פרויקט הצ'אט, סוף קורס תקשורת ומחשוב – אוניברסיטת אריאל בשומרון - סמסטר א תשפ"ב

הקדמה

המטרה הייתה ליצור מערכת צ'אט בו יש חדר צ'אט אחד אליו יכולים להיכנס כמה משתמשים, ולדבר ביניהם בין בהודעות פרטיות אחד לשני, ובין בהודעות פומביות המיועדות לכלל המשתמשים המחוברים כעת.

בנוסף, כל לקוח יכול להוריד קבצים שיושבים על השרת, ובאפשרותו לבחור אם להוריד את הקובץ מעל TCP, או מעל UDP – והדרך האחרונה מהווה את עיקר הלימוד התקשורתי עבור הפרויקט.

בתור התחלה, הדבר הראשון שעשינו הוא פרוטוקול אפליקטיבי, שהוא בעצם החוזה בין השרת ללקוח..

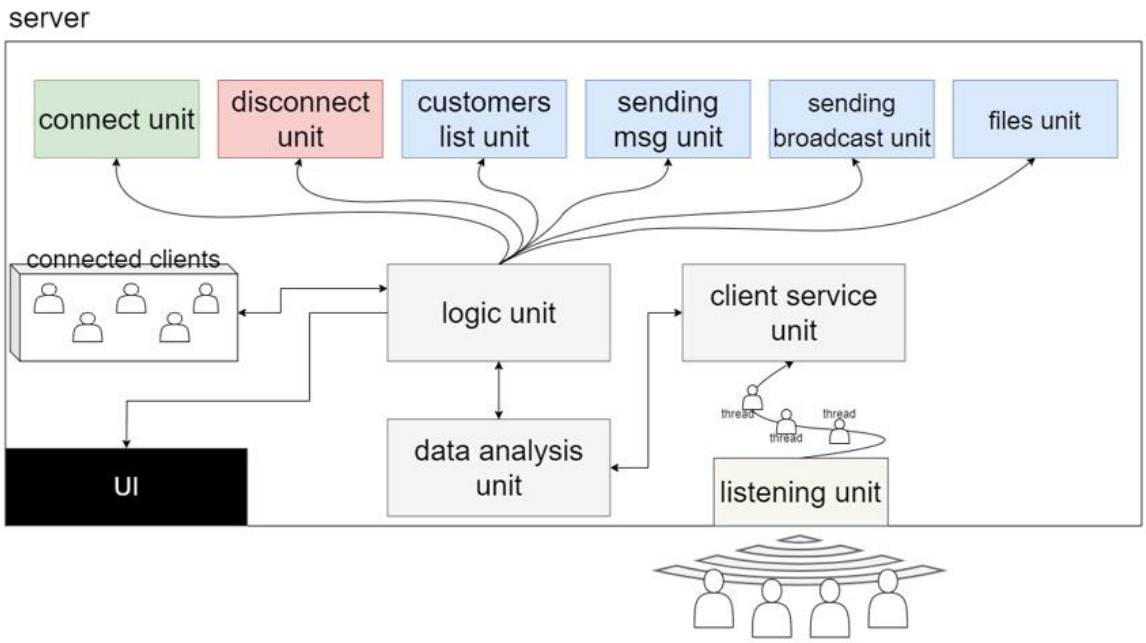
אין טעם להתחיל את העבודה ללא הפרוטוקול, כמובן.

עבור בניית הפרוטוקול הטקסטואלי קיבלנו השראה מפקטות ICMP שמתחילות תמיד ב CODE ו TYPE, ובאופן כללי קיבלנו השראה מצורת הפרוטוקולים של מערכת השכבות, שבנויות כך שהמחשב שקורא את ההודעה יודע על סמך פרוטוקול מסוים כיצד לקרוא את הפרוטוקול את עצמו, ואת זה שעובר מעליו. למשל, בקריאת השדה TYPE של פרוטוקול Ethernet נוכל לדעת אם נצטרך לקרוא עכשיו את פרוטוקול IP, או ARP – למשל. דוגמא נוספת (או המשך דוגמא) השדה הראשון של פרוטוקול IP מציין האם השדות הם של IPv4 או IPv6 וכך בזמן הקריאה של הפקטה אפשר להמשיך לקרוא אותה בצורה נכונה.

מצורף קישור למסמך התיעוד של חוקי הפרוטוקול כאן

למרות הקישור המצורף, מידי פעם עבור ההסברים יצורפו צילומי מסך ממסמך התיעוד של הפרוטוקול.

לאחר בניית הפרוטוקול התחלנו לתכנן את תמונת הארכיטקטורה הגדולה עבור תוכנת השרת ועבור תוכנת הלקוח. תמונת "מבט העל" על המערכת, ללא המימוש – רעיוני בלבד עדיין – מצורף כעת עבור השרת:



בשרת קיימת יחידת האזנה, הממתינה למשתמשים חדשים.

גם קיימת יחידת שירות לקוחות - כל לקוח שמתחבר מקבל ת'רד שמטפל בו על ידי היחידה הזאת.

הפונקציות שמאפשרות את הטיפול בלקוח נקראות על ידי יחידת הלוגיקה, ואלו מטפלות במגוון הבקשות מהלקוחות. על מנת להבין מה הלקוח ביקש בכלל, קיימת יחידת הניתוח, שהיא בעצם מספרת ליחידת הלוגיקה מה הלקוח רצה, על פי המידע של הפרוטוקול האפליקטיבי שזרם מהסוקט.

את הלוגים השונים על פעולות השרת נוכל לראות מודפסים בטרמינל. (UI)

דוגמא לבקשה שהגיע מהלקוח, ועל השרת לטפל בה, היא בקשה להתחבר לצ'אט עצמו.

זו הבקשה וההסבר שלה במסמך הפרוטוקול:

מזהה הודעה	מטרה	מבנה	הסבר מבנה ההודעה לאחר המזהה
100	בקשת התחברות לשרת	100XX<USERNAME>	XX – אורך שם המשתמש USERNAME – השם שהמשתמש בוחר לעצמו.

(זו גם ההודעה הראשונה במסמך, לכן יש כותרות על העמודות. בצילומי מסך האחרים לא יהיו הכותרות)

ההודעה הזאת מגיעה מסוקט הלקוח לשרת, שמחכה בלופ להודעות מהסוקט הזה, והשרת צריך להבין על ידי יחידת הניתוח שמשתמש רוצה להיכנס לצ'אט עם השם שהוא בחר לעצמו.

