
פרויקט גמר

כיתוח תעבורת בפרוטוקול TCP/IP

מוגש על ידי: לירן דגן

תעודת זהות: 215609397

שם מרצה: ד"ר קוזקוב אנדריי

קבוצת הרצאה: 61305-3

תוכן עניינים

3	חלק ראשון
3.....	מבוא
3.....	יצירת קובץ ה-CSV
6.....	תיאור והסביר אריזת הפאקטות
10.....	ניתוח התעבורה באמצעות WIRESHARK
15	חלק שני
15.....	מבוא
16.....	ארכיטקטורת הפרוייקט
17.....	6.1 אפליקציית השרת
21.....	6.2 אפליקציית הלקוון
24.....	הוראות התקינה והרצה
30.....	דוגמאות קלט ופלט

חלק ראשון

אריזת נתונים ולבידת מנוגות ב-Wireshark

1. מבוא

בחלק הראשון של הפרויקט נדרשים אנו להכין קובץ CSV ובו עמודות שונות לניטוח בסיסי של מידע ברמת היישום. מטרת העל של הקובץ היא להציג הודעות שמקבל המחשב מאפליקציות אחרות שונות בהתאם ל프וטוקול יישומי נבחר. (כדוגמת HTTP, DNS, SMTP)

הודעות אלו הן כאמור מתקבלות על גבי פרוטוקול בשכבה האפליקציה/יישום, ולשם הפרויקט בחרתי ליבא הודעות שהועברו בפרוטוקול DNS (Domain Name System).

2. יצירת קובץ ה-CSV

לקובץ ה-CSV שמצוף לו"ח הוספה שדota של פאקטות, בין היתר סוג החיבור (TCP/UDP), גודל הפאקטה וט. Request/Reply

במובן שהקובץ מכיל גם את השדות להם נדרשו בתיאור הפרויקט והם מכילים:

- msg_id - מספר סידורי המיחיד כל הודעה
- app_protocol - פרוטוקול ברמת האפליקציה אשר בחרנו הלו הוא DNS
- src_port – מספר הפורט ממנו יצאתה הפאקטה מצד השולח
- dst_port – מספר הפורט אליו הגיעו הפאקטה מצד המקלט
- time_stamp – הרגע שבו הגיעו הפאקטה
- message – הודעה שנרצה להפיק מהפאקטה

הקובץ נוצר בשלהותו באמצעות לכידת תעבורה בתוכנת Wireshark והנתונים שבו הם למעשה פאקטות שהוקלטו ממכשירים ואפליקציות שהיו פועלים על המחשב שלי בעת ההקלטה. (אפשר למשל למצוא בקובץ התעבורת שהתקבלה הודעה משרת DNS של Google ושל Discord, שרצה על המחשב שלי בעת ההקלטה).

כדי לקבל את התעבורת שרציתי עבור פרוטוקול DNS השתמשתי בפילטר 'asap' כדי לסוג את הפאקטות הרלוונטיות. עבר כל שדה בפאקטה שרציתי להוסיף בעמודה ביצעתו לחיצה ימנית על העכבר ולחצתי על "apply as column". בך קיבלתי את כל השדות שרציתילייצא עבורי כל פאקטה. לאחר שאספתי מספיק תעבורת עצות את ההקלטה, לחצתי על תפריט File, ולאחר מכן על Export packet dissections ולאחר מכן על CSV as. בך יצאתי את הטבלה שנוצרה ב-Wireshark אל תוכנת Excel ומשם שמרתי את הטבלה בתור קובץ CSV שנשמר מקומית על המחשב.

עבור השדה message – הרכבתי אותו על ידי שרשור שdots:

(domain name | | is Response | | Transaction ID)

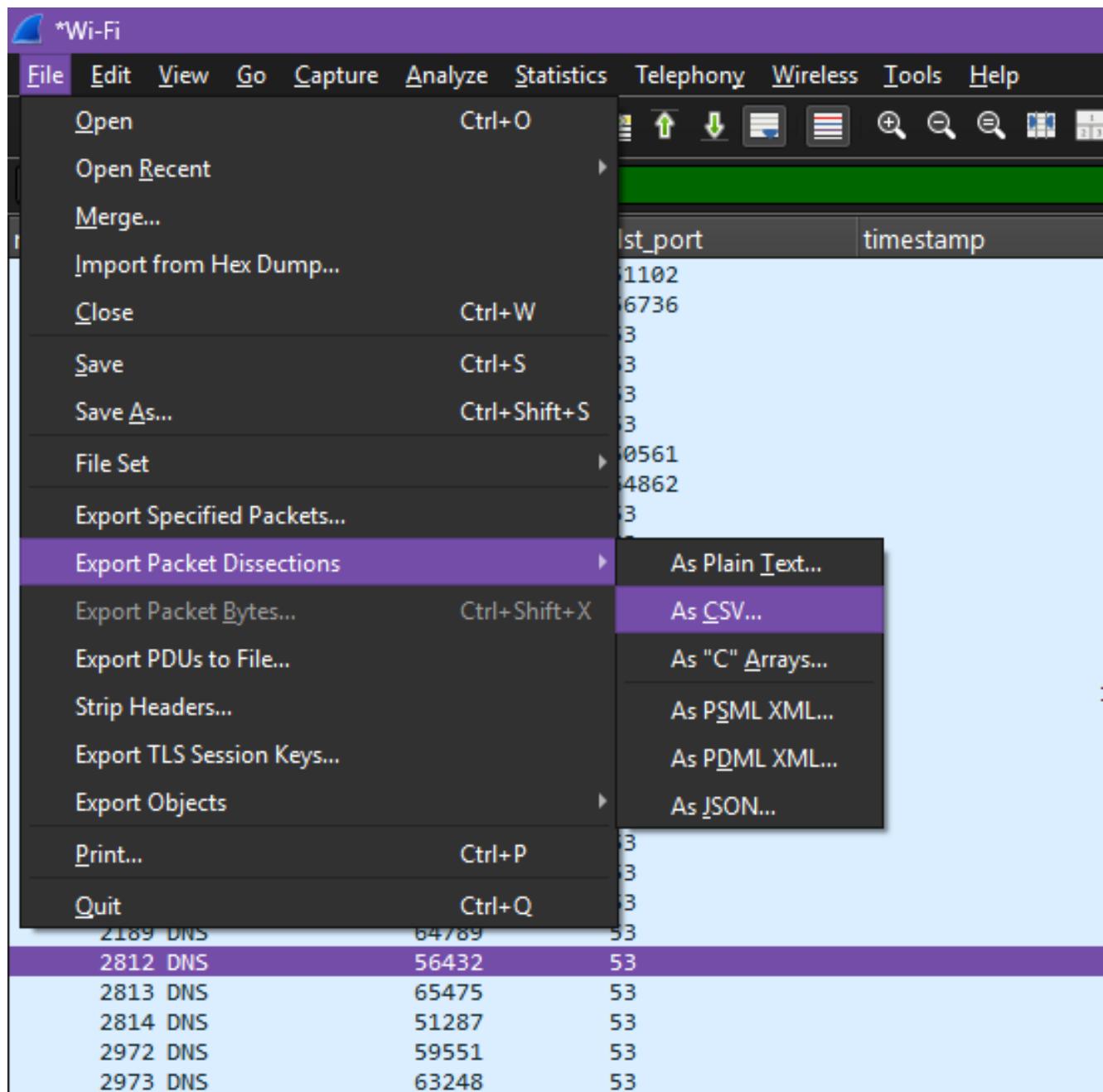
שאלן בולם שודות שמופיעים את פרוטוקול DNS. Domain name משמעו שם הכתובת של האפליקציה, is משמעו האם הפקטה היא בקשה לשרת (אפס) או תגובה של השרת (אחד).

ID Transaction משמעו מספר סידורי של פרוטוקול DNS עבור השאלה.

יצירת השדה message

Fields: dns.qry.name dns.flags.response dns.id		
stination Address	timestamp	message
1.1.1	0.000108	dns.google,0,0x03d8
1.1.1	0.000140	dns.google,0,0x0957
1.1.1	0.000336	dns.google,0,0xb42
1.1.1	0.000071	dns.google,0,0x162f
1.1.1	16.693004	dns.google,0,0x18d5
1.1.1	0.000361	dns.google,0,0x2100
1.1.1	0.895141	dns.google,0,0x3072
1.1.1	6.888908	dns.google,0,0x368f
1.1.1	29.761779	dns.google,0,0x3abc
1.1.1	1.014845	dns.google,0,0x41f8
	2.948365	dns.google,0,0x4261
	0.000235	dns.google,0,0x5954
1.1.1	0.000210	dns.google,0,0x60af
1.1.1	0.000120	dns.google,0,0x6d94
1.1.1	0.000145	dns.google,0,0x6f9b
	0.000129	dns.google,0,0x86f9
1.1.1	0.000216	dns.google,0,0x940d
	0.000334	dns.google,0,0x9473
	0.000388	dns.google,0,0x9cb5
	0.000256	dns.google,0,0xa3fa
	1.012652	dns.google,0,0xa7a7
1.1.1	0.000329	dns.google,0,0xe57d
1.1.1	0.882902	dns.google,0,0xf405
2.168.1.18	0.008372	dns.google,1,0x0957
2.168.1.18	0.000000	dns.google,1,0x162f
2.168.1.18	0.006692	dns.google,1,0x2100
2.168.1.18	0.001023	dns.google,1,0x368f
2.168.1.18	0.029216	dns.google,1,0x3abc
2.168.1.18	0.001168	dns.google,1,0x41f8
2.168.1.18	0.026362	dns.google,1,0x60af
2.168.1.18	0.066347	dns.google,1,0x6d94
2.168.1.18	0.006016	dns.google,1,0x6f9b
2.168.1.18	0.011840	dns.google,1,0x940d
2.168.1.18	0.001820	dns.google,1,0xe57d
2.168.1.18	0.020594	dns.google,1,0xf405
	14.580840	dns.msftncsi.com,0,0x3fc4
	0.100803	dns.msftncsi.com,0,0x3fc4
1.1.1	1.011470	dns.msftncsi.com,0,0x3fc4
	8.962245	

יצוא הפקטוט אל קובץ CSV



3. תיאור והסבר אריזת הפקטוות

השלב השני בחלק זה הוא טיענת קובץ CSV אל מחברת הג'ופיטר המצורפת שבא קוד פיתון לייצור תובורה מקומית של הודעות שנלקחו מה-CSV.

