Relazione per il progetto di "Programmazione di Reti"

Simone Rega
(0000915995)

24 luglio 2021

Indice

Capitolo 1

Introduzione

Il progetto consegnato rispecchia la Traccia 2 degli elaborati mostrati a lezione.

Il progetto si pone nello scenario di un insieme di device IoT, chiamati SmartMeter IoT, che mandano dati ad un Cloud Server, passando però prima da un Gateway si pone nel mezzo tra i due e ne gestisce le richieste.

In questo progetto si farà uso di socket e di protocolli UDP e TCP.

Capitolo 2

Descrizione Architetturale

2.0.1 IoT Devices

Gli SmartMeter IoT sono device che rilevano un numero ben definito di volte ogni giorno la temperatura e l'umidità del terreno in cui sono inseriti.

Questi Device si collegano una sola volta al giorno ad un Gateway per inviare le misure che hanno raccolto nelle 24 ore.

Per questa connessione client-server viene utilizzato un protocollo UDP.

```
import DeviceUtils as dev

#Constants
deviceIP = "192.168.1.2"
deviceSubnet= "255.255.255.0"
filePath = "DataDevice1.txt"
gatewayAddress = ("localhost", 6969)

#Generate random measures, param = quantity
dev.getRandomMeasures(4)
#Obtain the data from the file
message = dev.getDataFromFile(filePath)
#Sending the data to the gateway
dev.sendDataToGateway(deviceIP,deviceSubnet,gatewayAddress, message)
```

I quattro client IoT sono stati realizzati separatamente *(Device1.py, Device2.py, Device3.py, Device4.py)* i quali usufruiscono di due classi utility: DeviceUtils e AddressTools.

Le funzioni principali di DeviceUtils sono:

- **getRandomMeasures(quantity)**: dato un numero in ingresso che indica la quantità, la funzione provvederà a generare 4 file, o a sovrascriverli nel caso esistano già, e li popolerà con delle misure verosimili generate random.

- **getDataFromFile(filePath)**: dato un percorso di un file, la funzione procede ad aprire un file in lettura e leggerne il contenuto, restituendo una stringa in output contenente le rilevazioni.

- **sendToGateway(deviceIP,deviceSubnet,gatewayAddress, message)**: la funzione crea un socket per una connessione UDP con il Gateway. Una volta stabilita la connessione viene inviato un pacchetto, tramite un buffer di 1024 Byte, contenente IP del device, Subnet Mask del device e il messaggio contente le misure rilevate.

Le funzioni principali di AddressTools sono:

- **getAddressEncoded**:restituisce l'indirizzo IP e la subnet mask contatenati e convertiti in byte, quindi in totale 16 Bytes.

- **convertBytesToIP(ipBytes, subnetBytes)**: dati in input un IP e una Subnet Mask in Bytes, la funzione si preoccupa di restituire un oggetto AddressTool contenente un ip e una subnet come stringhe.

2.0.2 Gateway

Il Gateway fa da tramite tra i Device IoT e il Cloud Server. Prima si mette in ascolto sull'interfaccia di rete in cui ci sono i Device, attendendo che questi gli inviino i dati, per poi aprire una connessione TCP con il Cloud Server inviando tutte le rilevazioni ottenute dai vari Device.

Il Gateway è stato realizzato nel modulo Gateway.py, usufruendo delle librerie socket e time.

2.0.3 Cloud Server

Il Cloud Server apre semplicemente una connessione TCP cpn il Gateway sull'interfaccia 10.10.10.0, mettendosi in ascolto e aspettando la ricezione del messaggio del Gateway contenente i dati delle rilevazioni.

```python
def startGateway():
    message=""
    NUM_DEVICE=int(4)
    socket_device = socket(AF_INET, SOCK_DGRAM)
    socket_device.bind(("localhost",6969))
    print('Ready to Serve. Listening for devices...')

    # Waiting devices's detections
    ipAlreadyListened=list()
    try:
        #Keep listening until my list is filled with all the devices
        while len(ipAlreadyListened)<NUM_DEVICE:
            data, address = socket_device.recvfrom(1024)
            deviceIP=AddressTools.convertBytesToIP(data[:4],data[4:8])
            deviceData=(data[8:]).decode("utf8")
            #Checking if an IP is already in my list or subnet isn't 255.255.255.0
            if (deviceIP.ip not in ipAlreadyListened) or (deviceIP.subnet!="255.255.255.0"):
                ipAlreadyListened.append(deviceIP.ip)
                indexDevice=len(ipAlreadyListened)
                print("{}° Device = {}".format(str(indexDevice),deviceIP.ip))
                messageReply = "Detection arrived"
                socket_device.sendto(messageReply.encode(), address)
                #Filling the final message with the current device data
                for line in deviceData.split('\n'):
                    if(line!=""):
                        message+=("{} - {}\n".format(deviceIP.ip,line))
    except Exception as e:
        print(e)
    finally:
        socket_device.close()
    print("Measures delivered!")
    return message
```

Figura 2.1: Metodo che fa partire il gateway e lo mette in ascolto dei socket IoT

```python
def sendToCloudServer(message):
    PORT=6942
    BUFFER = 1024
    print("Trying to connect to Cloud Server on port {}".format(PORT))
    try:
        socket_cloud = socket(AF_INET, SOCK_STREAM)
        socket_cloud.connect(('localhost', PORT))
        initialTime = time.perf_counter()
        socket_cloud.send(message.encode())
        data = socket_cloud.recv(BUFFER)
        print("Waiting the Cloud Server's answer...")
        elapsedTime = round(time.perf_counter() - initialTime,5)
        print("Received Message: {}" .format(data.decode("utf8")))
        print("TCP message's sending time {} and the size of used buffer is {}" .format(elapsedTime, BUFFER))
    except Exception as e:
        print(e)
    finally:
        print("Closing connection")
        socket_cloud.close()
```

Figura 2.2: Metodo che manda i dati al Cloud Server

```python
def connectToGateway():
    serverSocket = socket(AF_INET, SOCK_STREAM)
    # Bind Socket to port & ip
    serverSocket.bind(("localhost", PORT))

    # Listen Client request
    serverSocket.listen(1)
    print("SERVER CLOUD")
    print("Waiting on interface 10.10.10.0 on port {}..." .format(PORT))

    # Waiting Gateway connection
    gatewayConnection, address = serverSocket.accept()
    print("Gateway connected!")
    try:
        # TCP Server socket
        print("All the measures from devices down here :\n")
        message = gatewayConnection.recv(BUFFER)
        print(message.decode("utf8"))
        gatewayConnection.send(("Ok, measures received!").encode())
    except Exception as e:
        print(e)
    finally:
        print("Closing connections")
        gatewayConnection.close()
        serverSocket.close()
```

Capitolo 3

Spiegazione dettagliata del Funzionamento

- **1° Passo**: Avviamo il Gateway e il Cloud Server che aspettano entrambi in ascolto sulla propria interfaccia che qualcuno gli invii dei dati
- **2° Passo**: Avviamo i 4 Device IoT che leggono le misure rilevate e le inviano al Gateway, indicando anche il tempo impiegato per la connessione UDP e il buffer utilizzato
- **3° Passo**: Una volta fatti partire tutti i moduli pyton, il Gateway riceverà le misure dei quattro Device IoT (è presente un controllo che non permette ai Device di mandare più volte la misura lo stesso giorno, in modo da poter leggere tutti e quattro i socket). Alla fine il Gateway stesso procederà a inviare al Cloud Server, tramite una connessione TCP, tutte le misure indicando anche il tempo e la dimensione del Buffer.