מכאן יחידת הלוגיקה תרצה להחזיר תשובה בהתאם – או אישור התחברות, או הודעת שגיאה (כמו שרת מלא או שם משתמש שתפוס כבר)

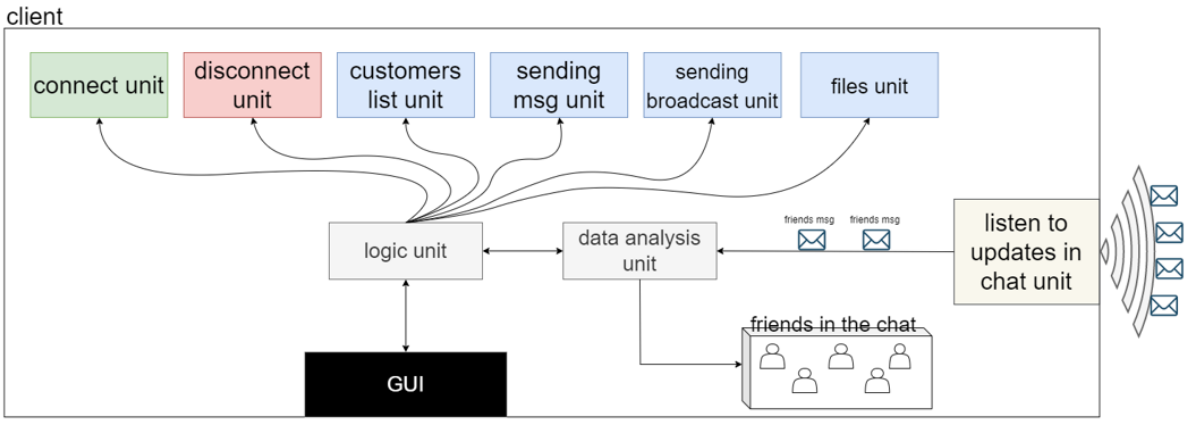
אז השרת ירצה ליצור למשל את הודעת האישור כניסה לצ'אט.

זו ההודעה שהפונקציות המתאימות ביחידת הלוגיקה תיצור לשליחה:

200	אישור התחברות לשרת	200XX<USERNAME><PORT>	XX – אורך שם המשתמש USERNAME – השם שהמשתמש בוחר לעצמו. PORT– הפורט בו השרת במיוחד עבור הלקוח מאזין וממנו יגיעו הודעות מהחברים בצאט (הודעות בקוד 3)

הודעה זו אומרת למעשה "קיבלתי את בקשתך להיכנס לצ'אט עם שם המשתמש שבחרת, ועכשיו התחבר מיד לפורט הנוסף שהקצתי לך, ותקשיב בלופ להודעות נכנסות כי זה בעצם "מסך הצ'אט" (=הודעות בלתי צפויות - כמו לקוח חדש שנכנס, יצא, הודעה שהגיע אליך ממשתמש אחר בפרטיות או הודעה שהגיעה כי היא הגיעה לכלל המחוברים."

באופן דומה, זהו "מבט העל" של ארכיטקטורת הלקוח, שהוא בהתאם יוצר בקשות, ומנתח את התגובות והעדכונים מהשרת:



היחידות הקיימות בלקוח מאוד דומות ליחידות של השרת – כלומר יש יחידת ניתוח של תגובות ועדכונים מהשרת, ויחידת לוגיקה שתפקידה ליצור בקשות בהתאם לפקודות מהממשק הגרפי – עליו כלל עוד לא דיברנו עדיין. יחידת הלוגיקה בעצם משתמשת בפונקציות המתאימות ליצירת הבקשות שהלקוח מעוניין בהם, אותם בקשות מנותחות בקצה על ידי השרת, וזה האחרון מחזיר תגובות אותם הוא בנה עם יחידת הלוגיקה שלו, אותם יחידת הניתוח של הלקוח מנתחת.

נקודה חשובה – כל לקוח מתקשר במספר סוקטים עם השרת:

סוקט ראשון דרכו עוברות הודעות 100 בפרוטוקול, וחוזרות תגובות 200 או 400. נקרא לו **"הסוקט המרכזי"**

סוקט שני הוא סוקט שהלקוח מקשיב בלופ למידע ממנו תמיד, ומידע זה הוא עדכונים מהשרת - אלו הודעות 300 של הפרוטוקול. נכנה סוקט זה בשם **"סוקט 300"**, זה היה הכינוי שלו במהלך הפיתוח. סוקט זה נוצר אחרי התחברות מוצלחת לחדר הצ'אט.

סוקט שלישי נוצר ונמחק עם כל שליחת קובץ, נקרא לו **"סוקט הקובץ"**, והוא יהיה מסוג TCP או UDP, בהתאם לבקשת המשתמש.

אציין גם שהשרת מחזיק "בריכת פורטים" פנויים, וכל פורט שהסתיים השימוש בו – חוזר לבריכה.

FAST reliable UDP with CC

עד נקודה זו החיים היו נוחים, TCP דאג לנו שהמידע יגיע בצורה אמינה.

כעת נדרשנו "לממש את TCP מעל UDP", כלומר במקום ששכבת התעבורה תדאג להעברה האמינה של המידע, שכבת האפליקציה הייתה צריכה לדאוג לכך. שינוי סדרי עולם :)

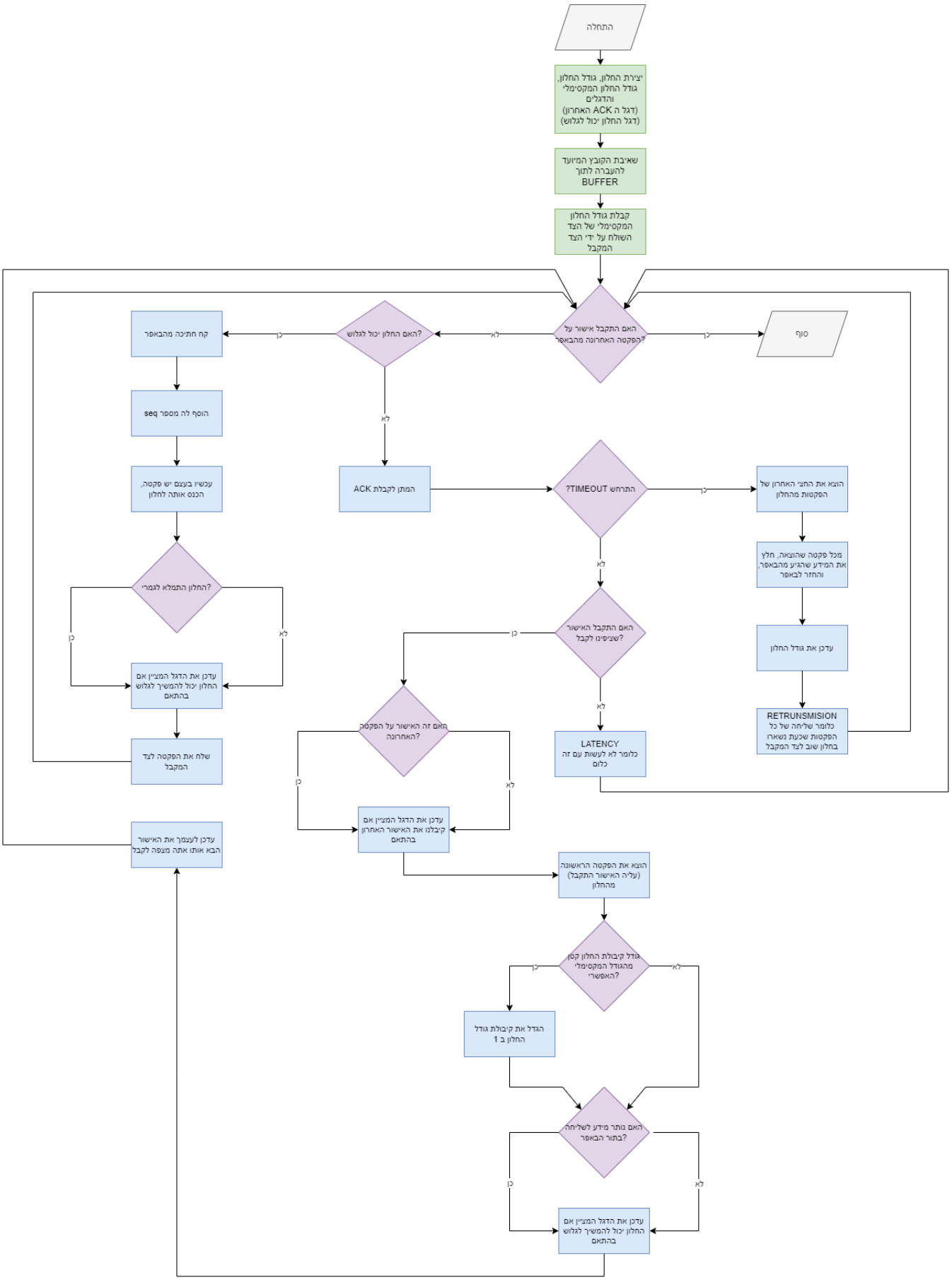
על מנת לממש את ה RDT היה צורך קודם ללמוד על פרוטוקולי sliding window.

