Iran University Of science And Technology

(IUST)

Summer Internship Project:

Create a direct communication channel between clients by a central control server

Company:
Atlas Compressor Kara

Programmer:
Pegah Torkamandi

2013

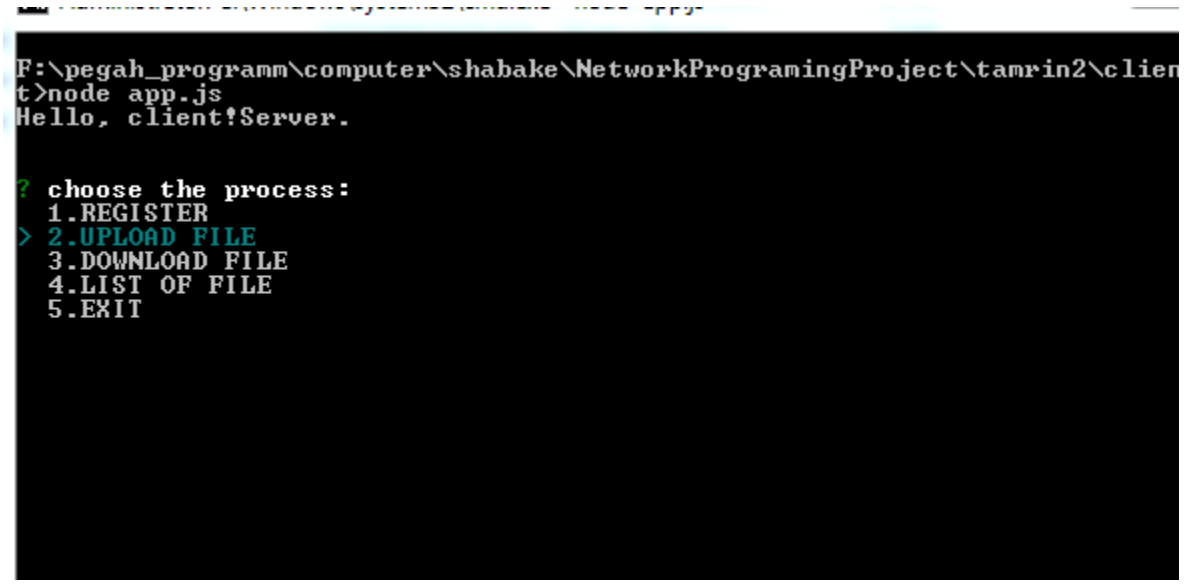
```
            tmp = tmp + line[:end]
            b_start = 0
            b_end =1

    if(b_end == 1):
        tmp = re.findall(r"[\w']+",tmp)
        nd=0
        for i in range(0,len(tmp)):
            tmp[i]=tmp[i].lower()
            tmp[i] = porter.stem(tmp[i])
            if(tmp[i] in list_stop):
                continue
            m.append((tmp[i],[doc_id,i]))
            nd+=1
        num_doc.append(pp,nd)
        b_end=0
        tmp=str()
        doc_id = doc_id + 1
    if sys.getsizeof(m)>4*1024*1024:
        total_temp_files=total_temp_files+1
        fname='partial-temp-'+str(total_temp_files)+'.dat'
        print 'saving temporary file ',fname,'...'
        m=sort_data(m)
        write_file(fname,m,int(0),len(m))
        total_records=total_records+len(m)
        m=[]
    position=fr.tell()
    line=fr.readline()
    _num+=1
total_temp_files=total_temp_files+1
fname='partial-temp-'+str(total_temp_files)+'.dat'
print 'saving temporary file ',fname,'...'
m=sort_data(m)
write_file(fname,m,int(0),len(m))
```

TCP CONNECTION:

In this project, there is a central server that controls the information and communication of each client. Each new client initially registers a TCP connection to the central server, and also opens its UDP port and provides the UDP port number with the requested information to the server.

The server stores the client information in the data base and registers the client and allows the client to receive or share new data on the network, and the client can request the list of all the shared information between other clients by sending a command to server.



TCP PORTOCOL:

Request from client:

Command\nlabel:statement\n\n

Command = register

reg\nclientname:name of client\nudpPort:port\n\n\n

Command = upload

Put\nfilename:name of file\nsha: $SHASUM^2$\n\n

Command = download

get\nfilename:name of file\nclientname:name of client\n\n

command = Shared Listings

lst\n\n

reply from server:

Register or upload done!

