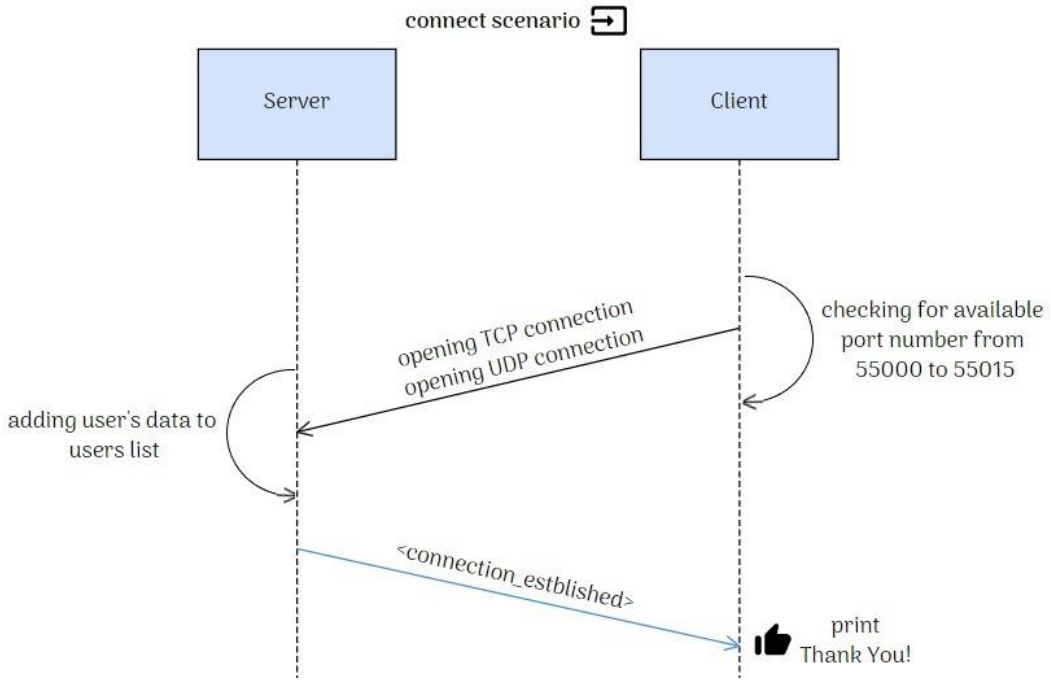
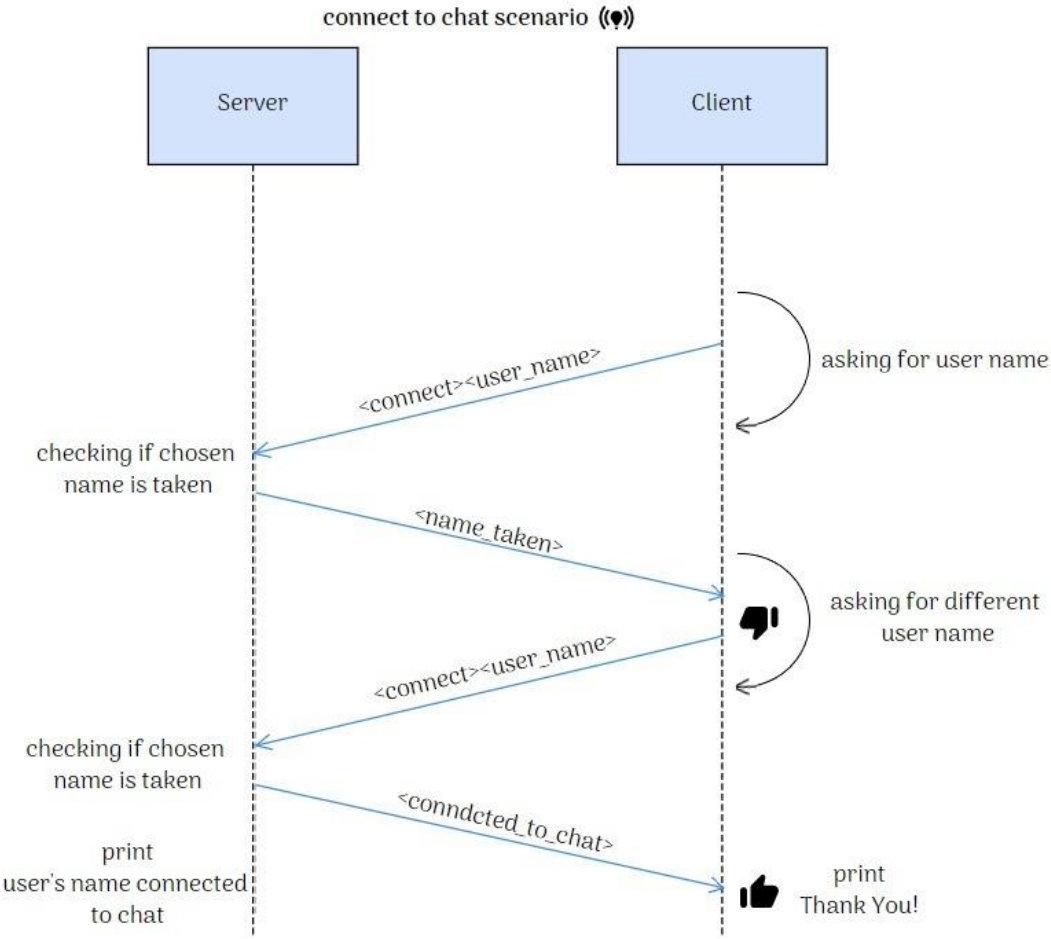
בס"ד

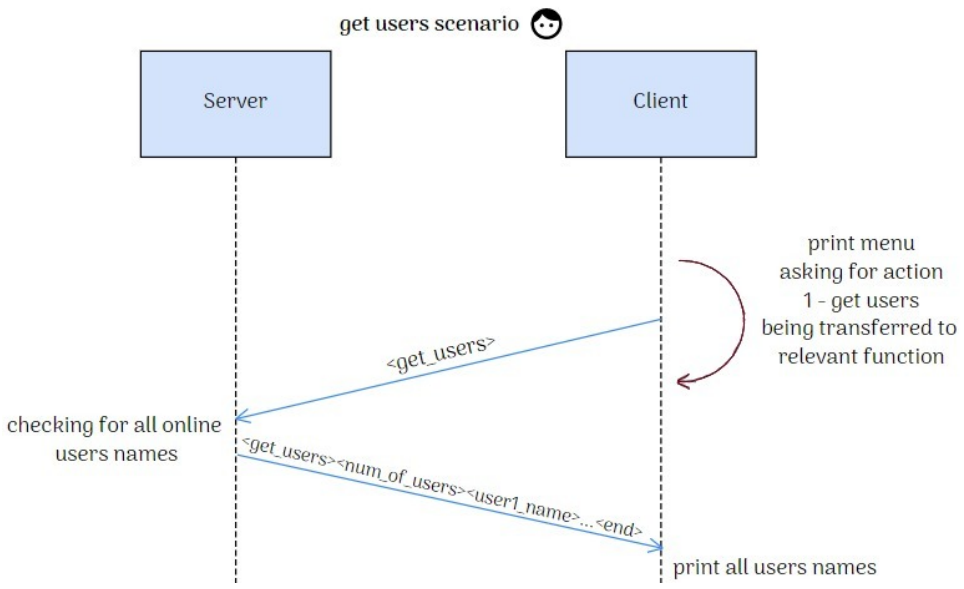
חלק ב'

