Universidad de los Andes

Infraestructura de Comunicaciones

2021-20

GRUPO 6

Juan Diego Cardona 201819447

Felipe Guzmán 201813791

Gabriela Caballero 201812917

Nicole Dueñas 201816242

Laboratorio 3

Tabla de Contenidos

Evidencia pertinente de los resultados de las actividades

4.2. Requerimientos del Servidor de Transferencia de Archivos

Se creo una clase en Python, llamada server_logs_hash.py, en donde en la siguiente imagen se puede evidenciar, que este servidor puede soportar 25 conexiones concurrentes, y que tiene 2 archivos de 100MB y 250MB para enviar a los clientes.

```
print("Archivos:")
print(" (1) 100MB.txt")
print(" (2) 250MB.mp4")
print("-----------------")
archivo = int(input("Archivo a enviar (1 o 2): "))
if archivo == 2:
    file_path = './archivos/2.dummy'
num_clientes = int(input("Ingrese numero de clientes: "))
print("Esperando {} clientes".format(num_clientes))

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 8080
buffer_size = 500*1024
s.bind((host, port))
s.listen(25)
print("Escuchando desde: {}:{}".format(host,port))
```

Se implemento una función, en donde se calcula un valor hash para el archivo transmitido.

```
def hash_file(filename):
   h = hashlib.sha1()
   with open(filename,'rb') as file:
       chunk = 0
       while chunk != b'':
           chunk = file.read(1024)
           h.update(chunk)
   return h.hexdigest()
```

Además, se creó otra función en donde, se crea un archivo log.

```
def threaded(c, thread_count):
    client_ready = c.recv(buffer_size).decode()
    if client_ready == "Preparado":
        synchronize(num_clientes)
        with c,open(file_path, 'rb') as f:
            filename = path_leaf(file_path)
            filesize = f'{os.path.getsize(file_path)}'

            c.sendall(str(thread_count).encode()+b'\n')
            c.sendall(filename.encode()+b'\n')
            c.sendall(filesize.encode() + b'\n')
            c.sendall(hash_file(file_path).encode()+b'\n')

            log = "C" + str(thread_count) + " | " + fecha + " | "
            log = log + filename + " | " + str(round((int(filesize)/1024),2)) + "KB"
            inicio = time.time()
            while True:
                data = f.read(buffer_size)
                if not data: break
                c.sendall(data)
            client_recibido = c.recv(buffer_size).decode()
            if client_recibido == "Recibido":
                print("Enviado con exito a cliente {}".format(thread_count))
                c.sendall(str(inicio).encode()+b'\n')
                tiempo = float(c.recv(buffer_size).decode())
                log = log + " | Tiempo: " + str(tiempo) + "s | Exitoso\n"
            else:
                print("Errores en el envio a cliente {}".format(thread_count))
                log = log + " | Error\n"
    c.close()
    log_file = open("./logs/"+fecha+".txt",'a')
    log_file.write(log)
    log_file.close()
```

4.3. Requerimientos del cliente TCP

Se creo una clase en Python, llamada **client_logs_hash.py**, en donde en la siguiente imagen se puede evidenciar, que este servidor puede soportar 25 conexiones concurrentes.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
port = 8080
buffer_size = 500 * 1024

s.connect((host, port))
s.listen(25)
print("Conexion establecida con {}:{}".format(host, port))
s.sendall("Preparado".encode())
```

Se implemento una función, en donde se calcula un valor hash para el archivo recibido.

```
def hash_file(filename):
    h = hashlib.sha1()
    with open(filename, 'rb') as file:
        chunk = 0
        while chunk != b'':
            chunk = file.read(1024)
            h.update(chunk)
    return h.hexdigest()
```

Además, se creó otra función en donde, se crea un archivo log.

