מטלת גמר תקשורת

מגישים:

יהונתן עמוסי - 209542349

ליאב לוי - 206603193

חלק א'

בחלק זה התבקשנו לכתוב מערכת צ'ט מרובת משתתפים (לפחות 5 משתמשים + שרת) כך שכל אדם בצ'ט יכול לשלוח הודעה לכל המשתתפים וגם יוכל לשלוח הודעה פרטית למשתמש ספציפי.

לצורך כך הקמנו 6 מכונות ווירטואליות- 1 מהן מהווה השרת ושאר המכונות מהוות 5 משתתפים שונים כתובת השרת ידועה מראש לכן בקובץ הקוד של השרת וה- GUI הכנסנו את הכתובת 10.0.2.4 כדלהלן:

```
import ...


class MyClientGUI:
    def login_f(self):
        self.user_name = self.e.get()
        self.module = user.module.user_module(self.address, self.user_name)
        self.module.connect(('10.0.2.4', 50000), self.user_name)
        self.login.destroy()

    def __init__(self, address):
```

```
import socket
from threading import local


from Server.server import Server

my_server = Server('10.0.2.4')
```

בנוסף כל מכונה קיבלה פורט שדרכו תתבצע התקשורת עם השרת בסדר עולה:

שרת IP- 10.0.2.4 פורט- 50000

מכונה א' IP- 10.0.2.5 פורט- 50001

מכונה ב' IP- 10.0.2.6 פורט- 50002

מכונה ג' IP- 10.0.2.7 פורט- 50003

מכונה ד' IP- 10.0.2.8 פורט- 50004

מכונה ה' IP- 10.0.2.9 פורט- 50005

כאשר התחלנו את הפרויקט הכנו תוכנית עבודה **שתתחיל** קודם בבניית השרת שיענה על הדרישות שנדרשנו שיענה עליהם וביניהם העברת הודעות בין שני לקוחות, העברת הודעה מלקוח אחד לכל שאר הלקוחות, שליחת רשימת האנשים שמחוברים לשרת, רשימת הקבצים הזמינים להורדה וכן הורדת הקבצים שנמצאים ברשותו (שני האחרונים יוצגו בחלק ב'), **תמשיך** בבניית המשתמש בצ'ט כולל הפעולות שהוא יכול לבצע המצורפות בטופס המטלה למשל שליחת הודעה למשתמש ספציפי, שליחת הודעה לכלל המשתתפים ובקשות מידע מהשרת כגון רשימת משתמשים, רשימת קבצים זמינים להורדה והורדת קובץ מהשרת (השניים האחרונים יוצגו בחלק ב') **ותסתיים** ביצירת ממשק גרפי להמחשה ויזואלית (השתמשנו בתבנית עיצוב MVC).

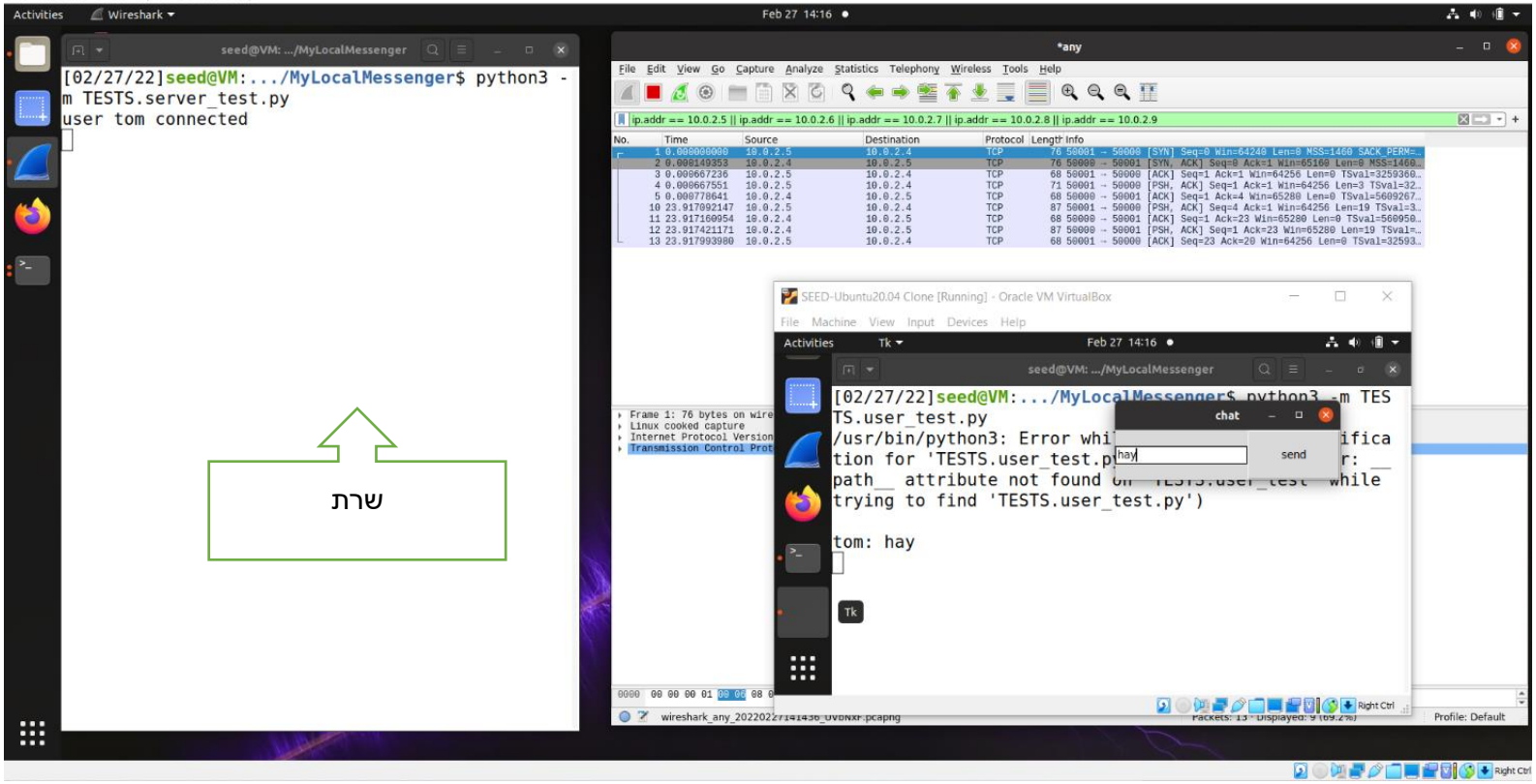
ראשית נציג את היכולת של לקוח לשלוח הודעה לכלל המשתתפים:

נבצע קודם את הפעולות הבאות:

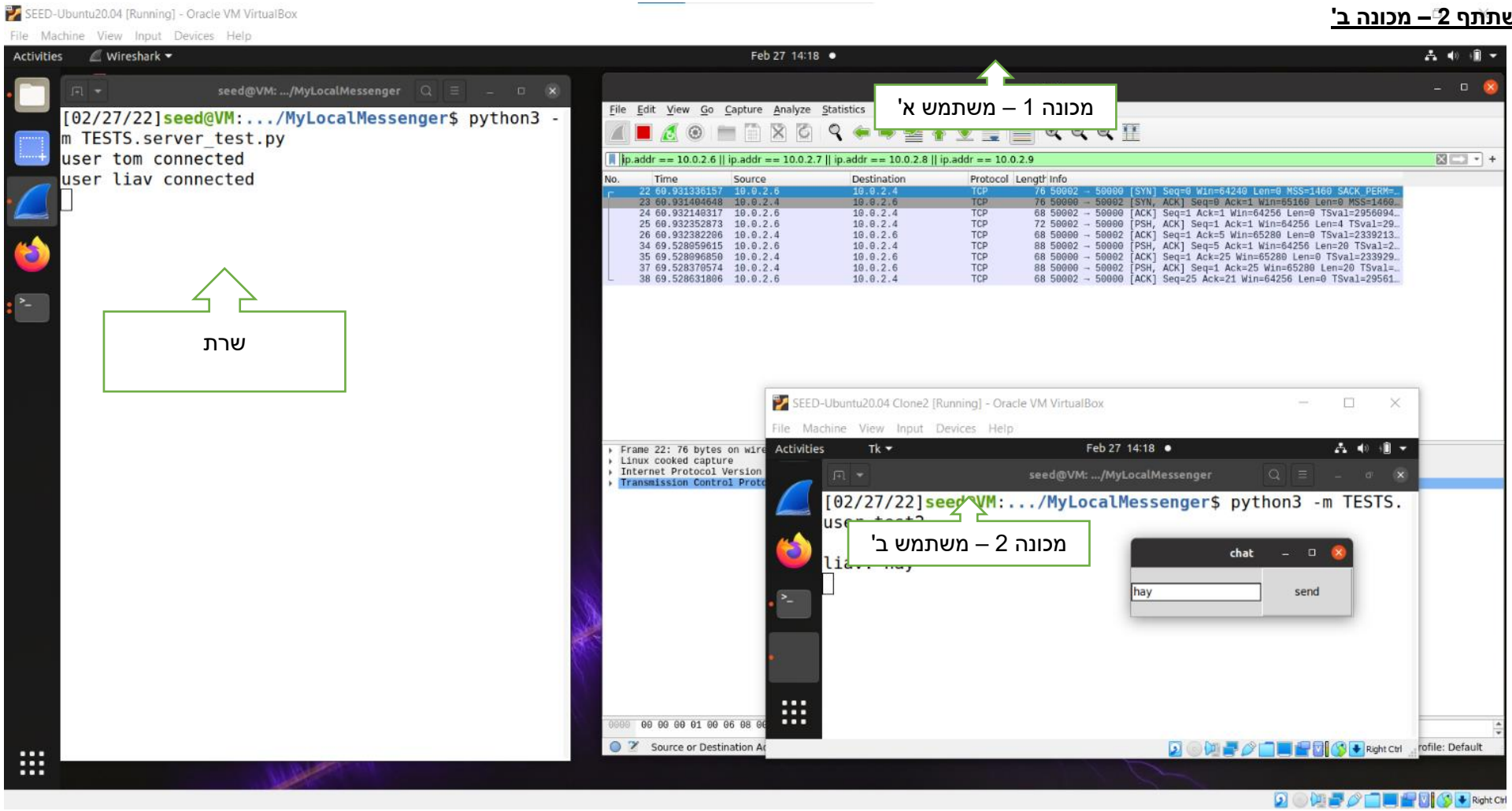
א. הפעלת השרת – יש להיכנס לתיקייה MyLocalMessenger לפתוח שם את הטרמינל ולהריץ את הפקודה הבאה- python3
ב. חיבור כל 5 המשתתפים כך שכאשר משתתף מתחבר לצ'ט הוא שולח הודעה hay , בכל

