Inlämning 2, ET1524

Linus Jansson, 001218-8694

May 16, 2022

Task A

0.0.1 Task A1

```
# UDP Sender
# Max Dahlgren, Linus Jansson

from random import randrange
import socket
from time import sleep

serverPort = 12000
receiverIp = "127.0.0.1"


def getData(length):
    string = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    out = ""
    for i in range(length):
        out += string[randrange(0, len(string)-1, 1)]
    return out


msgToSend = getData(1430)
nrOfMsg = 9999
msgSpeed = 1000000

print("UDP target IP: %s" % receiverIp)
print("UDP target Port: %s" % serverPort)
print("UDP target Message: %s" % msgToSend)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)


for i in range(nrOfMsg):

    failed = False
    randInt = randrange(0, 1000, 1)
    if randInt == 154:
        failed = True
        i += 1
    msg = f"{10000+i}{msgToSend}####"
    # print(msg)

    if failed:
        i -= 1
    sock.sendto(msg.encode(), (receiverIp, serverPort))
    sleep(1/msgSpeed)
```

0.0.2 Task A2

```
# Max Dahlgren, Linus Jansson

import socket

serverPort = 12000
```

```
ip = "0.0.0.0"

sock = socket.socket(socket.AF_INET,
                     socket.SOCK_DGRAM)
sock.bind((ip, serverPort))

nextId = 10000

while True:
    data, addr = sock.recvfrom(1500)
    msgId = int(data[0:5])
    msg = data[5:-4]
    if msgId != nextId:
        print(f"ERROR EXPECTED MSG {nextId} BUT RECVIED {msgId}")
        break;
    nextId += 1
    print(f"received message id: {msgId}: {msg[0:50]}...")
```

0.0.3 Task A3

Jag använde sleep funktionen för att implementera ett paket delay. Jag har en variabel i koden som heter msgSpeed, som anger antal medelanden per sekund. Med det talet så kan jag ta $1/msgSpeed$ för att få fram ett delay i milisekunder

Task B

0.1 Task B1

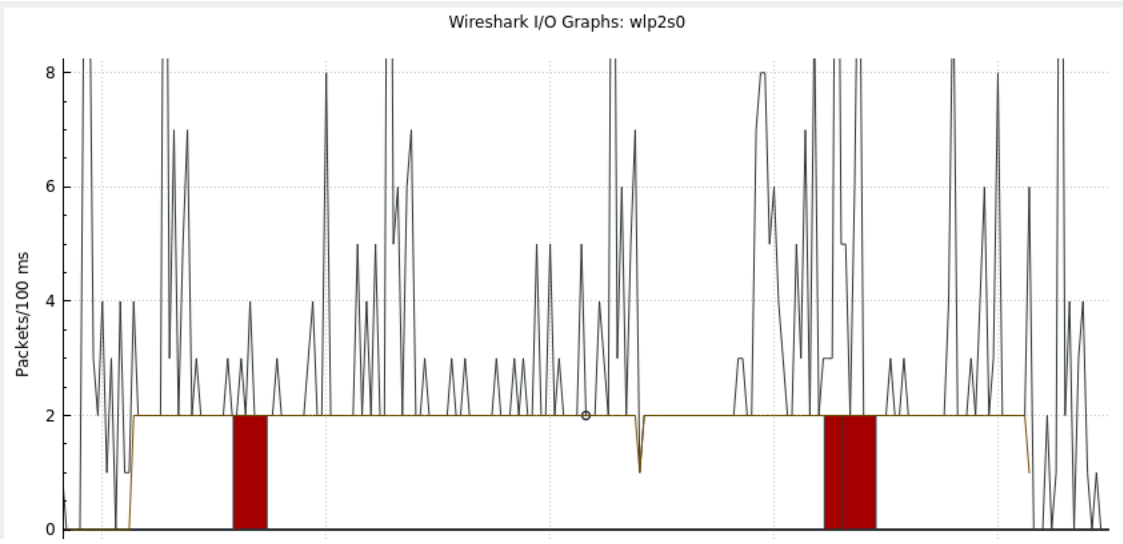
0.1.1 Task B1a

Eftersom den räknas ut som beskrivit innan, så valde jag 20 paket per sekund, och 400 totala paket, så då får sleep 50 ms.