- על מנת לעשות זאת קודם חרשנו סרטוני המחשה על הפרוטוקולים Stop & Wait, Go-Back-N, ו- selective repeat.
- החלטנו לא ישר ללכת על ה selective, כי מן הסתם גם במציאות קודם כל נוצר S&W, אחריו GBN, ורק בסוף ה selective repeat .
- לכן ה RDT הראשון אותו עשינו היה S&W, ואחרי שהתגברנו על כלל מקרי הקצה (חבילה לא מגיעה לנמען, חבילה מגיעה לנמען והאישור לא מגיע למוען, מתרחש timeout, מתרחש timeout ואחריו מגיע האישור באיחור, הנמען מקבל אותה חבילה יותר מפעם אחת) רק אז עברנו ל GBN.
- כיוון ש S&W הוא מקרה פרטי של GBN, בו ה N שווה ל-1, היינו צריכים לעשות את ההתאמות הנדרשות על מנת מעבר לעבודה עם GBN, שבגדול זה אומר ליצור חלון, והגדלת טווח ה seq-ים האפשריים.
- על מנת לבדוק את המערכת יצרנו שני מקרים מלאכותיים:
  1. איבוד חבילות, על ידי התעלמות אחת לכמה חבילות בצד המקבל, כך שלא באמת נקבל את החבילה ולא יישלח ack.
  2. Latency, על ידי "מנוחה" קצרה בצד המקבל אחת לכמה חבילות – כך שהאישור יגיע, אבל באיחור.
- על מנת לבדוק ממש את המערכת הרצנו אותה עם שינויים בגודל החלון, שלחנו קבצים גדולים, קטנים, קבצי טקסט וקבצים בינאריים וסתם מחרוזות.
- בדיקה נוספת של המערכת הייתה על ידי מבחן שנתנו לה לעשות:
  1. אני רוצה לשלוח 9 חבילות, גודל החלון הוא 3, כל פקטה חמישית הולכת לאיבוד – והשאר תקין. כמה פקטות בפועל נצטרך לשלוח? לאחר ספירה, המערכת ענתה את התשובה 16. זו גם התשובה הנכונה. תודה לסרטון הזה
  2. באותו אופן, כאשר לפי הנתונים רצינו לשלוח 10 פקטות, גודל החלון הוא 4 וכל חבילה שישית נאבדת, המערכת ספרה 17 פקטות סה"כ, שזו גם התשובה. תודה לסרטון הזה
- כעת החלטנו להטמיע את המערכת בתוך השרת-לקוח לפני שנעבור ל selective repeat.
אז קודם כל, לפרוטוקול האפליקטיבי נוסף מבנה הפקטה ותגובה לפקטה כאשר מעבירים אותה מעל UDP. ניתן לראות את ההוספה בסוף טבלת התיאור של הפרוטוקול.
- לאחר מכן הטמענו את המערכת של ה GBN לתוך השרת והלקוח עצמם.
- רק אחרי שראינו שהטמעת מערכת ה RDT הייתה טובה, הוספנו CC שדומה ל RENO,
כך שכאשר מתרחש timeout, קודם כל החצי השני של החלון חוזר לבאפר, ורק אז נעשה retransmission על הפקטות שנותרו. לאחר מכן, הגדלת החלון נעשית בצורה לינארית.
- בפועל, לא הספקנו להתקדם ל selective repeat, ולכן **במערכת שלנו קיים GBN עם CC סטייל RENO.**
- בשני העמודים הבאים יופיעו תרשימי זרימת ה BGN עד סיום שליחת הקובץ (buffer יותר נכון)
- לאחר מכן נענה על השאלות שבדף הדרישות עבור הפרויקט.