רעיוןינו אנחנו משתמשים בכימוס (Encapsulation) במודל OSI כדי לאזרז את הפקטוות ולהעבירן. האריזה מרכיבה את הפקטה מהשכבה העליונה (האפליקציה) עד לשכבות התחתונות. נתאר באופן כללי את תהליך האריזה ולאחר מכן נפרט.

תחילה ההודעה נוצרת בשכבת האפליקציה (שלוחים והודעה, טקסט שנבחר למסור) ולאחר מכן מועברת אל שכבת התעבורה. במחברת, האפליקציה שולחת את הודעות ה-CSV שלנו (או הודעת דמה). ככלומר המחברת שולחת כל הודעה משירות ה-CSV אל שכבת התעבורה.

שכבת התעבורה עוטפת את הודעתה בתווך SEGMENT לפענוח על ידי שכבת התעבורה מצד המקלט. פרוטוקול TCP משייר לפאקטה כותרת TCP Header שבה הוא מכיל:

Checksum source port, destination port, flags שימושים נוספים. בין הערבים נוסף שדה sum שמשמש את הצד המקלט לפענוח שגיאות. שכבת התעבורה מעבירה את הפקטוות לשכבת הרשות. עד בה הודעת ה-CSV שלנו נעטפה בסגמנט של שכבת התעבורה.

שכבת הרשות עוטפת את הסגמנט בתווך Datagram IP. ככלומר הוא מוסיף כותרת IP Shmcilla: source IP, destination IP, version, TTL משדרת ה-CSV שלנו נעטפה בסגמנט, שנעטף כותרת IP. הפאקטה לשכבת הקו. עד בה הודעת ה-CSV שלנו נעטפה כותרת הקו.

רעיוןינו שכבת הקו מוסיפה מסגרת לפאקטה והיא השכבה האחורונה שעוטפת אותה לפני שהיא משודרת להלה. היא מוסיפה בתובות MAC (מקור ויעד), פרוטוקול רשות (6 bytes) ומעבירה את הפאקטה לשכבה הפייזית, ככלומר משדרת אותה מהמכשור אל החוז. בפועל, במקרה שלנו התהילכים שתעשה המחברת הם לוקאים ולבן שכבת הקו לא Tosip בתובות MAC או מידע חדש אבל כן יש לה תפקיד בכך שהוא תshedר את הפקטוות אל לולאת-loopback, ככלומר מקומית בתחום המחשב.

אלו הצעדים הרעיוניים, מכאן והילך נסביר במפורט איך נוצרים כותרות ה-TCP_Header ו-

למעשה עד כאן פירטנו את הרעיון

שבאריזת המנות ומעבשו נפרט מה בפועל

כל שכבה עשויה בהתאם לקוד שכותב

במחברת. ככלומר ננתח את הקוד של

הפונקציות build_tcp_header ו-

build_ip_header. מדובר בمعنى העמeka

רחבה הרבה יותר... נתחייב לשכבת

האפליקציה וננד לשכבת הרשות.

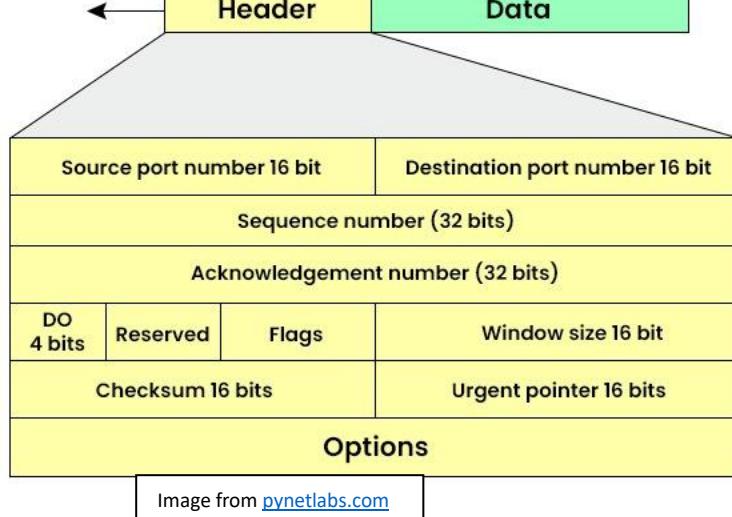
בשכבת האפליקציה תחילת אנחנו טוענים

את ה-CSV המקורי שלנו למחברת עם

שימוש בספריית pandas.

במחברת מוגדרות פונקציות לבניה "ידנית"

של כותרת TCP header של שכבת



התעבורה ולבניית כוורתת header-IP של שכבת הרשת, לשימוש מידת הצורך בהתאם למערכת הפעלה. לאחר מכן המחברת טעונה את ההודעה עבור כל שורה ב-CSV ושולחת אותה לוקאלית עם פונקציית `send`.

כדי לבנות את כוורתת-TCP אנחנו קודם כל בונים כוורתת `tcp_header` ראשונית שמכילה רק חלק מהשדות. הסיבה היא ששדה `checksum` משמש לביקורת המידע נושא על שאר השדות, וכך לחשב אותו נדרש להרכיב את שאר השדות לפני.

אז, בהינתן:

- מספרי הפורט של המקור ושל היעד
- מספר סידורי (number) שסופר כל בית שעובד (ומאותחל רנדומלית אם לא אותחל עידיון)
- מספר סידורי שמאשר את קבלת הנתונים עד הביטח הנוכחי (ack number)
- סוג הפקטה (flags), כולל רצף ביטים התואמים לדגלי Syn, Ack, Fin ודגלים נוספים המתארים את הפעולות שהפקטה אחראית להשלים.
- גודל החלון (Window) שמשמש לדעת כמה ביטים ניתן לשלוח ברצף פאקטות אחד ברגע זה למקבל.
- `Data Offset,Reserved`, שדות שימושיים להגדרת אורך הפקטה ולשמירת 4 ביטים נוספים.
- `urgentPointer`, שדה שמשמש לציין ביטים מסוימים שדחופים לעיבוד והוא מצביע על סוף רצף הביטים האלו. לנו אין נתונים שדחופים לעיבוד ולכן השדה ישאיר אפס.

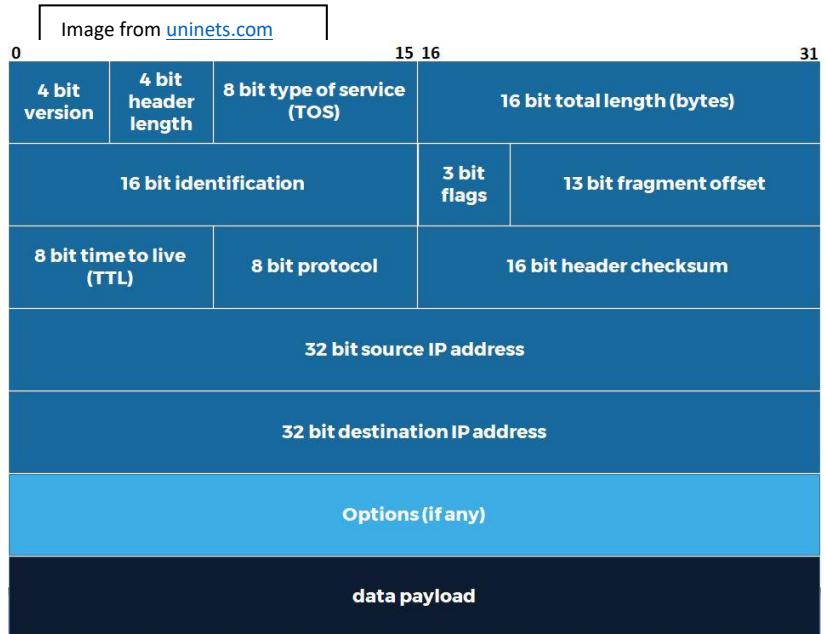
לאחר שאספנו את כל הנתונים אנחנו אורזים אותם לתוך כוורתת `tcp_header` ראשונית.

לאחר מכן אנחנו בונים `pseudo_header` עם חלק מהשדות שנכנסים לכוורתת-IP. (כמו בתובות ה-`checksum`). רק לאחר מכן מחשבים את שדה `checksum`.

- `Checksum`, שדה זה משמש את מקבל כדי לאמת את המידע שבפקטה. הוא עושה זאת על ידי חיבור כל השדות בפקטהividually עם שדות של `pseudo_header`. החיבור הוא של כל 16 ביטים. לאחר קבלת וצאת החיבור הופכים את הביטים זהה `checksum`. אם מקבל מבצע את אותו תהליך ומתקבל תוצאה שונה ממה ששמור בשדה זה הוא יגלה שהמידע שבפקטה השתבש במהלך העברה, וידע לזרוק את הפקטה או לבקש שליחת מחודשת.

רק לאחר שיחסבנו את שדה `checksum` אנחנו אורזים מחדש את `tcp_header` שלנו, הפעם עם שדה `checksum` ובכך השלמנו את אריזת `segment` של שכבת התעבורה.

כדי להסביר את שכבת הרשת ובניית כוורתת-IPv4, נשים לב למבנה של ה-Header:



בנויות בותרת ה-IP של שכבות הרשות. גם כאן נבנה header התחלתי כדי לבצע את חישוב הchecksum לאחר יותר.

בහינתן:

- כתובות ה-IP של המקור ושל היעד
- גודל הדאטה שהפאקטה נושא (בלומר – ה-payload שקיבלנו בשכבות האפליקציה וגודל הסגמנט שנוסף לה בשכבות התגובה).
- פרוטוקול התגובה (TCP)

אנחנו בונים את ה-header כז':

- אצלנו הגרסה היא 4 (4qv4) ולכן `version=4`.
- header_length הוא גודל ה-Header_IP שאנחנו בונים עבשו. הגודל נמדד במילימ' ולא בבייטים. לכן אם הגודל הוא למשל 5 (밀ימ'), הגודל בפועל הוא $5 \cdot 4 = 20$ בייטים.
- TOS הוא שדה שנועד לציין לצד המქבל כיצד לטפל בפאקטה עם רצפי ביטים מיוחדים כלפי (בדרכם כל ציון דרישות דילוי, תפוקה, אמינות מידע וכו'). לנו אין דרישות מיוחדות עבור הפאקטות ולכן השדה ישאר אפס.
- total_length הוא הגודל של הפאקטה כולה, כלומר הגודל header_length בבייטים פלוס הגודל של ההודעה והסגמנט שקיבלנו משכבות האפליקציה והתגובה.
- identification השדה זהה משמש את הדאטה להתפצל למקטעים. למשל, במידה והפאקטה גדולה מדי עבור הקישור שצריך לשדר אותה החוצה, שכבת הרשות תפצל את הדאטה למקטעים נפרדים, עם IP_headers שונים אך עם שדות identification זהים.
- המקטעים ישלחו בנפרד לקישור, וביעד (שכבת הרשות של הצד המქבל) יתאפסו חזרה אל הפאקטה המקוריות שהם מרכיבים בלבד. הם ידועו להתאסף חזרה אל הפאקטה המקוריות לפי השדה identification המשותף שהם מחזקים.
- השדות header_flags, fragment_offset הם ביטים שמצוינים האם יש לפצל את הדאטה למקטעים ואם כן, מה המיקום של כל מקטע ביחס לפאקטה המקורי.

