הפקולטה להנדסה
ע"ש אלכסנדר קופקין
אוניברסיטת בר־אילן

Computer Networks and Internet 1

Programming Assignment

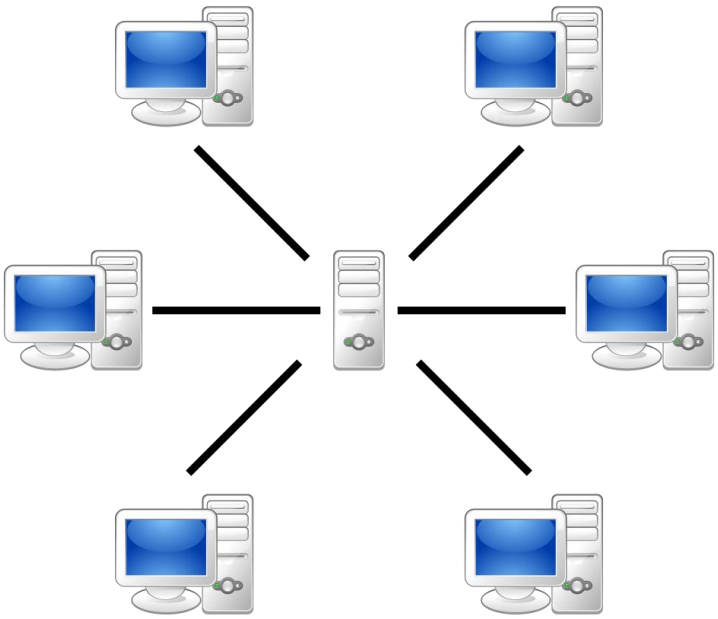
Full names and ID:

Hadas Yossef-Zada 213486764

Shoham Galili 208010785

Operating System: Windows 10

Language: Python3



חלק 1- רקע תיאורטי:

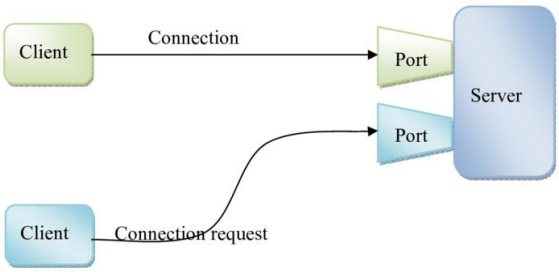
במטלה זו נדרשנו לכתוב קוד וליצור בעצם צ'אטים קבוצתיים וירטואליים. במהלך כתיבת הקוד נחשפנו לעומק שכבות התעבורה ושכבת האפליקציה (שכבות 4-5) תוך יישום ארכיטקטורת שרת-לקוח כפי שלמדנו בהרצאות.

ארכיטקטורת שרת-לקוח:

ארכיטקטורה מעין זו מורכבת משתי ישויות מרכזיות: הלקוח- אשר צורך שירותים ושרת- אשר מספק שירותים. המודל מחלק את המשימות או את עומס העבודה סין ספק השירות או המשאבים- השרת לבין מבקש השירות. עבור המקרה המתואר במטלה:

הלקוח ← הוא משתמש המתעניין בצ'אט קבוצתי. פתיחת צ'אט או הצטרפות לצ'אט קבוצתי קיים.

השרת ← הוא המחשב האחראי על ניהול הצ'אטים הקבוצתיים הללו ע"י מתן שירותים כגון: גישה לצ'אטים קבוצתיים, הדפסת הודעות חדשות וכו.



השרת הוא תוכנה פסיבית, המאזינה לרשת ומחכה לקבל בקשות. הלקוח לעומתו בדרך כלל מהווה את ממשק המשתמש, כלומר הוא מופעל על ידי המשתמש ופונה לשרת כאשר הוא זקוק למידע או שירותים ממנו. בדרך כלל, תוכנות השרת והלקוח רצות על גבי מחשבים שונים והתקשורת ביניהן מתבצעת על גבי רשת מחשבים. עם זאת, תוכנות השרת והלקוח יכולות לפעול גם על גבי אותו מחשב .

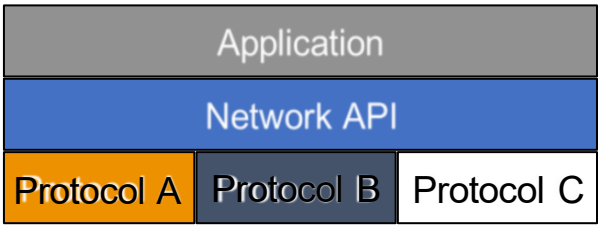
דוגמאות מרכזיות ליישומים המשתמשים במודל שרת-לקוח הן:

- שרת הדפסה.
- דואר אלקטרוני.
- הרשת העולמית.