הנחיה בתמונות הבאות השרת מופיע מאחורה מצד שמאל ה wireshark מצד ימין הטרמינל שהפעיל את השרת ועל גביו מימין באמצע תופיע המכונה שהריצה משתמש כלשהו:

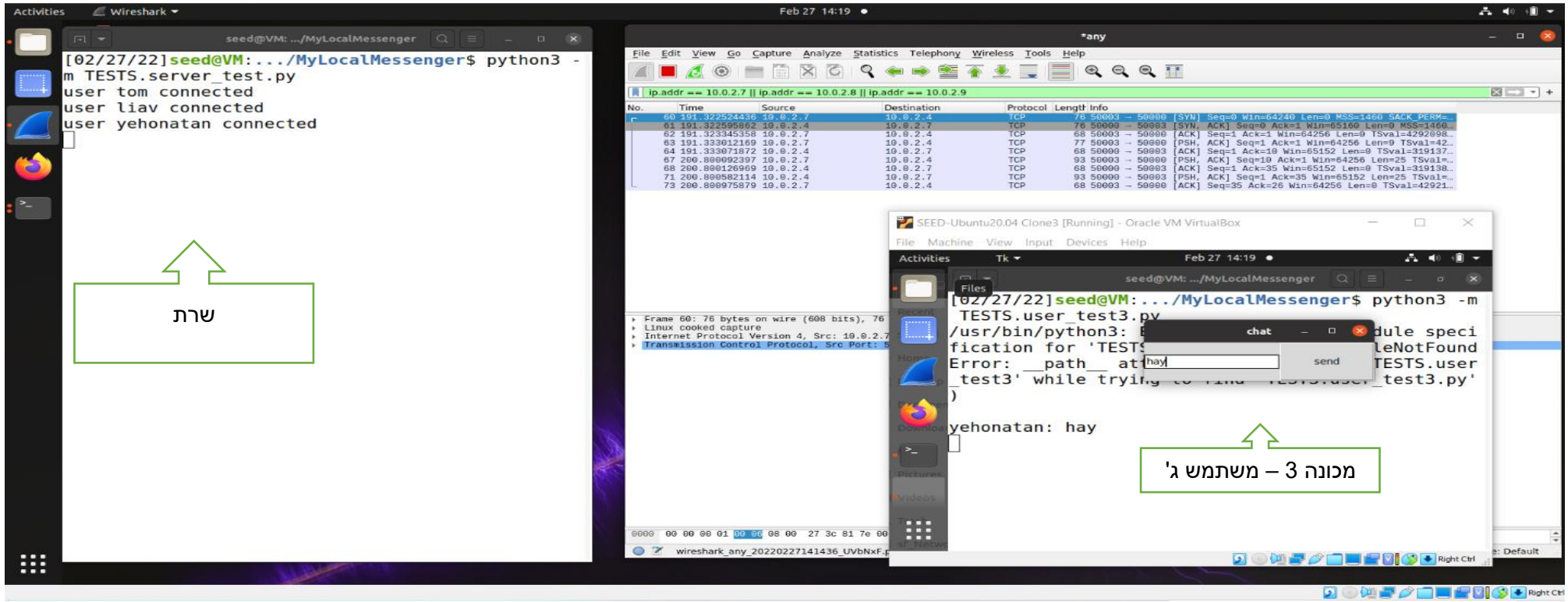
<u>משתתף 1 – מכונה א'</u>



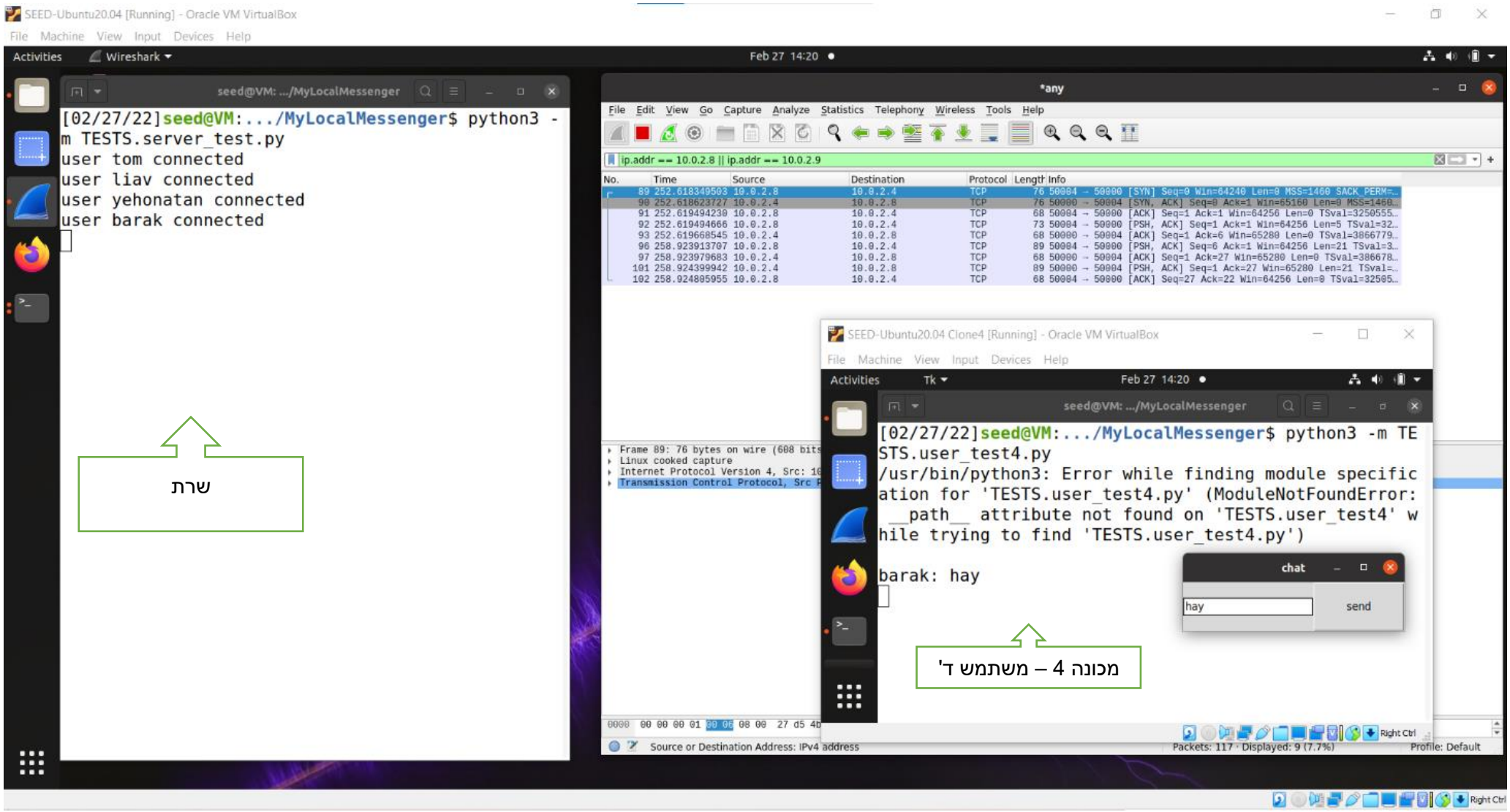
משתתף 2 – מכונה ב'