0.1.2 Task B1b

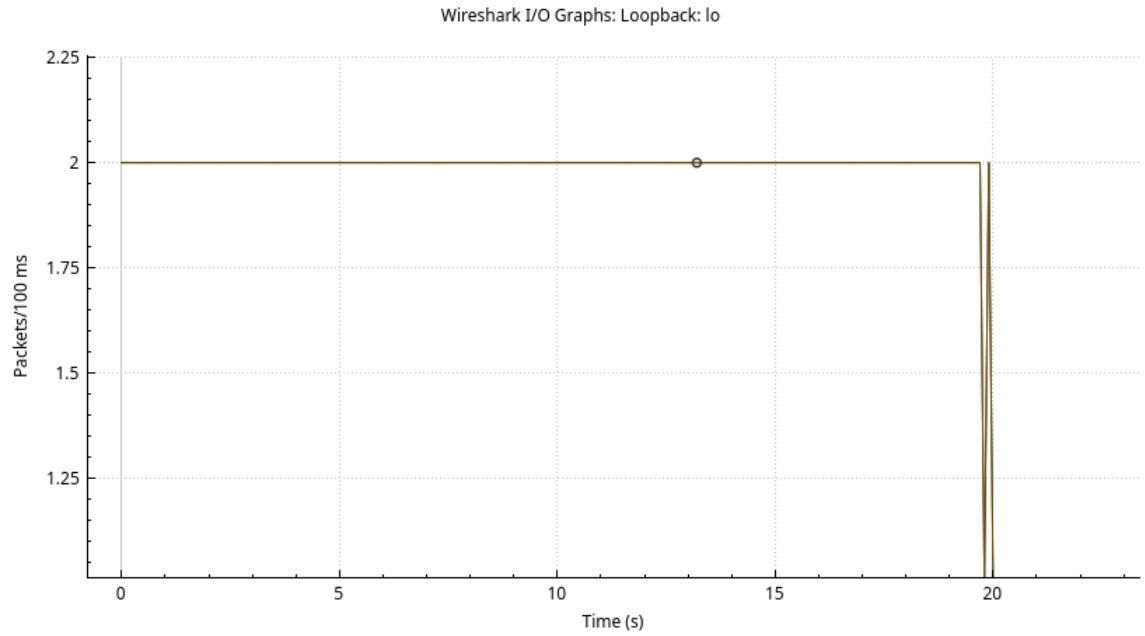
Sändningen är stabil, inga problem med 20 medelanden per sekund.

0.1.3 Task B1c



Skillnaden mellan de 2 graferna blir att på mottagarens graf, så ser vi mer paket per sekund, beroende på alla andra program som körs på datorn, men det finns alltid minst två paket.

0.1.4 Task B1d



0.1.5 Task B1e

Har inte märkt någon skillnad alls. Har stabilt nätverk, så har mycket bra anslutning. Alla resultat är tagna med Discord samtal och Youtube körandes samtidigt.

