Assignment:4 (DS)

Server.py

```
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time

# Data structure to store client address and clock data
client_data = {}

# Nested thread function to receive clock time from a connected client
def startReceivingClockTime(connector, address):
	while True:
	try:
	# Receive clock time
	clock_time_string = connector.recv(1024).decode()
	clock_time = parser.parse(clock_time_string)
	clock_time_diff = datetime.datetime.now() - clock_time

	client_data[address] = {
		"clock_time": clock_time,
		"time_difference": clock_time_diff,
		"connector": connector
	}

	print("Client Data updated with:", str(address))
	time.sleep(5)
	except Exception as e:
	print(f"Error receiving clock time from {address}: {e}")
	break

# Master thread function to accept clients over a given port
def startConnecting(master_server):
	while True:
	try:
	# Accepting a client
	master_slave_connector, addr = master_server.accept()
	slave_address = f"{addr[0]}:{addr[1]}"
	print(f"{slave_address} connected successfully")

	current_thread = threading.Thread(
```

```
                target=startReceivingClockTime,
                args=(master_slave_connector, slave_address)
            )
            current_thread.start()
            except Exception as e:
            print(f"Error accepting client connection: {e}")
            break

# Function to fetch average clock difference
def getAverageClockDiff():
            if not client_data:
            return datetime.timedelta(0, 0)

            time_difference_list = [client['time_difference'] for client in client_data.values()]
            sum_of_clock_difference = sum(time_difference_list, datetime.timedelta(0, 0))
            average_clock_difference = sum_of_clock_difference / len(client_data)
            return average_clock_difference

# Function to generate cycles of clock synchronization
def synchronizeAllClocks():
            while True:
            print("New synchronization cycle started.")
            print("Number of clients to be synchronized:", len(client_data))

            if client_data:
            average_clock_difference = getAverageClockDiff()
            for client_addr, client in client_data.items():
                try:
                synchronized_time = datetime.datetime.now() + average_clock_difference
                client['connector'].send(str(synchronized_time).encode())
                except Exception as e:
                print(f"Error sending synchronized time to {client_addr}: {e}")
            else:
            print("No client data. Synchronization not applicable.")

            print("\n")
            time.sleep(5)

# Function to initiate the Clock Server
def initiateClockServer(port=8080):
            try:
            master_server = socket.socket()
            master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
            print("Socket at master node created successfully")
```

```
        master_server.bind(('', port))
        master_server.listen(10)
        print("Clock server started...")

        # Start making connections
        print("Starting to make connections...")
        master_thread = threading.Thread(target=startConnecting, args=(master_server,))
        master_thread.start()

        # Start synchronization
        print("Starting synchronization parallelly...")
        sync_thread = threading.Thread(target=synchronizeAllClocks)
        sync_thread.start()
        except Exception as e:
        print(f"Error initializing Clock Server: {e}")

# Driver function
if __name__ == '__main__':
        initiateClockServer(port=8080)
```

Client.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

# Client thread function used to send time at the client side
def startSendingTime(slave_client):
        while True:
        try:
        # Provide server with clock time at the client
        slave_client.send(str(datetime.datetime.now()).encode())
        print("Recent time sent successfully\n")
        time.sleep(5)
        except Exception as e:
        print(f"Error sending time: {e}")
        break

# Client thread function used to receive synchronized time
def startReceivingTime(slave_client):
        while True:
```

```
        try:
        # Receive data from the server
        synchronized_time = parser.parse(slave_client.recv(1024).decode())
        print(f"Synchronized time at the client is: {synchronized_time}\n")
        except Exception as e:
        print(f"Error receiving synchronized time: {e}")
        break

# Function used to synchronize client process time
def initiateSlaveClient(port=8080):
        try:
        slave_client = socket.socket()
        # Connect to the clock server on local computer
        slave_client.connect(('127.0.0.1', port))

        # Start sending time to server
        print("Starting to send time to server\n")
        send_time_thread = threading.Thread(target=startSendingTime, args=(slave_client,))
        send_time_thread.start()

        # Start receiving synchronized time from server
        print("Starting to receive synchronized time from server\n")
        receive_time_thread = threading.Thread(target=startReceivingTime,
args=(slave_client,))
        receive_time_thread.start()
        except Exception as e:
        print(f"Error initializing Slave Client: {e}")

# Driver function
if __name__ == '__main__':
        # Initialize the Slave / Client
        initiateSlaveClient(port=8080)
```

Output:

```
Client Data updated with: 127.0.0.1:56488
New synchronization cycle started.
Number of clients to be synchronized: 1


Client Data updated with: 127.0.0.1:56488
New synchronization cycle started.
Number of clients to be synchronized: 1


Client Data updated with: 127.0.0.1:56488
New synchronization cycle started.
Number of clients to be synchronized: 1


Client Data updated with: 127.0.0.1:56488
New synchronization cycle started.
Number of clients to be synchronized: 1


Client Data updated with: 127.0.0.1:56488
New synchronization cycle started.
Number of clients to be synchronized: 1
```

```
Synchronized time at the client is: 2025-03-31 21:20:57.029699

Recent time sent successfully

Synchronized time at the client is: 2025-03-31 21:21:02.033952

Recent time sent successfully

Synchronized time at the client is: 2025-03-31 21:21:07.038677

Recent time sent successfully

Synchronized time at the client is: 2025-03-31 21:21:12.042658

Recent time sent successfully

Synchronized time at the client is: 2025-03-31 21:21:17.045455

Recent time sent successfully

Synchronized time at the client is: 2025-03-31 21:21:22.050307

Recent time sent successfully
```