```
log = datetime.now().strftime("%d-%m-%Y | %H:%M:%S") + " | "

with s, s.makefile('rb') as serverfile:
    thread_num = serverfile.readline().strip().decode()
    filename = serverfile.readline().strip().decode()
    length = int(serverfile.readline())
    log = log + filename + " | " + str(round((length / 1024), 2)) + "KB"
    hash_recibido = serverfile.readline().strip().decode()
    print("Hash recibido: {}".format(hash_recibido))
    initial_length = length
    print(f'Descargando {filename}')
    with open('C' + thread_num + "-" + filename, 'wb') as f:
        while length:
            chunk = min(length, buffer_size)
            data = serverfile.read(chunk)
            if not data: break
            f.write(data)
            length -= len(data)
            print("\r{}% completado".format(round((1 - (length / initial_length)) * 100, 2)), end='', flush=True)
        print("")
        fin = time.time()
        if length != 0:
            print('Descarga invalida')
            log = log + " | Error en descarga"
        else:
            print('Descarga terminada')
            hash_local = hash_file('C' + thread_num + "-" + filename)
            if hash_local == hash_recibido:
                print("Hash Correcto - Enviando Recibido")
                s.sendall("Recibido".encode())
                inicio = float(serverfile.readline().strip().decode())
                tiempo = round(fin - inicio, 2)
                print("Tiempo total: {} segundos".format(tiempo))
                s.sendall(str(tiempo).encode() + b'\n')
                log = log + " | Tiempo: " + str(tiempo) + "s | Exitoso"
            else:
                print("Error en el Hash")
                log = log + " | Error en Hash"