0.2 Task B2

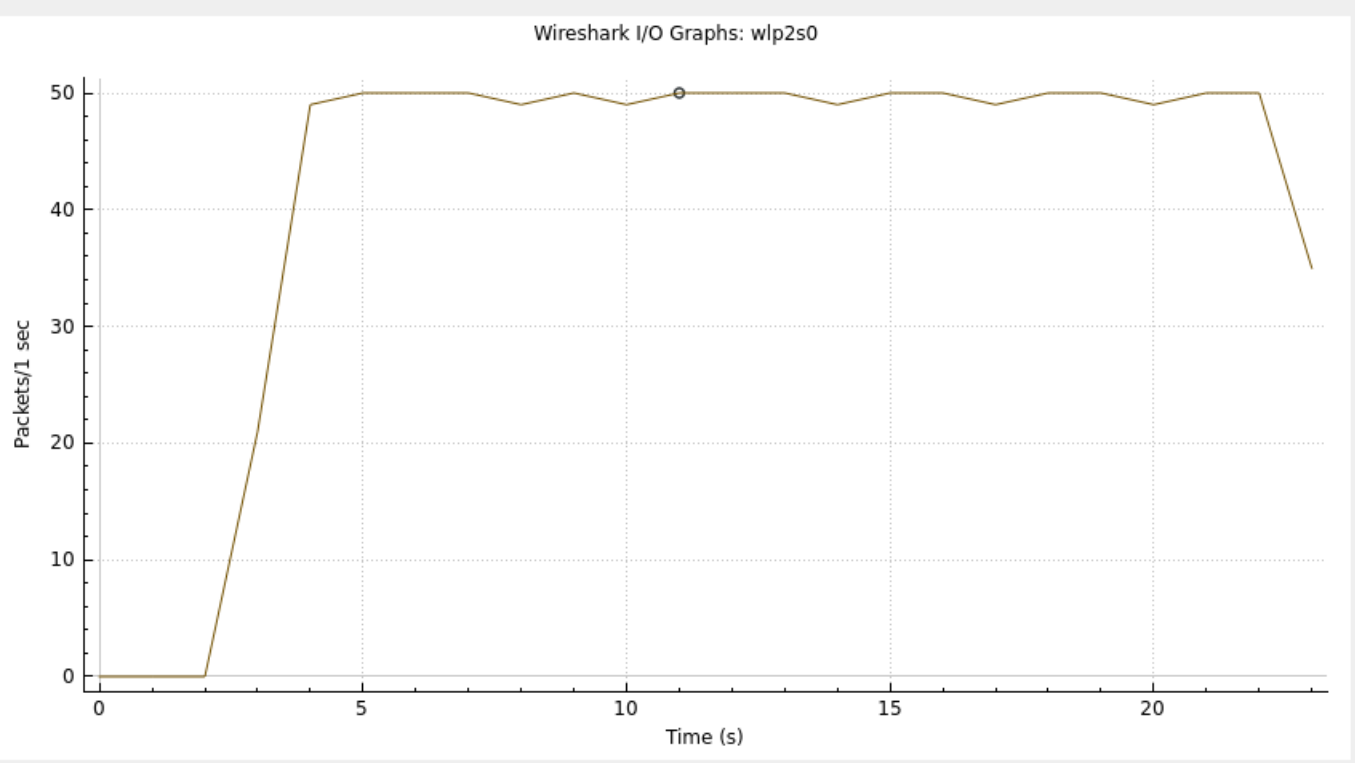
0.2.1 Task B2a

Eftersom den räknas ut som beskrivit innan, så valde jag 20 paket per sekund, och 1000 totala paket, så då får sleep 20 ms.

0.2.2 Task B2b

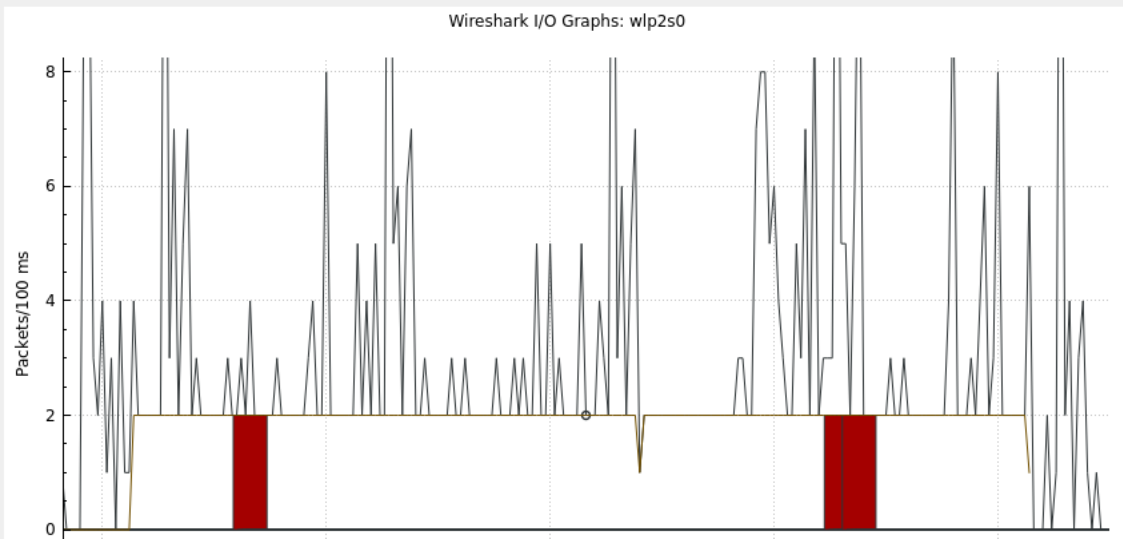
Sändningen är stabil, inga problem med 50 medelanden per sekund.

