**CN PROJECT REPORT**

**NAME:** NITTALA SATYA SAI KIREETI

**REGNO:** RA1911027010096

**TEAM MEMBERS:**

1.HASWANTH (RA1911027010093)

2.KIRAN (RA1911027010074)

3.PAVAN (RA1911027010084)

4.HEMANTH (RA1911027010095)

5.KIREETI (RA1911027010096)

**1.ABSTRACT:**

Multi-Threading refers to concurrently executing multiple threads by rapidly switching the control of the CPU between threads (called context switching).

A thread is a unit of execution within a process.

**2.INTRODUCTION:**

A multithreaded file transfer client-server program build using a python programming language. The server has the capability to handle multiple clients concurrently at the same by using threading. The server assigns each client a thread to handle working for that client.

The server program can handle an arbitrary number of concurrent connections and file exchanges, only limited by system configuration or memory. The server is started without any parameters and creates a TCP socket at an OS-assigned port. It prints out the assigned port number and stores it in a local file port, which is used when starting clients. The server listens on its main socket and accepts client connections as they arrive. Clients perform an upload or download operation, or instruct the server to terminate.

AN ALTERNATIVE SOLUTION:

DRAWBACKS IN MULTITHREADING :

1 .Threads are difficult to code,debugs and sometimes they have unpredictable results.

2.Overhead switching of context.

3.Not scalable for large number of clients.

4.Deadlocks can occur.

SELECT() : A better way to handle multiple clients using select() linux command.

HOW THIS IS BETTER ?

Select command allows to monitor multiple file description,waiting until one of the file descriptors become active.

For example, if there is some data to be read on one of the sockets select will provide that information.Select works like an interrupt handler, which gets activated as soon as any file descriptor sends any data.

**3. REQUIREMENT ANALYSIS:**

Hardware Requirements

Processor : 2.4 GHz Clock Speed

RAM : 1 GB

Hard Disk : 500 MB (Minimum free space)