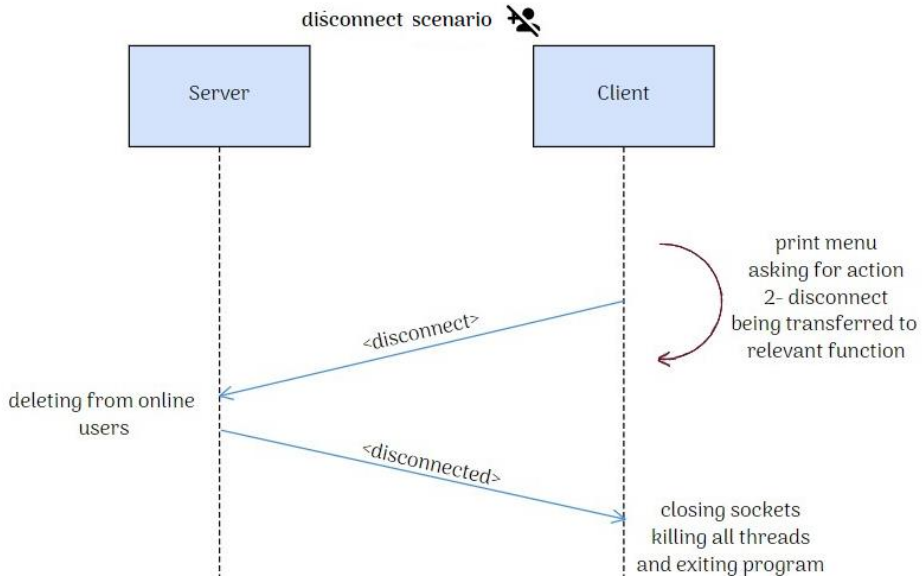
דיאגרמות:

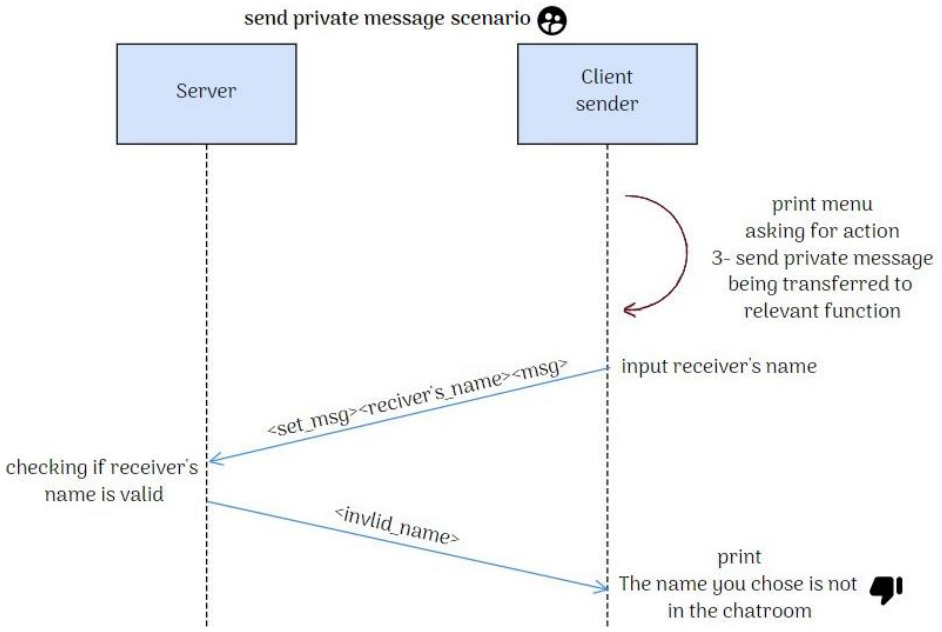




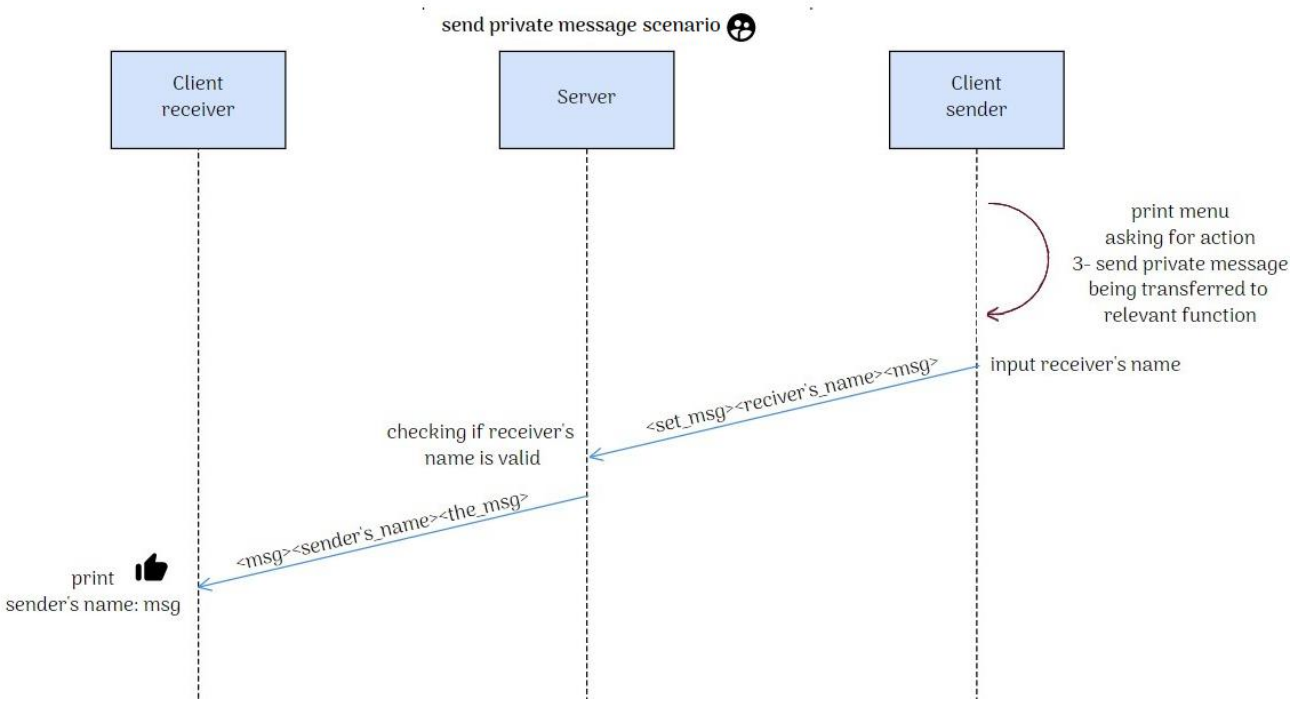
בס"ד

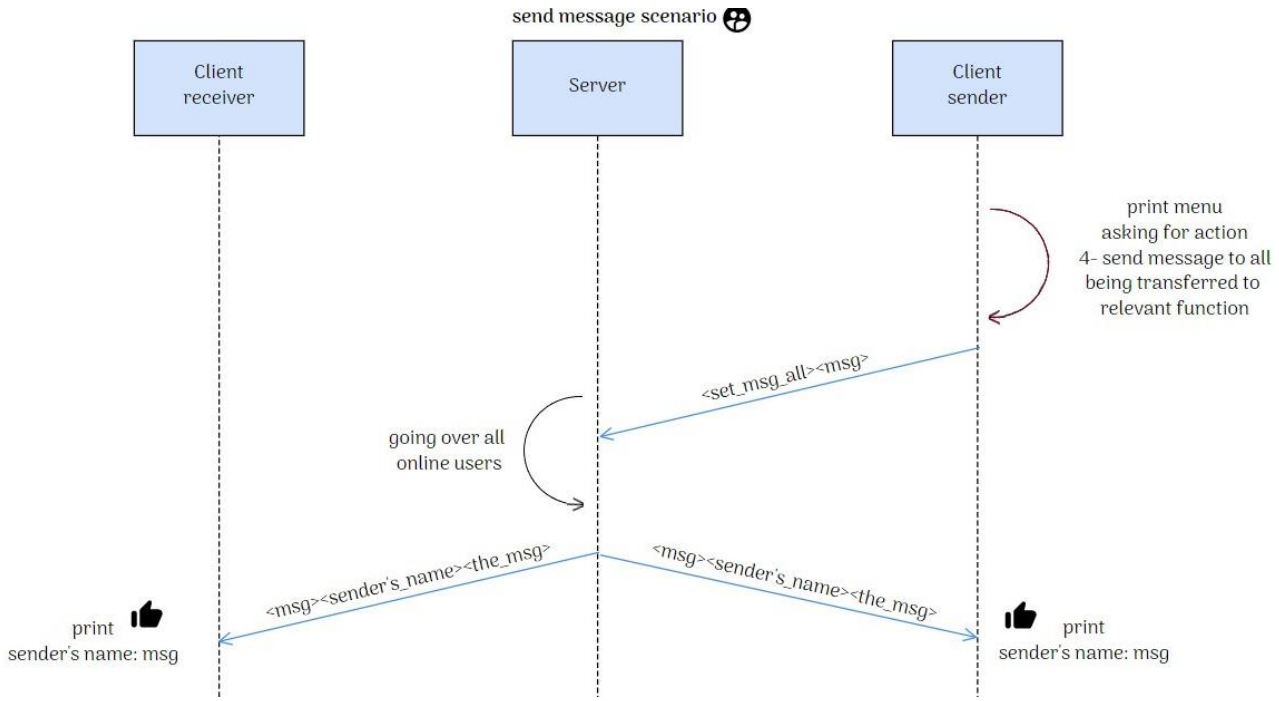

get users scenario
Server
Client
print menu
asking for action
1 - get users
being transferred to
relevant function
<get_users>
checking for all online
users names
<get_users><num_of_users><user1_name>...<end>
print all users names


disconnect scenario
Server
Client
print menu
asking for action
2- disconnect
being transferred to
relevant function
<disconnect>
deleting from online
users
<disconnected>
closing sockets
killing all threads
and exiting program

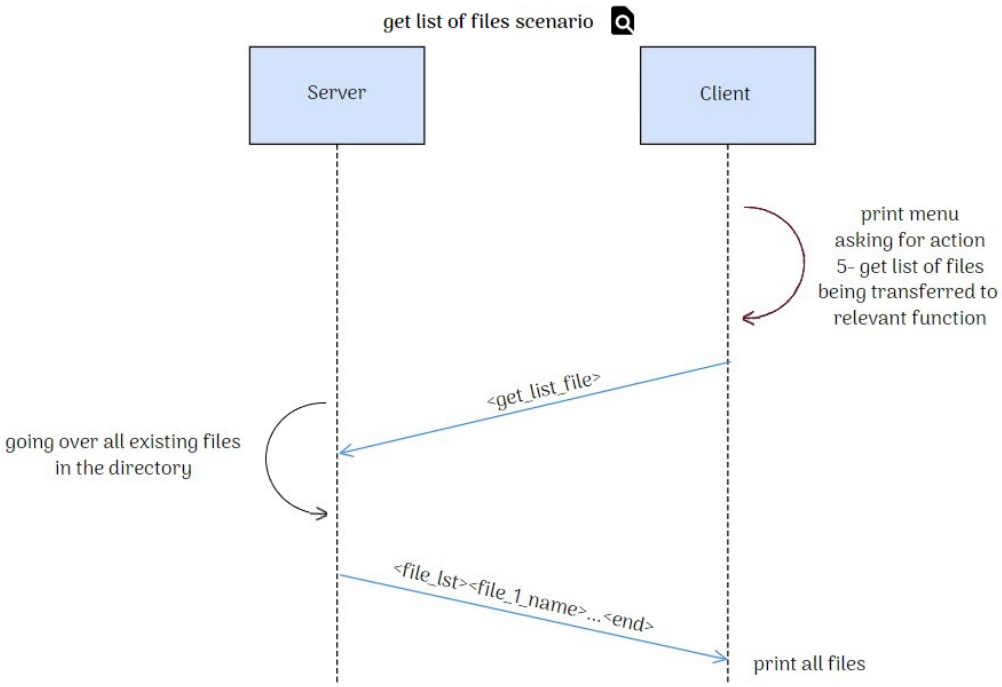

send private message scenario
Server
Client
sender
print menu
asking for action
3- send private message
being transferred to
relevant function
input receiver's name
<set_msg><reciver's_name><msg>
checking if receiver's
name is valid
<invlid_name>
print
The name you chose is not
in the chatroom

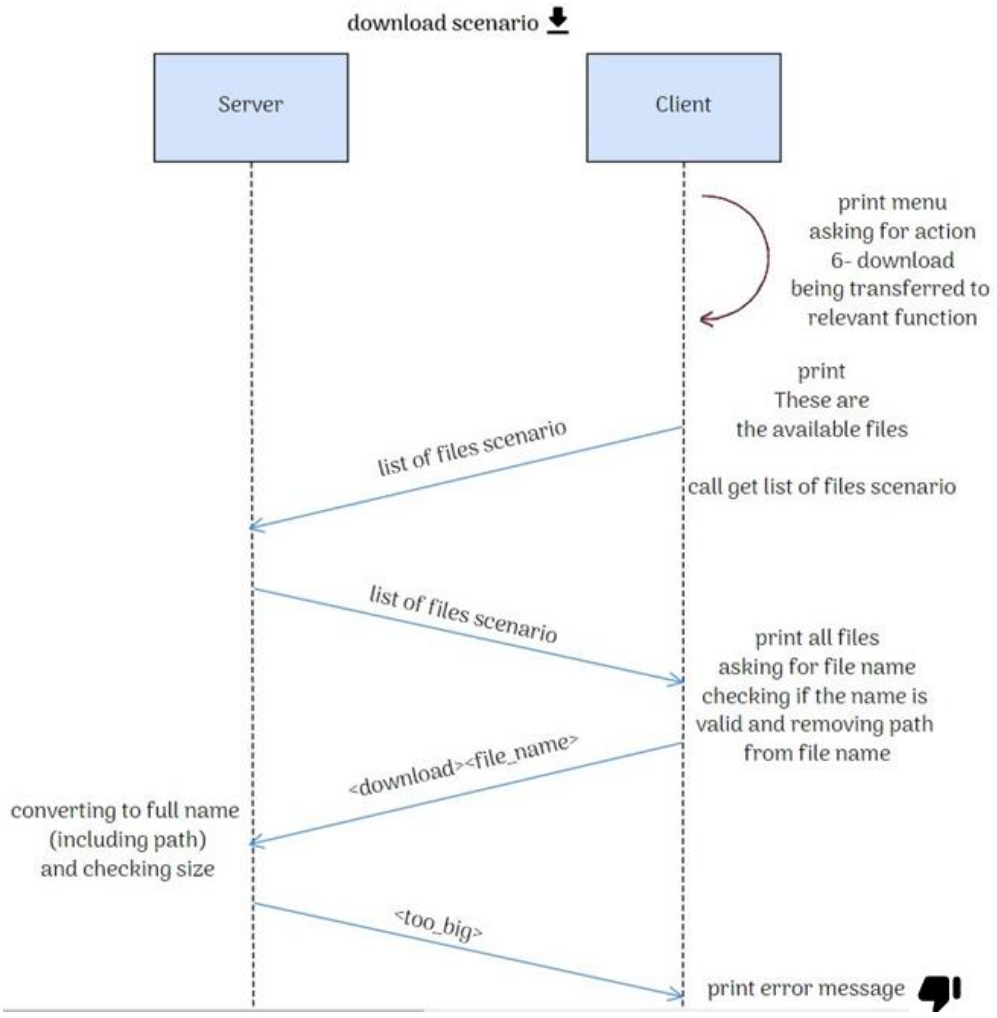
בס"ד


send private message scenario
Client receiver
Server
Client sender
print menu asking for action 3- send private message being transferred to relevant function
input receiver's name
<set_msg><reciver's_name><msg>
checking if receiver's name is valid
<msg><sender's_name><the_msg>
print sender's name: msg