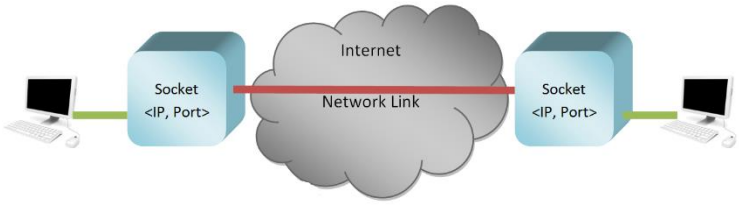
מהו SOCKET?

הSOCKET הוא ממשק בין האפליקציה והרשת, באמצעותו האפליקציה יכולה לשלוח/לקבל נתונים ומידע אל/מהרשת וכך לייצר תקשורת.

Network socket הוא מעין מבנה וירטואלי- תוכנתי בתקשורת המחשב המשמש כנקודת קצה לשליחת וקבלת נתונים ברחבי הרשת.



המידע ברשת מועבר ע"י כתובות. כתובת מורכבת מזוג: (IP,Port). הSocket מספק ממשק לזוג המייצג כתובת, ומוגדר על ידו.



מהו PORT?

Port הינו מבנה לוגי אשר מזהה תהליך ספציפי או סוג של שירות ברשת. קיימים מספרי יציאות ספציפיים השמורים לזיהוי שירותים ספציפיים, באמצעותם ניתן להעביר בקלות חבילה שמגיעה לאפליקציה הנמצאת בהרצה.

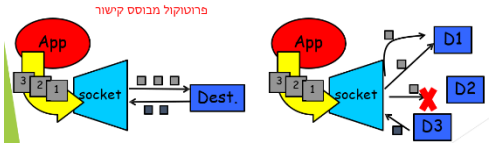
ע"י פקודת Netstat– CMD ניתן לבדוק את כל חיבורי הsocket הזמינים במחשב. לדוגמא:

```
C:\Users\USER>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.1.12:50061     172.217.22.137:443     ESTABLISHED
  TCP    192.168.1.12:50083     216.239.38.120:443     TIME_WAIT
  TCP    192.168.1.12:50428     69.46.36.6:80          ESTABLISHED
  TCP    192.168.1.12:50439     69.46.36.10:4005       ESTABLISHED
  TCP    192.168.1.12:50467     216.58.201.238:443     TIME_WAIT
  TCP    192.168.1.12:50563     34.194.177.112:443     ESTABLISHED
  TCP    192.168.1.12:50671     216.58.204.98:443      TIME_WAIT
  TCP    192.168.1.12:50728     216.58.204.99:443      ESTABLISHED
  TCP    192.168.1.12:50809     216.58.213.174:443     ESTABLISHED
  TCP    192.168.1.12:50811     69.46.36.6:80          TIME_WAIT
  TCP    192.168.1.12:50812     216.58.205.3:443       ESTABLISHED
  TCP    192.168.1.12:50813     216.58.204.97:443      ESTABLISHED
  TCP    192.168.1.12:50814     216.58.207.227:443     ESTABLISHED
  TCP    192.168.1.12:50815     216.58.215.33:443      ESTABLISHED
  TCP    192.168.1.12:50819     69.46.36.6:80          TIME_WAIT
```

כיצד השרת מאפשר ללקוחות להתחבר ולשלוח בקשות והודעות?

