Netze Projekt 1

Schäfer Pascal
Assel Anas

April 28, 2023

Experimental Setup

We first with fallocate -l 100M example.txt in testfile Folder for sending. Then we send The file with packages of siez 100,1000 and 60000.

Wir haben jeweils ein File sender.py zum senden von Daten und ein File empfaenger.py zum Empfangen von Daten
Aufruf mit: - python sender.py TranmissionID PORT IPAdresse Filename(Path)
- python empfaenger.py

Implementation

Transmitter: Aufruf mit: python sender.py TranmissionID PORT IPAdresse Filename(Path)

Wir lesen Argumente TranmissionID PORT IPAdresse Filename(Path) ein, danach erstellen wir einen Datagram socket(UDP) und erzeugen das erste Paket das gesendet werden soll. Dieses enthält die übergebene TranmissionID die aktuelle Sequence Nummer die maximale Sequenznummer und den Filnamen. Danach werden die Datenpakete gesendet mit TransmissionID und Sequenznummern. Am Ende wird das Kontrollpaket versandt welches den Hash zur kontrolle enthält.

```python
def main():
    if (len(sys.argv)==5):
        execute_send(sys.argv[1],sys.argv[2],sys.argv[3],sys.argv[4])
    else :
        print('\nuse format TransimmionID, Port,IP-Adress,Filename with path ')
        sys.exit()




if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print('\naborting transmission')
```

```python
def execute_send(transmissionid,PORT,ipadress,thisfilnamepath):



    # Definiere Konstanten
    BUFFER_SIZE = 1000
    UDP_IP = ipadress     #'localhost'    127.0.0.1'
    UDP_PORT = int(PORT)   #5005

    # Wähle die zu sendende Datei
    filenamebase = os.path.basename(thisfilnamepath)      #'example.txt'
    #absPath/example100MB.txt
    #or just example100.txt when file in same dir
    filenameabs= thisfilnamepath
    # Erstelle UDP-Socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bestimme die maximale Sequenznummer

    max_seq = os.path.getsize(filenameabs) // BUFFER_SIZE
    if os.path.getsize(filenameabs) % BUFFER_SIZE != 0:
        max_seq += 1


    #Für letztes Paket mit md5
    max_seq +=1

    # Sende das erste Paket mit der Dateiinformation
    trans_id = int(transmissionid) #1234  Wähle eine Transmission ID
    seq_num = 0
    filename_encoded = filenamebase.encode()
    if (len(filename_encoded)<8 | len(filename_encoded)>2048):
        print('\nuse filenameabs longer 8 and smaller 2048 bytes')
        sys.exit()


    header = struct.pack('!HLL', trans_id, seq_num, max_seq) + filename_encoded

    #start time transit
    starttransmit = time.time()
    print("Start sending...")
    sock.sendto(header, (UDP_IP, UDP_PORT))
```

```python
    # Sende die Datenpakete
    with open(filenameabs, 'rb') as f:
        packets_sent=0
        for i in range(max_seq-1):
            data = f.read(BUFFER_SIZE)
            seq_num += 1
            packet = struct.pack('!HL', trans_id, seq_num) + data
            sock.sendto(packet, (UDP_IP, UDP_PORT))
            packets_sent+=1
            if (packets_sent % (max_seq//10) == 0):
                percentage_sent = round(packets_sent/max_seq*100)
                print(f'{percentage_sent}%of packets sent')



    # Sende das letzte Paket mit dem MD5-Hash
    with open(filenameabs, 'rb') as f:
        data = f.read()

    md5 = hashlib.md5(data).digest()
    packet = struct.pack('!HL', trans_id, max_seq) + md5
    sock.sendto(packet, (UDP_IP, UDP_PORT))
    endtransmit = time.time()

    print(f'Transmission time:{endtransmit-starttransmit}')
    # Schließe den Socket

    sock.close()
```

Receiver: Aufruf mit: python empfaenger.py
Wir erzeugen einen Socket und binden diesen an Port und Ip-Adresse. Danach empfangen wir das erste Paket mit Filename und max Sequenznummer. Im anschluss daran werden die empfangenen Sequenznummern abgeglichen und die Daten geschrieben. Zum Schluss empfangen wir den md5 Kontrollhash und vergleichen ihn mit unserem eigenen.

```python
# Definiere Konstanten
BUFFER_SIZE = 1024
UDP_IP = 'localhost' #  127.0.0.1'
UDP_PORT = 5005

# Erstelle UDP-Socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((UDP_IP, UDP_PORT))

# Empfangen des ersten Pakets
header, addr = sock.recvfrom(BUFFER_SIZE*2+10)
trans_id, seq_num, max_seq= struct.unpack('!HLL', header[:10])
filename = header[10:].decode()




with open(filename, 'wb') as f:
    seq_num=1
    # Empfangen der Datenpakete
    while seq_num < max_seq:
        data, addr = sock.recvfrom(BUFFER_SIZE+6)
        trans_id_recv, seq_num_recv  = struct.unpack('!HL', data[:6])

        # Überprüfe Transmission ID und Sequenznummer
        if trans_id != trans_id_recv or seq_num != seq_num_recv:
            print('Falsches Paket empfangen')
            break

        # Schreibe Daten in Datei
        f.write(data[6:])
        if seq_num > max_seq:
            break

        seq_num +=1



    # Empfangen des letzten Pakets mit dem MD5-Hash der Datei
    data, addr = sock.recvfrom(BUFFER_SIZE+6)
    trans_id_recv, seq_num= struct.unpack('!HL', data[:6])
    file_md5_recv = data[6:].hex()



    # Überprüfe Transmission ID, Sequenznummer und MD5-Hash
    if trans_id != trans_id_recv or seq_num != max_seq :
        print('Fehler beim Empfangen der Datei')
```

```
    # Überprüfe Transmission ID, Sequenznummer und MD5-Hash
    if trans_id != trans_id_recv or seq_num != max_seq :
        print('Fehler beim Empfangen der Datei')



f.close()


with open(filename, 'rb') as f:
    data_for_hash_compare = f.read()
md5 = hashlib.md5(data_for_hash_compare).hexdigest()
if md5 == file_md5_recv :
    print('Korrekter Hash')
else:
    print('Falscher Hash')


print(md5)
print(file_md5_recv)

f.close()
sock.close()
```

Messungen

Größe des Files sind 100 MB:

Messreihe mit Packetgröße 100

Messung	Runtime [sec]
1 Messung	83.52
2 Messung	84.31
3 Messung	83.77
4 Messung	82,13(falscher Hash)
5 Messung	83.56
6 Messung	82.95
7 Messung	83.01
8 Messung	80.87 (falscher Hash)
9 Messung	83.46
10 Messung	82.86

Messreihe mit Packetgröße 1000

Messung	Runtime [sec]
1 Messung	8.32(falscher Hash)
2 Messung	9.12
3 Messung	8.92
4 Messung	9.32
5 Messung	8.35 (falscher Hash)
6 Messung	9.31
7 Messung	9.20
8 Messung	9.33
9 Messung	9.37
10 Messung	9.18

Messreihe mit Packetgröße 60000

Messung	Runtime [sec]
1 Messung	0.414
2 Messung	0.403
3 Messung	0.391(falscher HasH)
4 Messung	0.408
5 Messung	0.410
6 Messung	0.425
7 Messung	0.406
8 Messung	0.391(falscher Hash)
9 Messung	0.3912(falscher Hash)
10 Messung	0.413