0.2.3 Task B2c



Märker inte att något inte stämmer. Alla paket verkar komma fram i tid och i ordning. Att ha andra program igång gjorde ingen skillnad.

0.2.4 Task B2d



0.2.5 Task B2e

Har inte märkt någon skillnad alls. Har stabilt nätverk, så har mycket bra anslutning. Ingen paketförlust än.

0.3 Task B3

0.3.1 Task B3a

Med sleep(0) får jag en medelandes frekvens 20000 medelanden per sekund.

0.3.2 Task B3b

Utan sleep så får jag en medelandes frekvens 20000 medelanden per sekund också, samma som med sleep(0)

0.3.3 Task B3c

Antagligen mitt nätverkskort, eftersom det ska hantera 40000 paket per sekund. Efter lite snabb räkning så blir det cirka 750 mbit, så ganska nära kortets maximala 1000 mbit.

Task C

0.4 Task C1

0.4.1 Task C1a

```
from random import randrange
import socket
from time import sleep

serverPort = 12000
receiverIp = "127.0.0.1"
```

```
def getData(length):
    string = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
    out = ""
    for i in range(length):
        out += string[randrange(0, len(string)-1, 1)]
    return out


msgToSend = getData(1431)
nrOfMsg = 1000
msgSpeed = 50

print("UDP target IP: %s" % receiverIp)
print("UDP target Port: %s" % serverPort)
print("UDP target Message: %s" % msgToSend)

#asd
for i in range(nrOfMsg):
    data = None
    msg = f"{10000+i}{msgToSend}####"
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((receiverIp, serverPort))
        s.sendall(msg.encode())



    sleep(1/msgSpeed)
```

0.4.2 Task C1b

```
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
well_known_port = 12000
sock.bind(('127.0.0.1', well_known_port))
sock.listen(100000)
nextId = 10000

numPackets = 0
now = None
try:
    while 1:
        newSocket, address = sock.accept()
        # loop serving the new client
        while 1:
            try:
                data = newSocket.recv(1500)
                if not data:
                    break
                msgId = int(data[0:5])
                msg = data[5:-4]
                if msgId != nextId:
                    print(f"ERROR EXPECTED MSG {nextId} BUT RECVIED {msgId}")
                    break
                nextId += 1
                numPackets += 1
                if numPackets % 100 == 0:
```

