

Relazione per l'elaborato di Programmazione di Reti

Alessandro Brasini

22 luglio 2021

Indice

1	Introduzione	2
2	Descrizione	3
	Device (Smart IoT Meter)	3
	Gateway	4
	Server	5
3	Moduli utilizzati	6
4	Modalità di avvio	7

Introduzione

La traccia scelta è la numero 1, ovvero realizzare una simulazione di uno scenario IoT dove ci sono diversi Smart IoT Meter che rilevano la temperatura e umidità del terreno in cui sono installati. Questi si connetteranno una volta al giorno per inviare ad un gateway tramite una connessione UDP tutte le misure fatte durante il giorno. Il gateway, a sua volta, invia tutte le misure raccolte ad un server, il quale mostra su console tutte le misure dei vari device.

Descrizione

Device (Smart IoT Meter)

I device sono stati realizzati su 4 moduli separati (*device1.py*, *device2.py*, ...) dentro ai quali sono definiti, mediante costanti, i seguenti dati:

- Il numero di misure random (*N_MEASURES*) che si vogliono generare per poi inviare al gateway
- L'indirizzo IP del device (*IP_ADDRESS_DEVICE*)
- La subnet mask del device (*SUBNET_MASK_DEVICE*)
- L'indirizzo del gateway (*GATEWAY_ADDRESS*)
- La porta del gateway (*GATEWAY_PORT*)

Successivamente viene creato un oggetto **IP_Address** (definito nel modulo *IP_Address*) che descrive la configurazione IP del dispositivo (Indirizzo IP e subnet mask).

In seguito viene creato l'oggetto **device** al quale viene passato l'*ID* del dispositivo e l'oggetto *IP_Address* definito prima. Vengono poi chiamate le funzioni **generate_random_measures**, la quale genera delle misure randomiche, la funzione **print_info**, la quale stampa le informazioni del dispositivo creato, e la funzione **send_data**, la quale invia i dati letti al server. Nella funzione **send_data** si creerà un messaggio (codificato) composto nel seguente modo:

ip + subnet mask + tempo di inizio invio del pacchetto + misure da inviare

L' *IP* e la *subnet mask* sono codificati in un byte array grazie alla funzione *encode_ip_and_subnet* dichiarata nel modulo *IP_Address*, il tempo di inizio invio del pacchetto è acquisito grazie alla funzione *perf_counter* del modulo *time* per poi essere codificato, sempre in byte array con la funzione *pack* contenuta nel modulo *struct*. Le misure da inviare, invece, vengono lette dal corrispondente **file csv** per poi essere codificate.

Gateway

L'implementazione del gateway è all'interno del modulo **gateway.py**. Qua vengono istanziati due oggetti **IP_Address** che descrivono le due "interfacce" di rete del gateway, una rivolta verso la rete dei device e una verso la rete del *server*. Verrà creato un socket UDP, con porta specificata nella costante **GATEWAY_DEVICE_SIDE_PORT**, al quale poi si collegheranno i vari **device**. Quando verrà ricevuto un pacchetto da un dispositivo, si procederà a "spacchettarlo" nel seguente modo:

- I primi 4 byte contengono l'indirizzo IP del dispositivo
- I successivi 4 byte contengono la subnet mask del dispositivo
- Gli 8 byte successivi contengono il tempo di inizio di invio del pacchetto da parte del dispositivo
- I restanti byte contengono le misure inviate

Con i primi 8 byte, quindi, si crea l'oggetto **IP_Address** del dispositivo che ha inviato i dati grazie alla funzione **bytes_to_IP** (contenuta sempre nel modulo *IP_Address*). Gli 8 byte del tempo, invece, vengono convertiti in **double** grazie alla funzione **unpack** contenuta nel modulo *struct*. Una volta estratti i dati, il gateway verifica se il dispositivo che ha inviato i dati è all'interno della stessa sottorete e, per fare questo, si avvale della funzione **is_in_same_network** presente nel modulo *IP_Address*, passandogli l'oggetto *IP_Address* creato in precedenza. All'interno di questa funzione viene eseguito un **AND logico** tra l'indirizzo IP e la subnet mask del gateway, in modo da ricavare l'indirizzo di rete, e viene confrontato, a sua volta, con l'indirizzo di rete del dispositivo che ha inviato i dati (ottenuto in modo analogo).

Se i due indirizzi di rete sono uguali, il gateway accetterà le misure inviate dal dispositivo (e le formatterà secondo la modalità richiesta), altrimenti le scarterà e mostrerà un messaggio di errore.

Il gateway ripeterà questa operazione finché il numero di dispositivi univoci che hanno inviato i dati è uguale alla costante

NUMBER_OF_DIFFERENT_CLIENTS.

Se un dispositivo che ha già inviato le proprie misure cercherà di rinviarle, il gateway scarterà il messaggio. Se il dispositivo che ha inviato il pacchetto soddisfa le due condizioni scritte prima (è all'interno della stessa rete e non ha ancora inviato un pacchetto), provvederà a mostrare sulla console un messaggio di conferma e il tempo impiegato dal pacchetto per arrivare al gateway (*tempo di inizio di invio del pacchetto - tempo fine ricezione*). Ricevute tutte le misure, il gateway provvederà a stabilire una connessione con il **server** mediante la creazione di un socket TCP. Una volta stabilita provvederà a comporre il pacchetto in modo analogo a come faceva il *device*, quindi *ip + subnet + tempo di inizio di invio del pacchetto + tutte le misure dei device*, dove questa volta, come ip e subnet utilizzerà quella dell'interfaccia che comunica con il server (**GATEWAY_IP_SERVER_INTERFACE**). Una volta inviate le misure al server, chiuderà il socket verso il server.

Server

L'implementazione del server è all'interno del modulo **server.py**

Viene creato un socket TCP che sta in ascolto sulla porta designata dalla costante **SERVER_PORT**. Quando riceverà una richiesta di connessione, quest'ultima verrà accettata e resterà in attesa di un pacchetto. Quando finirà di ricevere il pacchetto, memorizzerà il tempo attuale e procederà con lo stesso procedimento che esegue il gateway. Se tutto è andato a buon fine, mostrerà su console le misure nel formato indicato.

Moduli utilizzati

- time: utilizzato per l'utilizzo del metodo *perf_counter()*
- os: utilizzato verificare se esiste
- csv: utilizzato leggere i dati scritti dai device
- struct: utilizzato per eseguire la conversione di un double in un byte array (utilizzato per inviare il tempo di inizio invio del pacchetto al gateway/server)
- sys: utilizzato "uccidere il processo"
- signal: utilizzato aggiungere un *event listener* di quando viene premuta la combinazioni di tasti **CTRL-C** in modo poi da richiamare un metodo che chiude in modo sicuro il socket/connessione aperte.
- random: utilizzato per generare misure e orari random
- datetime: utilizzato per comporre un'orario.
- socket

Modalità di avvio

Si esegue per primo il modulo **gateway.py** e poi **server.py** (funziona anche il viceversa). Successivamente si avviano i moduli **device1.py**, **device2.py**, **device3.py**, **device4.py**. E' presente anche un modulo **device5.py** il quale però ha come subnet mask "255.255.254.0", quindi non è all'interno della stessa sottorete dei device: cercando di eseguire questo modulo, il device invierà il pacchetto al gateway ma quest'ultimo lo scarterà, in quanto non è nella stessa sottorete. Si può modificare anche l'ip in, per esempio, "192.168.2.1" e subnet mask "255.255.255.0" ma verrà sempre prontamente scartato dal gateway.