- השדה ttl (Time To Live) מגדיר את מספר רכיבי הרשת המxisימי שהפאקטה יכולה לעבור דרכם לפני שתיזרק. בך השדה מגדיר את תוחלת החיים של הפאקטה.

לאחר שהגדכנו את כל השדות הללו אורזים header_IP ראשוני. לאחר מכן אנחנו מחשבים את שדה checksum על ידי חיבור כל השדות של header_IP (ב-16 ביטים) והפיכת הביטים של הסכום.

לאחר מכן אנחנו אורזים מחדש את header_IP ביחד עם שדה checksum ובכך השלמנו את האזיה של שכבת הרשת.

בכך השלמנו את האזיה של שלושת השכבות, אפליקציה (הודעות CSV שהמחברת טוענת ושולחת), התעבורה (הסמנט שנבנה, tcp_header) והרשת (ה-ip datagram שנבנה, ip_header). בך כל תעבורה שנמצאה מהמחברת תקרה לאחר Encapsulation של שלושת השכבות האל. התחלנו מرمת האפליקציה וארכזו את ההודעות בשכבות עד ליצירת פאקטות שנושאות את ההודעות.

4. ניתוח התעבורה באמצעות WIRESHARK

בשלב הראשון ננתח תעבורה ראשונית שהתקבלה מהפעלת הפונקציה:

```
[30]: def demo_send(num_packets: int=3, delay_sec: float=1.0, flags: int=0x02):
    for i in range(num_packets):
        payload = f'Hello Packet {i}'.encode()
        transport.send(payload, flags=flags)
        time.sleep(delay_sec)
```

עבור כל אחת מהפקודות:

```
demo_send(num_packets=3, delay_sec=1.0, flags=0x02)
demo_send(num_packets=3, flags=0x18)
demo_send(num_packets=3, flags=0x10)
demo_send(num_packets=3, flags=0x01)
demo_send(num_packets=3, flags=0x04)
```

aczin שמספר הפורט 25215 הוא פורט שהוקצה למשתמש (אני, שה:right את Wireshark) על ידי מערכת הפעלה ומספר הפורט 12345 הוא מספר הפורט של האפליקציה (מחברת ה-Jupyter-h).

- `demo_send(num_packets = 3, delay_sec=1.0, flags=0x02)`

זו הגדרה לפאקטת TCP שבה רק הדגל SYN פועל. במיילים אחרים זו פאקטה שמהוות את השלב הראשון בתהליך 3-way-handshake. אבל לפאקטה אין בתובות MADEINA, במיילים אחרים אין כאן באמת חיבור TCP בין שני צדדים, אין תקשורת sockets אלא העברת סתמיית של פאקטות בין בתובות פורט בלבד שהוקצתה על ידי מערכת הפעלה לבין המחברת. וכן נצפה שלאחר הפאקטה הריאונה, תישלח פאקטה שתבהיר על Reset, כי אין אפשרות להתחילה חיבור. ובאמת ניתן לראות שלאחר כל פאקטה כזו שנשלחת למחברת (ומזהות על ידי 0 Hello Packet 1 או Hello Packet 2 או Hello Packet 2 בשדה ה-payload (TCP payload) מקבלים מהמחברת בתגובה פאקטת TCP שבה

.Ack=1, Reset=1

No.	Source Port	Destination Port	Time	Protocol	Syn	Acknowledgment	Push	Reset	Fin	Flags	TCP payload
349	25215	12345	11.084745	TCP	1	0	0	0	0	0x0002	48656c6c6f205061636b65742030
350	12345	25215	11.084795	TCP	0	1	0	1	0	0x0014	
355	25215	12345	12.088255	TCP	1	0	0	0	0	0x0002	48656c6c6f205061636b65742031
356	12345	25215	12.088322	TCP	0	1	0	1	0	0x0014	
374	25215	12345	13.092120	TCP	1	0	0	0	0	0x0002	48656c6c6f205061636b65742032
375	12345	25215	13.092247	TCP	0	1	0	1	0	0x0014	

0000	02 00 00 00 45 00 00 36	00 01 00 00 40 06 7c bf	... E 6 ... @
0010	7f 00 00 01 7f 00 00 01	62 7f 30 39 00 00 00 00 b.09....
0020	00 00 00 00 50 02 20 00	a1 b6 00 00 48 65 6c 6c	... P. Hell
0030	6f 20 50 61 63 6b 65 74	20 30	o Packet 0

- `demo_send(num_packets=3, flags=0x18)`

זו הגדרה לפאקטת TCP שבה רק הדגלים ack,push,reset מפעילים. כמו מקודם אין באמת תקשורת בין הצדדים וכן נצפה לקבל חזרה Reset=1. הפעם נשים לב שנקבל 0. ack=0. הסיבה היא שהמחברת לא קיבלה לפני כן פאקטת SYN מהמשתמש ולכן היא לא מודעת או לא חושדת בקיום קשר.

20313	25215	12345	2116.209752	TCP	0	1	1	0	0	0x0018	48656c6c6f205061636b65742030
20314	12345	25215	2116.209836	TCP	0	0	0	1	0	0x0004	
20325	25215	12345	2117.214729	TCP	0	1	1	0	0	0x0018	48656c6c6f205061636b65742031
20326	12345	25215	2117.214837	TCP	0	0	0	1	0	0x0004	
20331	25215	12345	2118.217473	TCP	0	1	1	0	0	0x0018	48656c6c6f205061636b65742032
- 20332	12345	25215	2118.217552	TCP	0	0	0	1	0	0x0004	

שאר הפקודות מתנהגות באופן דומה למעט האחרון:

demo_send(num_packets=3, flags=0x04) •

זו פקודה שבה רק הדגל Reset פעועל. בשפaket זה נשלחת אפיו לא נקלט פאקטות בתגובה מהמחברת. הסיבה היא שפaket מהסוג זה לא מצפה לתגובה בחזרה אלא מבקשת מהמחברת לסגור את החיבור הנוכחי. אבל למעשה אין חיבור אמיתי ולכן לא נקלט תגובה מהמחברת.

Sequence Number: 0 (relative sequence number)	0000 02 00 00 00 45 00 00 36 00 01 00 00 40 06 7c bf	... E 6 ... @ .
Sequence Number (raw): 0	0010 7f 00 00 01 7f 00 00 01 62 7f 30 39 00 00 00 00 b.09.....
[Next Sequence Number: 14]	0020 00 00 00 00 50 04 20 00 a1 b4 00 00 48 65 6c 6cP.Hello
Acknowledgment Number: 0	0030 6f 20 50 61 63 6b 65 74 20 30	o Packet 0
Acknowledgment number (raw)		

התמונה הבאה מדגימה כיצד נראה הסגמנט של הפאקטוט בפי שמייצגות WIRESHARK.

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 25215, Dst Port: 12345, Seq: 0, Len: 14

Source Port: 25215
Destination Port: 12345 **PORTS**

[Stream index: 21]
[Stream Packet Number: 1]

[Conversation completeness: Incomplete (45)]
[TCP Segment Len: 14]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 0
[Next Sequence Number: 15 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
0101 = Header Length: 20 bytes (5)

Flags: 0x002 (SYN)

000. = Reserved: Not set
...0 = Accurate ECN: Not set
.... 0.... = Congestion Window Reduced: Not set
.... .0.... = ECN-Echo: Not set
.... ..0.... = Urgent: Not set
.... ...0.... = Acknowledgment: Not set
....0.... = Push: Not set
....0.... = Reset: Not set
....1.... = Syn: Set
....0.... = Fin: Not set
[TCP Flags:S.]

Window: 8192
[Calculated window size: 8192]

Checksum: 0xa1b6 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0

[Timestamps]
[Client Contiguous Streams: 0]
[Server Contiguous Streams: 1]
TCP payload (14 bytes)

Data (14 bytes)
Data: 48656c6c6f205061636b65742030
[Length: 14]

FLAGS

ניתן לראות גם את ה-TCP Payload (בתמונה – “Hello World 0” – TCP Payload

0000 02 00 00 00 45 00 00 36 00 01 00 00 40 06 7c bf	... E 6 ... @ .
0010 7f 00 00 01 7f 00 00 01 62 7f 30 39 00 00 00 00 b.09.....
0020 00 00 00 00 50 02 20 00 a1 b6 00 00 48 65 6c 6cP.Hello
0030 6f 20 50 61 63 6b 65 74 20 30	o Packet 0

בשלב השני ננתה תובורה ממחברת ה-CSV שהתקבלה בתוצאה מהעברת ה-CSV:

Send Messages from CSV file

Iterate over the rows and send message by message

```
[43]: #Send messages from CSV file
for index, row in messages_df.iterrows():
    # Extract message details from the DataFrame row
    message = row['message']
    message = f"test message {index}" if not message else message
    # Send the message using the RawTcpTransport class
    # (You may need to adjust flags and other parameters as needed)

    #TODO: uncomment the line below to send the messages
    transport.send(message.encode(), flags=0x18) # Example with PSH+ACK flags

    time.sleep(0.1) # Optional delay between messages
```

למעשה שלוחים פאקטות TCP עם הדגמים ack=1,push=1. ניתחנו מוקדם את התנagoות התובורה של פאקטות אלו. ההבדל הוא שבאן אנחנו מחלכים מקובץ ה-CSV שטרפנו בכל פעם את שדה ה-`message` ושולחים אותו. אבן ניתן לראות שככל הפאקטות שמתקבלות בעלות בעלות בוטרת `message`

```
[Stream index: 0]
└ Transmission Control Protocol, Src Port: 25215, Dst Port: 12345, Seq: 1, Ack: 1, Len: 32
  Source Port: 25215
  Destination Port: 12345
  [Stream index: 0]
  [Stream Packet Number: 17]
  ↳ [Conversation completeness: Incomplete (40)]
  ↳ [TCP Segment Len: 32]
  Sequence Number: 1      (relative sequence number)
  Sequence Number (raw): 0
  [Next Sequence Number: 33      (relative sequence number)]
  Acknowledgment Number: 1      (relative ack number)
  Acknowledgment number (raw): 0
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x018 (PSH, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    ....0.... .... = Congestion Window Reduced: Not set
    ....0... .... = ECN-Echo: Not set
    ....0. .... = Urgent: Not set
    ....1.... = Acknowledgment: Set
    ....1... .... = Push: Set
    ....0... .... = Reset: Not set
    ....0....0. = Syn: Not set
    ....0....0. = Fin: Not set
    [TCP Flags: .....AP....]
  Window: 8192
```

