

IT Technology

Assignment 22

Synchronised transmission from client to server



.....

University College

Authors

Martin E. Kristensen

mart59p5@edu.eal.dk

Sebastian Thomle Mason

seba7286@edu.eal.dk

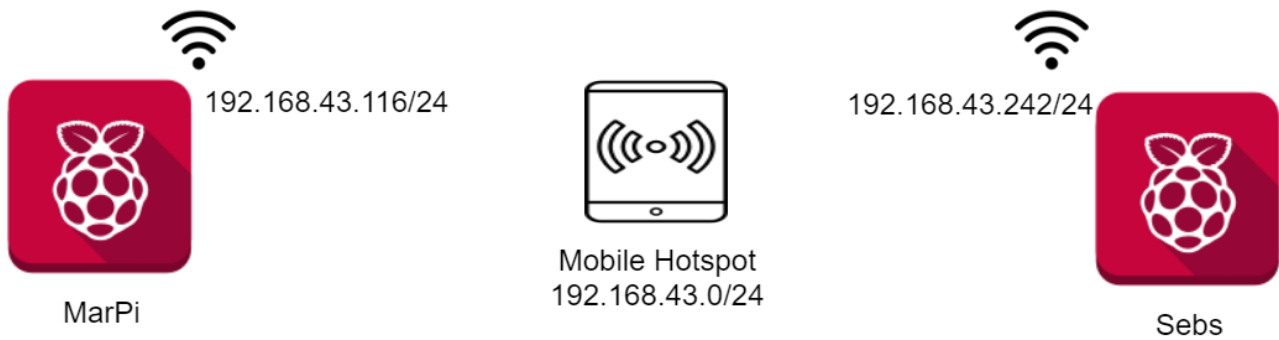
Sunday 09 Dec 2018

Table of Contents

A network diagram with brief explanation	2
Illustrate graphically and explain the developed application layer protocol	3
Show and explain the output from the server program when client and server are executed	4
Show in Wireshark that the transmitted data can be monitored in plain text	5
Appendix	6
Client:	6
Server:	7

A network diagram with brief explanation

The diagram shows two Raspberry Pi's and a mobile hotspot. Each Pi can be either the server or the client and they are connected to the same mobile hotspot.

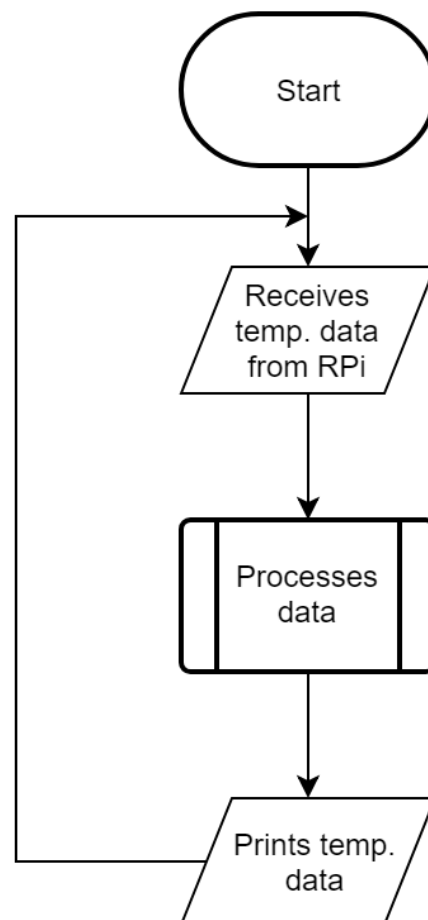


Illustrate graphically and explain the developed application layer protocol

The server receives the data one character at a time and saves each character to a list. It does this until it encounters the string “q” (short for quit) in the datastream. The server then removes the “q” entry from the list, and prints the list. It finally resets the list, emptying it.

To print the list, the server decodes every character and then prints the combined string. The user can interrupt the program at any time.

See the code in the Appendix.