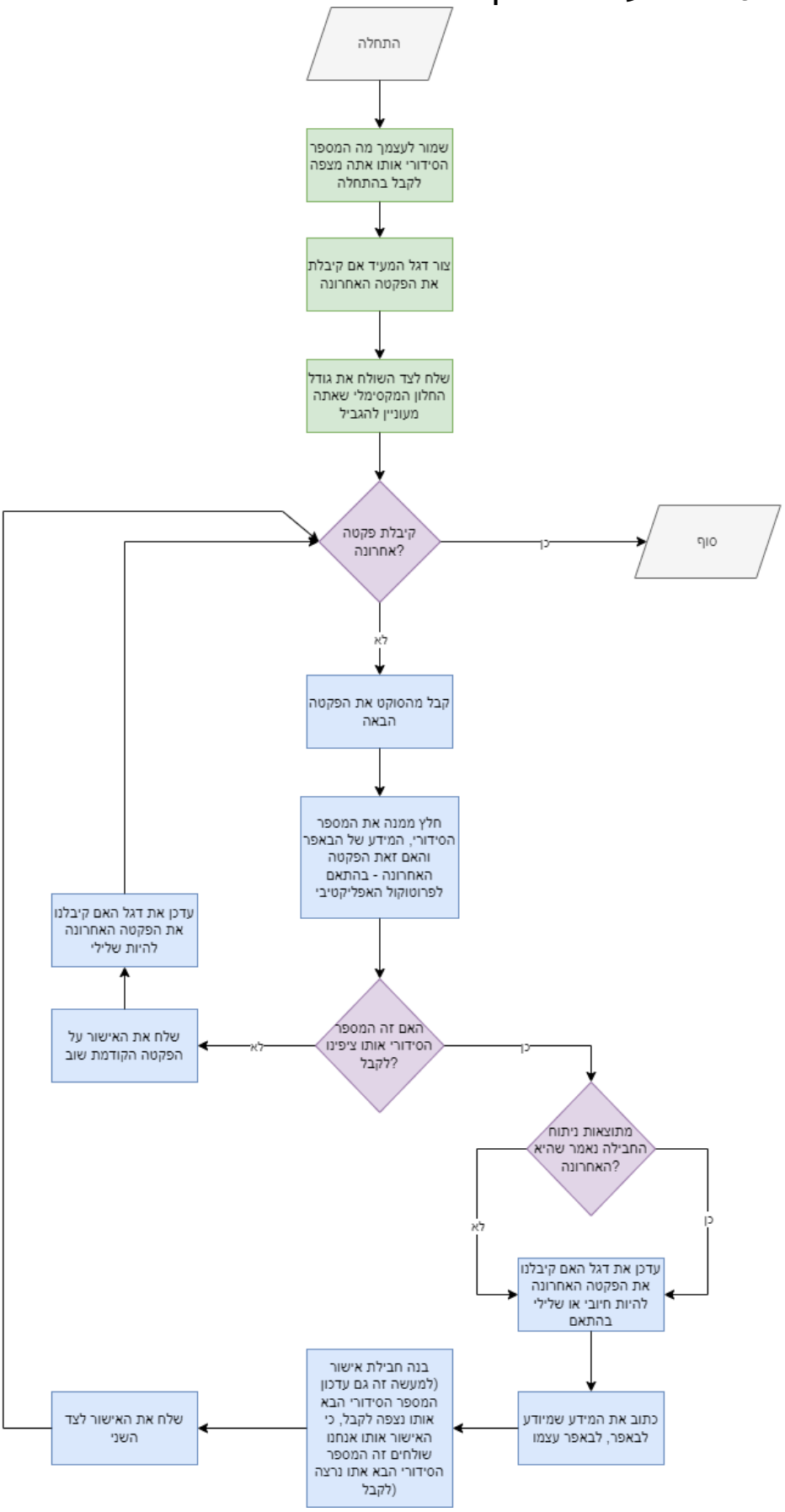
בס"ד

מגישים: 323104562 ו- 324942077

GBN של הצד השולח:



תרשים זרימה עבור הצד המקבל



- המערכת מתגברת על איבוד חבילות כך:
  - בצד השולח מתרחש retransmission, גודל החלון קטן פי 2, והחבילות שנותרו בו נשלחות שוב.
  - בצד המקבל – אם נשלח חלון, ונאבדה למשל חבילה והרסה את הרצף בצד המקבל, החבילות שהיו אחריה באותו חלו, יזכו להתעלמות בצד המקבל, כי הסדר של ה seq נשבר, ויישלח ack עבור הנקודה שחסרה לו עדיין. למעשה גודל החלון של הצד המקבל ב GBN הוא 1, ולכן פשוט יישלחו ack-ים על מה שכבר נשלח ack ממקודם

- המערכת מתגברת על בעיות latency כך:
  - בכל זמן נתון קיים משתנה השומר את ה ack אותו אנחנו מצפים לקבל. אם קיבלנו ack שלא תואם את מה שאנחנו רוצים, פשוט נתעלם ממנו ונמשיך כאילו כלום לא קרה.
  - בצד המקבל יש משתנה דומה, שדואג שלא נכניס לקובץ (ל buffer הקבלה יותר נכון) אותו מידע פעמיים, אלא רק את המידע עם ה seq הרצוי באותו הרגע.

הוראות התקנה והפעלה

- דאגו מראש שיותקן לכם python3 על המחשב, בין להרצת שרת ובין להרצת לקוח.
  מומלץ להתקין git.
- באמצעות git, עשו clone עם הקישור הבא: https://github.com/amirg00/Simple-Chat.git
- אנו מבקשים לעיין גם ב readme שבגיט.
- כעת ירדה לכם תיקיה בשם simple-chat
  - על מנת להריץ את השרת, כנסו לתיקיית server והריצו את הקובץ main.py
  - על מנת להריץ את הלקוח, כנסו לתיקיית client, והריצו את הקובץ GUI.py

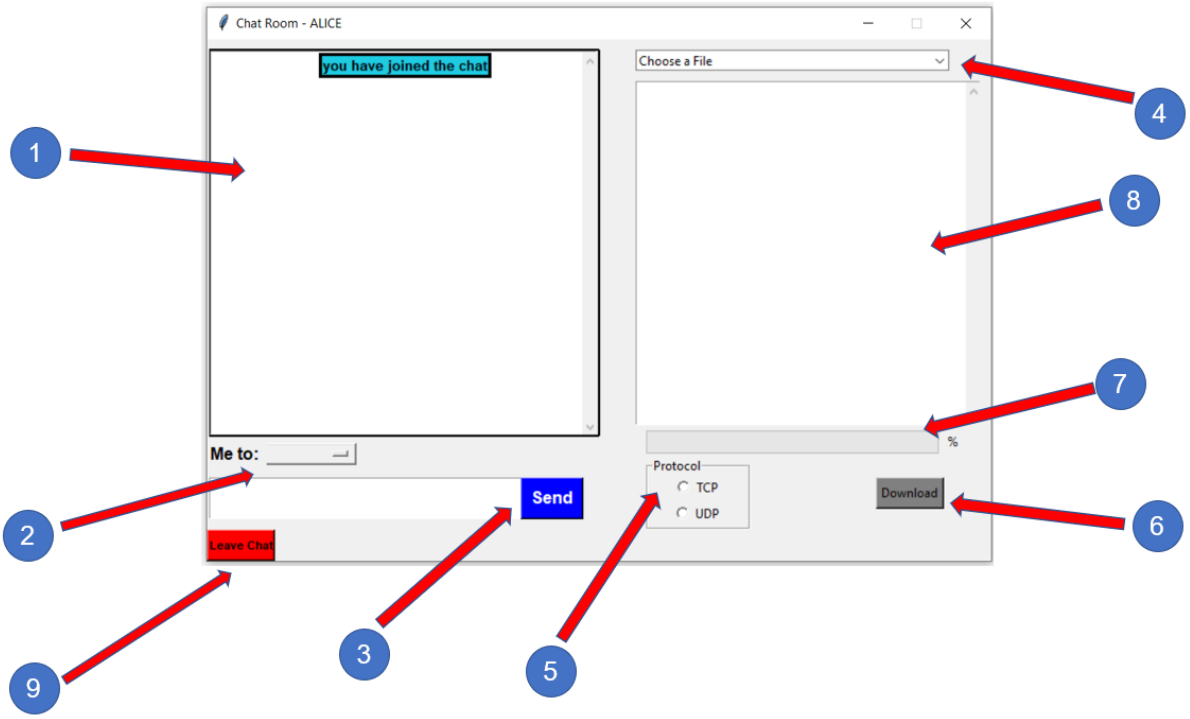
כעת נדגים שימוש בתכנה.

סיפור הרקע הוא שאליס רוצה לשלוח הודעה לבוב, שכבר מחובר לצ'אט.