קובץ הלכידה לאחר שליחת הودעות CSV בטור המחברת נראה כך:

No.	Source Port	Destination Port	Time	Protocol	Syn	Acknowledgment	Push	Reset	Fin	Flags	TCP payload
1	25215	12345	0.000000	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307862626430
2	12345	25215	0.000051	TCP	0	0	0	1	0	0x0004	
3	25215	12345	0.104457	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307865346462
4	12345	25215	0.104562	TCP	0	0	0	1	0	0x0004	
5	25215	12345	0.209886	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307862626430
6	12345	25215	0.209951	TCP	0	0	0	1	0	0x0004	
7	25215	12345	0.315308	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307865346462
8	12345	25215	0.315410	TCP	0	0	0	1	0	0x0004	
9	25215	12345	0.419121	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307862626430
10	12345	25215	0.419220	TCP	0	0	0	1	0	0x0004	
11	25215	12345	0.524244	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c302c307865346462
12	12345	25215	0.524342	TCP	0	0	0	1	0	0x0004	
13	25215	12345	0.631054	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c312c307865346462
14	12345	25215	0.631209	TCP	0	0	0	1	0	0x0004	
15	25215	12345	0.734618	TCP	0	1	1	0	0	0x0018	697076362e6d736674636f6e6e656374746573742e636f6d2c312c307862626430
16	12345	25215	0.734695	TCP	0	0	0	1	0	0x0004	
17	25215	12345	0.840747	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307835633234
18	12345	25215	0.840873	TCP	0	0	0	1	0	0x0004	
19	25215	12345	0.945010	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307866636531
20	12345	25215	0.945091	TCP	0	0	0	1	0	0x0004	
21	25215	12345	1.048312	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307835633234
22	12345	25215	1.048476	TCP	0	0	0	1	0	0x0004	
23	25215	12345	1.154639	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307866636531
24	12345	25215	1.154743	TCP	0	0	0	1	0	0x0004	
25	25215	12345	1.260281	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307866636531
26	12345	25215	1.260364	TCP	0	0	0	1	0	0x0004	
27	25215	12345	1.365673	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c302c307835633234
28	12345	25215	1.365783	TCP	0	0	0	1	0	0x0004	
29	25215	12345	1.470111	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c312c307866636531
30	12345	25215	1.470203	TCP	0	0	0	1	0	0x0004	
31	25215	12345	1.575103	TCP	0	1	1	0	0	0x0018	7777772e6d736674636f6e6e656374746573742e636f6d2c312c307835633234
32	12345	25215	1.575235	TCP	0	0	0	1	0	0x0004	
33	25215	12345	1.680408	TCP	0	1	1	0	0	0x0018	646973636f72642e636f6d2c302c307838306634
34	12345	25215	1.680546	TCP	0	0	0	1	0	0x0004	

ונימן לראות שכל פאקטה בעלת TCP Payload שאינו ריק נושא את שדה message של השורה המתאימה בקובץ CSV. למשל, הפאקטה הראשונה נושא את ההודעה:

0000	02 00 00 00 45 00 00 49	00 01 00 00 40 06 7c ac	E I @
0010	7f 00 00 01 7f 00 00 01	62 7f 30 39 00 00 00 00 b 09
0020	00 00 00 00 50 18 20 00	de f8 00 00 69 70 76 36 P ipv6
0030	2e 6d 73 66 74 63 6f 6e	6e 65 63 74 74 65 73 74	.msftcon necttest
0040	2e 63 6f 6d 2c 30 2c 30	78 62 62 64 30	.com,0,0 xbbd0

ושורת CSV הראשונה נושא את שדה message:

L	K	J	I	H	G	F	E	D	C	B	A
Connection Protocol	Response	Type	Length	message	timestamp	Destination Address	Source Address	dst_port	src_port	app_protocol	msg_id
	Message is a query	A	104	ipv6.msftconnecttest.com,0,0xbbd0	53	64155	DNS	1 2

הפאקטה ושורת CSV שתיהן נושאות את אותה ההודעה:

.ipv6.msftconnecttest.com,0,0xbbd0

ולכן שורה מס' 1 בקובץ CSV מתאימה לפאקטה מס' 1 בקובץ הלכידה.

דוגמה נוספת, בקובץ הלכידה (שמצויר לדו"ח), פאקטה מס' 63 נשאת את ה-Payload

0000	02 00 00 00 45 00 00 3b 00 01 00 00 40 06 7c ba	
0010	7f 00 00 01 7f 00 00 01 62 7f 30 39 00 00 00 00	
0020	00 00 00 00 50 18 20 00 7d 2c 00 00 64 6e 73 2e	
0030	67 6f 6f 67 6c 65 2c 31 2c 30 78 36 30 61 66	dns.google,1,0x60af

שורה ה-CSV מס' 32 נשאת את שדה ה-message:

Protocol	Type	Source IP	Destination IP	Port	Timestamp	Size	Protocol	Action
UDP	Message is a response	AAAA	130	dns.google,1,0x60af	0.026362	192.168.1.18	1.1.1.1	59365 53 DNS 32

הפקטה ושורת ה-CSV שתיהן נשאות את ההודעה:

dns.google,1,0x60af

ולכן שורה מס' 32 בקובץ ה-CSV מתאימה לפאקטה מס' 63 בקובץ הלכידה.

באופן דומה ניתן לזהות כל הודעה בקובץ CSV גם בקובץ הלכידה.

חלק שני

פיתוח יישום רשת ללקוח/שרת ב프וטוקול TCP

5. מבוא

נדרשו לבתוב פרויקט המימוש צ'אט רב משתמשים שבו קיימים לקוחות שמתקשרים אחד עם השני ושרת שמנהל את החלפת המסרים ביניהם. ככלומר, בהינתן השרת, לקוח א' יציין את שם הלקוח ב' אליו ירצה לפנות, והשרת ידאג להעביר את המסרים של לקוח א' אל לקוח ב'. בנוסף השרת יעביר ללקוח הודעות שקיבל מלקוחות אחרים שכן התקשרות בין הלקוחות אמורה להיות רב-כיוונית; ללקוח אמורא להיות היכולת גם לשלוח הודעות לכל לקוח אחר וגם לקבל הודעות מכל לקוח אחר. לדע"ח זה מצורפים שתי אפליקציות שמרכיבות את היישום, האחת היא עבור הלקוח ששלוח בקשות לשרת ומתקבל מסרים מהשרת. השנייה היא עבור השרת שמנהל את רוב העבודה בתהילcis. הוא זה שמקבל קלטים ממספר לקוחות ויודיע להתנהל לפיהם, יודיע להעביר הודעות מלוקח פונה א' אל לקוח נמען ב' ומתקשר הוא עצמו עם לקוחו.

את היישום בחרתי לבתוב בשפת C++ ובסביבת Visual Studio 2022.

כאמור בנוסך לדע"ח מצורפים האפליקציות בתיקיות:

- Server_interface
- Client_interface
- Client_2
- Client_3
- Client_4
- Client_5

באשר server_interface היא אפליקציית השרת, client_interface היא אפליקציית הלקוח, ובכל השאר הם עותקים של אפליקציית הלקוח client_interface שנועדו לרוץ במקביל כדי לקיים שיח. אין הבדל בין התיקיות האלו.

ניתן להריץ את הפרויקט גם עם שניים, שלושה או ארבעה עותקים מתוך החמישה שצורפו. ניתן גם לשכפל את התיקיות ולהריץ הרבה יותר לקוחות מהחמשה שצורפו. הקורא יוכל להריץ כמה עותקים שיירצה, אך לשם הנוחות מצורפים חמישה עותקים, ניתן להשתמש בכלם או בחלקם לפי מידת הצורך.

כל קבצי הקוד בשתי האפליקציות server_interface, client_interface מלאוים בתיעודים ובהסבירם לנוחיות הקורא/ת.

6. ארכיטקטורת הפרויקט

כדי ליזור תקשורת בין יישומים בשפת C++ אנחנו משתמשים בספריית WinSock2 שמאפשרת לנו גישה לכל פעולות ה-Sockets להן נדרש כדי לאפשר תקשורת. כאשר מדובר בתקשורת מבוססת sockets אנחנו נדרשים לביצוע התהליכים טכניים לפני שנוכל לתקשר עם אפליקציות אחרות ולאחר מכן לתקשר עימן. התהליכים משתנים כמעט לגמרי בין הלקוח לשרת, ואלו הם:

עבור הלקוח נבצע:

- אתחול ספריית Winsock
- יצירת socket שישתמש את הלקוח להתחבר לשרת (פונקציית יצירת socket (socket))
- ציון בתובת השרת וחיבור בין socket הלקוח לבין השרת (פונקציית connect (connect))
- ניהול תקשורת עם השרת באמצעות שליחת וקבלת הודעות (recv, send)
- בסוף השימוש, קרי התנטקות של הלקוח מהשרת או יצא מהצ'אט, נסגור את ה-socket (פונקציית closesocket)
- קרייה לפונקציית cleanup של ספריית Winsock

עבור השרת נבצע:

- אתחול ספריית Winsock
- יצירת socket שיძין ללקוחות נכנסים (פונקציית socket (socket))
- קישורת ה-socket המאזין אל הכתובת שבה ניצב את השרת, ככלומר אל כתובות ה-ip ואל מספר הפורט שנרצה לו (פונקציית bind (Bind))
- רק לאחר שייצרנו את ה-socket וקשרנו אותו לכתובת אחרת, נתחליל בהازנה ללקוחות נכנסים.
- בעת האזנה, נקבל ללקוחות חדשים ונשמור אותם בסיס נתונים שיפורט בהמשך (פונקציית listen (listen))
- לאחר קבלת הלקוח ננהל תקשורת באמצעות קבלה ושליחת הודעות בין הלקוח לשרת (פונקציות recv, send) (recv, send). חשוב לציין שככל לקוח פועל בצורה א-סינכרונית. ככלומר הפעולות של לקוח אחד אינן תלויות בזמן הפעולה של לקוח שני. התהליכים של כל לקוח קורים במקביל.
- באופן עקרוני השרת אמר לעובוד תמיד (ריצה אינסופית) כי בלבדו הצ'אט לא קיים, בשונה מהלקוח שבן אמר להפסיק מהצ'אט בשלב מסוים (לא רץ אינסופית). עם זאת, אנחנו מתעסקים עם sockets ולבן למן הסדר הטוב נציין שיש לסגור את ה-socket בסוף השימוש. (פונקציית closesocket (closesocket)).
- בהמשך לסעיף הקודם, קרייה לפונקציית cleanup של ספריית Winsock

באמור הפרויקט מורכב משתי אפליקציות, ללקוח ושרת וכן נקייש פירוט לכל אחת.

