Relazione Programmazione Di Reti
Traccia 1 - Progetto IOT

Alex Testa

A.A 2020-2021

Indice

Capitolo 1

Scelte di Progetto

Le scelte di progetto effettuate si scompongono in tre fasi, quelle relative alle centraline IOT, quelle relative al gateway e quelle relative al server centrale. In generale ho preferito mantenere ogni singola esecuzione su un singolo thread poichè i dati da inviare o da elaborare sono quantitivamente pochi e all'interno di un circuito moderno, non causerebbero ritardi di elaborazione significativi. (Inoltre preciso che la rete verrà simulata solo tramite localhost, gli ip assegnati non sono reali)

1.0.1 IOT Client

Il client IOT ha il compito di inviare tramite socket `UDP` i dati relativi alle rilevazioni del terreno, al gatway. Per simulare una tale realtà ho deciso di prendere i dati dalla località in cui mi trovo (Riccione) e questo mi è permesso tramite un `API` concessa dal sito `https://openweathermap.org/` la quale mi forniva dati veritieri sull'umidità e temperatura corrente. Inoltre ho deciso di simulare i 4 IOT in maniera che inviassero i dati l'uno dall'altro con un ritardo di 3 secondi, anche se da test effettuati, il server gatway è in grado di ricevere tutti i dati al medesimo istante. Nel caso in cui i client IOT non riuscissero a collegarsi al server, i dati non verranno persi, ma saranno salvati in un file locale `log.txt`, questo mi permette di non perdermi alcun informazione che potrebbe essere preziosa ai fini di una ricerca. In conclusione è stato settato anche un `timeOutInterval`, di 10 secondi, il quale mi permette, in caso di non risposta da parte del server, di non bloccare l'esecuzione del codice e di salvare i dati sul file di log.