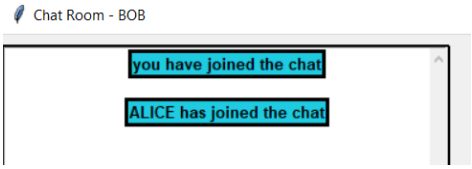
כל שאליס צריכה לעשות, הוא לדעת את כתובת ה IP והפורט של השרת על מנת להתחבר:

אל דאגה, אם שכחתם את הפורט, או שהכתובת לא נכונה, או שמשהו בפרטים לא הולך לעבוד, הממשק יידע לעדכן אתכם מה קרה ומה לעשות.

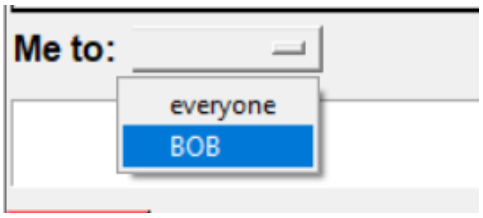
כעת נכנס למסך הצ'אט. המשך בעמוד הבא



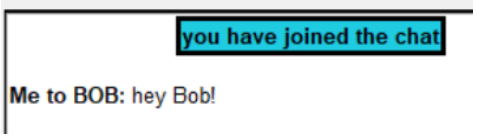
1- מסך הצ'אט, בו יופיעו הודעות ועדכונים. למשל, עכשיו כשאליס נכנסה, במסך של בוב תופיע ההודעה הבאה:



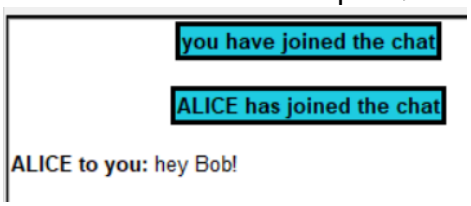
2- כפתור עליו לוחצים ובוחרים למי נרצה לשלוח הודעה. כאשר אליס תלחץ עליו, זה מה שהיא תראה. נניח שהיא גם תבחר עכשיו את בוב:



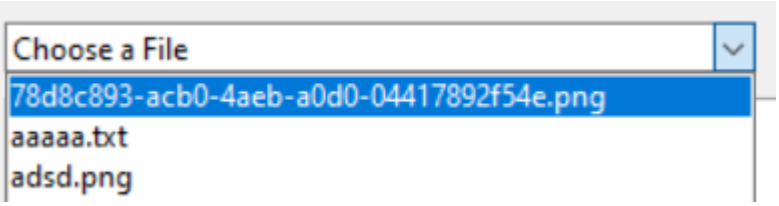
3- על מנת לשלוח הודעה, צריך לכתובת אותה בתיבת הטקסט וללחוץ send.
לאחר מכן, זה מה שיופיע במסך של אליס:



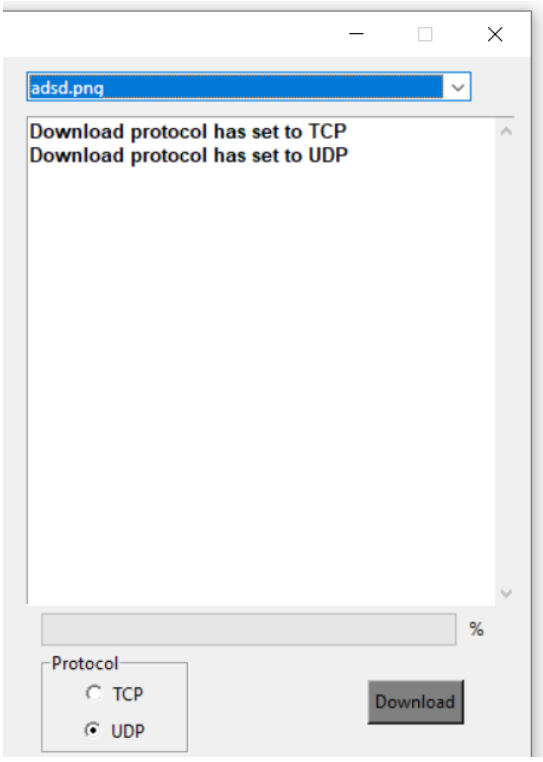
וזה מה שיופיע במסך של בוב:



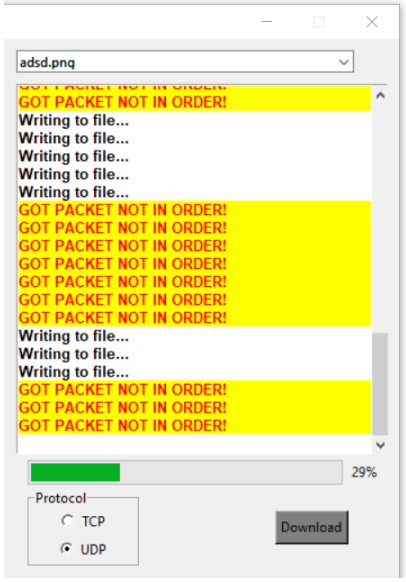
4- אם ברצונכם להוריד קובץ שיושב בשרת, עליכם לבחור אותו מהתפריט המוצג:



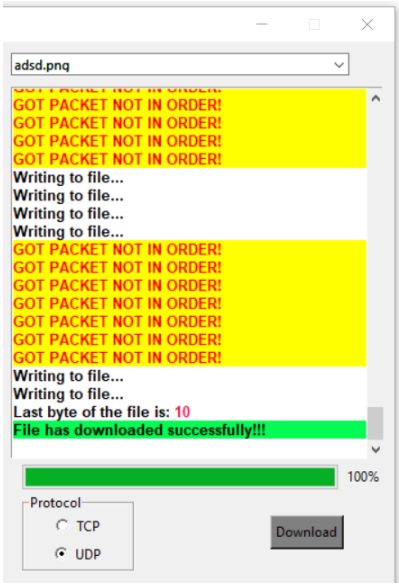
5- נניח שאליס רצתה לבחור הורדה של adsd.png מעל TCP, אבל אז היא החליטה לאתגר את המערכת ולהוריד דרך UDP. כל שעליה לעשות הוא לבחור את פרוטוקול שכבת התעבורה שברצונה להשתמש:



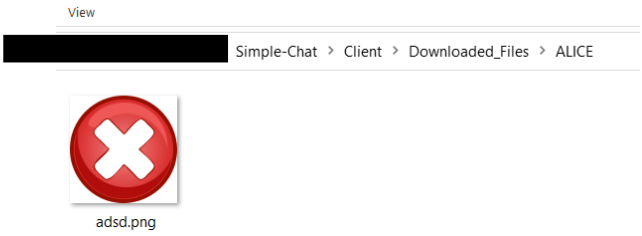
6- זהו, היא יכולה ללחוץ על כפתור ההורדה. (אגב, אם חסר הקובץ להורדה, המערכת תעדכן בהתאם. בנוסף – אם לא ייבחר פרוטוקול תעבורה, הפרוטוקול הדיפולטיבי יהיה דווקא UDP)