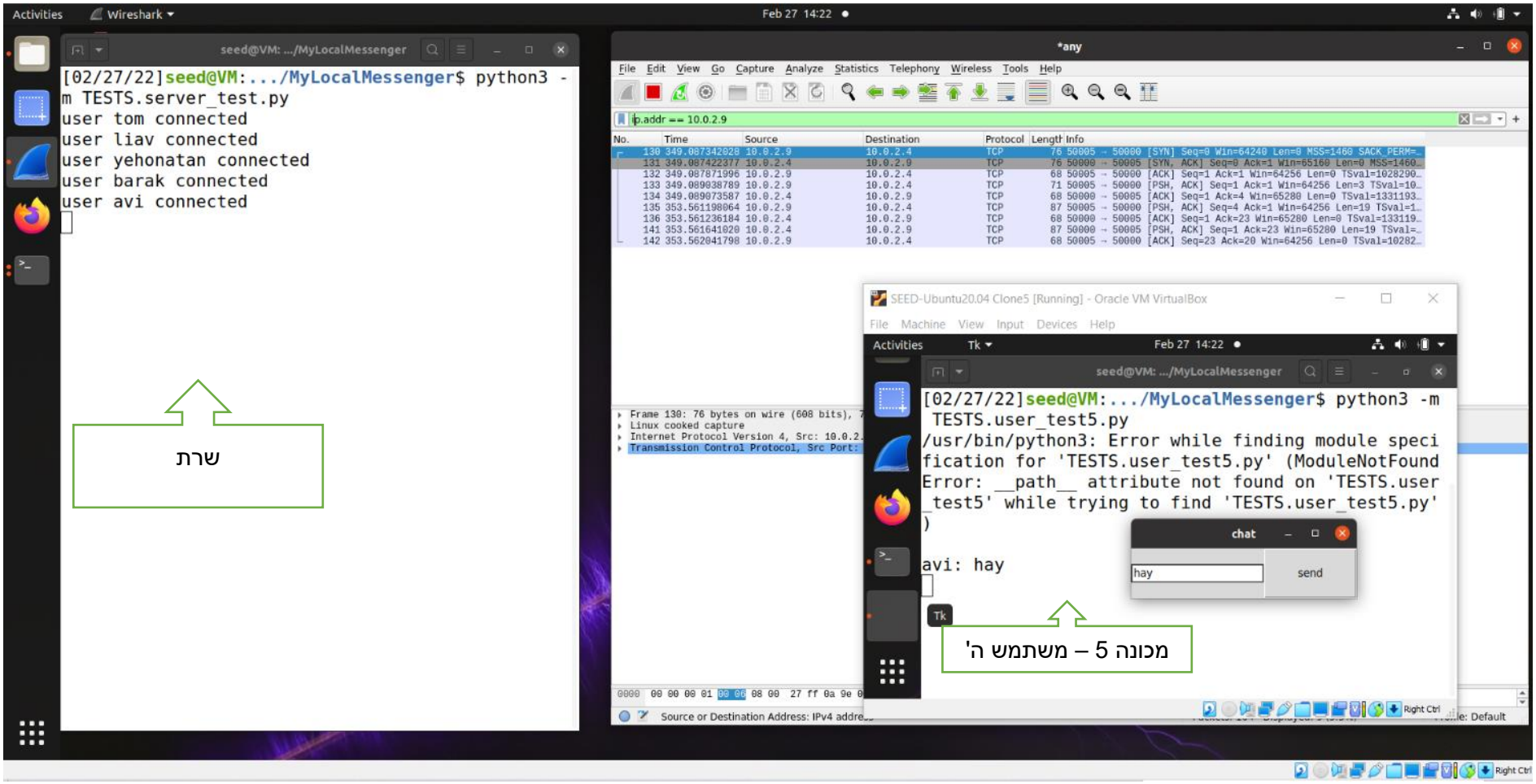
משתתף 3 – מכונה ג'



משתתף 4 – מכונה ד'

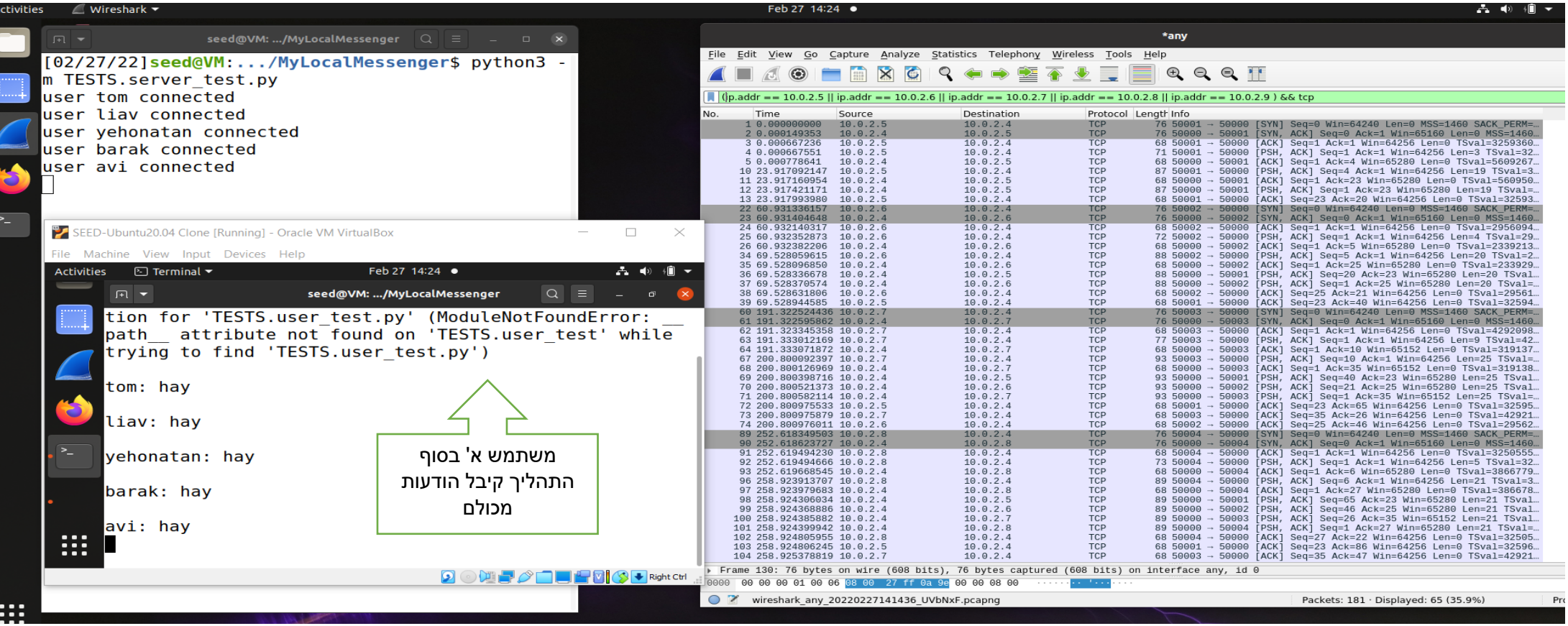


משתתף 5 – מכונה ה'



נשים לב שכשאר משתתף מתחבר לצ'ט מופיעה הודעה בשרת שהוא מחובר.

ניתן לראות שבסוף התהליך כל לקוח קיבל הודעה מכל מי שהתחבר אחריו כלומר התבצעה שליחה לכלל המשתתפים בצ'ט למשל הלקוח הראשון קיבל הודעה מכולם:

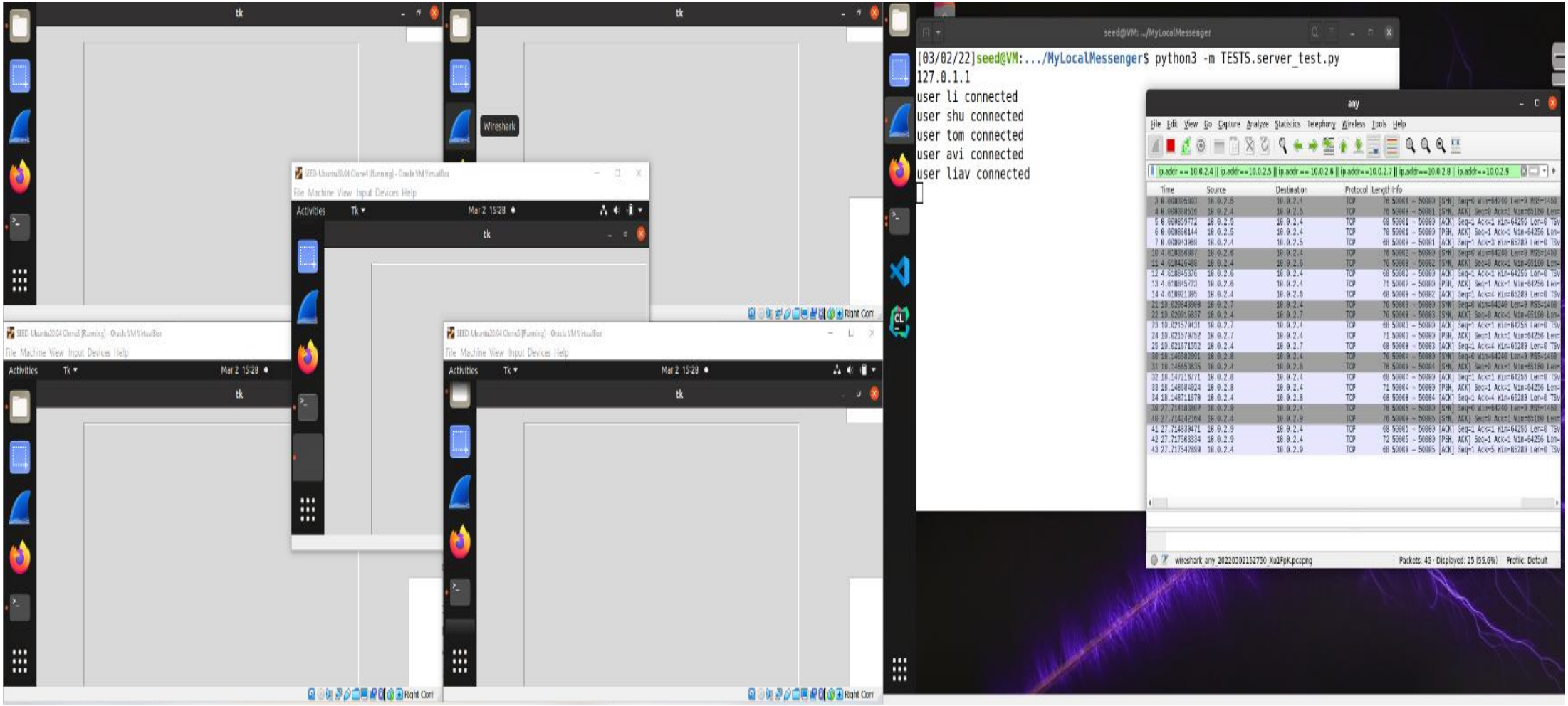


נשים לב שב wireshark מופיעה כל תמסורת הפאקטות בצ'ט (כל משתתף שהתחבר לצ'ט ושלח הודעה יופיע פה) ניתן לראות בפירוט בקובץ pcap המצורף באמצעות שימוש בפילטר (יש להכניס את אחד ממספרי ה IP שהוזכרו למעלה).בנוסף ניתן לראות שכל פעם שהלקוח האחרון שהתחבר שולח הודעה אזי ב wireshark נראה שאנו שולחים הודעה לכל שאר המכונות כלומר הודעה לכלל המשתתפים שמחוברים לצ'ט.

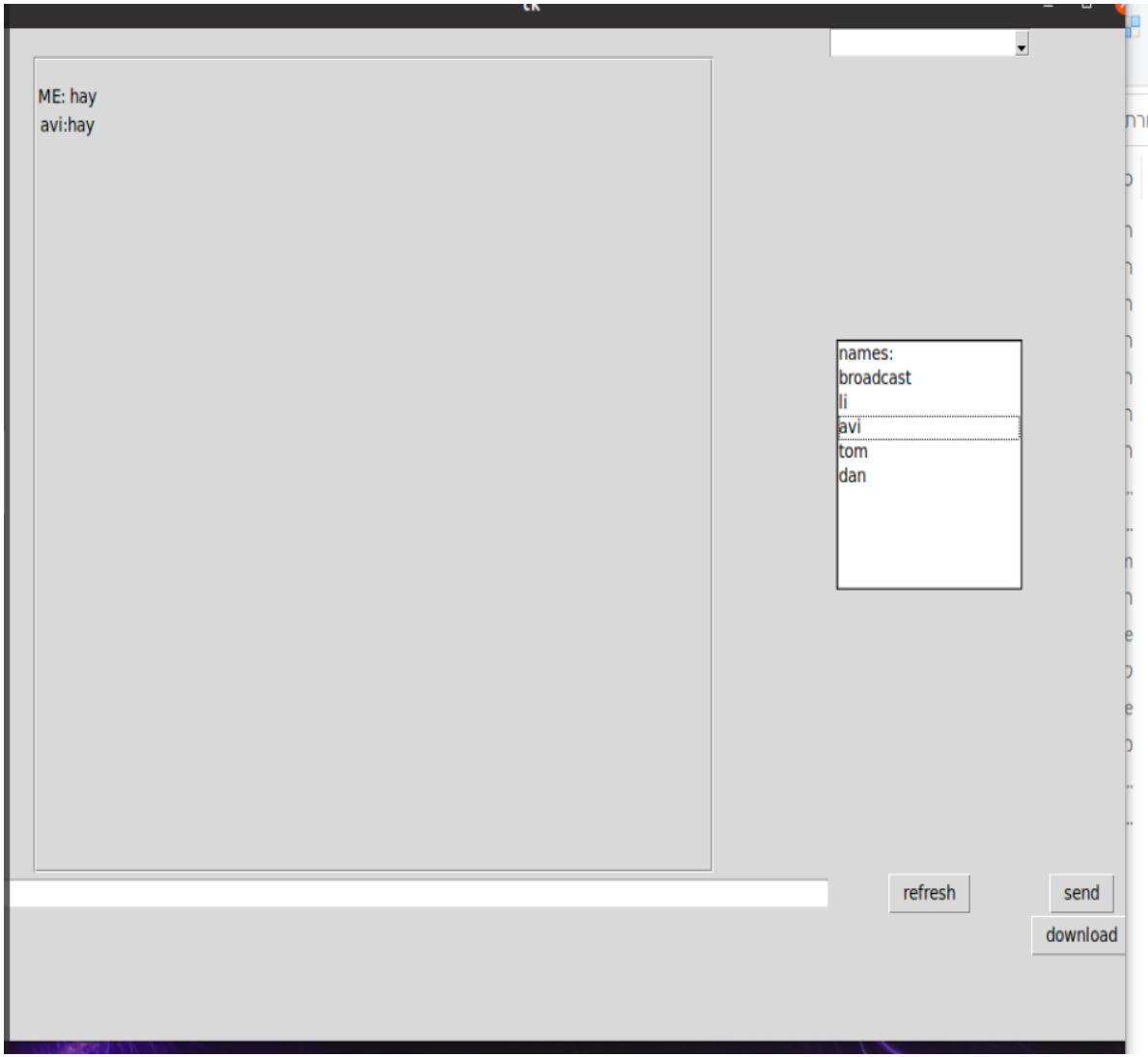
כעת נראה את היכולת של משתמש מסוים לשלוח הודעה למשתמש ספציפי:
לאחר הפעלת השרת והמשתמשים נכתוב הודעה כלשהיא ומצג ימין נבחר משתמש ספציפי שיקבל את ההודעה כפי שנראה רק המשתמש שנבחר יקבל את ההודעה
בנוסף נראה ב wireshark כי הפקאטה שהמשתמש שלח באמת נשלחת רק לנמען ואילו בשליחה לכל המשתמשים היא נשלחת לכולם.

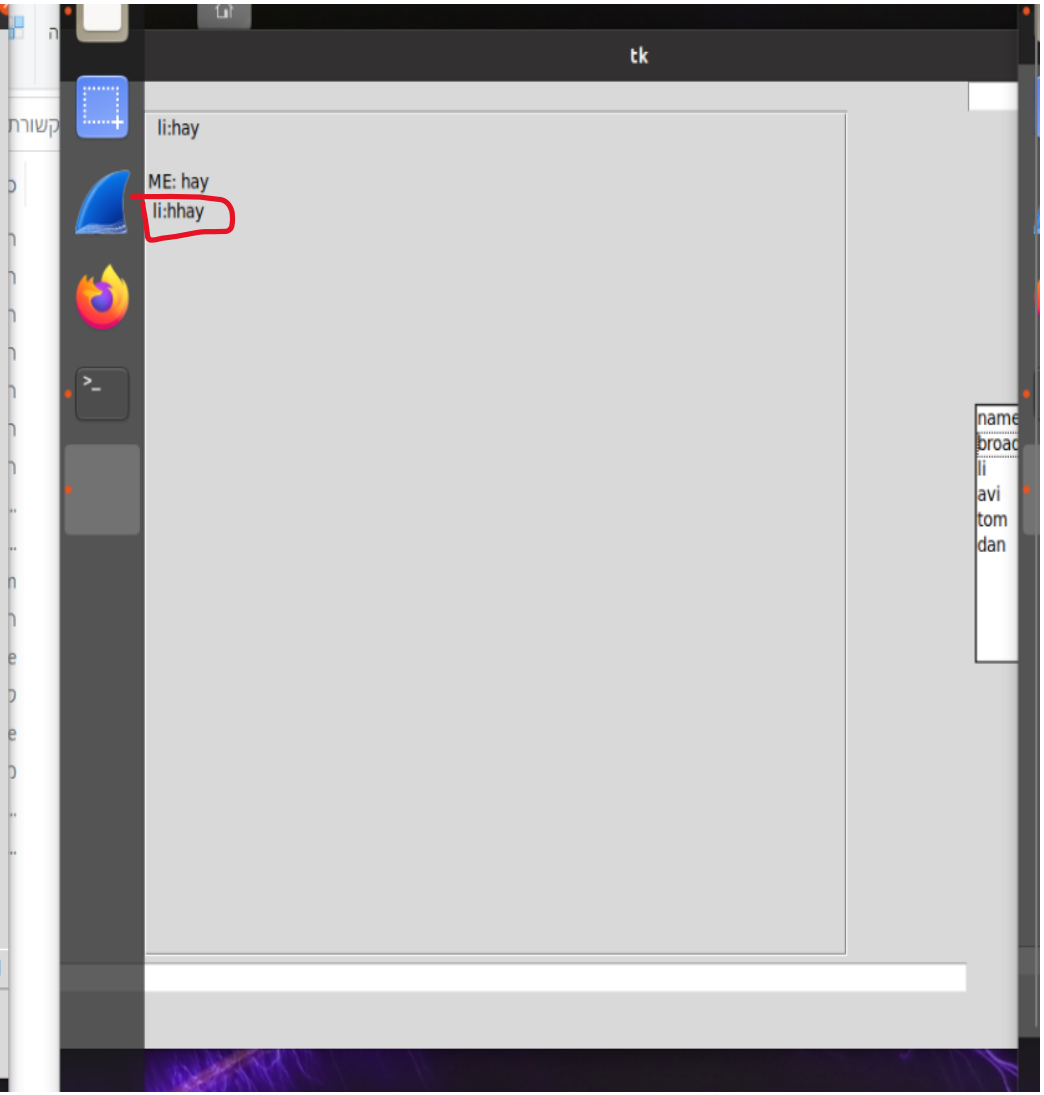
- את כל התמונות ניתן בצורה יותר טובה בתיקיית התמונות המצורפת לפרוייקט