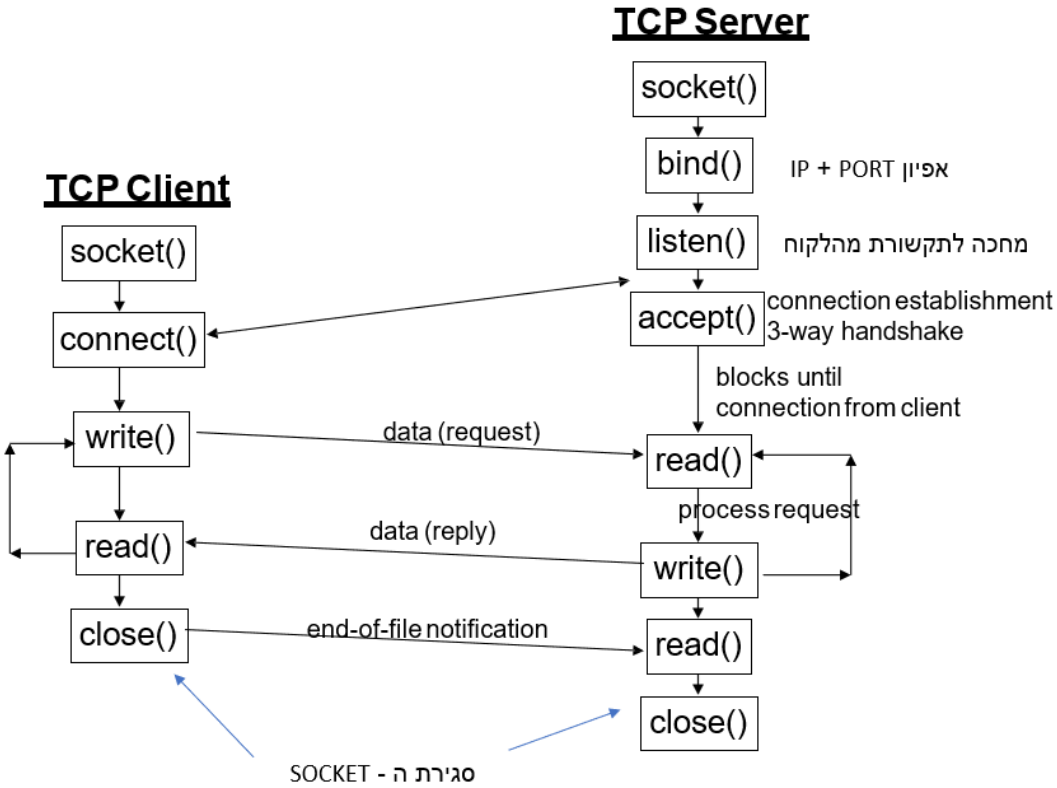


ע"י שימוש בשני סוגי Socket הכרחיים: TCP , UDP. מצורף הסבר על כל אחד מהם:

TCP- Transmission Control Protocol

הוא פרוטוקול הנמצא בשכבת התעבורה- Transport אשר עובד מעל IP. מטרתו היא לאפשר העברת מידע אמינה ורציפה בין שתי ישויות רשת, מעל תשתית שאינה אמינה.
הפרוטוקול אמין (לעומת UDP שאינו אמין), כלומר הוא משתמש ב Windowing על מנת לוודא את הגעת החבילות אליו ולהגביר את מהירות ההעברה ככל שהרשת מהירה יותר.

התקשורת באינטרנט היא בין שרת ששולח מידע ומחשב לקוח שמקבל מידע. השולחים והמקבלים ברשת האינטרנט אורזים את המסרים שלהם באמצעות פרוטוקול **TCP** לחבילות קטנות הנקראות "פקטות" **Packet switching**. פרוטוקול **TCP** שולח את הפקטות באמצעות פרוטוקול **IP** המנחה את הפקטה ליעדה. הפרוטוקולים **TCP** ו **IP** עובדים ביחד בשתי שכבות: שכבת **TCP** מפרקת לפקטות ו **IP** מנחה אל היעד. בגלל שהם עובדים ביחד לפרוטוקול קוראים **TCP/IP**.



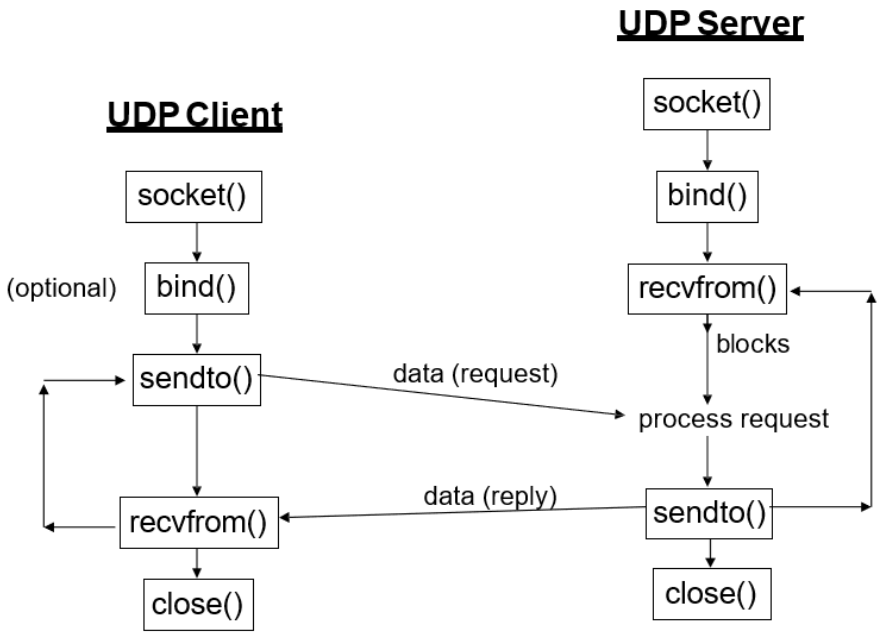
UDP - User Datagram Protocol

UDP הוא פרוטוקול העברת מידע לא אמין הנמצא בשכבת התעבורה של מודל ה OSI.

התואר "פרוטוקול תקשורת לא אמין" ניתן לפרוטוקול בעקבות ההבדלים בינו לפרוטוקול TCP:

- לא מכיל אפשרויות לבקרת זרימה.
- אין אישורים על העברת נתונים מיעד למקור.
- סגמנטים מגיעים לרוב לא לפי הסדר ובמסלולים שונים.