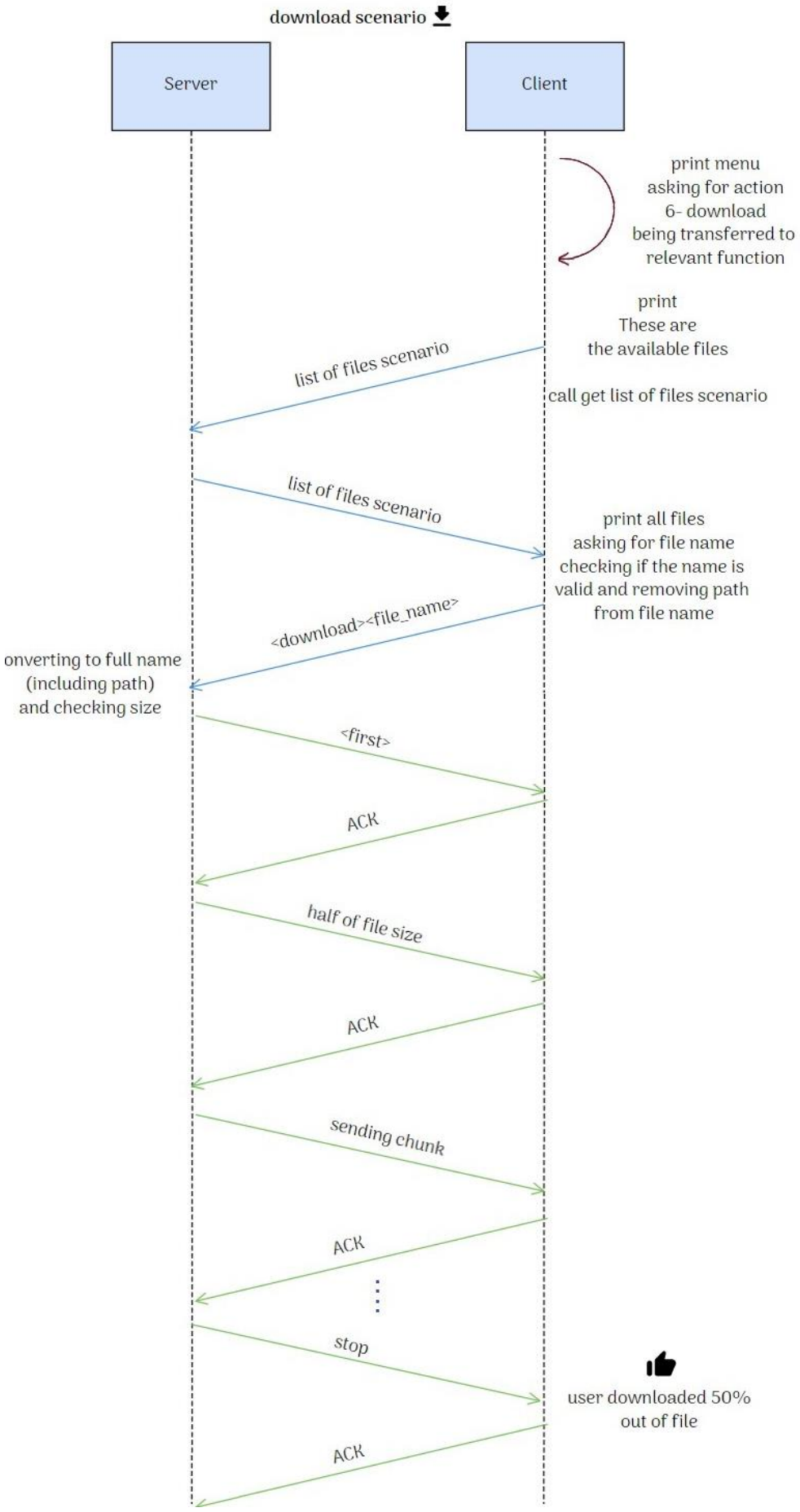

send message scenario
Client receiver
Server
Client sender
print menu asking for action 4- send message to all being transferred to relevant function
<set_msg_all><msg>
going over all online users
<msg><sender's_name><the_msg>
<msg><sender's_name><the_msg>
print sender's name: msg
print sender's name: msg

בס"ד

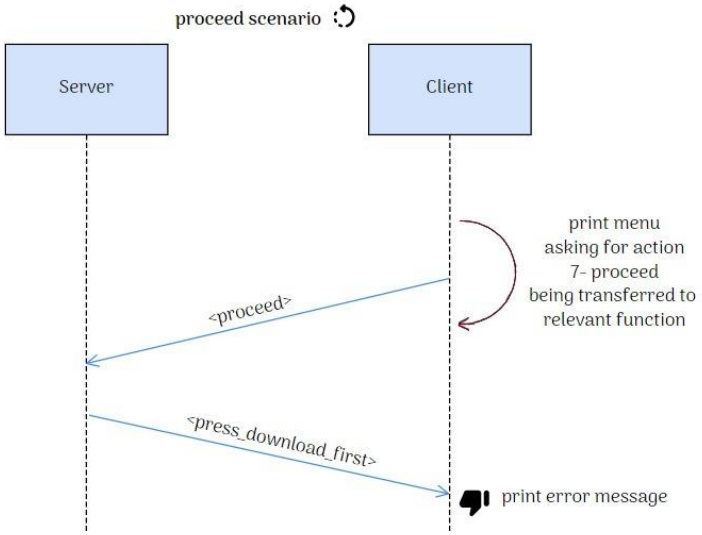

get list of files scenario
Server
Client
print menu
asking for action
5- get list of files
being transferred to
relevant function
<get_list_file>
going over all existing files
in the directory
<file_lst><file_1_name>...<end>
print all files

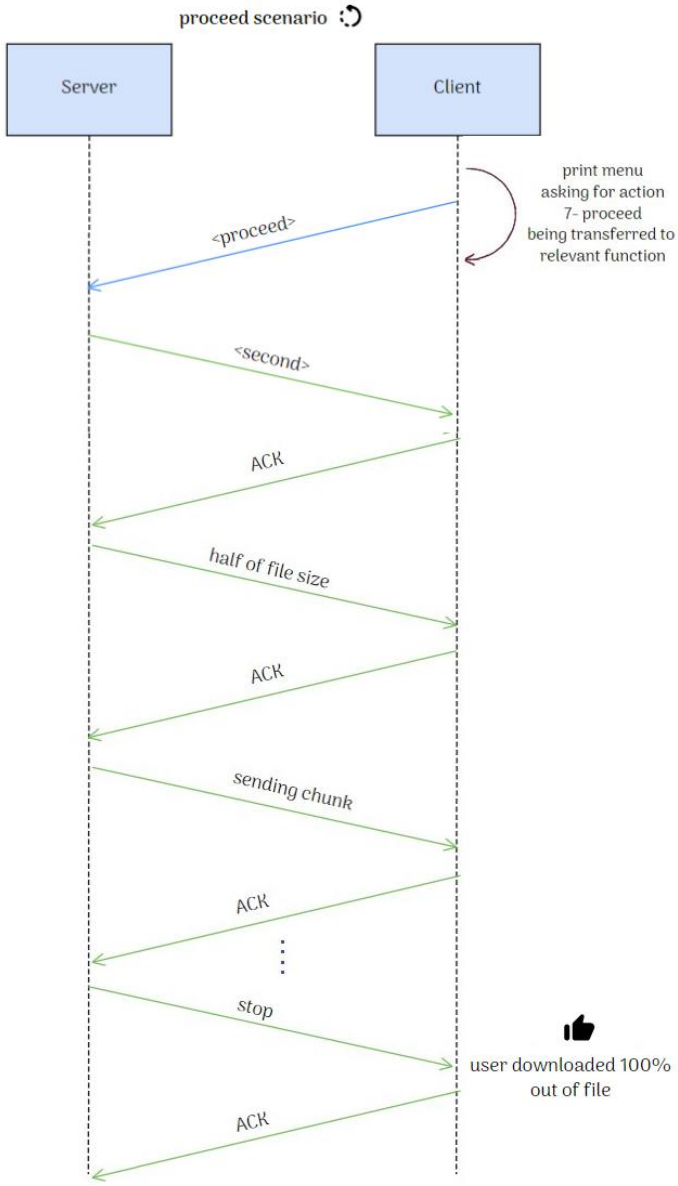

download scenario
Server
Client
print menu
asking for action
6- download
being transferred to
relevant function
print
These are
the available files
list of files scenario
call get list of files scenario
list of files scenario
print all files
asking for file name
checking if the name is
valid and removing path
from file name
<download><file_name>
converting to full name
(including path)
and checking size
<too_big>
print error message