6.1 – אפליקציית השירות Server_interface

תיקיה זו מורכבת מ-4 קבצי header שם מרוכדים הפונקציות עם תיעודים כלליים ו-4 קבצי cpp גם הם עם תיעודים מפורטים וביהם בפועל מיושמים התהיליכים. פירוט רחוב נמצא בתיעוד שבפרויקט, וכך גם כאן נציג את קבצי header.

header.h – בקובץ זה מוגדר struct עם פרטימ על הלוקוח, שבו אנחנו שומרים את השם של הלוקוח (השם ייחודי לכל לוקוח – דרישות הפרויקט מאפשרות לנו להניח זאת), ואת ה-socket של הלוקוח. נגידר גם פונקציה למציאת לוקוח בודד בתוך רשימה של לוקוחות בהתבסס על השם שלו. כפי שניתן להבין, בסיס הנתונים שבו נשמר את הלוקוחות יהיה רשיימה.

```
1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <iostream>
4  #include <WinSock2.h>
5  #include <string>
6  #include <list>
7
8  using namespace std;
9
10 // Each client has a name and a socket to which the server sends info to and receives info from
11 struct ClientInfo {
12     string name;
13     SOCKET socket;
14
15     bool operator==(const ClientInfo&);
16 };
17
18 ClientInfo* findByName(const string&, list<ClientInfo>&);
19
```

הלקוחות מתרחשות בנפרד בפונקציית ה-main שבקובץ server.h שיצג מיד.

שאנו צריכים לקיים כדי לאפשר תקשורת מבוססת sockets. בקובץ מופיעימוש בסיסי של אתחול ספריית Winsock, יצורת socket המאזין של השרת וקשרתו לכתובת השרת. ההאזנה וקבלת

```

1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <WinSock2.h>
4  #include <WS2tcpip.h>
5  #include <string>
6  #pragma comment(lib, "ws2_32.lib")
7
8  using namespace std;
9
10 /*
11 These are the steps the server handles to perform socket interactions:
12 1. Initialize winsock lib
13 2. Create the server socket
14 3. Bind ip (Which is 0.0.0.0) and port (which is 12345) to the socket
15 -----
16 4. Listen on the socket
17 5. Accept clients
18 -----
19 6. 'recv' for receiving messages from clients and 'send' for forwarding them to other clients
20 -----
21 7. Close the server socket when finished
22 8. Cleanup
23
24 The goal: Use 'recv' to get inputs from the users, Use 'send' to instruct users and
25 forward their messages onwards
26 */
27
28 // step 1: Initialize WinSock version 2.2
29 bool initialize();
30
31 // step 2: Create a listening TCP socket for the server to accept clients
32 bool createTCPsocket(SOCKET& listener);
33
34 // step 3.1: Create the server's Address details, ip and port
35 bool createAddress(sockaddr_in& serverAddr, const string& ip, const int portNumber);
36
37 // step 3.2: Bind our TCP listening socket to the address details we created
38 bool assignAddress(SOCKET& listenSocket, sockaddr_in& serverAddr);
39

```

– בקובץ זה מרכזות הפקציית שמהוות את הלוגיקה שבתקשרות עם כל לקוחות באשר אנחנו משתמשים בפקציית על interact בפקציית נוספת כדי לנהל את התהליכי מול הלקוחות.

```

1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <iostream>
4  #include <string>
5  #include <list>
6  #include "client_info.h"
7  using namespace std;
8
9  /* Given 2 strings, compare them case insensitively(for example "liRaN" == "LIRan") */
10 bool equalStrings(const string& str1, const string& str2);
11
12 /* Given a client's name and socket we create and save his info in a 'ClientInfo' struct and add him to
13 | a list containing all active chatters */
14 ClientInfo createClient(string& name, SOCKET& socket, list<ClientInfo>& all_clients);
15
16 /* Given any client's message to the server we first have to check if he entered 'quit'
17 | If he did, then we notify the client on his disconnection and return true
18 | If he didn't, we return false */
19 bool isQuit(string& message, ClientInfo& client);
20
21 /* Given a client's input of a target he wants to message, the server checks if the target
22 | is valid and sends feedback to the client. Errors to be detected are:
23 | 1. target is the client himself (logical error, client is trying to message himself).
24 | 2. target doesn't exist (technical error, the target can not be found)
25 | If there is no error we notify the client that we found the target, else we
26 | send the error to the client. */
27 bool isError(string& targetName, ClientInfo& client, list<ClientInfo>& all_clients);
28
29 /* If the client disconnected (due to 'quit' message or due to connection error)
30 | Then we discard his socket and remove him from the list of active chatters */
31 void deleteClient(ClientInfo& client, list<ClientInfo>& all_clients);
32
33
34 /* 'Interact' is the main function that manages the interactions between the server and the client.
35 | It Handles the server's actions with a single client and forwards messages from the client
36 | to a target client.
37 | If needed, the server sends feedback messages regarding errors.
38 | Interact utilizes all the functions from above.
39
40 In short, we do the following:
41 0. We got a message from the client
42 1. Check for 'quit' (isQuit), if the client didn't send quit we proceed to 2., else we stop running
43 and remove him (deleteClient).
44 2. Use a 'mode' variable that gets 0, 1 or 2 to distinguish between actions as followed:
45 mode = 0 -> We receive client's name and save his details (createClient)
46 mode = 1 -> we receive the target client and check for errors (isError)
47 mode = 2 -> We got a message to deliver and we send it to the target client by finding his socket in the clients list
48 3. return to 0.
49 */
50
51 void Interact(SOCKET clientSocket, list<ClientInfo>& all_clients);

```

– בקובץ זה פונקציית ה-main, בה אנחנו מבצעים את כל פעולות ה-socket המקידימות כדי לאפשר לשרת לתקשר עם הלוקחות. לעומת זאת, אנחנו מבצעים את כל הפעולות הקיימות בקובץ socket_setups.h ומוסיפים האזנה וקבלת ליקוחות.

כדי לנו להל את כל הלוקחות במקביל אנחנו צריכים לשמר מידע על כלם. לעומת זאת אנחנו צריכים לבנות מנגנון שיאפשר להוסיף, למחוק ולהפוך ליקוחות בתוך המאגר. בפרויקט שלנו נזדקק לבנייה נתונים שיאפשר להוסיף, למחוק ולהפוך ליקוחות בתוך המאגר. נוכל לבצע זאת עם המונט סוגים של מבנים. אפשר למשה לשמר עצ AVL מאוזן שמסדר את הלוקחות לפי סדר האלף-בית של שמותיהם, או להשתמש בדרכים מתוחכמת יותר לניהול המאגר. עם זאת, זו לא גולת הכוחות של הפרויקט הזה, שכן נפשט מעט את השימוש ואת ההסבירים. כאמור נשמר את הלוקחות בתוך רשיימה מקוישת (list) מסוג ClientInfo אותה נאתחל בהתחלה בראיקה. כל לקוח שמתחבר לשרת יתווסף לרשיימה, כל לקוח שייתנק מהשירות ימחק מהרשימה ואם נרצה להעביר הודעה אל לקוח נמען יוכל פשוט לחפש אותו ברשיימה ולקבל פוינטර אל הפרטים שלו.

חשוב לציין – עברו כל לקוח שנבוצע עליו accept thread שמרתתו לפצל את הקש של הרשת לנמה ליקוחות במקביל, פונקציית ה-thread היא Interact מהקובץ interact_with_client.h לה אנחנו נותנים את ה-socket של הלוקח הנכנס וறרנס לבנייה הנתונים, הרשיימה, ששומרת את כל הלוקחות. בדרך זו נוצרת א-סינכרוניות ומשתמשים אינם תלויים זה זה!

```

1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <iostream>
4  #include <tchar.h>
5  #include <thread>
6  #include <list>
7  #include "client_info.h"
8  #include "socket_setups.h"
9  #define MAX_USERS 100
10 using namespace std;
11
12 /* In main we do all socket procedures from socket_setups.h along with listening and
   accepting clients. We check for errors in each procedure. We then create an empty
   list of type ClientInfo in which we will store all the active clients.
   Once a client comes in and we accept him, we send his socket to the 'Interact'
   function from interact_with_client.h by a thread to isolate the server and him, along with a
   reference to the list of clients we created.
   By using threads we split the server's attention to each client in particular creating desynchronization
   which is essential for chatting between one and another */
13 int main();
14
15
16
17
18
19
20

```

מתיק :server.cpp

```

list<ClientInfo> clients; // Stores the names & sockets of all clients participating in chat
while (true) {
    // step 5: Accept and start communicating with clients
    SOCKET clientSocket = accept(listenSocket, NULL, NULL); // A client tries to access the server
    if (clientSocket == INVALID_SOCKET)
    {
        cout << "Client socket is invalid" << endl;
    }
    else // We split the server's attention to each reaching client, so we get a multi client system
    {
        thread T(Interact, clientSocket, ref(clients)); // Each client gets his own treatment simultaneously with threads
        T.detach();
    }
}
closesocket(listenSocket);
WSACleanup();
return 0;

```

6.2 – אפליקציית הלוקו Client_interface

תיקיה זו מורכבת מ-3 קבצי header שם מרוכדים הפונקציות עם תיעודים כלליים ומ-3 קבצי cpp גם הם עם תיעודים מפורטים וביהם בפועל מיושמים התהיליכים. פירוט רחוב נמצא בתיעוד שבפרויקט, כאן נציג את קבצי ה-.header.