כמו כן, הוא מכיל בתוכו את המקור, היעד, אורך החבילה ו CRC או Checksum לבדיקה שהחבילה אכן מגיעה ללא שגיאות. היתרון הגדול של הפרוטוקול הוא מהירות העברת הנתונים הגדולה שלו (בעקבות חוסר "ההפרעות") והקלות שבה מחשבים והתקני רשת מסוגלים לעבד את הנתונים, בעיקר בגלל גודלו של הסגמנט.



חלק 2- הסבר על קטע הקוד:

את קטע הקוד ביצענו בשני חלקים: ה-Server וה-client. קטעי הקוד מובאים בנוסף לקובץ הוורד.

קוד ה-Server:

```
# Imports
import socket
import threading

# Define constants
import time

HOST = '127.0.0.1'  # Standard loopback IP address (localhost)
PORT = 5555  # Port to listen on (non-privileged ports are > 1023)
FORMAT = 'utf-8'  # Define the encoding format of messages from client-server
ADDR = (HOST, PORT)  # Creating a tuple of IP+PORT
group_id_counter = 1
clients_in_groups=[[]]
groups_id_list=[]
password_groups=[]
```

הקוד מתחיל בייבוא הספריות socket, threading, time שישמשו אותנו בהמשך הקוד. לאחר מכן הגדרנו מספר מרכיבים שיעזרו לנו בעת יצירת ה-group chat:

- ה-HOST זהו כתובת IP סטנדרטית.
- נעבוד בפורט 5555 (פורט רנדומלי).
- הפורמט יהיה utf-8.
- לאחר מכן יצרנו משתנה טאפל שיהיה הכתובת שמחזיקה את כתובת ה-IP והפורט שבו נאזין ללקוח.
- group_id_counter זהו משתנה הסופר את מספר ה-id של הקבוצה הבאה שתיווצר כאשר אתחלנו אותו ב-1.
- groups_id_list זוהי רשימה הכוללת את ה-id של קבוצות הצ'אט שכרגע קיימות.
- password_groups זוהי רשימה הכוללת את הסיסמאות של קבוצות הצ'אט שכרגע קיימות.
- clients_in_groups זוהי מטריצה כך שלכל קבוצת צ'אט תהיה רשימה של ה-sockets שנמצאים בה.

לאחר ההגדרות התחלנו בבניית הפונקציות שינהלו את קבוצות הצ'אט.

<u>פונקציית handle_client1</u>:

פונקציה זו מנהלת את מה שקורה כאשר לקוח חדש נכנס למערכת.

```
# Function that handles a single client connection
# Operates like an echo-server
def handle_client1(conn, addr):
    global group_id_counter
    global groups_id_list
    global password_groups
    global clients_in_groups

    print('[CLIENT CONNECTED] on address: ', addr)  # Printing connection address
    # total_messages = conn.recv(1024).decode(FORMAT)  # Receiving from client # of messages to expect
    # received = 0
    conn.send("Hello client, please choose an option: \n "
              "1. Connect to a group chat. \n 2. Create a group chat. \n "
              "3. Exit the server. \n ".encode(FORMAT))
    msg = conn.recv(1024).decode(FORMAT)
```

תחילה הגדרנו את משתני העזר שהסברנו קודם כגלובליים – כלומר שבפונקציה יכירו בהם.

לאחר מכן התרענו ל-Server כי לקוח חדש נכנס למערכת ואת מיקומו שמתקבל כפרמטר לפונקציה.

כאשר הלקוח ייכנס הוא יקבל תפריט אפשרויות בו הוא יחליט מבין 3 אופציות:

1) הצטרפות לקבוצת צ'אט.
2) יצירת קבוצת צ'אט.
3) יציאה מהמערכת.

את הבחירה שלו אנו מקבלים דרך ה-socket שלו המתקבל כפרמטר לפונקציה ושומרים אותו כמשתנה בשם msg. כעת אנו בודקים מה התקבל.

אם התקבל 1:

```
if msg == "1":
    conn.send("Enter your name: \n".encode(FORMAT))
    name = conn.recv(1024).decode(FORMAT)
    # getting the group id:
    conn.send("Enter group ID: \n".encode(FORMAT))
    gro_id = conn.recv(1024).decode(FORMAT)
    while int(gro_id) not in list(groups_id_list):
        conn.send("There is no group id like this,try again".encode(FORMAT))
        conn.send("Enter group ID: \n".encode(FORMAT))
        gro_id = conn.recv(1024).decode(FORMAT)
    # here the group id is okay
    conn.send("Enter password: \n".encode(FORMAT))
    pass1 = conn.recv(1024).decode(FORMAT)
    while int(pass1) not in list(password_groups):
        conn.send("There is no group password like this,try again".encode(FORMAT))
        conn.send("Enter password: \n".encode(FORMAT))
        pass1 = conn.recv(1024).decode(FORMAT)
    # here the group id and the password is okay
    conn.send(("You're connected to group chat #" + gro_id + "\n").encode(FORMAT))
    # adding the new client to the chat:
    clients_in_groups[int(gro_id) - 1].append(conn)
    # starting the conversation
    handle_conv(conn, gro_id, name)
```