בצהוב – ציון פקטות שהגיעו לא לפי הסדר.
יש לציין שבשביל הפרויקט עשינו איבוד פקטות והשהייה מובנית **בכוונה,** כדי להראות שזה עובד. ניתן כמובן לבטל את האופציה (בקוד)
בסוף ההורדה נראה את זה:



את הקובץ נמצא בתקיה עם שמה של ALICE (השם הוא ID לא ניתן להתחבר לשרת עם שם שכבר נתפס. במסך הכניסה המשתמש לא יצליח להיכנס אם שמו נתפס, והוא יעודכן בהתאם)

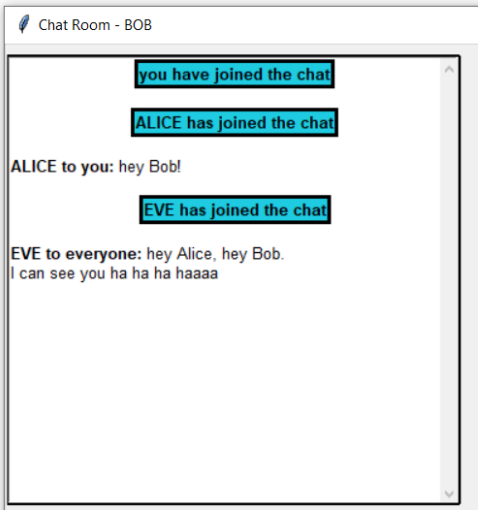


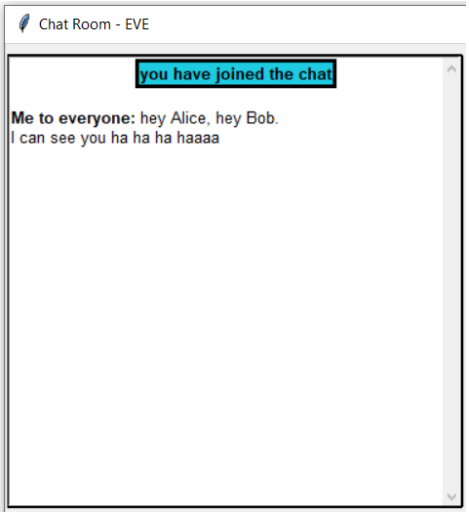
נראה לי חשוב לציין פה שה X זה התמונה עצמה 😊 ולא שווינדוס אומר שהקובץ מקולקל :)

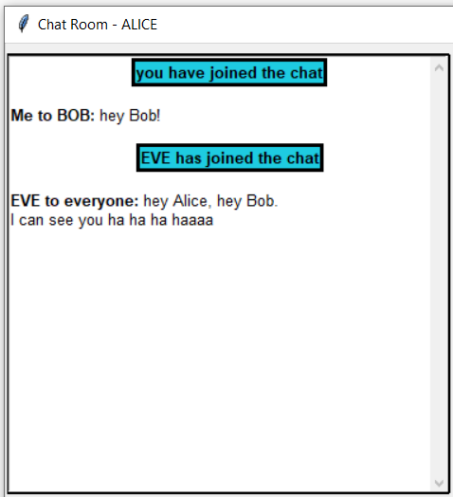
7- זה המסך עם הלוגים על מצב ההורדה. אין מה להרחיב מעבר למה שכבר נאמר קודם.

8 – מעיד על כמה אחוזים מהקובץ כבר ירדו בזמן אמת.

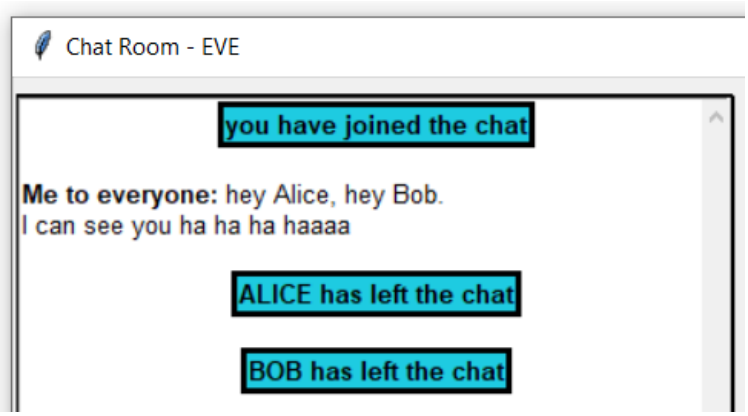
8- כפתור יציאה. נניח לרגע ש EVE נכנסה ושלחה הודעה לכולם. זה ייראה כך:







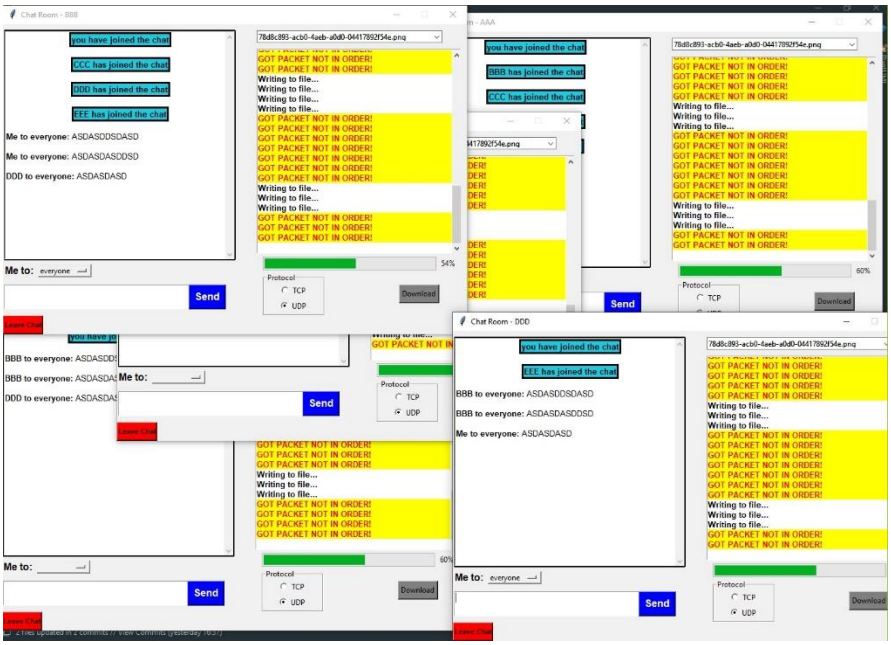
אבל כאשר EVE נכנסת, תמיד קורים דברים רעים.. אז אליס ובוב החכמים והטובים יצאו מיד. לצורך חיסכון במקום בדף, נראה רק את התוצאה על הצ'אט של EVE:



לבסוף, נדרשנו בפרויקט לאפשר לכמה לקוחות במקביל להוריד קבצים.

למעשה יש כאן יותר מזה – כל לקוח יכול להוריד כמה קבצים במקביל! כל מה שהוא צריך לעשות זה שבזמן שיורד קובץ אחד, לבחור קובץ אחר ולעשות "הורדה". במקרה זה, מד האחוזים יראה כל רגע מה מצב כל אחד מהקבצים.