log_file = open('C' + thread_num + "-" + datetime.now().strftime("%d-%m-%Y_%H_%M_%S.txt"), 'w')
log_file.write(log)
log_file.close()
```

4.4. Análisis de desempeño del servicio

1. Capturas disponibles en: https://uniandes-my.sharepoint.com/:f:/g/personal/fg_guzman_uniandes_edu_co/EolFxCxUPeNMlNvjE3ga7OcBJx8nc9Wxta_sIlonyOztIw?e=tm8K9o

2.

Prueba 1

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	102.4MB	0.69 s	148MB/s	65528	102.4MB	0.64 s	8080

Prueba 2

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	256MB	1.48 s	173MB/s	49161	256MB	1.48 s	8080

Prueba 3

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	102.4MB	1.23 s	83.25MB/s	49169	102.4MB	1.23 s	8080
2	Si	102.4MB	1.25 s	81.92MB/s	49170	102.4MB	1.25 s	8080
3	Si	102.4MB	0.81 s	126MB/s	49171	102.4MB	0.81 s	8080
4	Si	102.4MB	1.05 s	97.5MB/s	49172	102.4MB	1.05 s	8080
5	Si	102.4MB	1.25 s	81.92MB/s	49173	102.4MB	1.25 s	8080

Prueba 4

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	256MB	2.42 s	105.8MB/s	49183	256MB	2.42 s	8080
2	Si	256MB	1.97 s	192.9MB/s	49184	256MB	1.97 s	8080
3	Si	256MB	3.23 s	79.26MB/s	49185	256MB	3.23 s	8080
4	Si	256MB	1.95 s	131.3MB/s	49186	256MB	1.95 s	8080
5	Si	256MB	2.84 s	90.14MB/s	49187	256MB	2.84 s	8080

Prueba 5

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	102.4MB	1.94 s	52.78MB/s	49194	102.4MB	1.94 s	8080
2	Si	102.4MB	1.84 s	55.65MB/s	49195	102.4MB	1.84 s	8080
3	Si	102.4MB	2.06 s	49.71MB/s	49196	102.4MB	2.06 s	8080
4	Si	102.4MB	1.73 s	59.19MB/s	49197	102.4MB	1.73 s	8080
5	Si	102.4MB	2.61 s	39.23MB/s	49198	102.4MB	2.61 s	8080
6	Si	102.4MB	1.45 s	70.62MB/s	49199	102.4MB	1.45 s	8080
7	Si	102.4MB	1.12 s	91.43MB/s	49200	102.4MB	1.12 s	8080
8	Si	102.4MB	1.87 s	54.76MB/s	49201	102.4MB	1.87 s	8080
9	Si	102.4MB	2.09 s	49MB/s	49202	102.4MB	2.09 s	8080
10	Si	102.4MB	0.73 s	140.27MB/s	49203	102.4MB	0.73 s	8080

Prueba 6

Transferencia	Exitosa	Bytes Cliente	Tiempo de transferencia cliente	Tasa de trasnferencia	Puerto cliente	Bytes servidor	Tiempo de transferencia servidor	Puerto servidor
1	Si	256MB	5.08s	50.39MB/s	49211	256MB	5.08s	8080
2	Si	256MB	5.52s	46.38MB/s	49212	256MB	5.52s	8080
3	Si	256MB	4.52s	56.64MB/s	49213	256MB	4.52s	8080
4	Si	256MB	5.95s	43.03MB/s	49214	256MB	5.95s	8080
5	Si	256MB	4.37s	58.58MB/s	49215	256MB	4.37s	8080
6	Si	256MB	3.73s	68.63MB/s	49217	256MB	3.73s	8080
7	Si	256MB	5.5s	46.55MB/s	49218	256MB	5.5s	8080
8	Si	256MB	4.39s	58.31MB/s	49219	256MB	4.39s	8080
9	Si	256MB	4.92s	52.03MB/s	49220	256MB	4.92s	8080
10	Si	256MB	6.56s	39.02MB/s	49221	256MB	6.56s	8080

3. Analice cada prueba realizada. Considerando como cambian las métricas de acuerdo con el número de clientes conectados, el tamaño del archivo transmitido y la captura de tráfico realizada durante la prueba.

Prueba 1

La prueba fue exitosa, el tiempo de transferencia fue levemente menor del servidor hacia el cliente por 0,04s.

Prueba 2

Fue una prueba exitosa, el tamaño de Bytes transmitidos fue mayor que el de la prueba 1 y la Tasa de transferencia aumento con respecto a la prueba 1, el tiempo de transferencia entre servidor y cliente son iguales

Prueba 3

La prueba global fue exitosa, aumento el número de clientes con respecto a las pruebas 1 y 2, la tasa de transferencia por cliente disminuyo, ya que se transfirieron menos Bytes por segundo, sin embargo, se observa que es el mayor tiempo de transferencia es de 1,25s lo que indica que se la tasa de trasferencia global es de 409,6MB/s

Prueba 4

La prueba fue exitosa, con respecto a la prueba 3 aumento la cantidad de Bytes trasmitidos la tasa de trasferencia individual aumento, y se observa una disminución en la tasa de trasferencia global 396,28MB/s

Prueba 5

La prueba fue exitosa, con respecto a las pruebas 4 y 5 se duplico la cantidad de clientes y la tasa de trasferencia bajo sin embargo se observa una trasferencia global de 400 MB/s la cual es mayor a la de la prueba 4 pero inferior a la prueba 3

Prueba 6

Fue exitosa, se duplico la cantidad de información transmitida con respecto a la prueba 5, se observa una degradación de transferencia a nivel de cliente, pero a nivel global se observa que es igual a la a mejor tasa de trasferencia de la prueba 3

Nota:

$$Transferencia\ global = \frac{Bytes\ Cliente\ \times\ \#Clientes}{Mayor\ Tiempo\ de\ Transferencia\ Cliente}$$

Se puede ver que, en las pruebas con más de un cliente, los puertos de cliente son diferente para cada uno.

4. Realice un análisis global tomando el resultado del conjunto de pruebas realizado. Apóyese en tablas y graficas que le permitan dar una mejor explicación a su análisis.

# Prueba	Tasa de transferencia global
1	148 MB/s
2	173 MB/s
3	409,6 MB/s
4	396,28 MB/s
5	400 MB/s
6	409,6 MB/s

Primero que todo se observa que todas las pruebas fueron exitosas, la mejor tasa de transferencia se obtiene con la prueba 3 y 6, la 3 es la mejor para clientes de 102,4 MB y para clientes de 256 MB es la 6.

Además, se observa que la duplicar el número de clientes y la cantidad de Bytes aumenta la tasa de transferencia a excepción de las pruebas 3 y 4.

Video

IMPLEMENTACIÓN DE SERVIDOR Y CLIENTE TCP GRUPO 6

Repositorio GitHub

https://github.com/juandiegocardona/LabsInfracomu/tree/main/Lab3

Referencias

- Rockikz, A. (2019, 19 agosto). *How to Transfer Files in the Network using Sockets in Python - Python Code*. PythonCode. https://www.thepythoncode.com/article/send-receive-files-using-sockets-python

Universidad de los Andes

Infraestructura de Comunicaciones

2021-20