כעת הלקוח מבקש להיכנס לקבוצת צ׳אט. תחילה, נבקש מהלקוח את שמו ונשמור במשתנה name. לאחר מכן, נבקש ממנו את ה-group id שאליו הוא רוצה להיכנס. במידה וה-group id לא קיים במערכת (לא נמצא ברשימה groups_id_list) נאמר למשתמש שאין כזו קבוצה ושינסה שוב. אנו עושות זאת בעזרת לולאת while עם התנאי: int(gro_id) not in list(groups_id_list)

כלומר כל עוד ה-id שהתקבל מהלקוח לא נמצא ברשימה. לאחר שיצאנו מלולאת ה-while, אנו יודעים כי התקבל id תקין מהלקוח. כעת נותר לנו לבדוק את הסיסמא.

במידה והסיסמא לא קיימת במערכת כלומר לא תואמת (לא נמצא ברשימה password_groups) נאמר למשתמש שאין כזו קבוצה ושינסה שוב. אנו עושות זאת בעזרת לולאת while עם התנאי: int(pass1) not in list(password_groups)

כלומר כל עוד הסיסמא שהתקבלה מהלקוח לא נמצאת ברשימה.

לבסוף לאחר שהכל נמצא תקין, נוסיף את הלקוח לקבוצת הצ׳אט על ידי הוספתו לרשימה של הלקוחות בצ׳אט במטריצה clients_in_groups ונשלח את ה- socket של הלקוח, שמו וה-id של הקבוצה לפונקציה handle_conv שנסביר עליה בהמשך הדוח.

אם התקבל 2:

```
elif msg == "2":
    conn.send("Enter your name: \n".encode(FORMAT))
    name = conn.recv(1024).decode(FORMAT)

    conn.send("Enter password for the group: \n".encode(FORMAT))
    password = conn.recv(1024).decode(FORMAT)
    password_groups.append(int(password))

    the_group_id = group_id_counter
    group_id_counter += 1 # for the next group
    groups_id_list.append(int(the_group_id))

    conn.send(("The id of the group is:" + str(the_group_id) +
               "\nYou're connected to group chat #" + str(the_group_id) + "\n" +
               "You can start to talk!").encode(FORMAT))
    # starting the conversation
    clients_in_groups[int(the_group_id) - 1].append(conn)
    handle_conv(conn, the_group_id, name)
```

כעת הלקוח מבקש ליצור קבוצת צ'אט. תחילה, נבקש מהלקוח את שמו ונשמור במשתנה name ונבקש את הסיסמא שהוא רוצה שתהיה לקבוצה ונשמור במשתנה password. את הסיסמא נצרף לרשימת הסיסמאות ונקצה לקבוצה שנוצרה id בעזרת ה-group_id_counte .

את המונה נקדם ב-1 לקבוצה הבאה שתיווצר ואת ה-id של הקבוצה נצרף לרשימת ה-id של הקבוצות. לבסוף נדפיס ללקחו את ה-id הנבחר לקבוצה, נצרפו לקבוצה על ידי הוספתו לרשימה של הלקוחות בצ'אט במטריצה clients_in_groups ונשלח את ה- socket של הלקוח, שמו וה-id של הקבוצה לפונקציה handle_conv שנסביר עליה בהמשך הדוח.

אם התקבל 3:

```
elif msg == "3":
    conn.send("Exit the chat \n".encode(FORMAT))
    print("[CLIENT DISCONNECTED]")
    conn.close()
    exit()
```

הלקוח מבקש כעת לצאת מהמערכת ולכן נדפיס לו ול-server כי הלקוח התנתק ונעשה ל-socket שלו סגירה ויציאה.

אחרת:

```
else:
    conn.send("You chose an invalid option... \n".encode(FORMAT))
```

אם התקבל תו אחר פשוט נדפיס ללקוח שהוא בחר אופציה לא חוקית.

פונקציית handle_conv:

```
# Function that starts the conversation
def handle_conv(conn, gro_id, name):
    global group_id_counter
    global groups_id_list
    global password_groups
    global clients_in_groups
    time.sleep(0.1)
    for client in clients_in_groups:
        # Start Handling Thread For Client
        thread = threading.Thread(target=handle_msg, args=(conn, gro_id, name))  # Create a new thread to every Client on SERVER side
        thread.start()
```

הפונקציה הזו מנהלת את השיח בין הלקוחות שבאותה קבוצת צ'אט. היא מקבלת את ה-socket של הלקוח שקרא לה, את ה-id של הקבוצה בה הוא נמצא ואת שמו של הלקוח.