socket_setups.h – בדומה לאפליקציית השרת, גם באפליקציית הלוקו נגידר קובץ עם פעולות מקדימות שאנו צריכים לקיים כדי לאפשר תקשורת מבוססת sockets. בקובץ מופיע מימוש בסיסי של אתחול ספריית Winsock, יצירה socket התחברות לשרת וההתחברות לשרת.

```
1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <WinSock2.h>
4  #include <WS2tcpip.h>
5  #include <string>
6  #pragma comment(lib, "ws2_32.lib")
7  using namespace std;
8
9  /*
10 These are the steps the client handles to perform socket interactions:
11 1. Initialize winsock
12 2. Create comuunicating socket
13 3. Connect to server
14 4. 'send' for delivering messages to server and 'recv' for receiving messages from server
15 5. Close the socket when finished
16 6. Cleanup
17
18 The goal: Use 'send' and 'recv' methods with the server in order to
19     1. forward messages to a specified client
20     2. get messages from other clients
21
22 */
23
24 // step 1: Initialize WinSock version 2.2
25 bool initialize();
26
27 // step 2: Create a TCP socket to contact the server with
28 bool createTCPsocket(SOCKET& serverSocket);
29
30 // step 3.1: Create the address details of the server the client connects - ip and port
31 bool createAddress(sockaddr_in& serverAddr, const string& ip, const int portNumber);
32
33 // step 3.2: Connect to the server using the 'connect' function
34 bool connectToServer(SOCKET& serverSocket, sockaddr_in& serverAddr);
35
```

בקובץ זה מרכזת כל הלוגיקה של אפליקציית הלקוח משום שבקובץ זה מפורטות כל הפעולות בין הלקוח לשרת מצד הלקוח. הפונקציות שבקובץ זה מהוות את הלוגיקה שבתקשרות באשר אנחנו משתמשים בפונקציות על `sendMessage`, `receiveMessage` שנעזרות בפונקציות נוספות כדי לנהל תהליכי מול השרת.

```

1  /* Made by Liran Dagan 215609397 */
2  #pragma once
3  #include <WinSock2.h>
4  #include <iostream>
5  #include <thread>
6  #include <string>
7
8  using namespace std;
9
10 // return str1==str2 case insensitive ('a'=='A' for example)
11 bool equalStrings(const string&, const string&);
12
13 // Send client's name to the server to be known. Return success (true) or fail (false)
14 bool registerClient(SOCKET& server);
15
16
17 /* Send name of another client we want to message. Server will respond with a feedback message.
18 | Return success (true) or fail (false) */
19 bool searchTarget(SOCKET& server);
20
21
22 /* A thread function. Here we handle all of client's messages to the server in correlation to 'mode'
23 'sendMessage' utilizes all the functions above.
24 In short we do the following:
25 1. Use the 'mode' variable that gets 0, 1 or 2 to distinguish between actions as followed:
26    mode = 0 -> Send client's name to server (registerClient)
27    mode = 1 -> Specify target (searchTarget)
28    mode = 2 -> Send message to server to deliver to target
29 2. If connected==true meaning client didn't quit and no connection error occurred then go to 1.
30    else quit function */
31 void sendMessage(SOCKET server);
32
33
34 /* A thread function. Here we handle all of server's messages to the client including error feedbacks
35 and messages from other clients.
36 In short we do the following:
37 0. Get message/prompt from server
38 1. if got sameUser or UserNotFound prompts then stay in mode = 1 for sendMessage
39 2. if got userFound prompt then proceed to mode = 2 for sendMessage
40 3. if got Quit prompt or a disconnection error then connected=false and 6.
41 4. if none of the above then we actually got a message from another client and not a prompt, so we display it
42 5. Go to 0.
43 6. quit both thread functions sendMessage, receiveMessage and return to main to finish program */
44 void receiveMessage(SOCKET server);
45

```

בקובץ זה פונקציית ה-main, בה אנחנו מבצעים את כלפעולות ה-socket המקידימות כדי לאפשר ללקוח לתקשר עם השירות. אנחנו מבצעים את כל הפעולות הקיימות בקובץ.h.socket_setups.h: חשוב לציין – אנחנו יוצרים שני thread כאשר פונקציות ה-thread הן: interact_with_server.h, sendMessage וreceiveMessage. שילוב שני ה-threads מאפשר לנו לשלוח הודעות ולקבל הודעות socket שאיתו מתקשרים עם השירות. בז'מינית.

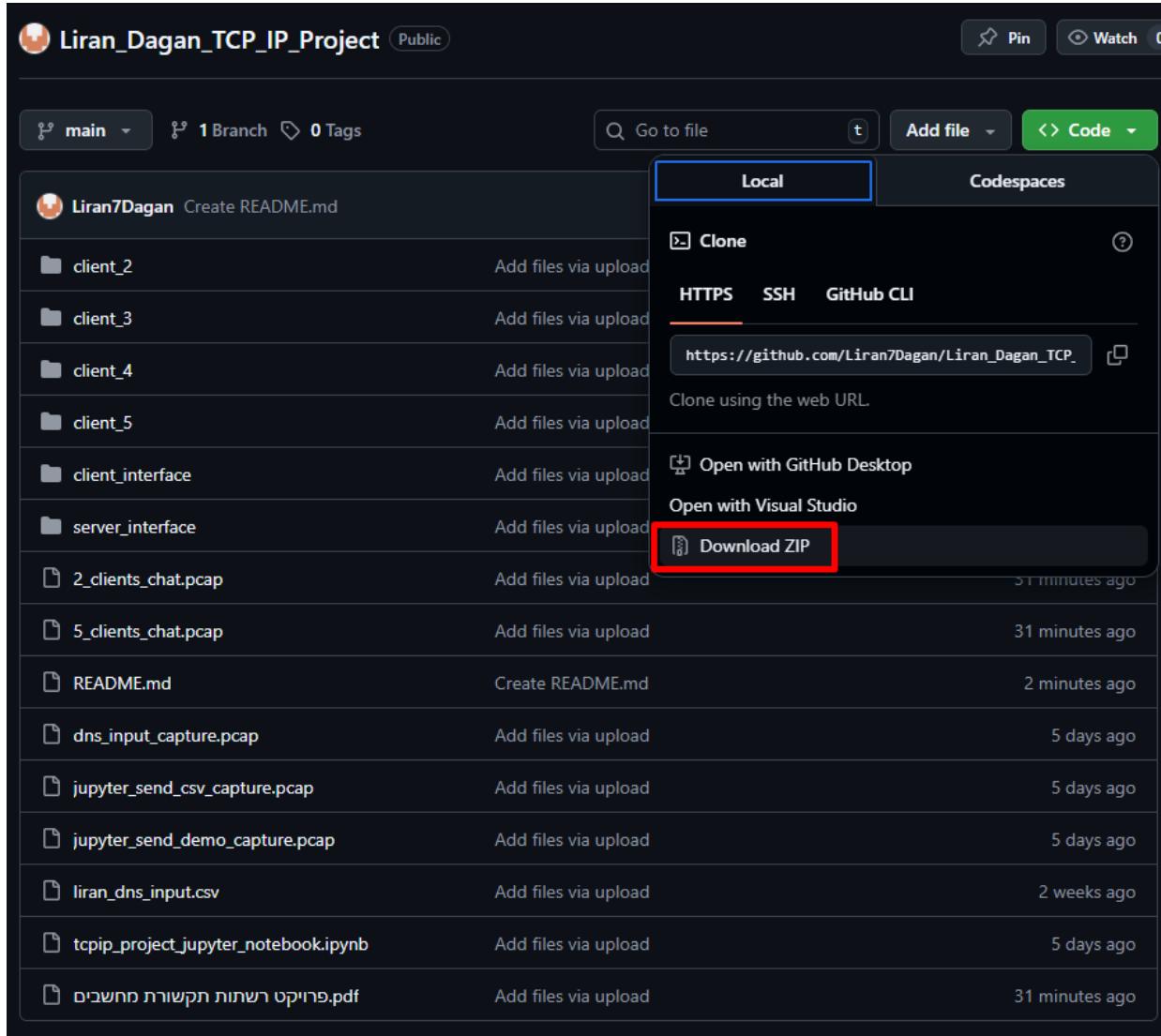
```
1  /* Made by Liran Dagan 215609397 */
2  #include "socket_setups.h"
3  #include "interact_with_server.h"
4  using namespace std;
5
6  /* In main we do all socket procedures from socket_setups.h.
7   * We check for errors in each procedure.
8   * We then create 2 threads, one for each function: sendMessage, receiveMessage
9   * and we give both of them the socket with which we communicate with the server.
10  By joining the threads we communicate with the server while having the two functions enable
11  each other. receiveMessage reacts to the server's feedback to sendMessage, and
12  sendMessage adjusts it's actions by reacting to receiveMessage's reponse to the server's
13  feedback. As a result we get one coherent application that sends and receives messages
14  at the same time. */
15 int main();
16
```

7. הוראות התקנה והרצה

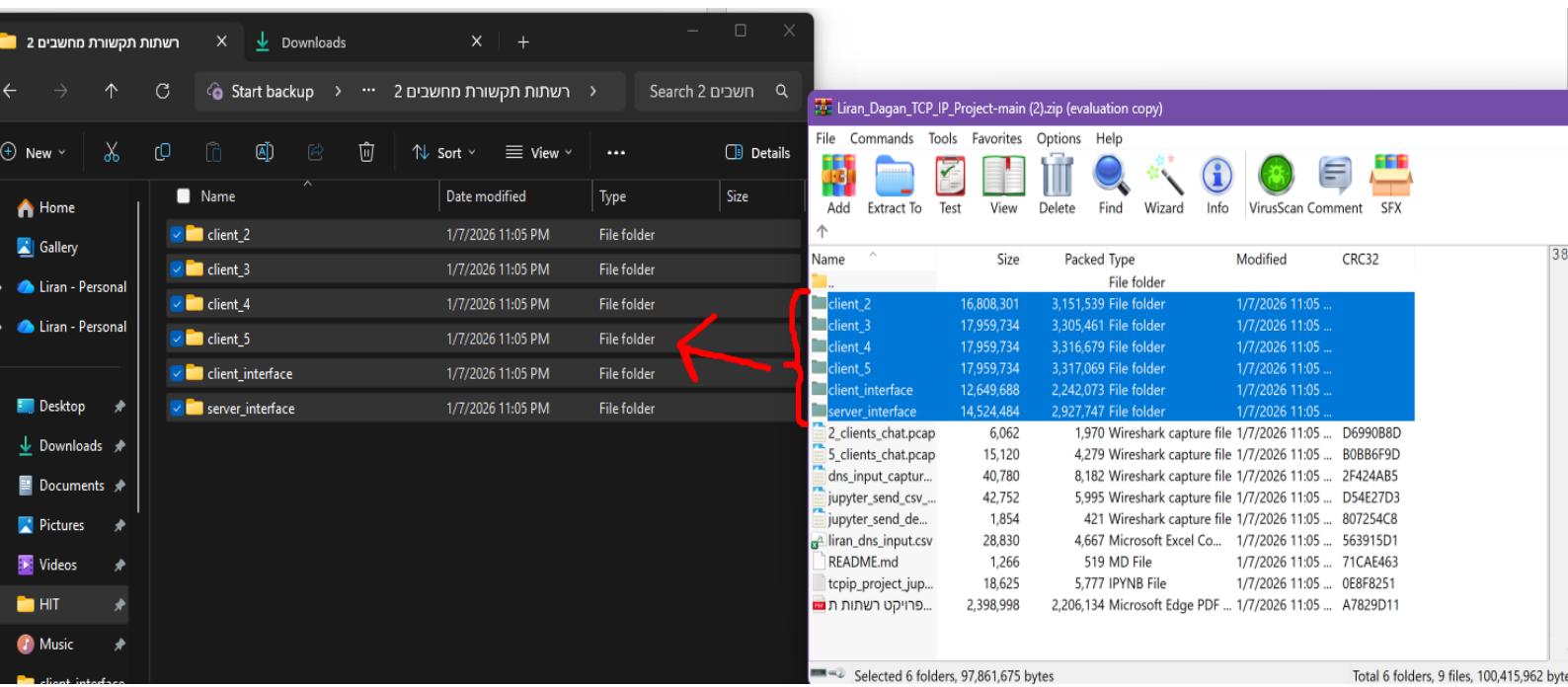
לגייטהאב מצורפים התקיות client_interface, server_interface, client_2 ו-client_1 שצויינו קודם לכן. כאמור שלושת האפליקציות נכתבו בשפת C++ ולכן ניתן על המחשב יש לפתח את קבצי ה-solution שלhn ביזואל סטודיו.