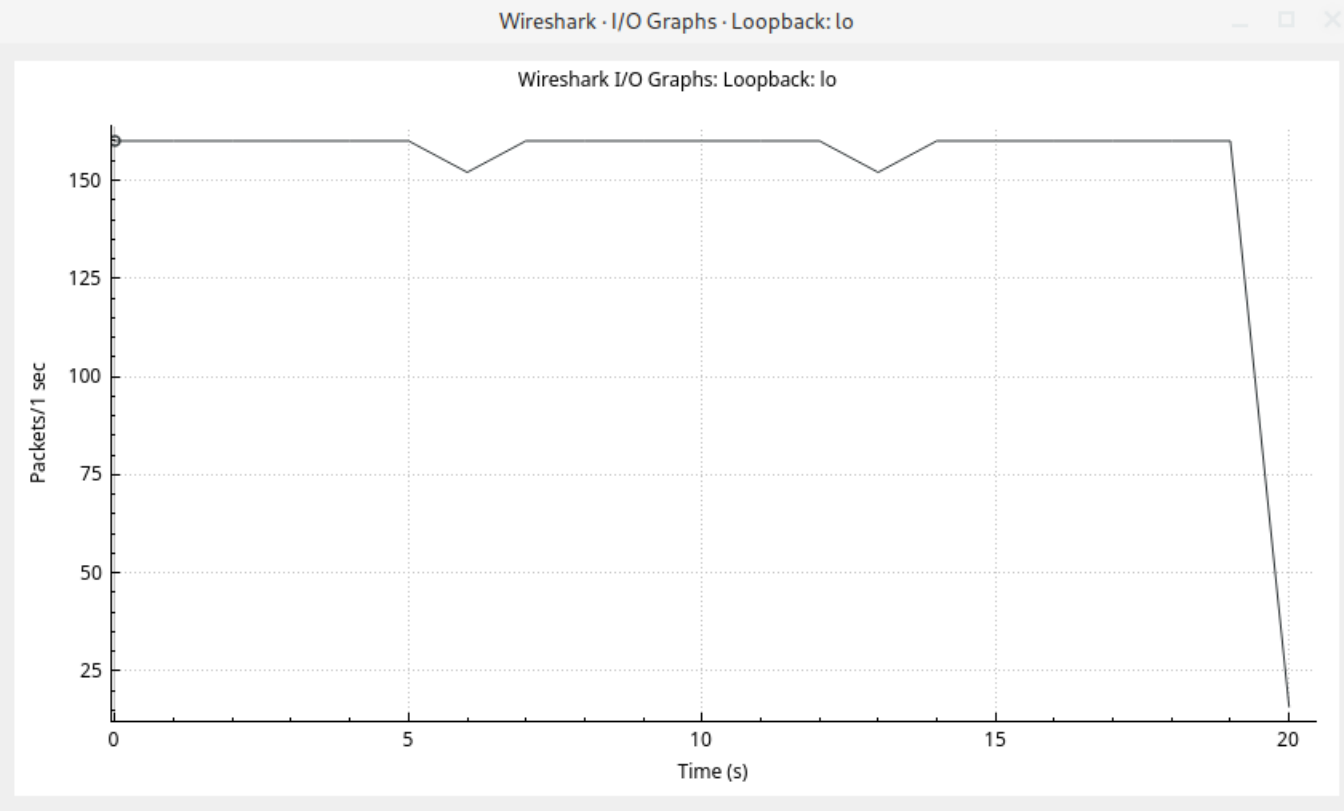
```
                print(f"{numPackets} packets received")
            print(f"received message id: {msgId}: {msg[0:50]}...")
            newSocket.close()
        except OSError:
            break
finally:
    sock.close()
```

0.5 Task C2

0.5.1 Task C2a

Sändningen är stabil, inga problem med 20 medelanden per sekund. Blir mycket fler paket för tcp så blir antagligen långsammare när man skickar fler paket.

