IT Technology

Assignment 21 Extended application layer protocol

University College

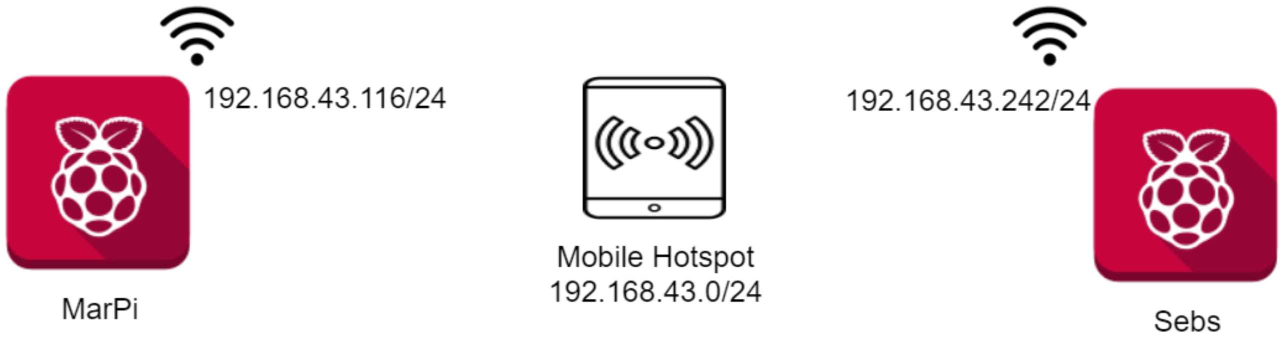
Authors
Martin E. Kristensen
mart59p5@edu.eal.dk
Sebastian Thomle Mason
seba7286@edu.eal.dk

Sunday 02 Dec 2018

Table of Contents

A network diagram with brief explanation

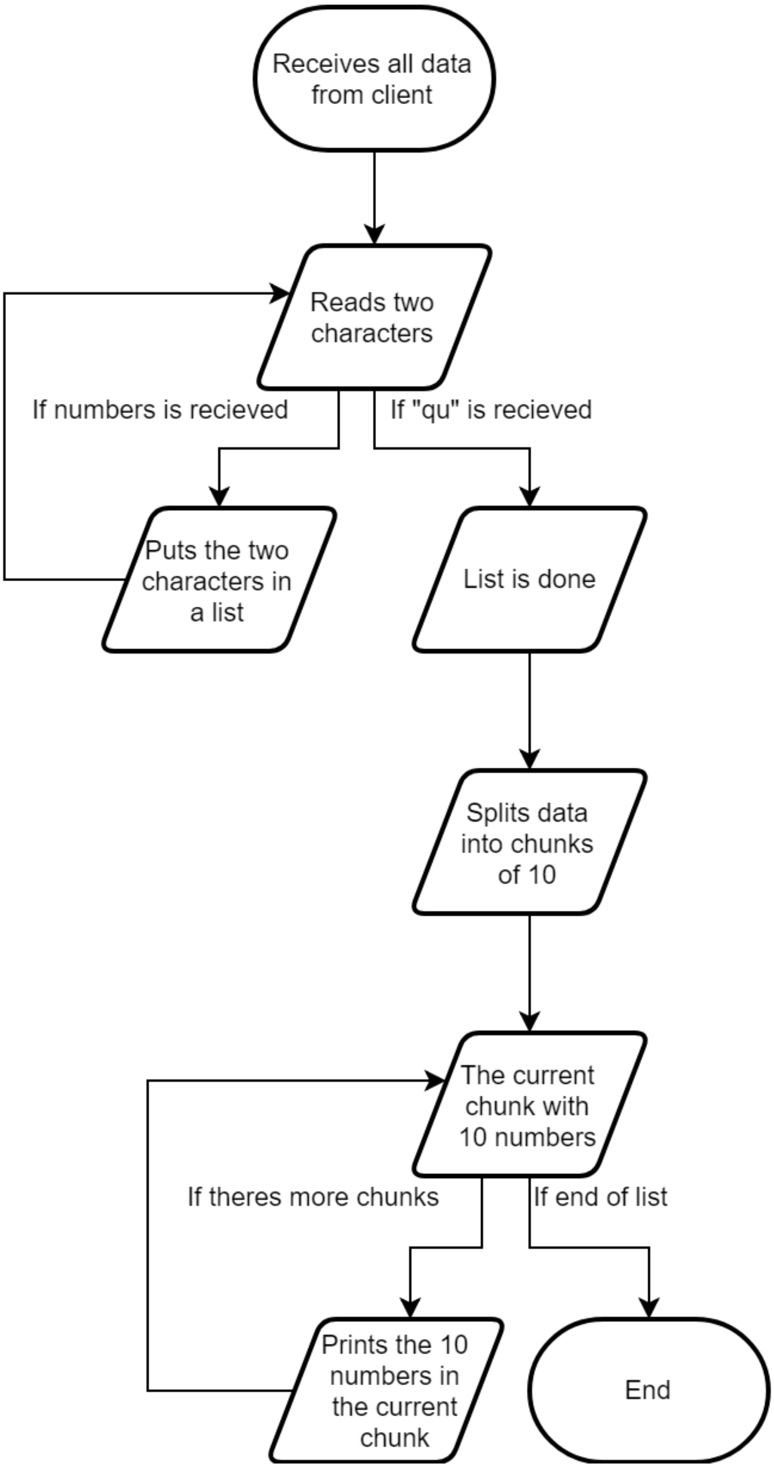
The diagram shows two Raspberry Pi’s and a mobile hotspot. Each Pi can be either the server or the client and they are connected to the same mobile hotspot.