תחילה הגדרנו את משתני העזר שהסברנו קודם כגלובליים – כלומר שבפונקציה יכירו בהם.

לאחר מכן לכל הלקוחות הנמצאים באותה קבוצת צ'אט ניצור thread ביניהם לבין הלקוח שקרא לפונקציה כאשר קטע הקוד שמנהל את השיח עצמו יהיה handle_msg שמוסבר כעת:

פונקציית handle_msg:

פונקציה זו מנהלת קבלה של הודעה.

```
# Function that handles the msg
def handle_msg(conn, gro_id, name):
    global group_id_counter
    global groups_id_list
    global password_groups
    global clients_in_groups
    while len(list(clients_in_groups[int(gro_id) - 1])) >= 1:
        msg = conn.recv(1024).decode(FORMAT)
        index = 0  # the index of the client in the group
        while (clients_in_groups[int(gro_id) - 1][int(index)] != conn):
            index += 1
```

תחילה הגדרנו את משתני העזר שהסברנו קודם כגלובליים – כלומר שבפונקציה יכירו בהם.

לאחר מכן פתחנו בלולאת while שתנהל את כל קטע הקוד במידה ובקבוצת הצ'אט קיימים לקוחות בכלל. נקבל הודעה מהלקוח שקרא לפונקציה ונרצה למצוא בעזרת לולאת while את מיקומו ברשימה של קבוצת הצ'אט. את מיקומו שמרנו במשתנה index.

בדיקת ההודעה:

```
# checking the msg:
if msg == "Exit":
    # we want to remove this client from the group
    clients_in_groups[int(gro_id) - 1][index] = -1# convert to some int
    clients_in_groups[int(gro_id) - 1].remove(-1)
    conn.send("You chose exit so you removed from the group. \n".encode(FORMAT))
    # printing a msg for everybody that the client has left:
    # print to the server:
    print(f'{name} has removed from the group: {gro_id} \n')
    # print to all the clients in the group:
    for client in list(clients_in_groups[int(gro_id) - 1]):
        if(client != conn):
            client.send(f'{name} has removed from the group \n'.encode(FORMAT))
    print("[CLIENT DISCONNECTED]")
    conn.close()
```

אם ההודעה של הלקוח הייתה Exit אז נרצה להוציא אותו מהקבוצה. נוציא אותו מרשימת הלקוחות בקבוצת הצ׳אט על ידי שינויו למספר מסוים ואז הסרתו (כי לא ניתן להסיר משתנה מסוג socket), נשלח לו הודעה כי הוא בחר לצאת ולכן הוסר מהקבוצה, ונשלח ל-server ולכל הלקוחות האחרים בקבוצה כי הוא הוסר מהקבוצה. לבסוף נסגור את ה-socket שלו.

```
# checking if the group has 0 clients:
if len(list(clients_in_groups[int(gro_id) - 1])) == 0:
    group_id_counter -= 1
    groups_id_list.remove(groups_id_list[int(gro_id) - 1])
    password_groups.remove(password_groups[int(gro_id) - 1])
    print("Group number: "+gro_id+" has deleted!\n") # print to the server that the group has deleted
# closing the thread
return
```

אם לאחר יציאתו לא נשארו יותר לקוחות בקבוצת הצ׳אט (כלומר גודל הרשימה של הלקוחות בצ׳אט שווה ל-0) אז נמחק את הקבוצה מהרשימות הרלוונטיות: ה-id שלה, סיסמתא והורדה ב-1 של מונה ה-id. לבסוף נשלח הודעה ל-server כי הקבוצה נמחקה ונסגור את ה-thread.

```
# now we check if it's a regular msg:
if msg != '':
    for client in list(clients_in_groups[int(gro_id) - 1]):
        if(client != conn):
            client.send(f'{name}: {msg}'.encode(FORMAT))
```

אם זו הודעה רגילה שאיננה הודעה ריקה פשוט נשלח לכל הלקוחות שבקבוצה שהם איננו השולח את ההודעה ששלח ביחד עם שמו.

<u>פונקציית start_server:</u>

פונקציה זו מתחילה את ה-server.

```
def start_server():
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind(ADDR)  # binding socket with specified IP+PORT tuple
    print(f"[LISTENING] server is listening on {HOST}")
    server_socket.listen()  # Server is open for connections
    while True:
        connection, address = server_socket.accept()  # Waiting for client to connect to server (blocking call)
        thread = threading.Thread(target=handle_client1, args=(connection, address))  # Creating new Thread object.
        # Passing the handle func and full address to thread constructor
        thread.start()  # Starting the new thread (<=> handling new client)
    # when all end we want to close the server socket:
    server_socket.close()
```

נאתחל את ה-server על ידי הפונקציות מהספרייה socket, נדפיס כי ה-server מאזין ונבצע את פעולת ההאזנה של השרת. לבסוף בעזרת לולאת while אינסופית נחכה ללקוחות שייכנסו ונקשר אותם באמצעות thread שקורא לפונקציה handle_client1 שהגדרנו לשרת.