בס"ד


download scenario
Server
Client
print menu
asking for action
6- download
being transferred to
relevant function
print
These are
the available files
list of files scenario
call get list of files scenario
list of files scenario
print all files
asking for file name
checking if the name is
valid and removing path
from file name
<download><file_name>
onverting to full name
(including path)
and checking size
<first>
ACK
half of file size
ACK
sending chunk
ACK
stop
user downloaded 50%
out of file
ACK

בס"ד


proceed scenario
Server
Client
print menu
asking for action
7- proceed
being transferred to
relevant function
<proceed>
<press_download_first>
print error message


proceed scenario
Server
Client
print menu
asking for action
7- proceed
being transferred to
relevant function
<proceed>
<second>
ACK
half of file size
ACK
sending chunk
ACK
stop
user downloaded 100%
out of file
ACK

בדיאגרמה זו אנו רואים את כלל המצבים של המערכת אך בנוסף ישנם שלושה טרדים בקליינט:

- טרד ראשון אחראי על ACTIONS (הטרד האדום בדיאגרמות)– מקבל את הפקודה שהיוזר רוצה לממש ודואג לשלוח את הפקודה (טמפלט) הרלוונטית לשרת.
- טרד שני אחראי על קבלת הודעות TCP (הטרד הכחול בדיאגרמות) מהשרת וניהול התגובה בהתאם.
- טרד שלישי אחראי על קבלת הודעות UDP (הטרד הירוק בדיאגרמות) מהשרת וניהול התגובה בהתאם ובנוסף לכך הוא שולח אקים.

ובצד השרת עבור כל לקוח יש טרד. הטרד אחראי להקשיב לפקודות (טמפלט) הנשלחות אליו מהלקוח ולנהל אותן בהתאם.

<u>תשובה ל - כיצד המערכת מתגברת על איבוד חבילות?</u>

בחרנו לוודא את אמינות המערכת על ידי שימוש ב Reliable UDP התמקדנו במיוחד בצורת שליחת האקים.

לטובת העניין יצרנו פונקציה ספציפית שמוודאת קבלה אצל הלקוח בעזרת קבלת הודעת ACK ממנו המנוהלת ע"י חריגת TIMEOUT. במידה וקפצה חריגה (ה TIMEOUT המגודר עבר) הפונקציה שולחת מחדש את ההודעה ומחכה שוב לתגובה.

<u>תשובה ל – כיצד המערכת מתגברת על בעיות LATENCY?</u>

בשביל להתגבר על בעיות LATENCY (זמן השהייה של המערכת קבענו את הדברים הבאים:

TIMEOUT קצר (שנייה) לווידוא קבלת ACK.

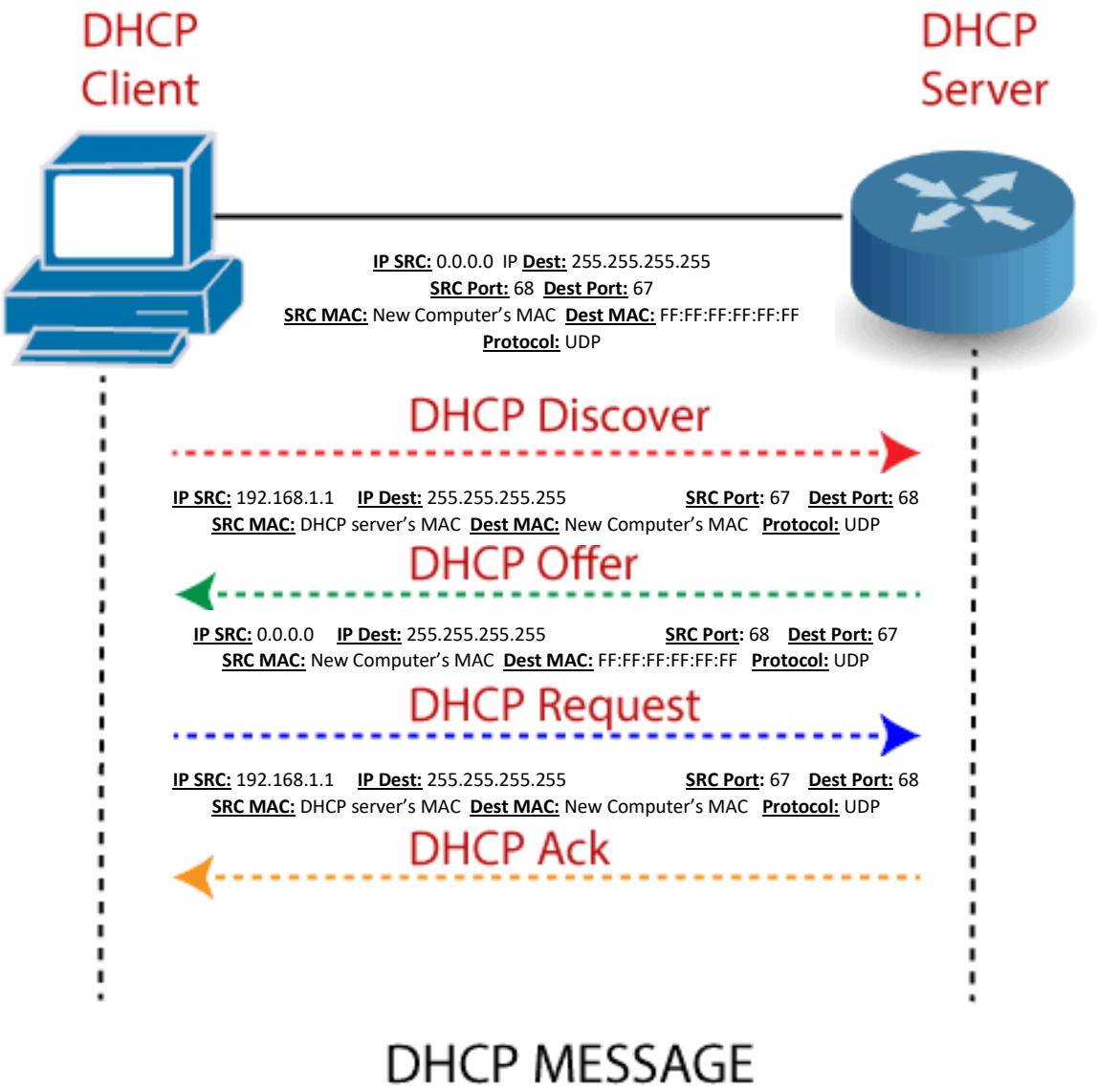
מכיוון שישנו כבר שלב התחברות לצ'אט המתבצע בעזרת TCP אין צורך לבצע את שלב "לחיצת היד" ב UDP (בקשת התחברות, קבלת אישור).

בס"ד

חלק ג'

תשובה לשאלה 1:

שרת DHCP – קונפיגורציה התחלתית המחלקת כתובת IP זמנית ונותנת את כתובת שרת ה DNS, כתובת SUBNET וכתובת הנתב הראשון שמוציא אותנו החוצה מהרשת הפנימית.



DHCP DISCOVER: המחשב החדש (0.0.0.0 – מכיוון שלא הוגדר IP) שולח הודעת ברודקאסט (255.255.255.255) עם ה MAC שמוגדר לו (קבוע לכל מחשב) ובודק האם יש שרת DHCP באזור. ההודעה נשלחת עפ"י פרוטוקול התעבורה UDP. הפורטים נבחרו על סמך זה ש PORT 67 מתייחס לשרת DHCP ו PORT 68 מתייחס לקליינט המבקש שירות DHCP.

בס"ד

DHCP OFFER: שרת DHCP (192.168.1.1 – דיפולטיבי) שולח הודעת ברודקאסט (255.255.255.255 - מכיוון שעדיין לא הוגדר IP לקליינט) חזרה לקליינט עם ה MAC שמוגדר לו ונותן הצעה לכתובת IP עבור המחשב החדש. ההודעה נשלחת עפ"י פרוטוקול התעבורה UDP וכתגובה הפורטים כמובן יתחלפו.

