Relazione Progetto Programmazione di Reti

Alessia Lombardi

A.A. 2020/2021

Indice

Capitolo 1

Introduzione

Traccia 1 - Progetto IoT: Si immagini di avere uno scenario di Smart Meter IoT che rilevano la temperatura e umidità del terreno in cui sono posizionati. I 4 dispositivi sono indicati nella figura a fianco come DEVICE. Questi dispositivi si collegano 1 volta al giorno con una connessione UDP verso il Gateway. Tramite questa connessione i dispositivi inviano le misure che hanno raccolto durante le 24 ore precedenti. Le misure consistono di un file che contiene l'ora della misura e il dato di temperatura e umidità. Una volta che i pacchetti di tutti i dispositivi sono arrivati al Gateway, il gateway instaura una connessione TCP verso un server centrale dove i valori vengono visualizzati sulla console del server nel seguente modo:
Ip address device1 - ora misura - valore temperature - valore umidità
I 4 Dispositivi IoT 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 dispositivi il cui IP Address appartiene alla stessa network dei dispositivi mentre l'interfaccia che parla con il server ha indirizzo ip appartenente alla classe 10.10.10.0/24, classe a cui appartiene anche l'IP address del server centrale. Si realizzi un emulatore Python che sfruttando il concetto dei socket visti in laboratorio consenta di simulare, utilizzando l'interfaccia di loopback del proprio PC, il comportamento di questo sistema. Si devono simulare le conessioni UDP dei device verso il Gateway e la connessione TCP del Gateway verso il Server mostrando sulla Console del server la lista dei messaggi ricevuti nel formato indicato sopra. Inoltre indicare la dimensione dei buffer utilizzati su ciascun canale trasmissivo, il tempo impiegato per trasmettere il pacchetto UDP ed il tempo impiegato per trasmettere il pacchetto TCP.

Capitolo 2

Elementi del progetto

Di seguito sono riportati gli elementi principali del progetto.

2.1 Cloud

Il Cloud svolge una funzione di server centrale a cui il Gateway invia i dati sulle rilevazioni effettuate dai Device tramite una connessione TCP. Esso mostrerà poi tali valori sulla console.
Il codice si compone di una sola funzione principale:
TCPconnessioneClient: tale funzione prende in input l'indirizzo del server (server address) e la dimensione del buffer (buffer size). Come prima cosa crea il socket TCP e lo associa al server address; dopodichè definisce la lunghezza della coda di backlog (ovvero il numero delle connessioni in entrata che sono state completate dallo stack TCP/IP ma non ancora accettate dall'applicazione).
In seguito si connette con il Gateway (Client) e prova a ricevere il messaggio. Se quest'ultimo viene ricevuto, il Cloud stampa i dati e invia un messaggio di OK al Gateway; in caso contrario il Cloud invia un messaggio di errore. Dopodichè viene chiusa la connessione.
La parte restante di codice inizializza server address, server IP e buffer size per poi stabilire la connessione TCP con il Gateway richiamando la funzione descritta sopra.

```python
#Utilizzo questa funzione per stabilire una connessione TCP
#con il Client (Gateway)
def TCPconnessioneClient(server_address, buffer_size):

    #Creo il socket e lo associo con il server_address
    serverSocket = sk.socket(sk.AF_INET, sk.SOCK_STREAM)
    serverSocket.bind(server_address)

    #Definisce la lunghezza della coda di backlog
    serverSocket.listen(1)
    #Avviso che il socket è pronto
    print ('Il Cloud è pronto...')
    connessioneGateway, indirizzo = serverSocket.accept()
    print('Connessione con il Gateway avvenuta con succeso!')

    #Provo a ricevere il messaggio
    try:

        print("Caricamento sondaggi...")
        messaggio = connessioneGateway.recv(buffer_size)
        #Stampo il messaggio
        print(messaggio.decode('utf8'))
        print("Messaggio ricevuto")
        #Avviso il Gateway che ho ricevuto il messaggio
        connessioneGateway.send("Messaggio ricevuto".encode())
        #Chiudo la connessione
        print("\nChiusura connessione\n")
        connessioneGateway.close()

    #Se non riesco a ricevere il massaggio lancio un eccezione
    except IOError:
        #Invia messaggio di risposta per file non trovato
        connessioneGateway.send(bytes("Errore","UTF-8"))
        #•Chiudo la connessione
        print("\nChiusura connessione\n")
        connessioneGateway.close()
```

Figura 2.1: Funzione per la connessione TCP al Client

2.2 Gateway

I Gateway ha due interfacce di rete:

- Svolge il ruolo di Server nella connessione UDP con i Device (Client)
- Svolge il ruolo di Client nella connessione TCP con il Cloud (Server)

Per questo motivo ho deciso di raggruppare i due ruoli principali del Gateway in due funzioni:

- **UDPconnessioneClient:** serve per stabilire una connessione UDP con i Client (Device). Prende in input l'indirizzo del server (server address), il messaggio contenente le rilevazioni (msg) e la dimensione del buffer (buffer size). Inizialmente creo il socket e lo associo a server address. Poi, per ogni Device, il Gateway riceve il messaggio contenente le rilevazioni, lo stampa e invia un messaggio di OK al Cloud. Infine la funzione chiude il socket e ritorna il messaggio completo (quello composto da tutte le rilevazioni dei diversi Device).

```
#Utilizzo questa funzione per stabilire una connessione UDP
# con i Client (Devices)
def UDPconnessioneClient(server_address, msg, buffer_size):

    #Creo il socket
    sock = sk.socket(sk.AF_INET, sk.SOCK_DGRAM)

    # associamo il socket alla porta
    print ('\n\r Inizio  %s sulla porta %s' % server_address)
    sock.bind(server_address)

    for i in range(NUMDEVICE):
        #Attendo di ricevere il messaggio
        print('\n\r In attesa di ricevere il messaggio...')
        data, indirizzo = sock.recvfrom(buffer_size)
        #Ricevo i dati
        print('Ricevuti %s bytes da %s' % (len(data), indirizzo))
        msg = msg + data.decode('utf-8') + "\n"
        #Stampo il messaggio con i sondaggi
        print (data.decode('utf-8'))
        #Aspetto 2 secondi poi rispondo
        time.sleep(2)

        sent = sock.sendto("Messaggio arrivato".encode(), indirizzo)
        print ('Inviati %s bytes al Device' % (sent))

    #Chiudo il socket
    print ('\nChiusura del socket\n')
    sock.close()
    return msg
```

Figura 2.2: Funzione per la connessione UDP al Client

- **TCPconnessioneServer:** utilizzo questa funzione per instaurare una connessione TCP tra il Gateway (Client) e il Cloud (Server). L'input della funzione è composto dall'indirizzo del server (server address), dal messaggio con i dati da trasmettere (msg) e dalla dimensione del buffer (buffer size). Prima di tutto si cra il socket e si tenta la connessione al

Cloud (Server). Se la connessione riesce, viene salvato il tempo attuale nella variabile "time in" dopodichè i dati vengono inviati al Cloud attendendo l'OK. La funzione salva il tempo attuale nella variabile "time fine", e calcola il tempo impiegato per la trasmissione nella variabile "time tot".

Se, in caso contrario, la connessione non riesce, la funzione stampa l'eccezione.

La funzione termina con la chiusura del socket.

```
def TCPconnessioneServer(server_address, msg, buffer_size):
    #Creo il socket
    clientsocket = sk.socket(sk.AF_INET, sk.SOCK_STREAM)

    #Provo a connettermi al Cloud
    try:
        clientsocket.connect(server_address)
        print("Connessione con il Cloud... ")
        print("... Invio i sondaggi al Cloud... ")
        #Attendo 2 secondi prima di inviare la richiesta
        time.sleep(2)
        time_in = time.time()
    #Se la connessione fallisce invio un messaggio di errore
    except Exception as data:
        print (Exception,":",data)
        print ("Connessione fallita. Ritenta\r\n")
        sys.exit(0)

    #Invio i dati al Cloud
    clientsocket.send(msg.encode())
    #Stampo il messaggio
    print(msg.encode())
    print("In attesa di risposta dal Cloud...")
    response = clientsocket.recv(buffer_size)
    #Prendo il tempo corrente
    time_fine = time.time()
    #Calcolo il tempo necessario ad inviare il messaggio al Cloud
    time_tot = time_fine - time_in

    print("Messaggio ricevuto: {}" .format(response.decode("utf8")))
    print ("La dimensione del buffer è: ",buffer_size)
    print ("Il tempo impiegato per trasmettere il messaggio TCP è: ",time_tot)

    #Chiudo la connessione
    print("\nChiusura connessione\n")
    clientsocket.close()
```

Figura 2.3: Funzione per la connessione TCP al Server

Il restante codice inizializza le variabili da passare alle funzioni e richiama quest'ultime.

2.3 Devices

I Device sono i dispositivi che si occupano di effettuare le rilevazioni e che, una volta al giorno, tramite una connessione UDP inviano al Gateway (Server) i dati raccolti durante le 24 ore precedenti. Al fine di evitare ripetizioni di codice per ciascuna implementazione di Device, ho pensato di raggruppare in **interfacciaDevice** le funzioni comuni a tutti i dispositivi:

- **UDPconnessioneServer:** questa funzione permette di connettere i Device (Client) al Gateway (Server) tramite una connessione UDP. L'input è composto dall'indirizzo del Server (server address), dal messaggio contenente le rilevazioni (msg) e dalla dimensione del buffer (buffer size).

