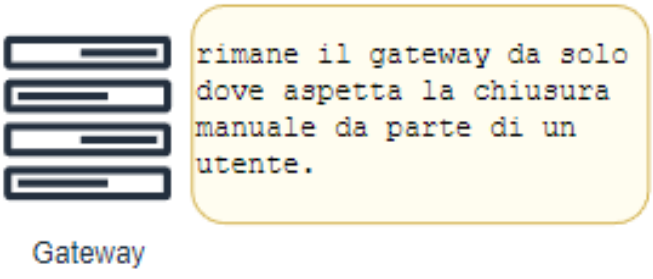
Fase 3

Il Client invia un messaggio in broadcast sulla rete dei droni con cui chiede la chiusura, infine si chiude.





Ogni drone presente sulla rete riceve il messaggio.



Guida

1. Avvio
    1.1. Avviare il sorgente Gateway.py
    1.2. Avviare il sorgente Client.py e scegliere se in modalità debug o no.
    1.3. Avviare uno o piu DroneX.py e scegliere se in modalità debug o no
2. Comunicazione
    2.1. Scegliere dal menu del client quale comando inviare
3. Chiusura
    3.1. Inviare dal client il comando CLOSE
    3.2. si potrà verificare l'effettiva chiusura dei droni e del client
    3.3. premere ENTER nella console del gateway per la chiusura definitiva.