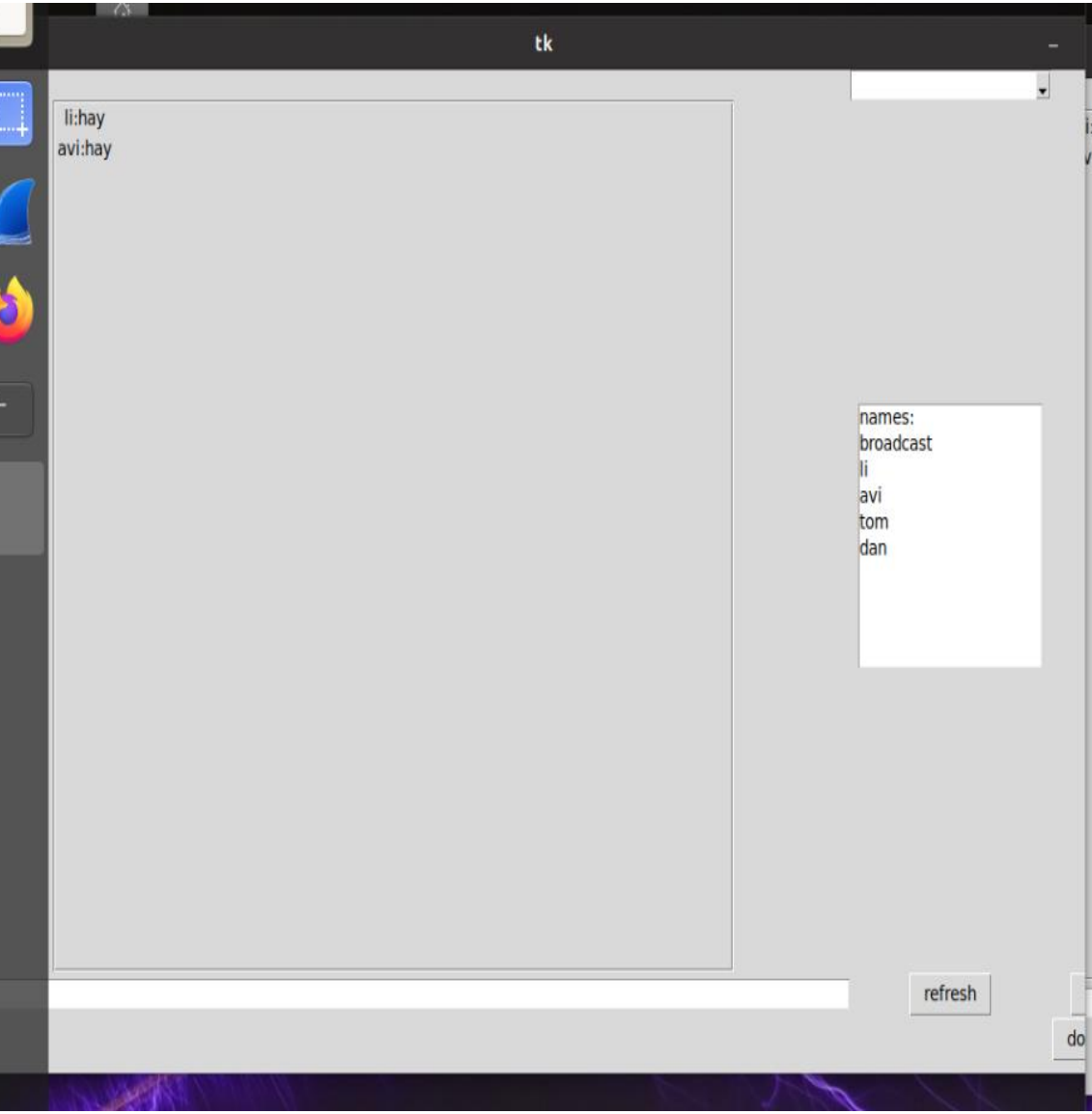
חיבור כלל המשתמשים לשרת:

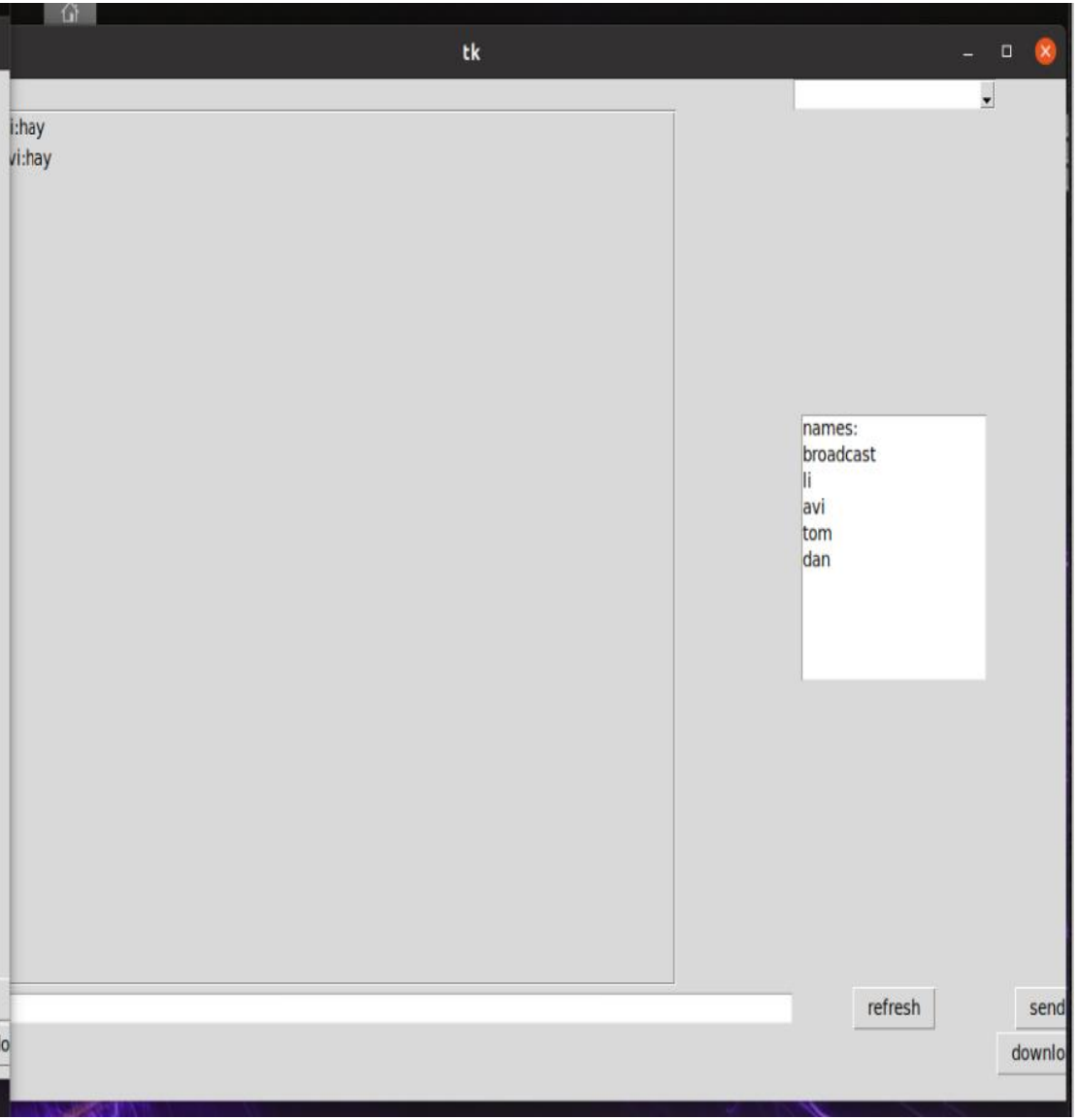


שליחת הודעה למשתמש ספציפי:
ניתן לראות בתמונה הנוכחית שהמשתמש הימני קיבל את ההודעה אולם בתמונה בעמוד הבא נראה ששאר המשתמשים לא קיבלו אותה