OK\nDESCRIPTION: description

Lst done!

Shared Listings

Download done!

OK\nip:port:sha

In the event of an error:

ERR\nDESCRIPTION: description

In this project, if the client has already registered and disconnected and wants to register again, the server will update the IP and UDP port of the client, as well as erase the list of files uploaded by the client in its data base.

If the client sends the download command to the server, the server sends the information of the client, which has the data, such as the UDP port, IP and SHA file to the requesting client, and finally the client sends a download request message to the client, which contains the data, via the UDP port.

UDP CONNECTION:

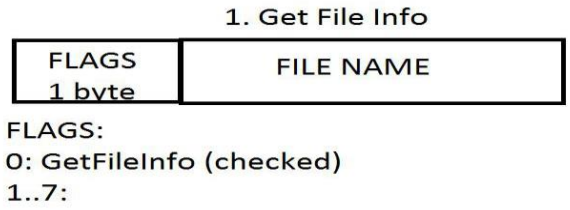
The client, which contains the data, dividing the data into parts and each part contains 5 packets with 256 bytes length.

The messages that the clients transfer by UDP port contain a flag, which is one bit and specifies the type of message.

A -> the client requests the file

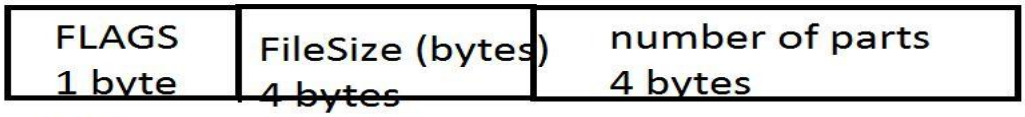
B -> the client contains the file

1) A requests the file from B with the below format message.



2) B sends the below message to A

2. Get FIle Info Response



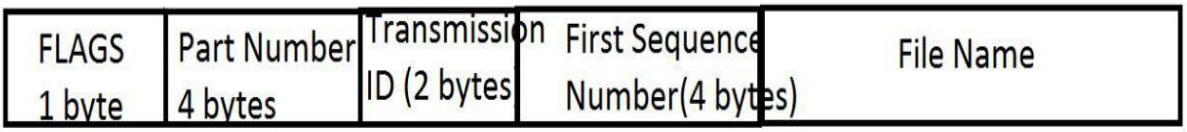
FLAGS:
1: GetFileInfo Repsonse(checked)
0,2..7:

NOTE: Size of each part can be determined by:
fileSize / numberOfParts

3) A sends the number of part and sequence number, which is the start number of the packet in the part, to B

The client assigns the unique transmission ID to the each download request.

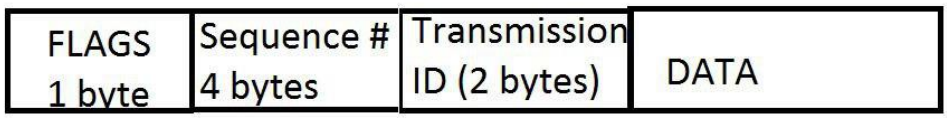
3. Request Part



FLAGS:
2: RequestPart (checked)
0..1,3..7:

4) B sends the data with the transmission ID and the number of the packet.

5. Part



FLAGS:

4: Part (checked)

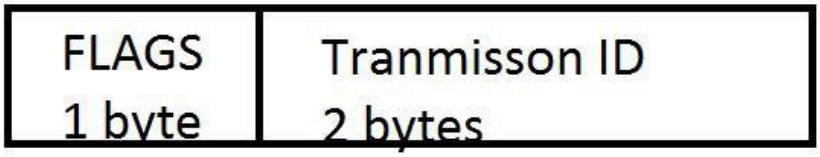
5: Last packet: (may be cheched alongside 4)

1..3,6..7:

5) A set the time for each request, so if B does not receive any reply in the specific time will send the request again.

If there is a problem, so the clients send the error message.

4. Request Part Error (new)



FLAGS:

3: Request Part Error

0..2,4..7:

Finally A compares the SHA of the received file with SHA, which receives from the server, if they are not same then A sends the error message and requests the file again.

In this project A and B show the time to upload and download file.

I order to run parallel client requests, we use the fork, which one of them is master and others are the child, in the server, also in the node js language we share the port, so they distribute the requests between themselves (load balancing).