Show and explain the output from the server program when client and server are executed

The client sends the RPi's CPU temperature, between 0 and 100, every second. The server then prints the temperature converted like shown below.

```
pi@sebs:~/Documents/Networking/ass22 $ python3 server.py
Awaiting connection on IP: 0.0.0.0 Port: 65433
Connection from: ('192.168.43.253', 38790)
44.008
45.084
44.546
```

Show in Wireshark that the transmitted data can be monitored in plain text

The image shows a Wireshark packet capture interface. The top pane displays a list of packets. Packet 563 is selected, showing details for Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The data field shows a sequence of bytes: 34 34 30 30 38 0a 71. The bottom pane shows the packet data in hexadecimal and ASCII. The ASCII column shows the text '44546q44008q44008q44008q44008q44008q'.

No.	Time	Source	Destination	Protocol	Length	Info
5...	10.5568...	192.168.43...	192.168.43...	TCP	66	65433 → 38788 [ACK] Seq=1
5...	11.5607...	192.168.43...	192.168.43...	TCP	73	38788 → 65433 [PSH, ACK]

Frame 563: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on i
Ethernet II, Src: Raspberr_65:9e:d9 (b8:27:eb:65:9e:d9), Dst: Raspberr_3f
Internet Protocol Version 4, Src: 192.168.43.253, Dst: 192.168.43.242
Transmission Control Protocol, Src Port: 38788, Dst Port: 65433, Seq: 29,
Data (7 bytes)
Data: 34343030380a71
[Length: 7]

Offset	Hex	ASCII
0000	b8 27 eb 3f c5 28 b8 27 eb 65 9e d9 08 00 45 00	. ' . ? . (. ' . e E .
0010	00 3b ae 40 40 00 40 06 b3 3c c0 a8 2b fd c0 a8	. ; . @ @ . @ . < . . + . .
0020	2b f2 97 84 ff 99 52 34 23 b9 21 0f 79 63 80 18	+ R4 # . ! . y c . .
0030	00 e5 2c f1 00 00 01 01 08 0a a6 4c cb 6f b4 19	. . , L . o . .
0040	94 d4 34 34 30 30 38 0a 71	. . 44008 . q

Wireshark · Follow TCP Stream (tcp.stream eq 1) · wlan0 — □ ×

44546
q44008
q44008
q44008
q44008
q44008
q

6 client pkts, 0 server pkts, 0 turns.

Appendix

Client:

```
#!/usr/bin/env python3

import socket
from time import sleep
from random import randint

HOST = '192.168.43.242'      # The server's hostname or IP address
PORT = 65433                # The port to send data to on the server
mySensorReadings = 'go'    # The application layer protoll

def temperatureSensor():
    with open("/sys/class/thermal/thermal_zone0/temp", "r") as f:
        t = f.readline()

    #out = randint(0, 100)
    return str(t) + "q"

def pad(i):
    if len(i) == 1:
        return "00" + str(i)
    elif len(i) == 2:
        return "0" + str(i)
    else:
        return str(i)

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

input("Press enter to begin")
while True:
    try:
        mySensorReadings = temperatureSensor()
        print(mySensorReadings)
        s.sendall(mySensorReadings.encode('utf-8'))
        sleep(1)

    except KeyboardInterrupt:
        s.close()
        exit()
```

Server:

```
#!/usr/bin/env python3
# TCP server. Will listen and receive data until client closes connection
# Adapted by Per dahlstrøm
import socket          # Fetch the socket module
from time import sleep

HOST = '' # Standard loopback interface address (localhost)
PORT = 65433 # Port to listen on (non-privileged ports are > 1023)
receivedData = []

def decode(s):
    return s.decode('utf-8')

def processData(lst):
    return list(map(decode, lst))

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((HOST, PORT))
    s.listen()
    print('Awaiting connection on IP: ', s.getsockname()[0],
          ' Port: ', s.getsockname()[1])
    connection, fromAddress = s.accept() # Wait and create connection object
    print('Connection from:', fromAddress)
    while True:
        receivedData.append(connection.recv(1))
        if receivedData[-1].decode("utf-8") == "q":
            receivedData.pop()
            receivedData = processData(receivedData)
            print("".join(receivedData))
            receivedData = []

except Exception as e:
    print(e)
    connection.close()
    print('Connection closed')
    s.close()
    print('Socket closed')
```