0.5.2 Task C2b



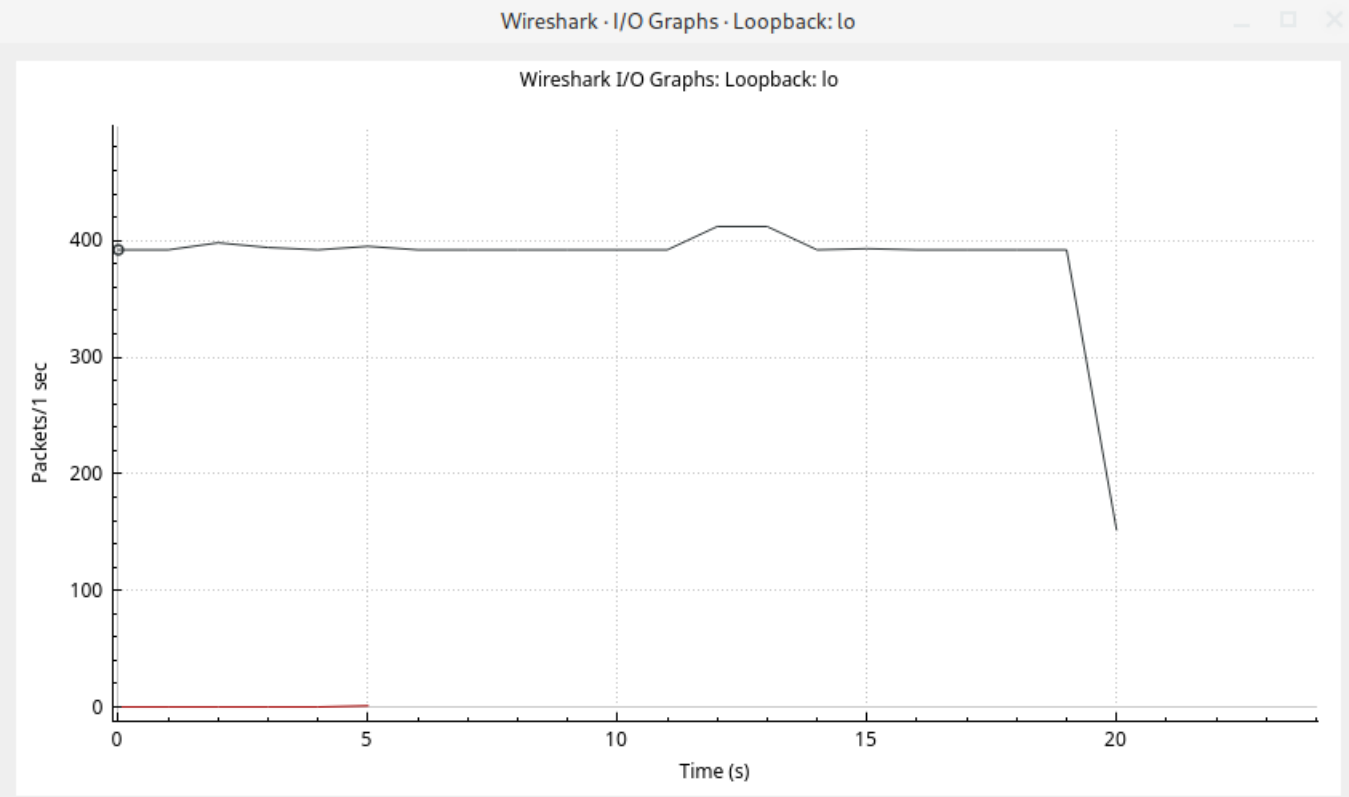
Ser inga paketomsändningar, eller paket fel, men ser att det definitivt blir fler paket för samma överföringar.

0.6 Task C3

0.6.1 Task C3a

Sändningen är stabil, inga problem med 50 medelanden per sekund heller. Trodde att det skulle vara ett problem, men ser bara ett fåtal omsändningar. Även men annat körande i bakgrunden

0.6.2 Task C3b



Ser inga paketomsändningar, eller paket fel, men ser att det definitivt blir fler paket för samma överföringar.

0.7 Task D

0.7.1 Task D1

UDP är oftast bättre när realtid är viktigt. Nackdelen är att vissa paket som skickas via UDP kan saknas eller vara korrupta, vilket kan resultera i att några bilder saknas i en video eller ljudet under en livestream. Den lilla missen brukar inte vara hela värden i de flesta fall. TCP är långsammare, men försäkar sig om att paketet kommer fram. Det visas också med testerna som gjorts.

0.7.2 Task D2

Uppgiften gick bra, men hade lite problem med att skapa en TCP server, eftersom jag försökte skriva om min UDP kod istället för att skriva om den ifrån början, så att hantera olika anslutningar blev inte så enkelt. Filtreringen i wireshark tog lite tid men efter att jag hittat loopback kortet, så blev det enklare.