צריך לפתח חלון ויזואל סטודיו אחד שירץ את השרת, חלון אחד שירץ את לקובץ 1 וחלון אחד שירץ את לקובץ 2. במידת הרצון ניתן לשכפל את תקיות הלקווח ולפתח עוד חלונות שיריצו עוד לקוביות.

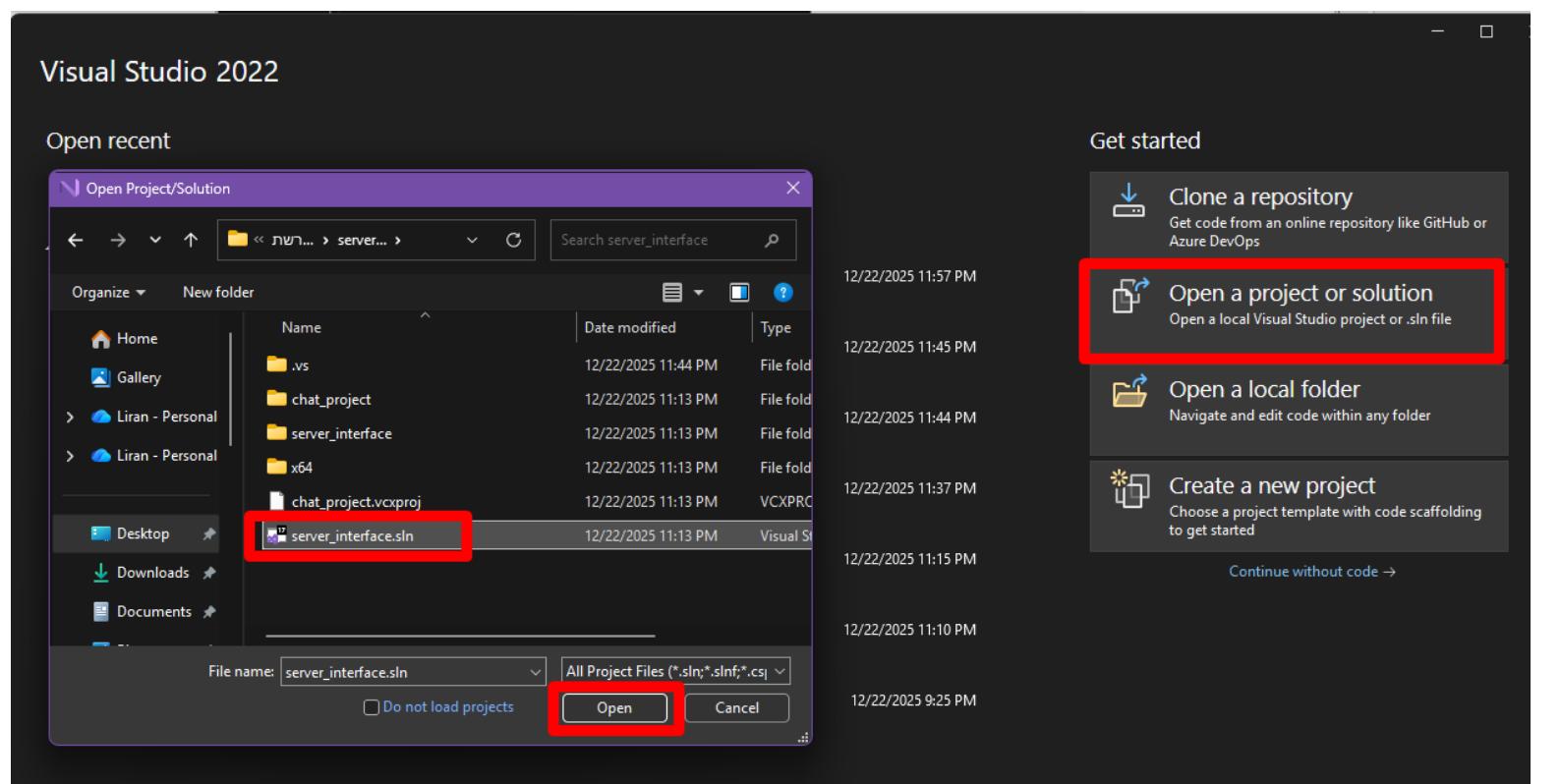
לכן אני מציע להורד בקובץ ZIP את תוכן הגיטהאב ולהלץ / Lager או שתי התקיות אל נתיב מועדף במחשב.



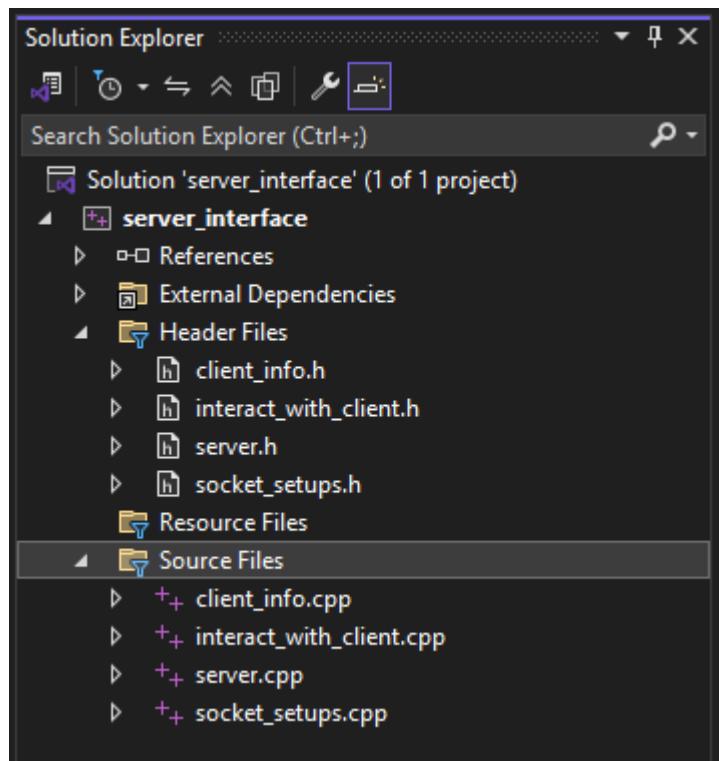
לאחר הורדת ה-ZIP, גירית התקינות:



לאחר מכן, בחלוןית של ויזואל סטודיו יש לפתוח אתไฟיל `server_interface.sln` שנמצא בתיקייה שנגירה.

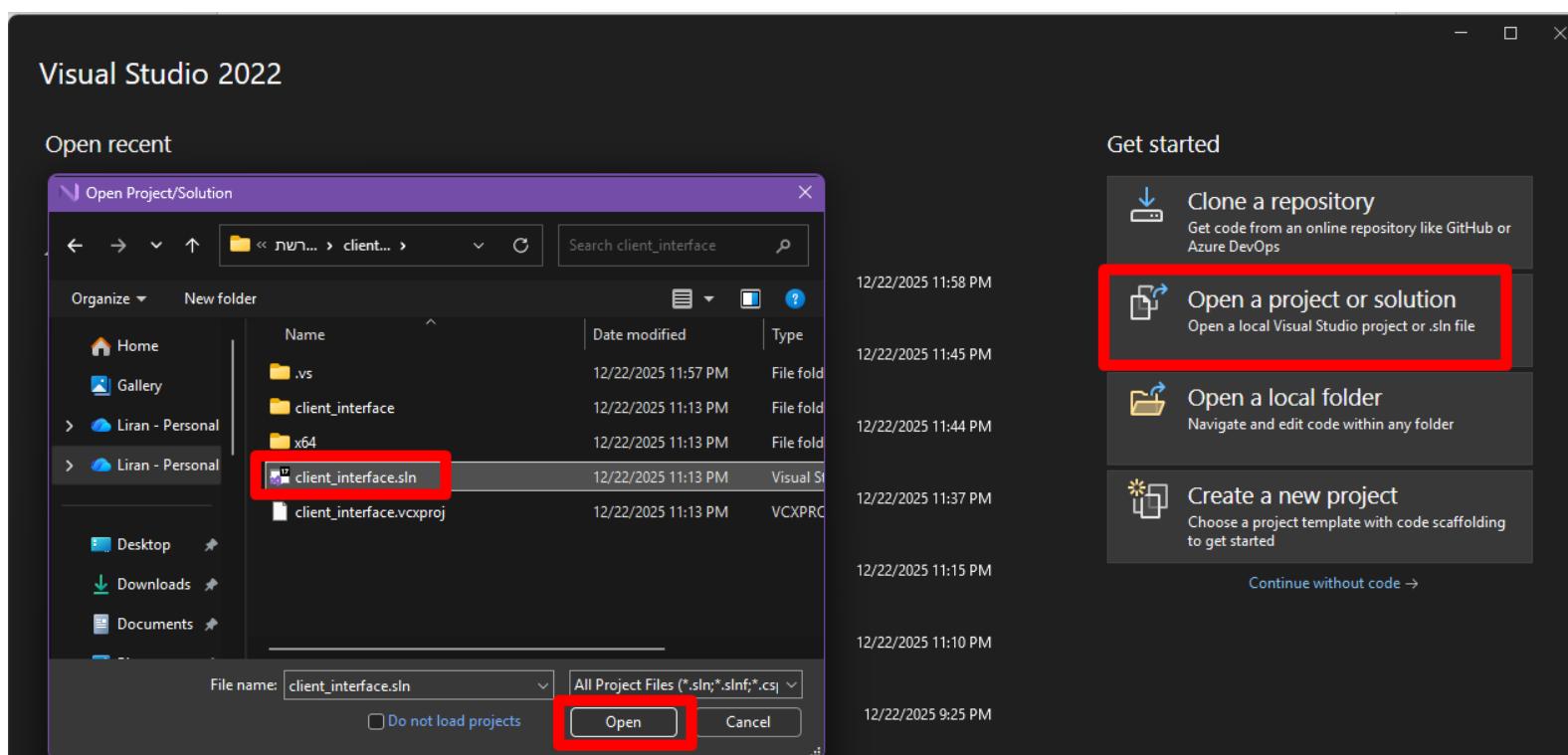


בשלב זה ניתן לפתח ולראות את כל קבצי הקוד - ה-`header` וה-`cpp` של השרת בתוך ויזואל סטודיו.