Illustrate graphically and explain the developed application layer protocol

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

To print the list, the server first splits the data into chunks of 10. Then it prints the 10 numbers in the current chunk and continues to the next chunk. This continues until the end of the data is reached.

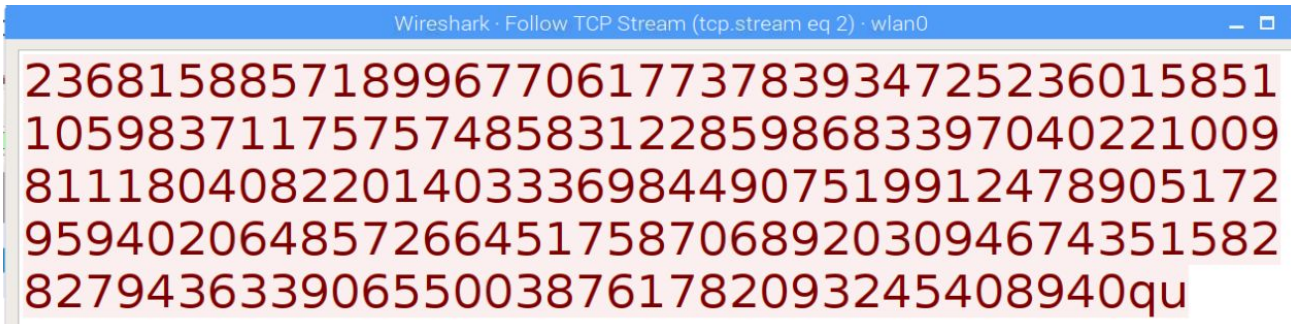

Receives all data from client
Reads two characters
If numbers is recieved
If "qu" is recieved
Puts the two characters in a list
List is done
Splits data into chunks of 10
The current chunk with 10 numbers
If theres more chunks
If end of list
Prints the 10 numbers in the current chunk
End

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

The client sends a list with 100 numbers, that contains random numbers between 0 and 99. The server prints the list like shown.

```
pi@sebs:~/Documents/Networking/ass21 $ python3 server.py
Awaiting connection on IP:  0.0.0.0  Port:  65433
Connection from: ('192.168.43.116', 37792)
23 68 15 88 57 18 99 67 70 61
77 37 83 93 47 25 23 60 15 85
11 05 98 37 11 75 75 74 85 83
12 28 59 86 83 39 70 40 22 10
09 81 11 80 40 82 20 14 03 33
69 84 49 07 51 99 12 47 89 05
17 29 59 40 20 64 85 72 66 45
17 58 70 68 92 03 09 46 74 35
15 82 82 79 43 63 39 06 55 00
38 76 17 82 09 32 45 40 89 40
```

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



```
23681588571899677061773783934725236015851
10598371175757485831228598683397040221009
81118040822014033369844907519912478905172
95940206485726645175870689203094674351582
82794363390655003876178209324540894 0qu
```

Appendix

Client:

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

import socket
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(n):
    lst = []
    for i in range(n):
      lst.append(randint(0,99))

    lst = list(map(pad,lst))
    lst.append("qu")

    return lst

def pad(i):
    if i < 10:
      return "0" + str(i)
    else:
      return str(i)


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


while True:
  n = input("How many numbers? ")
  if n == "it":
    s.sendall(n.encode('utf-8'))
    exit()
  try:
    n = int(n)
    mySensorReadings = temperatureSensor(n)
    mySensorReadings = "".join(mySensorReadings)
    s.sendall(mySensorReadings.encode('utf-8'))

  except:
    print("Please enter a valid number!")
```

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

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


def printData(lst):
    for i in range(0, len(lst), 10):
        for j in range(10):
            print(lst[i+j], end=" ")
        print()


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


receivedData = []
lst = map
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 DataCommingIn:
    receivedData.append(connection.recv(2))
    if receivedData[-1].decode('utf-8') == "qu":
        receivedData.pop()
        receivedData = list(map(decode, receivedData))
        printData(receivedData)
        receivedData = []

    if len(receivedData) > 0:
        if receivedData[-1].decode('utf-8') == "it":
            DataCommingIn = False

connection.close()
print('Connection closed')
s.close()
print('Socket closed')
```