DHCP REQUEST: המחשב החדש (0.0.0.0 – מכיוון שלא הוגדר IP) שולח הודעת ברודקאסט (255.255.255.255) עם ה MAC שמוגדר לו (קבוע לכל מחשב) ושולח הודעת ARP כדי לבדוק אם כתובת ה IP שהוצעה בשלב הקודם אכן פנויה (אם הכתובת לא פנויה המחשב יעבור לכתובת הבאה שהוצעה ויבדוק אם היא פנויה...) ובמידה והכתובת פנויה יתבצע השלב הבא. ההודעה נשלחת עפ"י פרוטוקול התעבורה UDP וכתגובה חזרה הפורטים כמובן יתחלפו.

DHCP Ack: שרת DHCP (192.168.1.1 – דיפולטיבי) שולח הודעת ברודקאסט (255.255.255.255 - מכיוון שעדיין לא הוגדר IP לקליינט) חזרה לקליינט עם ה MAC שמוגדר לו ומאשר את ההצעה לכתובת IP עבור המחשב החדש(לדוגמא: 10.0.0.1). ההודעה נשלחת עפ"י פרוטוקול התעבורה UDP וכתגובה חזרה הפורטים כמובן יתחלפו. כמו כן השרת שולח לקליינט כתובת שרת ה DNS, כתובת SUBNET, כתובת הנתב הראשון שמוציא אותנו החוצה מהרשת הפנימית ומשך זמן הקצאת הכתובת (במידה וביקש).

הערה: נשים לב שבמידה והשרת DHCP לא נמצא באותה הרשת של הקליינט ומדובר על שרת חיצוני אז התהליך יתבצע על ידי SWITCH שיעשה FLOODING.

לאחר שלבים אלה נרצה לבצע גישה לשרת (שכתובתו לצורך העניין היא 1.2.3.4), נרצה לברר מה היא כתובת ה MAC של אותו שרת. נניח בשאלה זו שהשרת מחוץ לרשת:
תשלח הודעת ARP ומכיוון שהשרת לא נמצא באותה הרשת ההודעה תגיע לנתב ה GATEWAY בשביל לקבל את כתובת ה MAC של השרת –

IP SRC: 10.0.0.1 **IP Dest:** 1.2.3.4
SRC MAC: New Computer's MAC **Dest MAC:** FF:FF:FF:FF:FF:FF **Protocol:** ARP (Request)

הודעת ARP חזור תראה כך:

IP SRC: Gateway's IP **IP Dest:** 10.0.0.1
SRC MAC: Gateway's MAC **Dest MAC:** New Computer's MAC **Protocol:** ARP (Response)

לאחר קבלת כתובת ה MAC מהודעת ה ARP הקודמת נוכל לבצע גישה לשרת, נניח כי הלקוח ביצע bind לport 55000 ושביצע חיבור TCP, בנוסף נניח כי השרת ביצע bind לport 50000, השרת והלקוח מתחילים בתהליך לחיצת היד, שלב ראשון – SYN מהלקוח לשרת

IP SRC: 10.0.0.1 **IP Dest:** 1.2.3.4 **Port SRC:** 55000 **Port Dest:** 50000
SRC MAC: New Computer's MAC **Dest MAC:** Server's MAC **Protocol:** TCP SYN

שלב שני – SYN ACK מהשרת ללקוח

IP SRC: 1.2.3.4 **IP Dest:** 10.0.0.1 **Port SRC:** 50000 **Port Dest:** 55000
SRC MAC: Server's MAC **Dest MAC:** New Computer's MAC **Protocol:** TCP SYN ACK

שלב שני –ACK מהשרת ללקוח

IP SRC: 10.0.0.1 **IP Dest:** 1.2.3.4 **Port SRC:** 55000 **Port Dest:** 50000
SRC MAC: New Computer's MAC **Dest MAC:** Server's MAC **Protocol:** TCP ACK

בס"ד

ניתן לראות את שלושת השלבים האלה בתצלום הwireshark הבא:

```
TCP        74 55015 → 50000 [SYN] Seq=0 Win=65495 [TCP CHECKSUM INCORRECT] …
TCP        74 50000 → 55015 [SYN, ACK] Seq=0 Ack=1 Win=65483 [TCP CHECKSUM …
TCP        66 55015 → 50000 [ACK] Seq=1 Ack=1 Win=65536 [TCP CHECKSUM INCOR…
```

כעת הלקוח (ששולח את ההודעה) ישלח בקשה לשליחת הודעה (נניח שהלקוח המקבל מחובר לצ'ט כבר, אם לא כל השלבים למעלה יחולו עליו גם), בנוסף נניח שכתובת הלקוח המקבל היא 192.0.0.2 והport שלו הוא 55001, כעת ההודעה שהלקוח השולח (נקרא לו לקוח א') רוצה לשלוח ללקוח המקבל (נקרא לו לקוח ב') תשלח תחילה לשרת,

הודעה מלקוח א' לשרת:

IP SRC: 10.0.0.1 **IP Dest:** 1.2.3.4 **Port SRC:** 55000 **Port Dest:** 50000
SRC MAC: New Computer's MAC **Dest MAC:** Server's MAC **Protocol:** TCP

השרת ישיב בACK (כי זה פרוטוקול TCP) וכך יסמן ללקוח א' שבקשתו לשליחת הודעה התקבלה.

הודעה משרת ללקוח א':

IP SRC: 1.2.3.4 **IP Dest:** 10.0.0.1 **Port SRC:** 50000 **Port Dest:** 55000
SRC MAC: Server's MAC **Dest MAC:** New Computer's MAC **Protocol:** TCP

כעת השרת ישלח את ההודעה של לקוח א' ללקוח ב',

הודעה משרת ללקוח ב':

IP SRC: 1.2.3.4 **IP Dest:** 192.0.0.2 **Port SRC:** 50000 **Port Dest:** 55001
SRC MAC: Server's MAC **Dest MAC:** Other Computer's MAC **Protocol:** TCP

ועכשיו לקוח ב' ישיב בACK לשרת (יסמן לו שההודעה התקבלה בהצלחה)

הודעה מלקוח ב' לשרת:

IP SRC: 192.0.0.2 **IP Dest:** 1.2.3.4 **Port SRC:** 55001 **Port Dest:** 50000
SRC MAC: Other Computer's MAC **Dest MAC:** Server's MAC **Protocol:** TCP

ניתן לראות דוגמה לארבעת ההודעות האחרונות בתצלום בwireshark הבא:

```
55000 → 50000 [PSH, ACK] Seq=16 Ack=44 Win=65536 [TCP CHECKSU…
50000 → 55000 [ACK] Seq=44 Ack=35 Win=65536 [TCP CHECKSUM INC…
50000 → 55001 [PSH, ACK] Seq=58 Ack=31 Win=65536 [TCP CHECKSU…
55001 → 50000 [ACK] Seq=31 Ack=73 Win=65536 [TCP CHECKSUM INC…
```

תשובה לשאלה 2:

CRC היא שיטה לזיהוי רצף של שגיאות בעת העברת חבילות. השיטה מחושבת בצורה הבאה:

נבדוק מהי החזקה הכי גדולה של הפולינום ונוסיף מס' אפסים זהה מימין למידע. לאחר מכן נבצע את חישוב החלוקה בפולינום. נשלח את המידע יחד עם שארית (r) כתקורה. ברגע שהתקבלה החבילה בצד השני, המקבל מחבר את המידע עם השארית ואז מבצע חלוקה בפולינום הידוע. במידה והתקבל 0 אז לא התבצעו שום רצף שגיאות בדרך. במידה והתוצאה שונה מ 0 ישנן רצף שגיאות שהתבצעו בדרך.

תשובה לשאלה 3:

HTTP1.0:

- מבוסס על פרוטוקול TCP.

בס"ד

- לא תומך בחילוק המידע לצ'אנקים משמע לא מאפשר שליחה של הדרים נוספים אחרי גוף ההודעה.
- אינו מאובטח.
- העברת המידע מתבצעת בחתיכה אחת. עבור כל חיבור TCP מתבצעת בקשה ותגובה יחידה. אם נרצה עוד בקשה נצטרך לפתוח חיבור חדש.