לאחר יציאה מהלולאה פשוט נסגור את ה- socket של השרת.

<u>פונקציית ה-main:</u>

```
# Main
if __name__ == '__main__':
    IP = socket.gethostbyname(socket.gethostname())  # finding your current IP address
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Opening Server socket
    print("[STARTING] server is starting...")
    start_server()
    print("THE END!")
```

בפונקציית ה-main נמצא את ה-IP הרלוונטי, ניצור לשרת socket, נדפיס כי השרת מתחיל לעבוד ואז נקרא לפונקציה start_server() שיצרנו שמפעילה את כל שאר הפונקציות האחרות.

לאחר שהיא נגמרת נדפיס "THE END!".

<u>קוד ה-Client:</u>

```
# Imports
import socket
import threading

# Define constants
import time

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 5555  # The port used by the server
FORMAT = 'utf-8'
ADDR = (HOST, PORT)  # Creating a tuple of IP+PORT
```

הקוד מתחיל בייבוא הספריות socket, threading, time שישמשו אותנו בהמשך הקוד. לאחר מכן הגדרנו מספר מרכיבים:

- ה-HOST זהו כתובת IP סטנדרטית שתואמת לכתובת ה-IP שהגדרנו לשרת.
- נעבוד בפורט 5555 שתואמת לפורט שהגדרנו לשרת.
- הפורמט יהיה utf-8.
- לאחר מכן יצרנו משתנה טאפל שיהיה הכתובת שמחזיקה את כתובת ה-IP והפורט.

פונקציית start_client:

```
def start_client():
    client_socket.connect(ADDR)  # Connecting to server's socket
    # creating a thread:
    thread = threading.Thread(target=receive)
    thread.start()

    while (True):
        # Get the client msg
        msg = input()
        client_socket.send(msg.encode(FORMAT))
```

פונקציה זו מקשרת את ה-socket של הלקוח לכתובת שלו. (הגדרת ה-socket של הלקוח תוגדר ב-main), ויוצרת לו thread עם הפונקציה receive שנסביר עליה בהמשך. בעזרת לולאת while אינסופית נחכה לקלט מהלקוח ונשלח ב-socket אל השרת.

פונקציית receive:

```
def receive(): #Recieve message from server:
    while True:
        # Receive Message From Server
        msg = client_socket.recv(1024).decode(FORMAT)
        if msg:
            print(msg)
```

אם נשלחת הודעה ללקוח נדאג שהלקוח יקבל אותה בעזרת לולאת while אינסופית ואם היא לא ריקה נדפיס אותה.

פונקציית ה-main:

```
if __name__ == "__main__":
    IP = socket.gethostbyname(socket.gethostname())
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    print("[CLIENT] Started running")
    start_client()
    print("Goodbye client:)")
```

בפונקציית ה-main נמצא את ה-IP הרלוונטי, ניצור ללקוח socket, נדפיס לשרת כי לקוח מתחיל לרוץ, ואז נקרא לפונקציה start_client() שיצרנו שמפעילה את כל שאר הפונקציות האחרות.

לאחר שהיא נגמרת נדפיס : "Goodbye client:()"

דוגמאות הרצה:

השרת בתחילת הרצה :

```
(venv) PS C:\network_project> python .\Server.py
[STARTING] server is starting...
[LISTENING] server is listening on 127.0.0.1
```

הלקוח בתחילת הרצה :

```
(venv) PS C:\network_project> python .\Client1.py
[CLIENT] Started running
Hello client, please choose an option:
 1. Connect to a group chat.
 2. Create a group chat.
 3. Exit the server.
```

בחירה באופציה 2 :

```
2
Enter your name:

Hadas
Enter password for the group:

12345
The id of the group is:1
You're connected to group chat #1
You can start to talk!
```

הלקוח הראשון הוא בשם הדס ובחר סיסמא : 12345 לקבוצה שה-id שלה הוא 1.

לקוח חדש עם בחירה באופציה 1 :

בחירה של id שגוי :

```
1
Enter your name:

Shoham
Enter group ID:

2
There is no group id like this,try again
Enter group ID:
```

בחירה של Id תקין וסיסמא שגויה:

```
Enter group ID:

1
Enter password:

123
There is no group password like this,try again
Enter password:
```

בחירה של Id תקין וסיסמא תקינה:

```
Enter password:

12345
You're connected to group chat #1
```

השיחה בין הלקוחות:

```
You're connected to group chat #1

Hello Hadas!!!!
```

אצל שהם

```
The id of the group is:1
You're connected to group chat #1
You can start to talk!
Shoham: Hello Hadas!!!!
```