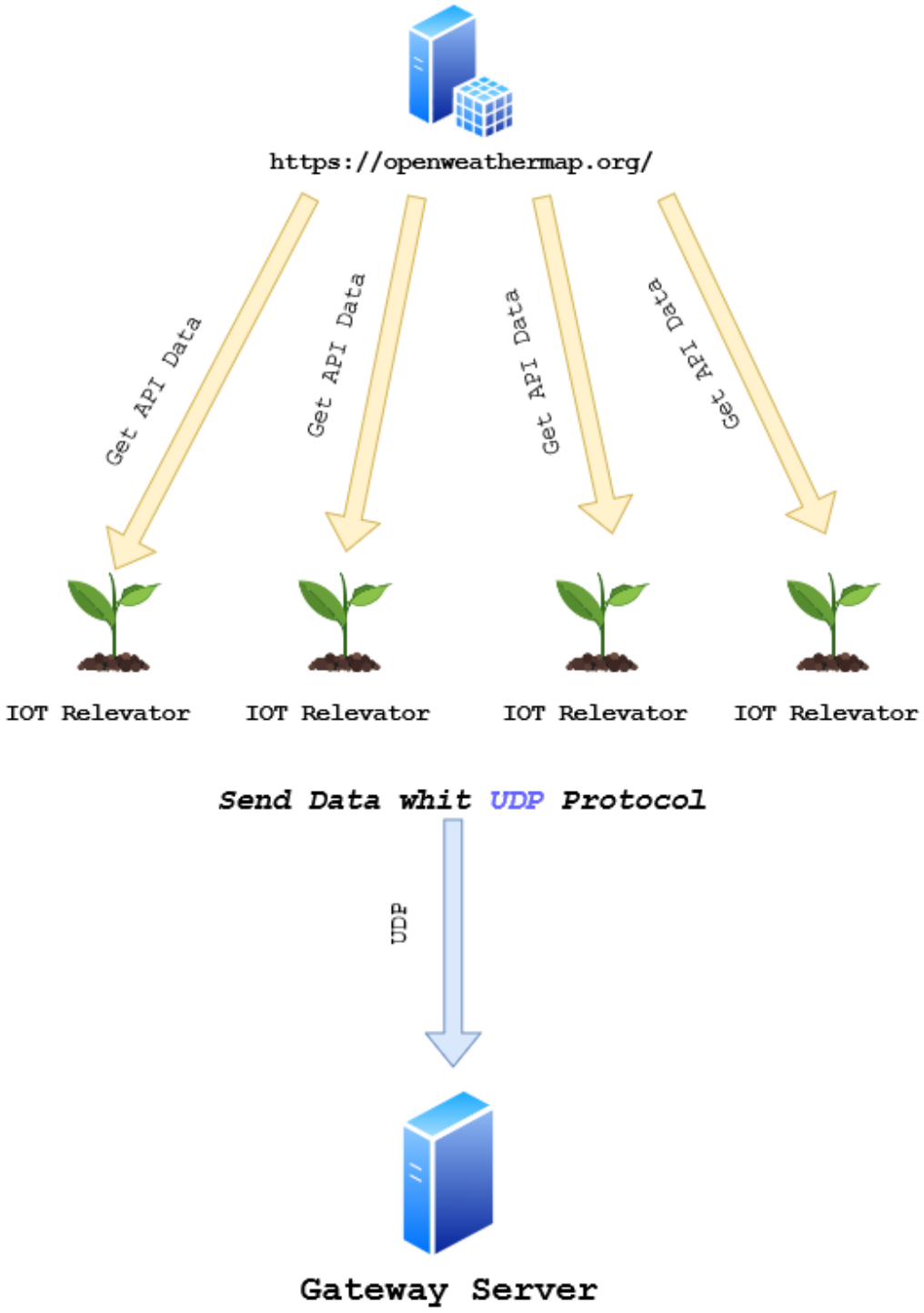

Figura 1.1: Schema esplicativo della soluzione parziale adottata

N.B ho deciso di separare alcune funzioni in un modulo distinto poichè mi permetteva una maggior chiarezza nel codice (scelta analoga è stata adottata per il gateway), inoltre ho cercato di dare ad ogni IOT un indirizzo ip differente.

1.0.2 Gateway e Server Centrale

Il `Gateway` ha il dovere di ricevere, lato server, i pacchetti `UDP` dai vari rilevatori, mentre lato client, deve gestire la connessione al server centrale tramite protocollo `TCP`. Ho deciso che il server gateway stabilirà e invierà una connessione al server centrale, solo alla ricezione dei 4 pacchetti inviati dai vari rilevatori, anch'esso nel caso di mancata connessione, o mancata risposta da parte del server centrale, salverà i dati in un file di log, aggiungendo la data corrente. Inoltre ha la funzionalità di sovrascrivere l'indirizzo ip all'interno del pacchetto, poichè come richiesto dal problema, è dotato di duplice interfaccia. Il `Server Centrale` si preoccupa soltanto di stabilire la connessione `TCP`, ricevere i dati e salvarli all'interno di un file.