התמונה הבאה סוגרת את הוראות הממשק, ובה ניתן לראות כמה לקוחות מורידים קבצים (גם אותו קובץ בדיוק אפשר) במקביל:



נציין שניתן להמשיך לקבל ולשלוח הודעות תוך כדי הורדה של קובץ, כי סוקט הקבצים, סוקט 300 והסוקט המרכזי אינם אותו סוקט ואינם באותו תהליכון.

הסבר תעבורה על ידי wireshark והצגת שימוש בפרוטוקול הטקסטואלי על ידי תוכנת ה telnet שהשתמשנו בה במהלך הפיתוח – puttyTM

תמונת הרקע

בוב והשרת על אותו מחשב, השרת עובד ובוב כבר בפנים. עכשיו אליס נכנסת, שולחת הודעה לבוב, בוב מחזיר לה תשובה, ויוצא. אחריו אליס יוצאת. ההסבר הוא מנקודת מבטה של אליס:

התהליך:

אליס מתחברת לשרת עם PUTTY (חיבור מבוסס סוקט)

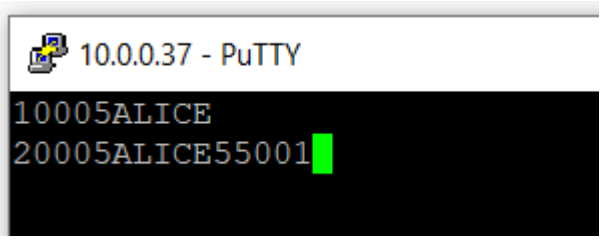
אז מבחינת "ממשק":

ובתעבורה נראה לחיצת יד:

Source	Destination	Protocol	Length	Info
10.0.0.239	10.0.0.37	TCP	66	55507 → 13337 [SYN] Seq=
10.0.0.37	10.0.0.239	TCP	66	13337 → 55507 [SYN, ACK]
10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=

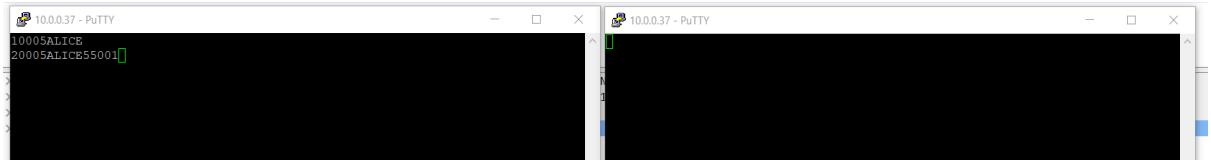
השרת הוא 10.0.0.37 מאזין ב 13337, ואליס היא 10.0.0.239. פורט המקור של הסוקט המרכזי שלה הוא 55507.

כעת אליס תרצה להתחבר לצ'אט עצמו, אז היא תשלח את ההודעה הבאה, לפי הפרוטוקול הטקסטואלי (מוסבר בנפרד!):



אליס שלחה, וקיבלה בתגובה הודעה שאומרת לה להתחבר לשרת בפורט 55001 בשביל להיכנס לצ'אט (מה שנקרא סוקט 300 אצלנו)

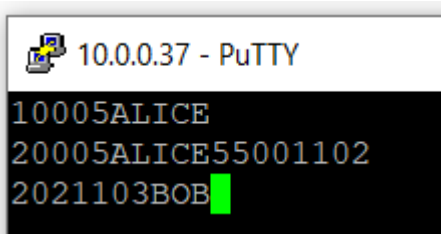
בשביל זה, נפתח עוד חיבור עם התוכנה בפורט 55001:



מבחינת התעבורה פשוט נראה עוד חיבור לשרת:

Source	Destination	Protocol	Length	Info
10.0.0.239	10.0.0.37	TCP	66	55507 → 13337 [SYN] Seq=2031769893 Win=64240 Len=
10.0.0.37	10.0.0.239	TCP	66	13337 → 55507 [SYN, ACK] Seq=3173468115 Ack=2031
10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=2031769894 Ack=3173468116
10.0.0.239	10.0.0.37	TCP	64	55507 → 13337 [PSH, ACK] Seq=2031769894 Ack=3173
10.0.0.239	10.0.0.37	TCP	56	55507 → 13337 [PSH, ACK] Seq=2031769904 Ack=3173
10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [ACK] Seq=3173468116 Ack=2031769906
10.0.0.37	10.0.0.239	TCP	69	13337 → 55507 [PSH, ACK] Seq=3173468116 Ack=2031
10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=2031769906 Ack=3173468131
10.0.0.239	10.0.0.37	TCP	66	55557 → 55001 [SYN] Seq=3147490410 Win=64240 Len=
10.0.0.37	10.0.0.239	TCP	66	55001 → 55557 [SYN, ACK] Seq=3549408049 Ack=3147
10.0.0.239	10.0.0.37	TCP	54	55557 → 55001 [ACK] Seq=3147490411 Ack=3549408050

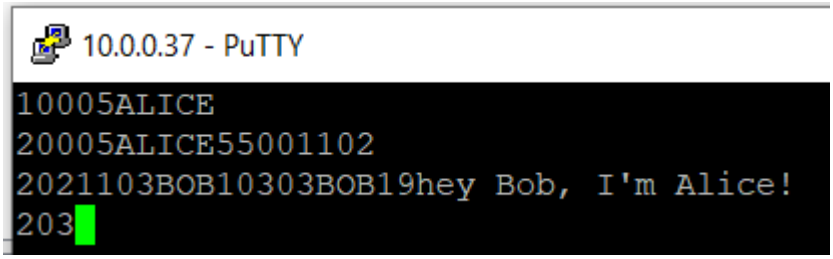
כעת אליס תבקש דרך הסוקט המרכזי את רשימת המחוברים בצ'אט, ותקבל רשימה עם משתמש אחד, בשם BOB.



מבחינת תעבורה נראה את הפקטות הבאות (בדיוק אחרי הפקטה האחרונה תמונה קודמת):

10.0.0.239	10.0.0.37	TCP	57	55507 → 13337 [PSH, ACK] Seq=2031769906 Ack=3173468131 Win=131328 Len=3
10.0.0.239	10.0.0.37	TCP	56	55507 → 13337 [PSH, ACK] Seq=2031769909 Ack=3173468131 Win=131328 Len=2
10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [ACK] Seq=3173468131 Ack=2031769911 Win=1051136 Len=0
10.0.0.37	10.0.0.239	TCP	64	13337 → 55507 [PSH, ACK] Seq=3173468131 Ack=2031769911 Win=1051136 Len=10
10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=2031769911 Ack=3173468141 Win=131328 Len=0