tk
li:hay
avi:hay
names:
broadcast
li
avi
tom
dan
refresh

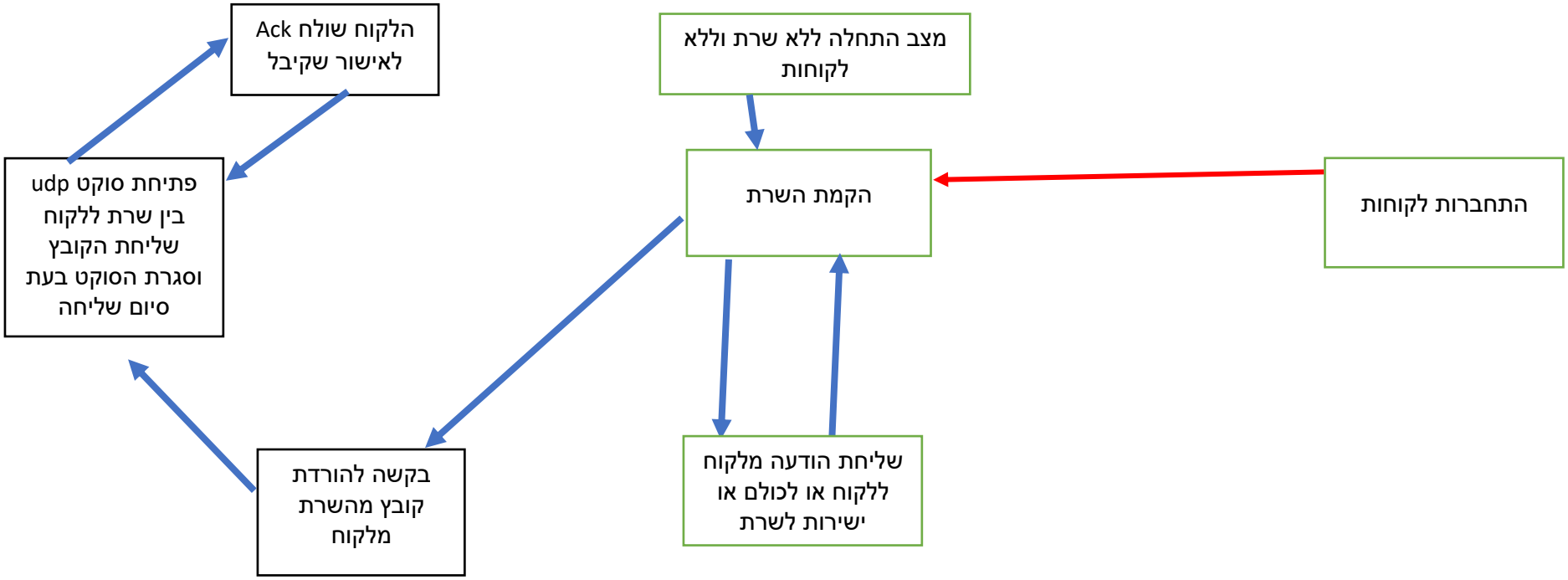

tk
i:hay
vi:hay
names:
broadcast
li
avi
tom
dan
refresh
send
downlo

חלק ב'

לאחר שהראינו שהצ'ט עובד ועונה על דרישות של שליחת המידע נראה עתה שהוא גם עונה על הדרישה של הורדת קבצים מהשרת.

לצורך כך הלקוח שולח הודעה לשרת שבה הוא מבקש קודם את רשימת הקבצים שזמינים להורדה (תהליך העברת הפאקטות לפי חלק א') לאחר שהלקוח קיבל את רשימת הקבצים ביכולתו לבחור לבצע הורדה של קובץ כלשהו מהרשימה ותוך כדי להמשיך ולדבר בצ'ט. לצורך כך יצרנו שני סוקטים חדשים אחד לשרת והשני ללקוח מסוג UDP . בנוסף על מנת שהתקשורת בין השרת והלקוח תהיה אמינה מימשנו את בקוד שלכל שליחת פאקטה ("מידע") אנו שולחים Ack מהלקוח כדי להבין שהוא הצליח לקבל את הפאקטה. בנוסף מימשנו זאת באמצעות selective-repeat כך שכשאר אובדת פאקטה או שמכל סיבה שהיא הלקוח לא קיבל פאקטה כלשהיא השרת ישלח את הפאקטה הספציפית שהלקוח לא קיבל.

דיאגרמת מצבים:



כיצד המערכת מתגברת על איבוד חבילות:

מימשנו כאן במערכת selective-repeat אשר עובד בצורה הבאה: כאשר פאקטה הלכה לאיבוד (או שקרתה בה שגיאה) והזמן שהוקצב לשליחה שלה ולקבלת Ack המאשר את קבלתה (כלומר התרחש timeout) אזי מבצעת שליחה חוזרת של אותה פאקטה ספציפית שלא קיבלנו עליה Ack שהתקבלה (וזאת בניגוד ל GoBackN ששם מתבצעת שליחה מלאה של כל החלון) וכך אנו נמנעים משליחה מיותר של פאקטות שכן קיבלנו עליהן Ack .

השרת מגיב בצורה הבאה:

- אם מתקבלת בקשה של הלקוח לקובץ מסוים השרת לוקח את הקובץ מפרק אותו לפאקטות ושולח אותך לפי הסדר ללקוח לפי גודל החלון שלו

- על כל קבלת Ack מהלקוח מתבצע סימון שאותה פאקטה התקבלה במידה וזו הפאקטה עם המספר הסידורי הכי נמוך (כלומר הראשונה בחלון שליחה) החלון שליחה יזוז ימינה עד הפאקטה הראשונה שלא קיבלנו עליה Ack .

- לכל פאקטה מוגדר timeout אישי משלה ולכן במידה ופאקטה אבדה או לא הגיעה בזמן ללקוח ולא קיבלנו מהלקוח Ack על אותה חבילה השרת ישלח שוב את אותה פאקטה בלבד (וזאת בניגוד ל GoBackN ששם מתבצעת שליחה מלאה של כל החלון) וכך אנו נמנעים משליחה מיותר של פאקטות שכן קיבלנו עליהן Ack .

הלקוח מגיב בצורה הבאה:

- לאחר שליחת בקשה להורדת קובץ מהשרת ואישורה, הלקוח מתחיל לקבל פאקטות של הקובץ המבוקש. על כל פאקטה שמתקבלת הלקוח שולח אישור שאותה חבילה התקבלה
- במידה ומדובר בפאקטה חדשה היא נשמרת
- אם הפאקטה שהלקוח קיבל היא הראשונה בחלון (כלומר הראשונה שאמורה לקבל אישור קבלה אם אין כלל בעיות) אזי החלון קבלה של הלקוח יזוז ימינה עד הפאקטה הראשונה שלא קיבלנו
- במידה והשרת מקבל Ack על חבילה שהוא כבר שלח וקיבל אישור עליה עוד קודם (כלומר השרת מאיזו שהיא סיבה שלח את החבילה פעמיים וקיבל אישור על החבילה הראשונה מבניהם) אזי הוא לא עושה כלום

כיצד המערכת מתגברת על בעיות latency :

בעיות latency הינן בעיות של עומס ברשת כך שפאקטות לא אובדות אך מגיעות באיחור ליעד בסיטואציה זו המערכת שלנו מגיבה לשתי סיטואציות:

- הפאקטה הגיעה לפני שהתרחש timeout במצב כזה המערכת עובדת כרגיל ולא מתרחש שום פעולה נוספת מצד השרת שלנו.
- הפאקטה הגיעה לאחר שהתרחש timeout במצב כזה המערכת שלנו מתייחסת לזה כאילו אבדה הפאקטה והיא נשלחת שוב מחדש על ידי השרת הלקוח לאחר שקיבל את הראשונה שולח ack לאישור ומתעלם מהפאקטה השנייה שנשלחה אליו ופותר אותה כמו שתיארנו לעייל.

<u>חלק ג' תשובות:</u>

1) בהינתן מחשב חדש המתחבר לרשת אנא תארו את כל ההודעות שעוברות החל מהחיבור הראשוני ל switch ועד שההודעה מתקבלת בצד השני של הצ'ט.
אנא פרטו לפי הפורמט הבא:
a. סוג הודעה, פירוט הודעה והשדות הבאים- כתובת IP מקור/יעד, כתובת פורט מקור/יעד, כתובת MAC מקור/יעד ,פרוטוקול שכבת התעבורה.

תשובה- בשביל לחבר את המחשב לרשת צריך קודם להשיג את הדברים הבאים- כתובת IP למחשב, כתובת הנתב הראשון וכתובת שרת ה DNS של הרשת לכן נשתמש בבקשת DHCP.
בקשת ה DHCP עטופה על ידי UDP שעטוף בתורו על ידי IP שעטוף גם הוא ב 802.3 (Ethernet) המחשב שולח הודעה מסוג broadcast (כתובת היעד 255.255.255.255) לכלל המשתמשים ברשת המקומית (LAN) על מנת שהיא תגיע גם לשרת ה DHCP שנמצא ברשת. כתובת המקור בבקשה זו יהיה 0.0.0.0 וכתובת היעד תהיינה 255.255.255.255 .
לאחר ששרת ה DHCP מקבל את ההודעה הוא שולח הודעת תגובת DHCP ACK שמכילה ועוטפת את כתובת IP למחשב, כתובת IP של הראוטר (נתב) הכי קרוב למחשב (נקרא first-hope router) ואת כתובת ה IP של שרת ה DNS של הרשת, הפריימים של ההודעה נשלחים דרך הרשת המקומית (ובתוך כך יש switch learning).
הלקוח כעת מקבל תגובת DHCP ACK וכעת יש לו את כל מה שצריך בשביל להיות "חלק" מהרשת אולם הוא עדיין לא יכול לשלוח הודעות למשתמשים אחרים שנמצאים ברשת כי הוא לא יודע מה הכתובות שלהם.
לכן לפני שליחת הודעה למשתמש המחשב ישלח הודעה לשרת ה DNS עם פרוטוקול DNS על מנת לקבל את הכתובת IP של המשתמש השני.
המחשב יוצר שאילתת DNS, שעטופה ב UDP, שעטופה ב IP, שעטופה ב Ethernet. אולם התחנה הראשונה ברשת היא הראטור שהזכרנו לעיל וכדי לשלוח אליו הודעה אנו צריכים את הכתובת MAC שלו (יותר נכון להגיד הכתובת של ה router interface) לצורך כך נשתמש בשאילתת ARP , שאילתת ה ARP מתקבלת על ידי הנתב, אשר מגיב עם תשובת ARP ונותן את כתובת ה MAC שלו.
כעת המחשב יודע את כתובת ה MAC של הנתב הראשון אליו יצטרך לשלוח את שאילתת ה DNS.
IP datagram מכיל שאילתת DNS המועברת דרך ה LAN switch מהלקוח אל הראוטר שהזכרנו לעייל. השאילתה מועברת מהרשת של הלקוח אל הרשת של השרת, (בעזרת טבלאות הנוצרות על ידי פרוטוקולי RIP, OSPF, BGP ועוד). השרת DNS מקבל את השאילתה ומחזיר תשובה אל הלקוח את כתובת ה IP של הנמען המבוקש וכעת למחשב יש את כתובת ה IP של הנמען.
כדי לשלוח את ההודעה לנמען, נצטרך לפתוח TCP socket אל השרת. נשלחת בקשת SYN אל השרת (שלב ראשון בה3-way handshake), השרת עונה עם SYNACK (שלב שני) וכעת נוצר חיבור TCP. כעת נשלחת דרך ה TCP socket. ה IP DATAGRAM המכיל את ההודעה מנותב אל השרת. השרת מגיב בחזרה שהוא קיבל את ההודעה ומעביר אותה לנמען המבוקש לאחר שהנמען מקבל את ההודעה הוא שולח הודעה לשרת שההודעה התקבלה.