אצל הדס

בנתיים בשרת:

```
(venv) PS C:\network_project> python .\Server.py
[STARTING] server is starting...
[LISTENING] server is listening on 127.0.0.1
[CLIENT CONNECTED] on address:  ('127.0.0.1', 63044)
[CLIENT CONNECTED] on address:  ('127.0.0.1', 63071)
```

שהם בוחרת לצאת מהקבוצה:

```
Hello Hadas!!!!
Exit
You chose exit so you removed from the group.
```

אצל שהם

```
Shoham: Hello Hadas!!!!
Shoham has removed from the group
```

אצל הדס

```
Shoham has removed from the group: 1

[CLIENT DISCONNECTED]
```

אצל השרת

הדס בוחרת לצאת מהקבוצה:

```
Exit
You chose exit so you removed from the group.
```

אצל הדס

```
[CLIENT DISCONNECTED]
Hadas has removed from the group: 1

[CLIENT DISCONNECTED]
Group number: 1 has deleted!
```

אצל השרת

כעת הקבוצה כולה נמחקה.

לקוח חדש עם אופציה 3:

```
(venv) PS C:\network_project> python .\Client1.py
[CLIENT] Started running
Hello client, please choose an option:
 1. Connect to a group chat.
 2. Create a group chat.
 3. Exit the server.

3
Exit the chat
```

לקוח חדש עם קבוצה חדשה:

```
(venv) PS C:\network_project> python .\Client1.py
[CLIENT] Started running
Hello client, please choose an option:
 1. Connect to a group chat.
 2. Create a group chat.
 3. Exit the server.

2
Enter your name:
```

```
Enter your name:

Bar
Enter password for the group:

123
The id of the group is:1
You're connected to group chat #1
You can start to talk!
```

קיבלנו קוצה עם id=1 כי הקודמת נמחקה.

עוד לקוח חדש עם קבוצה חדשה:

```
2
Enter your name:

Hadassss
Enter password for the group:

123456
The id of the group is:2
You're connected to group chat #2
You can start to talk!
```

כעת id=2 כי יצרנו עוד קבוצה.

אנו מזמינות אותך לנסות את הקוד שלנו גם בעצמך 😊.

חלק 3- Socket Handshake

בחלק זה נדרשנו לכתוב הסבר אודות socket handshake. כפי שנלמד בהרצאה ובתרגול, פעולת הHandshake קיימת רק בפרוטוקול TCP השתמשנו רבות במהלך כתיבת הקוד. תהליך זה אחראי על יצירת התקשורת (ראה הרחבה על פרוטוקול TCPברקע התיאורטי עמ' 4) ומאפשר יצירת קשר בין רשתות רבות ושונות, ללא תלות במבנה הרשת ובטכנולוגיה שעומדת בבסיסה.

חלק מהפקודות בהן השתמשנו מפרוטוקול TCP הן פקודות חוסמות, כלומר פקודות שהprocess לא ממשיך את פעולתו עד אשר יתבצע interrupt או אירוע מסוים.
נפרט על העיקריות בהן השתמשנו :

Connect: בביצוע פקודה זו, התהליך יחכה עד ליצירת החיבור מצד הלקוח ולאחר מכן ימשיך את המשך הקוד.

Accept: בביצוע פקודה זו, התהליך ייחסם ויחכה עד שנוצר חיבור מצד השרת ולאחר מכן ימשיך את המשך הקוד.

Recv, Recvfrom: בביצוע פקודה זו, התהליך ייחסם ויחכה עד שמתקבלת חבילה של נתונים ולאחר מכן ימשיך את המשך הקוד.

Send, Sendto: בביצוע פקודה זו, התהליך ייחסם עד שהנתונים יידחפו לבאפר של הsocket ולאחר מכן ימשיך את הקוד.

כמו כן, עבור תוכניות פשוטות יחסית, החסימה יחסית נוחה. עבור תוכניות מורכבות, קיימים חיבורים מרובים, השליחה והקבלה מתבצעת במקביל וכן ביצוע בו זמנית של עיבוד שאינו ברשת.

סיכום ומסקנות:

בכתיבת מטלה זו החכמנו ולמדנו רבות אודות ארכיטקטורת שרת-לקוח, אשר מרכזית וחשובה מאוד. הצלחנו ליצור תקשורת בין השרת- אשר מנהל את הצ׳אטים הקבוצתיים, ללקוחות- אשר משתמשים בשירותי השרת ונהנים משליחת וקבלת הודעות ביישום הקבוצתי. על ידי כך יכולים לתקשר ביניהם בקבוצות מוגדרות מראש.
וכן התעמקנו בשכבות 4-5 , שכבות התעבורה והאפליקציה, דבר אשר עזר לנו להבין וליישם בצורה טובה מאוד את הנלמד בהרצאות ובתרגולים!

יתר על כן, פיתחנו את יכולות התכנות שלנו ואף למדנו שפה חדשה ומגניבה- Python אשר אנחנו בטוחות שתשמש אותנו עוד רבות.