:HTTP1.1

- השרת משאיר את החיבור פתוח גם לאחר שליחת תגובה.
- הודעות HTTP עוקבות בין אותו לקוח ושרת ישלחו על אותו חיבור פתוח.
- תומך בהעברת מידע בחלקים.
- זמן ה RTT מתקצר בצורה משמעותית מכיוון והחיבור נשאר פתוח בין השרת ללקוח.
- מבוסס על פרוטוקול TCP.

הבעיה ב HTTP1.1 הוא שהשרת עונה לבקשות על פי הסדר. כך שאם הבקשה הראשונה גדולה אז שאר הבקשות נדחות במיידית.

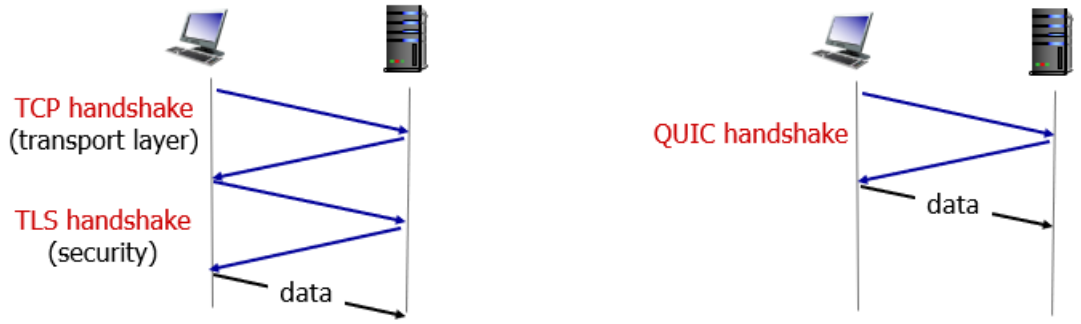
:HTTP2.0

- מפחית את ההשהיה עבור בקשות HTTP מרובות אובייקטים.
- מגביר את הגמישות של העברת האובייקטים בין השרת ללקוח.
- מתודות, רוב ההדרים של השדות נשארים זהים עוד מ HTTP1.1.
- סדר העברת המידע של האובייקטים המבוקשים מתבסס על העדפה ספציפית של הלקוח (בשונה מ HTTP1.1 שבו הבקשה הראשונה נענית ראשונה וכן הלאה על פי סדר הבקשות).
- דוחף אובייקטים שלאו דווקא התבקשו ללקוח. לדוגמא: ברגע שלקוח מבקש דף אינטרנט מסוים הכולל תמונות, הלקוח פותח בקשה עבור הדף ומקבל את ה HTML ובנוסף בלי לבקש מקבל את התמונות שבתוכו.
- קיים חיבור יחיד המחולק לסטרימים שונים.
- מבוסס על פרוטוקול TCP.

:QUIC

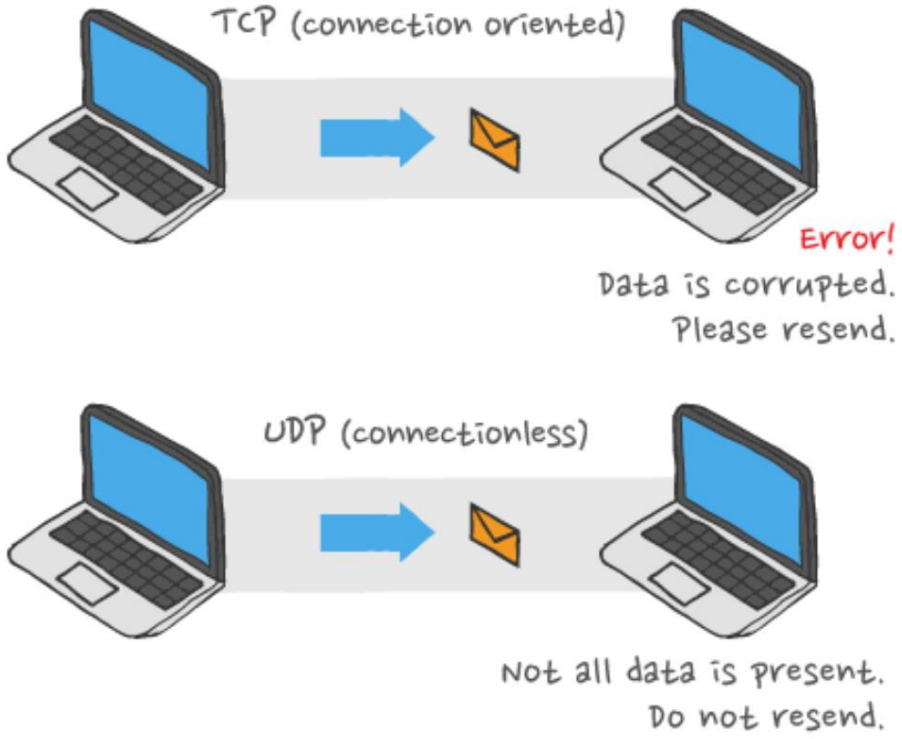
המטרה של HTTP3 היא לספק כמה חיבורים שונים, מאובטחים, ומהירים עבור כל המכשירים על ידי טיפול בבעייתיות של החיבור היחיד הנמצא ב HTTP2.

- מגביר ביצועים של HTTP.
- מבוסס על פרוטוקול UDP.
- לא חוסם את העברת זרימת המידע במידה ואבדו פאקטות בדרך.



ניתן לראות בתצלום לעיל את ההבדל בין תהליך העברת מידע של QUIC לתהליך העברת מידע של HTTP בגרסאות הקודמות. הזכרנו שההבדל הינו מתבצע בעיקר בביסוס הפרוטוקולים. כאשר מתבצעת העברת מידע של HTTP בגרסאות קודמות, פרוטוקול TCP, יש צורך לבצע חיבור התחלתי עם השרת. אך כאשר מדובר

ב QUIC, פרוטוקול UDP, אין צורך ב"לחיצת ידיים זו" ותהליך העברת המידע מתרחש בצורה אחידה בין אם המידע אבד או לא.



בתצלום לעיל, ניתן לראות מה קורה בעת קבלת מידע על ידי פרוטוקול TCP ומה קורה בעת קבלת מידע על ידי פרוטוקול UDP.

<u>תשובה לשאלה 4:</u>

מספרי הפורט הינם חשובים בהחלט מכיוון שהם משמשים כזיהוי חד חד ערכי של חיבור ברשת.
כל מכשירי ה network משתמשים במספרי פורט ורובם יכולים אפילו לשנות את המספרים כשנדרש.
הפורטים נוצרו כדי לאפשר לתוכנות שונות להשתמש באותה כתובת IP.

חשוב לציין שישנם פורטים שמורים עבור פרוטוקולים שונים לדוגמא: HTTP- PORT 80, DNS- PORT 53 ועוד.

עם זאת, לקוח מסוים יכול להתחבר לשרת המספק מס' תכונות שונות (עם מספר חיבורים שונה) ע"י מספרי פורטים שונים.

לדוגמא: נתב שמספק גם DHCP וגם DNS. לקוח יכול להתחבר לנתב עם שני בקשות שונות.

<u>תשובה לשאלה 5:</u>

ה Subnet מאפשרת לחלק את הרשת הגדולה לרשתות קטנות. לדוגמא: בחברה גדולה עם מס' רב של מחלקות ניתן לממש SUBNET עבור כל מחלקה וכך ניתן להגביר את היעילות.

לכל המחשבים באותו ה Subnet יש את אותו ה Prefix. דוגמא ליעילות: הפצת מידע בצורה אחידה וגורפת לכלל המחשבים עם אותו ה Prefix. דוגמא נוספת: ניתוב מהיר למחשב ספציפי.

כמובן שכל Subnet יכול לכלול בתוכו שרתים, לקוחות, ראוטרים, סוויטצ'ים ועוד..

<u>תשובה לשאלה 6:</u>

כתובת MAC היא כתובת המייחדת כל מחשב ואינה משתנה לעולם. כתובת IP היא כתובת מייחדת אך יכולה להשתנות עם כל חיבור לרשת שונה.