2) **הסבירו מה זה CRC ?**

תשובה- ה CRC הוא סוג מסוים של checksum אשר משומש כדי לזהות שגיאות או שינויים מקריים בנתונים גולמיים אשר מועברים בין מקור ליעד (למשל שרת ולקוח). הוא מורכב ממספר קבוע של סיביות בדיקה שמצורפות לסוף ההודעה, ולאחר השידור הלקוח או מקבל ההודעה בודק לפיו האם התרחשו שגיאות בעת העברת המידע.
ה CRC פועל בצורה הבאה:
בהינתן פולינום יוצר מדרגה r ובהינתן הודעה M שברצוננו לקודד נבצע את הפעולות הבאות:

א. נוסיף r אפסים מימין להודעה.
ב. נחלק בפולינום
ג. נחסר את השארית תוך שימוש ב xor
נצרף את התוצאה שקיבלנו מימין להודעה המקורית ונשלח, ה checksum של הצד המקבל יבצע את שלבים א ו- ב ויוודא ש r-הביטים האחרונים שנשלחו זהים לתוצאה שהתקבלה.
אם הם אינם זהים, אירעה שגיאה ויש להעביר את הנתונים מחדש.

3) **מה ההבדל בין http 1.0 ,http 1.1, http 2.0, QUIC ?**

<u>תשובה</u>- פרוטוקול http 1.0 הוא Non-persistent HTTP כלומר לאחר כל בקשה ותגובה החיבור בין השרת ללקוח ניסגר. המשמעות היא שאם השרת צריך לשלוח כמה אובייקטים ללקוח צריך לפתוח חיבור לכל אובייקט כזה ודבר זה לוקח משאבים וזמן מהמערכת.

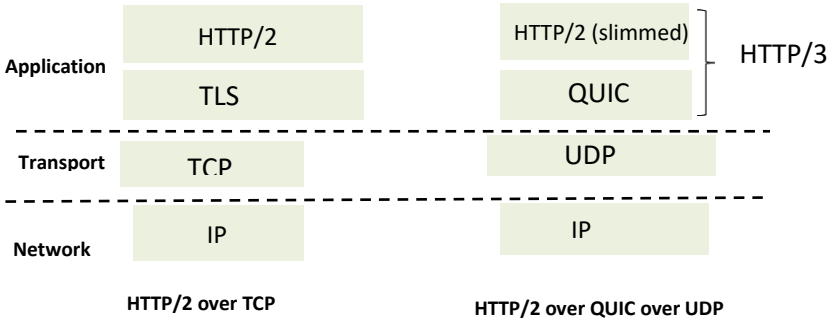
פרוטוקול http 1.1 הוא persistent HTTP כלומר אנחנו לא סוגרים את החיבור לאחר בקשה ותגובה אלא ממשיכים לבקש ולקבל אובייקטים על אותו חיבור ראשון שיצרנו, כלומר אנחנו נבקש כמה וכמה אובייקטים ביחד ונקבל אותם אחד אחרי השני על אותו חיבור לפי הסדר שבו ביקשו אותם.

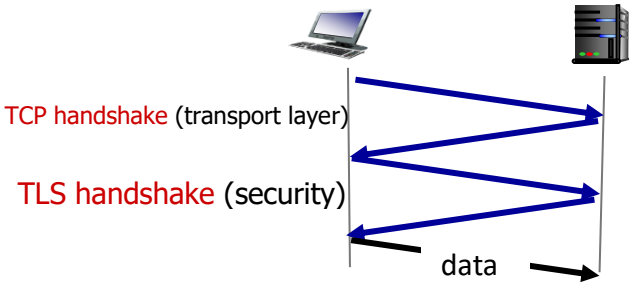
פרוטוקול http 2 הוא persistent HTTP כמו http 1.1 אך ב http 2 יש אפשרות לשרת לבצע push לאובייקטים ולשלוח אותם ללקוח גם אם הוא לא ביקש אותם במפורש. בנוסף השרת יכול לחלק את האובייקטים ל frames (כלומר אובייקט גדול יחולק להרבה frames קטנים) ויוצר מסגרת שליחה שלהם (schedule frames to mitigate HOL blocking) יכולת זו עוזרת כאשר packet אובדת אז אנו יכולים להמשיך לשלוח (בניגוד ל http 1.1 ששם עצרנו את השליחה עד שמצליחים לשלוח את ה packet שאבדה).

פרוטוקול QUIC מכיל הרבה מאוד מהמאפיינים של http 2 למשל:

- reliability, congestion control, authentication, crypto state
- multiple application-level “streams” multiplexed over single QUIC connection

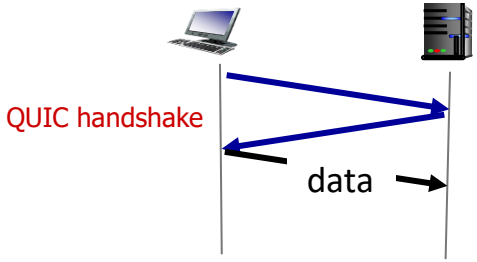
ההבדל העיקרי הוא שבעוד http 2 משתמש בפרוטוקול TCP לעומת זאת ה QUIC משתמש בפרוטוקול UDP כלומר ביצירת החיבור יש handshake 1 ואילו ב http 2 יש
2 serial handshake





TCP (reliability, congestion control state) +
TLS (authentication, crypto state)

- 2 serial handshakes



QUIC: reliability, congestion control, authentication, crypto state

- 1 handshake

4) **למה צריך מספרי port?**

<u>תשובה</u>- נניח ויש לנו חיבור שרת ללקוח, והשרת מספק מספר שירותים שונים למשל מייל ו web (למשל HTTP) אזי כאשר הלקוח שולח שתי בקשות אחת

של מייל ואחת של HTTP השרת צריך להפריד ביניהן ולשלוח כל בקשה לתוכנה (אפליקציה) המתאימה. על מנת שהשרת יצליח להפריד ביניהן ולשלוח אותן לאפליקציה המתאימה אנו צריכים מספר מזהה לכל אפליקציה כלומר לא מספיק שהלקוח יודע לפנות לשרת באמצעות IP של השרת אלא הוא גם צריך לספק מספר מזהה לסוג השירות אותו הוא מבקש (כלומר לאיזו אפליקציית שירות אתה פונה מבין כל השירותים שהשרת מספק). המספר המזהה הוא **הפורט**, באמצעות פנייה לפורט מסוים בבקשה השרת יכול לדעת איזה סוג שירות אנו מבקשים ולאיזו אפליקציה אנו פונים לדוגמה אם נשלח הודעה לפורט 80 באמצעות פרוטוקול TCP השרת יבין שסוג הבקשה הוא web (HTTP למשל).

5) **מה זה subnet ולמה צריך?**

תשובה- ה subnet נועד לעזור לנו לדעת מהו מזהה הרשת (הרשת הכללית של המחשב) שבה אני נמצא, כלומר הוא עוזר לנו להבחין בין הכתובת של הרשת הכללית של המחשב לבין הכתובת של הרשת הפרטית של המחשב.
הוא מגדיר כמה ביטים מתוך כתובת ה IP מייצגים את מזהה הרשת (הרשת הכללית של המחשב) 192.168.0.1/16 הוספת "/16" בסוף הכתובת מציינת שה subnet מכיל 16 ביטים כלומר הרשת שלי היא 192.168.0.0 וכתובת המחשב שלי היא 192.168.0.1 .
בנוסף ה subnet מאפשר שימוש של אותה כתובת בכמה מכשירים שונים שנמצאים ברשתות אחרות מה שמהווה פתרון לכך שאין מספיק כתובות בעולם.