Software Requirements

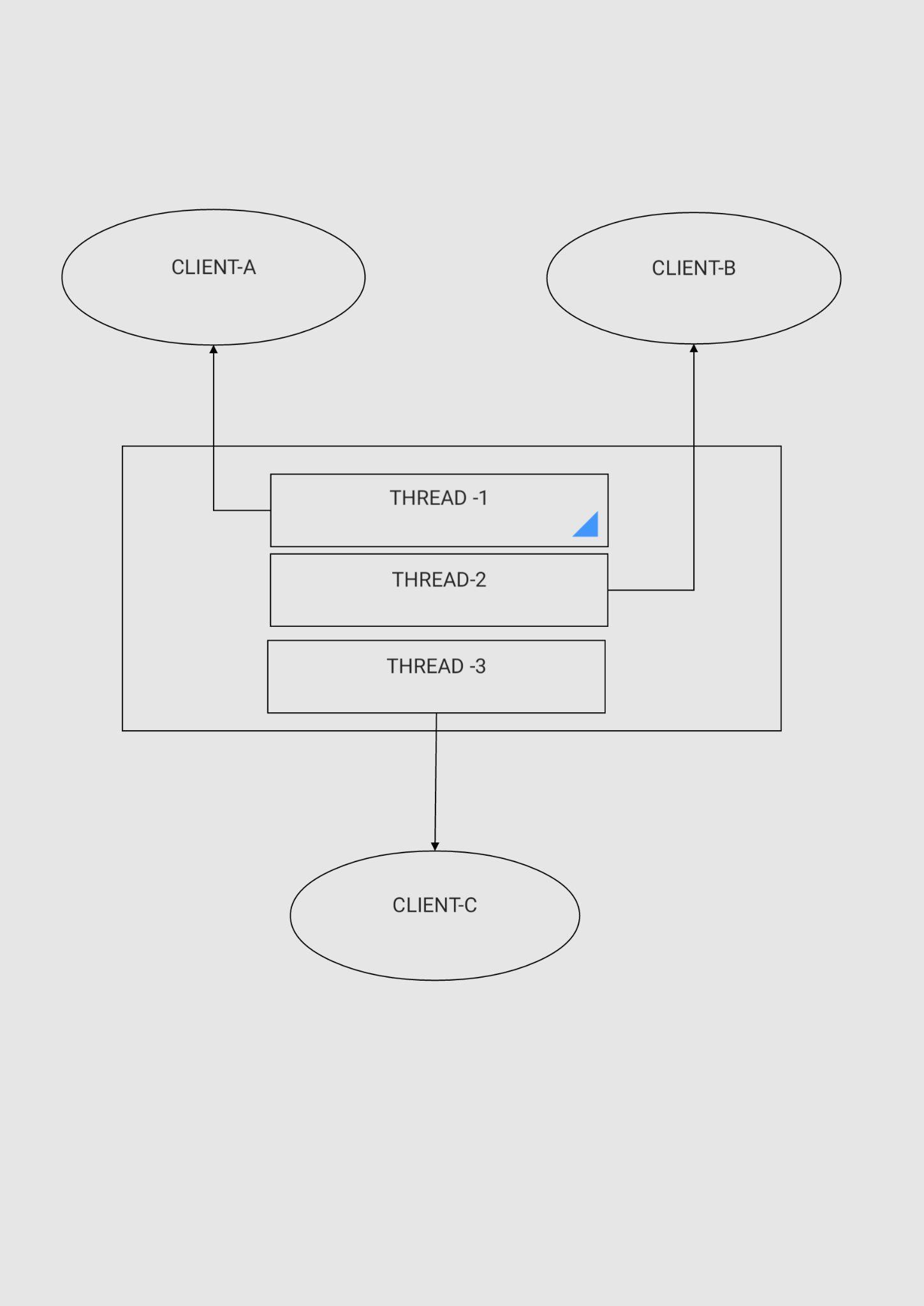
Operating System : Windows 7

Platform : Python

Special Tools : Opencv

Xuggle Server : Apache Tomcat

**4. ARCHITECTURE & DESIGN:**

****

**5. IMPLEMENTATION:**

SERVER CODE:

import os

import socket

import threading

IP = socket.gethostbyname(socket.gethostname())

PORT = 4456

ADDR = (IP, PORT)

SIZE = 1024

FORMAT = "utf-8"

SERVER\_DATA\_PATH = "server\_data"

def handle\_client(conn, addr):

print(f"[NEW CONNECTION] {addr} connected.")

conn.send("OK@Welcome to the File Server.\nEnter 'HELP' to get list of commands.".encode(FORMAT))

while True:

data = conn.recv(SIZE).decode(FORMAT)

data = data.split("@")

cmd = data[0]

if cmd == "LIST":

files = os.listdir(SERVER\_DATA\_PATH)

send\_data = "OK@"

if len(files) == 0:

send\_data += "The server directory is empty"

else:

send\_data += "\n".join(f for f in files)

conn.send(send\_data.encode(FORMAT))

elif cmd == "UPLOAD":

name, text = data[1], data[2]

filepath = os.path.join(SERVER\_DATA\_PATH, name)

with open(filepath, "w") as f:

f.write(text)

send\_data = "OK@File uploaded successfully."

conn.send(send\_data.encode(FORMAT))

elif cmd == "DOWNLOAD":

name = data[1]

filepath = os.path.join(SERVER\_DATA\_PATH, name)

with open(filepath, "r") as f:

text = f.read()

filepath = data[2]

with open(filepath, "w") as f:

f.write(text)

send\_data = "OK@File downloaded successfully."

conn.send(send\_data.encode(FORMAT))

elif cmd == "DELETE":

files = os.listdir(SERVER\_DATA\_PATH)

send\_data = "OK@"

filename = data[1]

print(f"[DISCONNECTED] {addr} disconnected")

conn.close()

def main():

print("[STARTING] Server is starting")

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(ADDR)

server.listen()

print(f"[LISTENING] Server is listening on {IP}:{PORT}.")

while True:

conn, addr = server.accept()

thread = threading.Thread(target=handle\_client, args=(conn, addr))

thread.start()

print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

CLIENT CODE:

import socket

import os

IP = socket.gethostbyname(socket.gethostname())

PORT = 4456

ADDR = (IP, PORT)

FORMAT = "utf-8"

SIZE = 1024

CLIENT\_PATH = "client\_data"

def main():

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(ADDR)

while True:

data = client.recv(SIZE).decode(FORMAT)

cmd, msg = data.split("@")

if cmd == "DISCONNECTED":

print(f"[SERVER]: {msg}")

break

elif cmd == "OK":

print(f"{msg}")

data = input("> ")

data = data.split(" ")

cmd = data[0]

if cmd == "HELP":

client.send(cmd.encode(FORMAT))

elif cmd == "LOGOUT":

client.send(cmd.encode(FORMAT))

break

elif cmd == "LIST":

client.send(cmd.encode(FORMAT))

elif cmd == "DELETE":

client.send(f"{cmd}@{data[1]}".encode(FORMAT))

elif cmd == "UPLOAD":

path = data[1]

with open(f"{path}", "r") as f:

text = f.read()

filename = path.split("/")[-1]

send\_data = f"{cmd}@{filename}@{text}"

client.send(send\_data.encode(FORMAT))

elif cmd == "DOWNLOAD":

filepath = os.path.join(CLIENT\_PATH, data[1])

client.send(f"{cmd}@{data[1]}@{filepath}".encode(FORMAT))

print("Disconnected from the server.")

client.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**6. EXPERIMENT RESULTS & ANALYSIS:**

1. LIST: List all the files from the server.

2. UPLOAD: Upload a file to the server.

3. DELETE : Delete a file from the server.

4. LOGOUT: Disconnect from the server

5. HELP: List all the commands

**7.REFERENCES:**Internet.