  Per prima cosa viene creato il socket e si prova ad inviare il messaggio contenente le rilevazioni al Gateway. Una volta inviato il messaggio, si attende l'OK dal Server.

  Per calcolare il tempo totale di trasmissione, ho sottratto alla variabile "time fine" (contenente il tempo dopo aver ricevuto la risposta dal Server) la variabile "time in" (contenente il tempo all'inizio della trasmissione).

  Se il messaggio non viene inviato al Gateway, stampo le informazioni relative all'eccezione.

  Infine viene chiuso il socket.

```python
#Utilizzo questa funzione per connettere un qualsiasi device al server
def UDPconnessioneServer(server_address, msg, buffer_size):
    #Creo il socket
    sock = sk.socket(sk.AF_INET, sk.SOCK_DGRAM)

    #Provo ad inviare il messaggio del device al gateway
    try:
        #Invio il messaggio
        print ('Invio del sondaggio al Gateway...')
        #attende 2 secondi prima di inviare la richiesta
        time.sleep(2)
        time_in = time.time()
        sock.sendto(msg.encode(), server_address)

        #Ricevo la risposta dal Gateway
        print('In attesa di risposta...')
        data, server = sock.recvfrom(buffer_size)
        #Prendo il tempo attuale
        time_fine = time.time()
        #Calcolo il tempo totale
        time_tot = time_fine - time_in
        #Aspetto 2 secondi poi stampo 'Mesaggio ricevuto'
        time.sleep(2)
        print ('Messaggio ricevuto: "%s"' % data.decode('utf8'))
        print ("La dimensione del buffer è: ",buffer_size)
        print ("Il tempo impiegato per trasmettere il messaggio UDP è: ",time_tot)

    except Exception as info:
        print(info)
    finally:
        print ('Chiusura del socket')
        sock.close()
```

Figura 2.4: Funzione per la connessione UDP al Server

- **sondaggi:** utilizzo questa funzione per estrapolare il messaggio da inviare dal file contenente i dati sulle rilevazioni del Device corrispondente. La funzione prende in input il nome del file e l'IP del Client, dopodichè cerca il file nella cartella 'Sondaggi' e lo apre leggendone il contenuto e formattando il messaggio. Una volta terminata la lettura, il file viene chiuso e la funzione ritorna il messaggio da inviare al Gateway.

```
#Utilizzo questa funzione per estrapolare il messaggio dal file contenente i messaggi
def sondaggi (file_name, client_IP):

    #Cerco il file nella cartella Sondaggi e lo apro
    filePath = "Sondaggi/" + file_name
    file = open(filePath, "r")
    print("Lettura dati...")
    #Aspetto 2 secondi prima di leggere
    time.sleep(2)
    messaggio = ""

    while True:
        #Leggo le righe del file
        riga = file.readline()
        #Controllo se sono arrivata alla fine del file ed in tal caso esco
        if (riga == ""):
            break;
        #Se non sono arrivata alla fine del file formatto il messaggio
        else:
            messaggio =  messaggio + client_IP + " " + riga + "\n"

    #Chiudo il file
    file.close();
    print("Lettura sondaggi completata")

    return messaggio
```

Figura 2.5: Funzione per estrapolare il messaggio dal file

Ora ciascun Device deve soltanto inizializzare le variabili da passare alle funzioni di interfacciaDevice, reperire il messaggio tramite **interfacciaDevice.sondaggi** e instaurare la connesione con **interfacciaDevice.UDPconnessioneServer**.

```python
import interfacciaDevice as iD

#Setto l'IP del Device1
client_IP = "192.168.1.4"
#Prendo l'indiizzo del server (Gateway)
server_address = ("localhost", 10005)
#Imposto la dimensione del buffer
buffer_size = 4096
#Leggo dal file 'SondaggiDevice1' i sondaggi effettuati
file_name = "SondaggiDevice1.txt"
messaggio = iD.sondaggi(file_name, client_IP)
#Richiamo la funzione per la connessione con il Server
iD.UDPconnessioneServer(server_address, messaggio, buffer_size)
```

Figura 2.6: Codice di Device1

Capitolo 3

Guida all'utente

Per prima cosa bisogna lanciare il Cloud che si mette in attesa della connessione con il Gateway.
Dopodichè si esegue il Gateway (in un'altra console) il quale si mette in attesa di ricevere i messaggi dai Device.
A questo punto si lanciano i vari Device, ognuno dei quali reperise i dati, li invia al Gateway, attende la conferma dell'arrivo del messaggio e stampa la dimensione del buffer e il tempo impiegato per la trasmissione.
Il Gateway una volta che ha ricevuto tutti e 4 i messaggi, si connette con il Cloud e invia i dati, poi chiude la connessione.
Infine il Cloud stampa sulla console i sondaggi e anch'esso chiude la connessione.