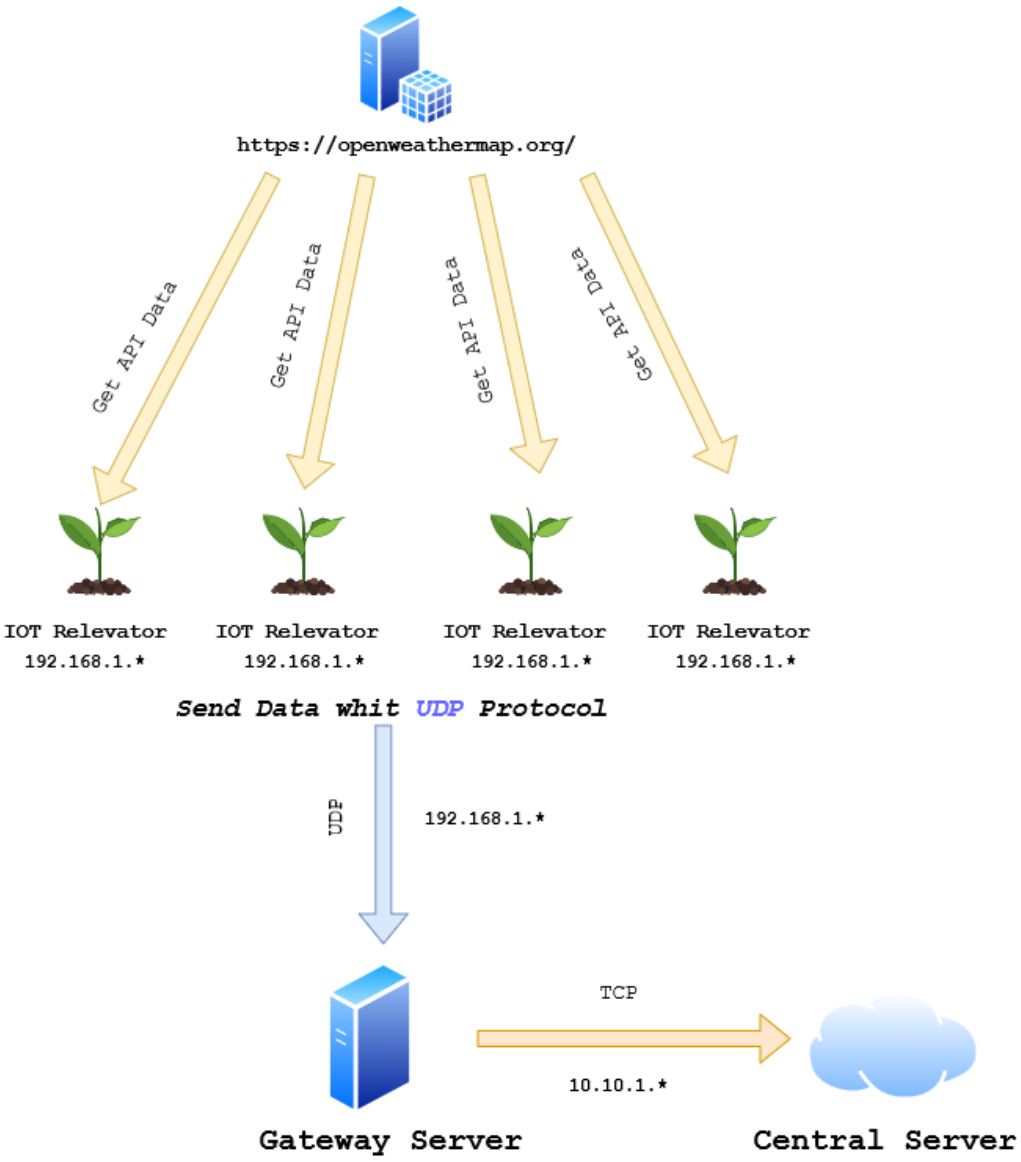

Figura 1.2: Schema esplicativo della soluzione adottata

Capitolo 2

Note Strutturali

Le principali librerie utilizzate :

- socket - per la creazione e gestione delle varie socket tcp o udp.
- datetime - per la gestione della data da salvare all'interno dei file di log.
- time - per la gestione della velocità di inoltro di un pacchetto.
- termcolors - per gestire i colori all'interno del terminale.

2.0.1 Threads

Come riportato nelle scelte di progetto, questa realtà attua tutte le sue componenti su un singolo thread poichè non necessita di multiprogrammazione.

2.0.2 Test

I test sono stati effettuati personalmente, grazie all'ausilio di un raspberry ,il quale aveva il compito di `Server Centrale`, e di un Arduino che fungeva da `Rilevatore`.

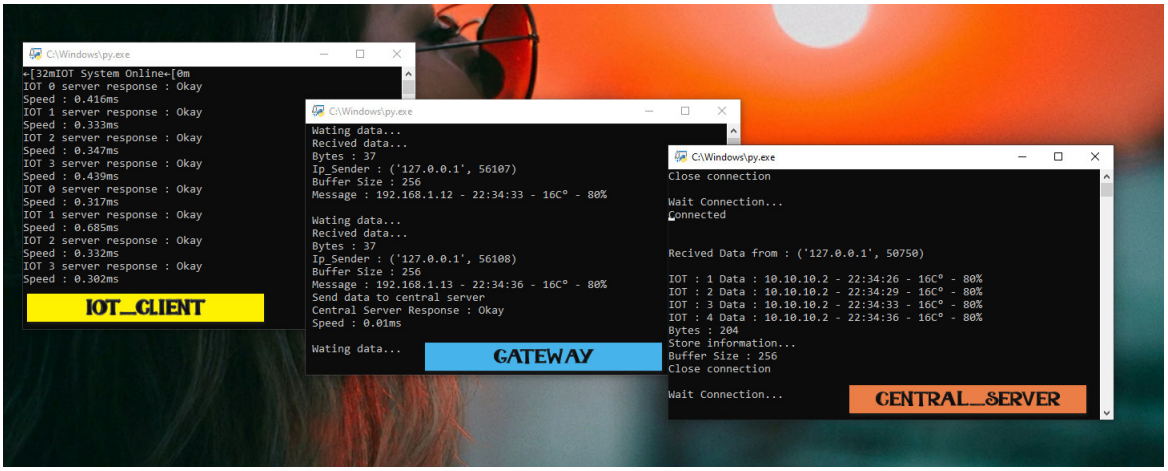

Figura 2.1: Test effettuato su windows

Per contribuire al progetto :
`https://github.com/AlexTesta00/Smart_Meter_IOT`