6) **למה צריך כתובות mac למה לא מספיק לעבוד עם כתובות IP ?**

תשובה- כתובות ה IP של מחשבים משתנות בצורה עקבית ומשומשות בשביל לתקשר בין רשתות שונות בכלל ובין רכיבים שונים ברשת בפרט. עקב כך נוצרת בעיה בתקשורת בין שני מחשבים מרשתות שונות כי כתובות ה IP שלהם משתנות ואינן ייחודיות.
לעומת זאת כתובות ה mac הינן קבועות תמיד וצרובות על NIC של המכשיר.
בנוסף כתובות mac הינן ייחודיות לכל מכשיר ומשומשות בשביל לתקשר בין רכיבים באותה הרשת ובמעבר בין שתי רשתות שונות במידת הצורך.
עקב כך כתובות ה mac הינן חיוניות ונחוצות על מנת שרכיבים יוכלו להזדהות בצורה חד משמעית וייחודית ברשת ובין רשתות.

7) **מה ההבדל בין Router Switch Nat ?**

תשובה- ה Router משמש לחיבור בין רשתות לוקאליות שונות (כלומר בין רשת לוקאלית -LAN לבין רשתות אחרות- World area network) כלומר מאפשר למחשבים שנמצאים ברשתות שונות לתקשר אחד עם השני. הוא מאחסן כתובות IP בטבלת הניתוב ושומר על כתובת IP משלו.

לעומת זאת, **ה Switch** משמש להעברת תקשורת בין רכיבים שנמצאים באותה הרשת ומקשר ביניהם והוא עושה זאת באמצעות כתובות MAC שיש לכל רכיב. לכל Switch יש טבלה שבה יש שיוך בין פורט פיזי לבין כתובת MAC .

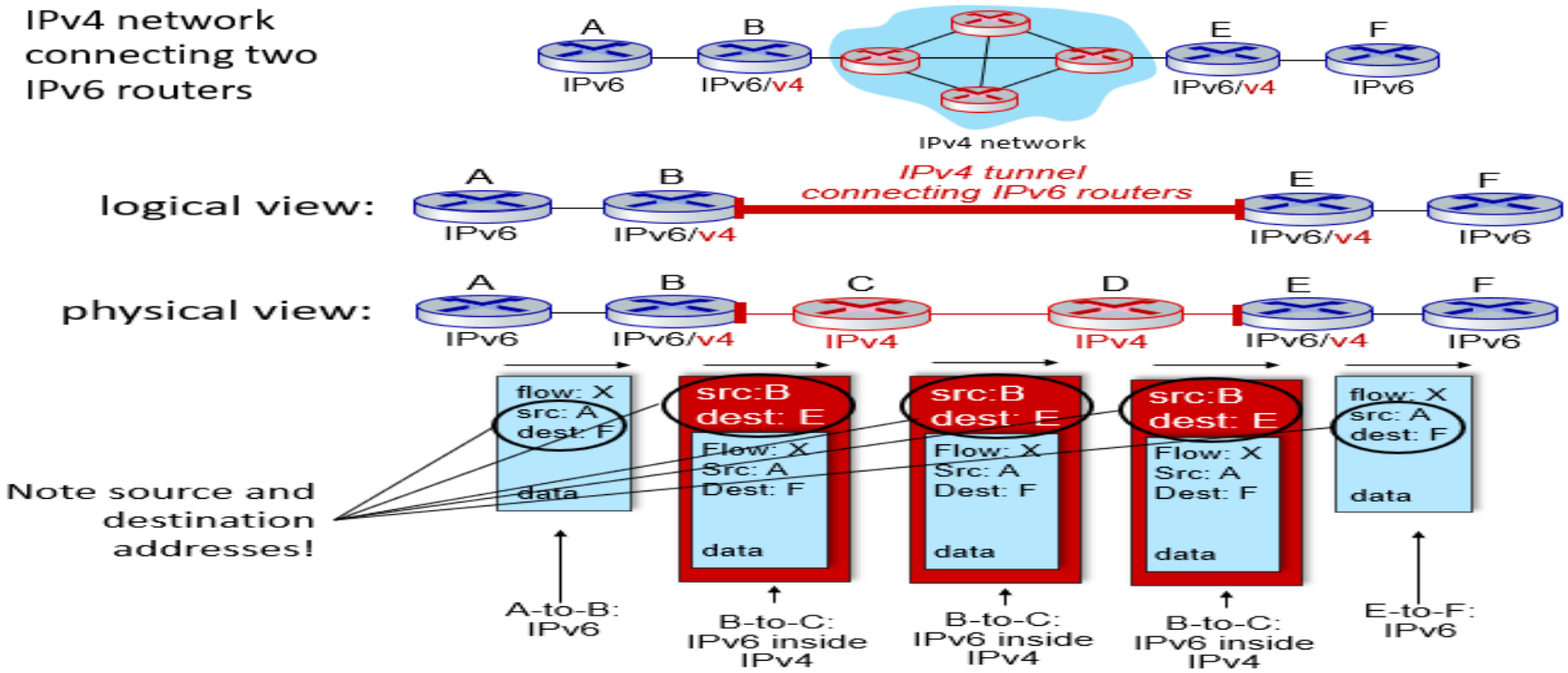
ה NAT (Network Address Translation) הוא בעצם תהליך של תרגום ומיפוי של מספר כתובות אישיות שונות לכתובת ציבורית אחת. כאשר הלקוח פונה לשרת הוא צריך לפנות דרך הכתובת הציבורית שלו על מנת שהוא יוכל לקבל בחזרה את התשובה לשאילתה שלו. אם ללקוח אין כתובת ציבורית אזי השרת לא ידע למי להחזיר תשובה לשאלה שהוא קיבל.

8) שיטות להתגבר על המחסור ב IPv4 ולפרט?

<u>תשובה</u>- החיסרון הבולט של IPV4 הוא שבסופו של דבר עקב הפופולריות שהולכת וגדלה של האינטרנט וריבוי המכשירים שצריכים לקבל כתובת IP אין מספיק כתובות לתת בשביל כל המכשירים בעולם כלומר נוצר מחסור בכתובות!

בשביל להתגבר על הבעיה פותחו מספר שיטות למשל **שימוש ב IPV6** שבו יש כמו הרבה יותר גדולה של כתובות (2 בחזקת 128 ! לעומת 2 בחזקת 32) אולם החיסרון בשיטה זו היא שזה פרוטוקול יחסית חדש וכיום רוב המערכות עדיין מושתתות על הפרוטוקול IPV4.

שיטה נוספת שפותחה היא שימוש ב NAT64 בשיטה זו אנו לוקחים כתובת IPV4 "ועוטפים" אותה ב IPV6 כך שבעצם אנו יכולים להשתמש באותה כתובת מסוג IPV4 ולשנות את כתובת "העטיפה" שהיא מסוג IPV6 (אנו בעצם מתרגמים את הכתובת מסוג IPV6 לכתובת ה IPV4 ולהפך).



משויכת אליו מתפנית בשביל לקוח אחר שיתחבר לרשת בעתיד. כלומר שרת ה DHCP מבצע ניהול דינאמי של כתובות IPV4 ושומר אילו כתובות פנויות ואילו תפוסות.

9) a,b,c,d נתונים על הרשת:

e) בעזרת איזה פרוטוקול לומד הנתב 3c על תת רשת x ?

<u>תשובה</u>- 3c נמצא ברשת AS3 ואילו תת רשת x נמצאת ב AS4 לכן כדי לעבור מ AS3 ל AS4 נשתמש בפרוטוקול BGP (שמשתמש בפרוטוקול TCP) ככה 3c יכול לקבל מידע מ 4c, ואילו 4c יקבל מידע על התת רשת x תוך שימוש בפרוטוקול RIP (אשר משתמש בפרוטוקול UDP) .

f) **בעזרת איזה פרוטוקול לומד הנתב 3a על תת רשת x ?**

<u>תשובה</u>- 3a נמצא ב AS3 ולכן יקבל מידע מ 3c תוך שימוש בפרוטוקול OSPF ואילו 3c מקבל מידע על תת רשת x כפי שהסברנו סעיף קודם (e) ולכן סה"כ הכל יש פה שימוש בפרוטוקולים RIP,BGP,OSPF

g) **בעזרת איזה פרוטוקול לומד הנתב 1c על תת רשת x ?**

1c נמצא ב AS1 ולכן כדי לקבל מידע על תת רשת x הוא קודם יצור קשר עם 3a (ראוטר קצה) שנמצא ב AS3 באמצעות שימוש בפרוטוקול BGP ואילו 3a מקבל מידע על תת הרשת x כפי שפירטנו סעיף קודם ולכן סה"כ הכל יש פה שימוש בפרוטוקולים הבאים- BGP,RIP,OSPF

h) **בעזרת איזה פרוטוקול לומד הנתב 2c על תת רשת x ?**

כיוון שאין חיבור פיזי בין AS2 ל AS4 2c יכול ללמוד על התת רשת x רק אם הוא יעבור דרך כל ה AS השונים בצורה הבאה:

2c יקבל מידע על תת רשת x מ 2a באמצעות פרוטוקול OSPF (כיוון ששניהם נמצאים ב AS2). 2a יקבל מידע על תת הרשת x מ 1b (שנמצא ב AS1) באמצעות פרוטוקול BGP . 1b יקבל מידע על תת הרשת x מ 1c באמצעות פרוטוקול RIP (שניהם נמצאים ב AS1) ואילו 1c יקבל מידע על תת הרשת x כפי שהסברנו בסעיף הקודם. סה"כ קיבלנו שאנו משתמשים בפרוטוקולים הבאים- BGP,RIP,OSPF