בשביל לזהות מכשיר ספציפי המבקש בקשות מסוימות אנו צריכים לדעת את שני הנתונים הללו (כתובת IP, כתובת MAC). כתובת IP – עבור לדעת את זיהוי המחשב ברשת. כתובת MAC – עבור לשייך את הזיהוי ברשת למחשב ספציפי.

תשובה לשאלה 7:

Switch	Router
פועל בשכבת ה LINK	פועל בשכבת ה Network
לא מאפשר NAT	מאפשר NAT
מאחסן טבלת כתובות MAC	מאחסן טבלת כתובות IP
מכשיר נטוורקינג בעל יכולת לפעול עם 24/48 יציאות	מכשיר נטוורקינג בעל יכולת לפעול עם 2/4/8 יציאות
החלטות ניתוב מסובכות ואיטיות יותר	עוזר למשתמש לבצע החלטות ניתוב פשוטות ומהירות
בחיבור LAN ה SWITCH מהיר יותר מה ROUTER	ישנן סביבות רשת שונות שבהן ה ROUTER מהיר יותר מה SWITCH
עובד עם חיבור קווי בלבד	עובד עם חיבור קווי או חיבור אלחוטי

Router: מחבר מספר רשתות ועוקב אחר התעבורה ביניהן. יש ל Router חיבור יחיד של אינטרנט וחיבור יחיד של רשת לוקאלית. בנוסף, יש מלא Router ים שמאפשרים להתחבר למספר מכשירים חוטיים. הרבה Router ים גם מכילים גלי רדיו אל חוטיים שמאפשרים להתחבר באמצעות מכשירי Wi-Fi.

Switch: משתמש בכתובות MAC להעביר מידע ליעד הנכון. מערכת ההפעלה בשלב ה LINK משתמשת בהחלפת חבילות בשביל לקבל, לעבד ולהעביר מידע.

NAT: טכניקת ניתוב ברשת מחשבים, מטרתה העיקרית היא לחסוך בכתובות IP על ידי צמצום כתובות האינטרנט לכתובת יחידה שתוצג כלפי שאר הרשתות החיצוניות. שיטה זו מתבצעת על פי השלבים הבאים:

שליחת מידע מהרשת הפנימית לחיצונית:

- הנתב ימלא בכתובת ה SRC את כתובתו.
- הנתב יחליף את ה SRC PORT למספר Port אחר שישמש כאינדקס בטבלת ה NAT.

קבלת מידע מהרשת החיצונית לפנימית:

- הנתב תחילה יחפש בטבלת ה NAT את ה DEST PORT של החבילה שהתקבלה.
- הנתב יחליף את ה DEST PORT ואת ה IP PORT בהתאם לאינדקס שצוין בטבלת ה NAT.
- הנתב ינתב את המידע למיקום הנכון.

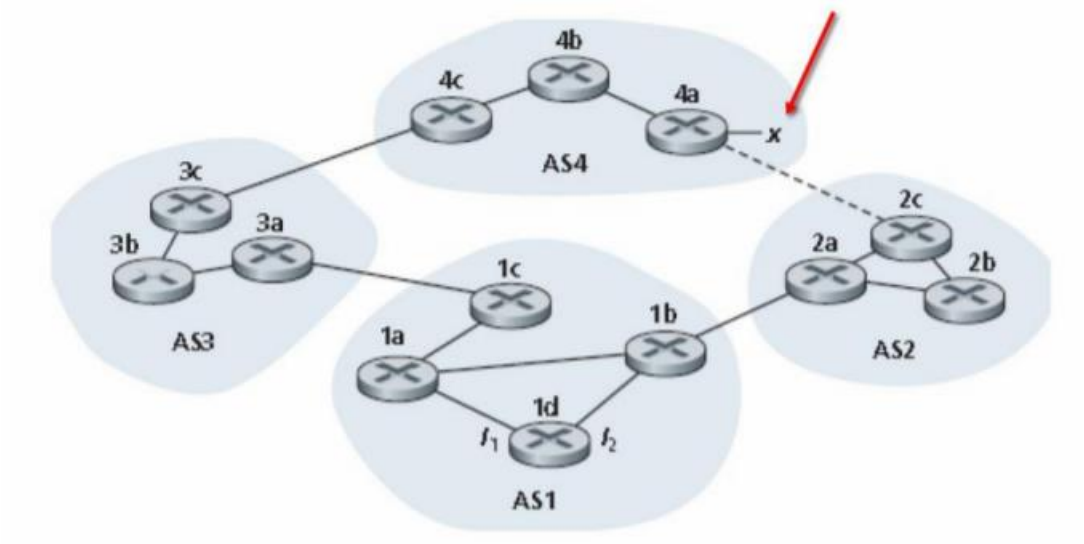
תשובה לשאלה 8:

ישנן 3 שיטות להתגבר על הבעייתיות של IPV4:

1. IPV6 – הוא פתרון לטווח הארוך שמכיל מספר כתובות מורחב יותר (ספרות הכתובות רבות יותר כך שניתנות יותר אופציות לבחירה).
הבעייתיות בפתרון זה היא: שמכיוון ש IPV4 כבר מוטמע בכל העולם ישנו צורך לעשות שינויים אפילו פיזיים בשביל להטמיע את IPV6. דבר שיכול לקחת זמן ומשאבים רבים ליישום. לא כל ספקי האינטרנט מציעים IPV6.
2. Network Address Translation (NAT) – כפי המוזכר בשאלה הקודמת זוהי שיטה לצמצום כתובות IPV4 ציבוריות ברשת מקומית לכדי כתובת אחת.

3. Carrier-grade Network Address Translation – שיטה זו מתרגמת כתובות אינטרנט על יד ספק האינטרנט. שיטה זו היא קצרת טווח ובדרך כלל מתרחשת כאשר יש לספק אינטרנט יותר לקוחות מאשר כתובות IP פנויות. בשיטה זו מספר לקוחות חולקים את אותה כתובת IP. שיטה זו לא מאפשרת חיבורים נכנסים כלל מכיוון שאין אופציה לדעת לאיזה לקוח משויך החיבור הנכנס. המחשבים המחוברים בשיטה זו אינם יכולים לשמש כספקים כלפי חוץ כך שאתרים שצריכים הרשאות מסוימות או מידע רלוונטי ממחשב ספציפי המחובר בשיטה זו אינם יכולים לגשת לכתובת הרצויה.

<u>תשובה לשאלה 9:</u>



לפי הנתון ידוע לנו ש:

- **AS2** ו **AS3** מריצים **OSPF** – על פי ספירת רוחב פס.
- **AS1, AS4** מריצים **RIP** – על פי ספירת צעדים.
- בין כל **ASs** ר. **BGP**.
- אין חיבור פיזי בין **AS2, AS4**.

הנתב **3c** לומד על תת רשת **x** על ידי פרוטוקול **BGP** מכיוון שהוא נתב ה**gateway** לכיוון **x** ולכן הוא לומד על תת הרשת דרך פרוטוקול התקשורת החיצוני.

הנתב **3a** לומד על תת רשת **x** על ידי פרוטוקול **OSPF**, כפי שהזכרנו קודם הנתב **3c** הוא הנתב **gateway** לכיוון **x** ולכן שאר הנתבים ב**AS3** כולל **3a** לומדים על **x** דרכו בעזרת פרוטוקול התקשורת הפנימי.

הנתב **1c** לומד על תת רשת **x** על ידי פרוטוקול **BGP** מכיוון שהוא נתב ה**gateway** לכיוון **x** ולכן הוא לומד על תת הרשת דרך פרוטוקול התקשורת החיצוני.

הנתב **2c** לומד על תת רשת **x** על ידי פרוטוקול **OSPF**, כי **2a** הוא הנתב **gateway** לכיוון **x** ולכן שאר הנתבים ב**AS2** כולל **2c** לומדים על **x** דרכו בעזרת פרוטוקול התקשורת הפנימי. (בנוסף נשים לב לנתון שאומר ש אין חיבור פיזי בין **AS2, AS4**)