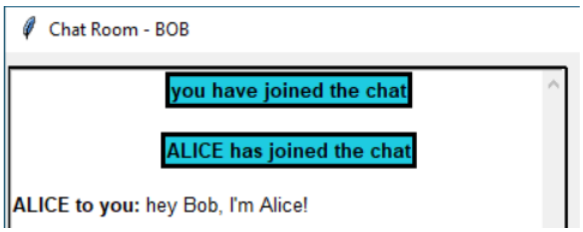
כעת אליס יכולה להרכיב הודעה לפי הפרוטוקול ולשלוח לבוב דרך הסוקט המרכזי. היא תקבל בתגובה הודעה שאומרת לה שההודעה הגיע לבוב, הכל לפי הפרוטוקול:



מבחינת תעבורה נראה את הפקטות הבאות (בדיוק אחרי הפקטה האחרונה תמונה קודמת):

10.0.0.239	10.0.0.37	TCP	83	55507 → 13337 [PSH, ACK] Seq=2031769911 Ack=3173468141 Win=131328 Len=29
10.0.0.239	10.0.0.37	TCP	56	55507 → 13337 [PSH, ACK] Seq=2031769940 Ack=3173468141 Win=131328 Len=2
10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [ACK] Seq=3173468141 Ack=2031769942 Win=1051136 Len=0
10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [PSH, ACK] Seq=3173468141 Ack=2031769942 Win=1051136 Len=3
10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=2031769942 Ack=3173468144 Win=131328 Len=0

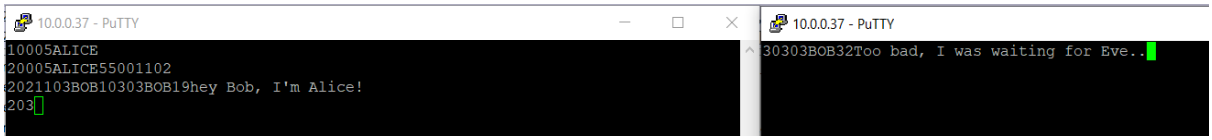
במסך של בוב זה מה שנראה:



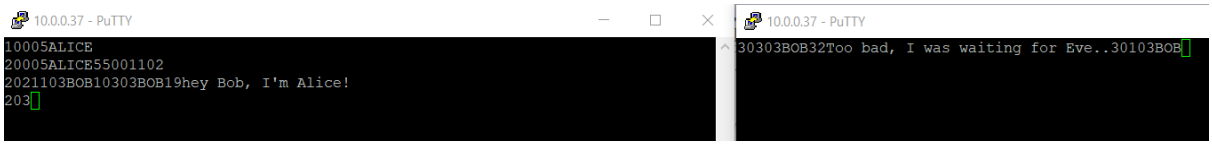
עכשיו בואו נגיד שבוב בכלל רצה את איב, אז הוא שולח לאליס את ההודעה

.. Too bad, I was waiting for Eve

ובחיבור עם סוקט 300, תתקבל אצל אליס ההודעה הבאה:



בעוד אליס המומה ופגועה, בוב עוזב את הצ'אט, והיא מקבלת עוד זבנג לפנים:

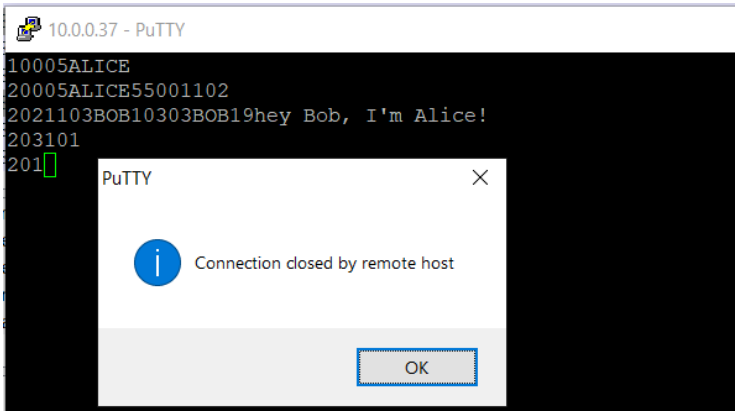


ההודעה של בוב דרך "סוקט 300" והעזיבה שלו נראים כך (מתקבל מהשרת דרך פורט 55001):

10.0.0.37	10.0.0.239	TCP	96	55001 → 55557 [PSH, ACK] Seq=3549408050 Ack=3147490411 Win=131328 Len=42
10.0.0.239	10.0.0.37	TCP	54	55557 → 55001 [ACK] Seq=3147490411 Ack=3549408092 Win=131328 Len=0
10.0.0.37	10.0.0.239	TCP	62	55001 → 55557 [PSH, ACK] Seq=3549408092 Ack=3147490411 Win=131328 Len=8
10.0.0.239	10.0.0.37	TCP	54	55557 → 55001 [ACK] Seq=3147490411 Ack=3549408100 Win=131328 Len=0

גם פקטות אלה הם המשך ישיר של התמונה הקודמת.

בסוף, אליס לא מסוגלת להתמודד עם הבושה ועוזבת את הצ'אט. רגע לפני הניתוק היא גם תקבל הודעת אישור עזיבה – הכל, לפי הפרוטוקול:



וזהו. מתרחשת סגירה של שני הסוקטים וכך זה נראה (שוב, המשך ישיר של הפקטות מתמונה קודמת):

44858	1515.615668	10.0.0.239	10.0.0.37	TCP	57	55507 → 13337 [PSH, ACK] Seq=2031769942 Ack=3173468144 Win=131328 Len=3
44859	1515.615695	10.0.0.239	10.0.0.37	TCP	56	55507 → 13337 [PSH, ACK] Seq=2031769945 Ack=3173468144 Win=131328 Len=2
44860	1515.616097	10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [ACK] Seq=3173468144 Ack=2031769947 Win=1051136 Len=0
44861	1515.617017	10.0.0.37	10.0.0.239	TCP	60	55001 → 55557 [FIN, ACK] Seq=3549408100 Ack=3147490411 Win=131328 Len=0
44862	1515.617056	10.0.0.239	10.0.0.37	TCP	54	55557 → 55001 [ACK] Seq=3147490411 Ack=3549408101 Win=131328 Len=0
44863	1515.617159	10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [PSH, ACK] Seq=3173468144 Ack=2031769947 Win=1051136 Len=3
44864	1515.617240	10.0.0.239	10.0.0.37	TCP	54	55557 → 55001 [FIN, ACK] Seq=3147490411 Ack=3549408101 Win=131328 Len=0
44865	1515.617606	10.0.0.37	10.0.0.239	TCP	60	55001 → 55557 [ACK] Seq=3549408101 Ack=3147490412 Win=131328 Len=0
44866	1515.656446	10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [ACK] Seq=2031769947 Ack=3173468147 Win=1051136 Len=0
46947	1594.385246	10.0.0.239	10.0.0.37	TCP	54	55507 → 13337 [FIN, ACK] Seq=2031769947 Ack=3173468147 Win=1051136 Len=0
46948	1594.385619	10.0.0.37	10.0.0.239	TCP	60	13337 → 55507 [ACK] Seq=3173468147 Ack=2031769948 Win=1051136 Len=0

סוף 😊

קובץ ההקלטה מצורף בנפרד.

תודה לחברת וואצאפ וחברת זום על ההשראה לעיצוב הצ'אט.