עבשינו צריך לפתח את אפליקציית הלוקו, בשבייל ברן צריך לפתח חלון נוסף של ויזואל סטודיו שייעבוד במקביל לחלון שכבר נפתח בעבר הרשות.

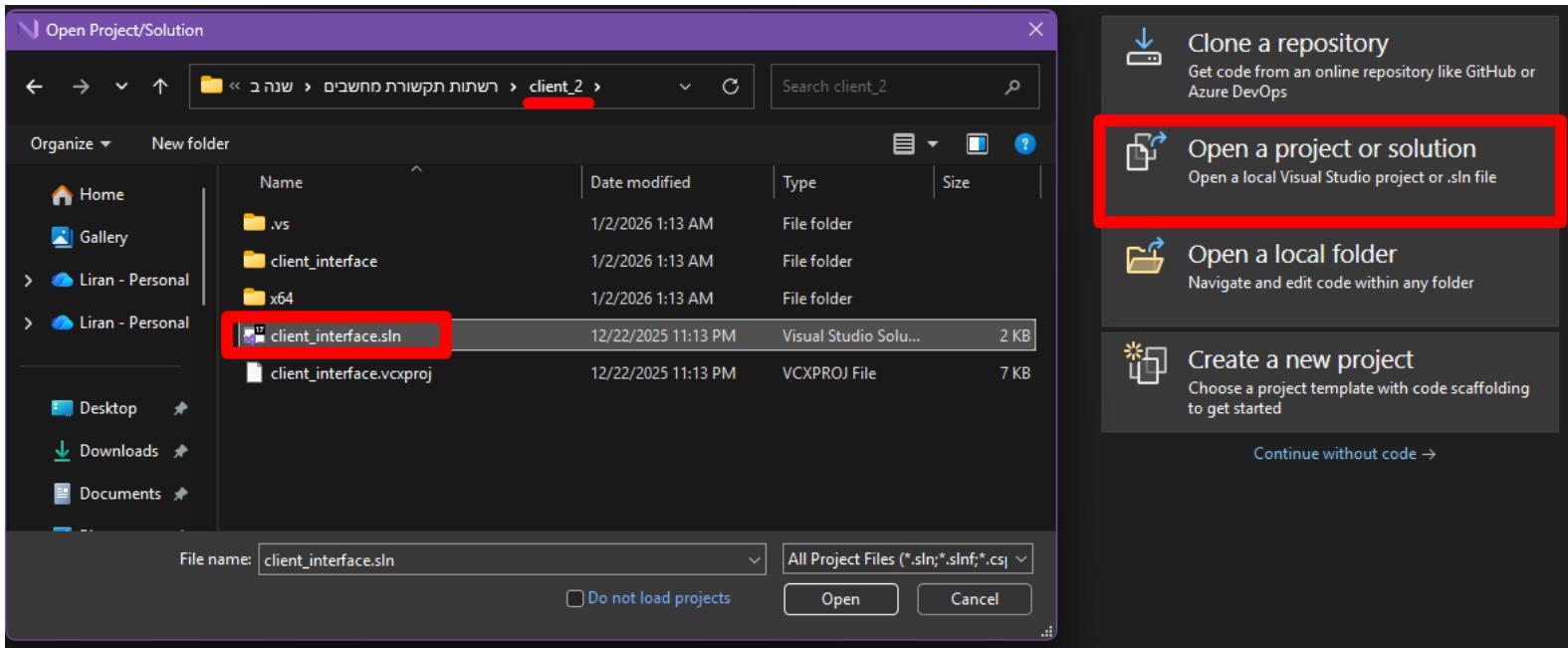
לאחר שנפתח חלון ויזואל סטודיו חדש נפתח את אפליקציית הלוקו באותו אוףן, נפתח את client_interface.sln שנמצא בתיקייה שנדרה:



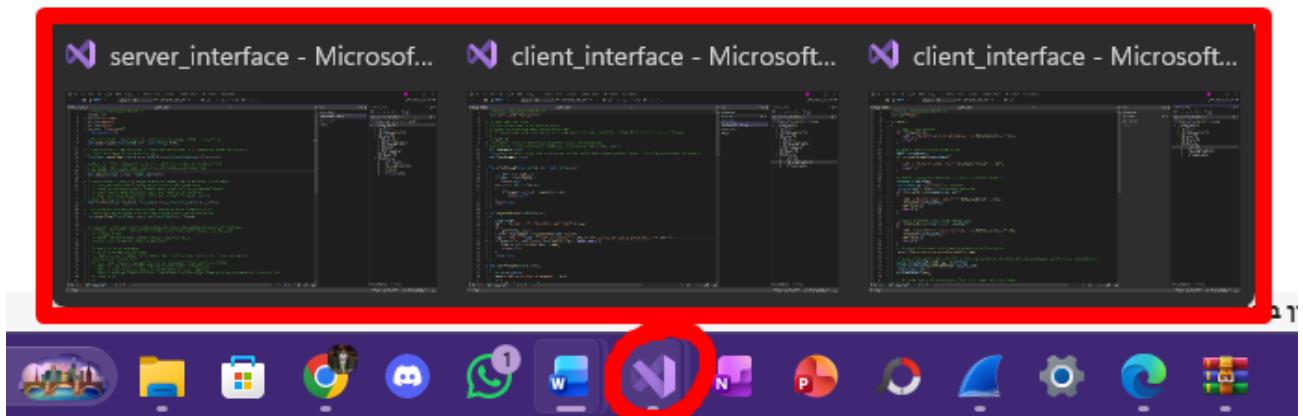
בשלב זה ניתן לפתח ולראות את כל קבצי הקוד – ה-.h ו-ה-.cpp של הלוקו בתוך ויזואל סטודיו.

כעת נוצר לפתוח גם את הלקוּח השני אם נרצה להריז שני משתמשים.

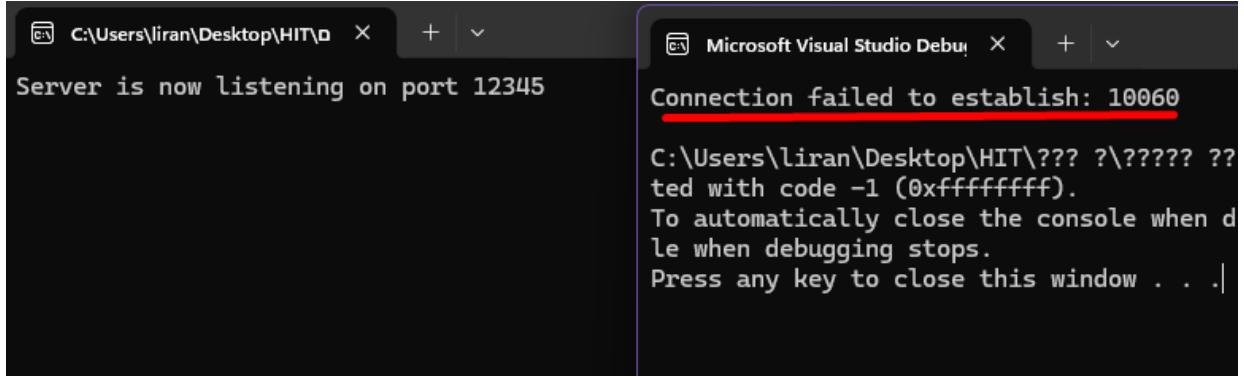
ובאופן זהה נפתח בחלון נפרד את `ls2.client` שנמצא בתיקייה שנגරה:



אם נרצה לפתח עודLKוחות, כאמור נשכפל את `TinyClient`client_interface ונפתח בחלונות ויזואל סטודיו נפרדים את קובץ `the.h`. של כל אחד. עד לקבלת שולחן עבודה שלושה חלונות ויזואל סטודיו שפועלם וניתנים להרצה במקביל. אחד בלבד עבור השרת ושניים או יותר עבור הלקוחות.



שגיאה צפiosa היא שגיאה מהסוג:



Server is now listening on port 12345

Connection failed to establish: 10060
C:\Users\liran\Desktop\HIT\??? ?\????? ??
ted with code -1 (0xffffffff).
To automatically close the console when d
le when debugging stops.
Press any key to close this window . . .

הסיבה: באפליקציית הלוקוח, בקובץ `client.cpp`, שורה 23:

```
1  /* Made by Liran Dagan 215609397 */  
2  #include "client.h"  
3  
4  int main()  
5  {  
6      // step 1: Start winsock  
7      if (!initialize()) {  
8          cout << "Failed to start the application." << WSAGetLastError() << endl;  
9          return -1;  
10     }  
11  
12     // step 2: Create TCP type server socket  
13     SOCKET serverSocket;  
14     if (!createTCPsocket(serverSocket))  
15     {  
16         cout << "Failed to create socket " << WSAGetLastError() << endl;  
17         return -1;  
18     }  
19  
20     // step 3.1: Assign the server address details we wish to connect to  
21     sockaddr_in serverAddr;  
22  
23     //const string ip = "127.0.0.1"; // UNCOMMENT FOR LOOPBACK  
24     const string ip = "192.168.1.15"; // local IP address, comment if needed  
25  
26     const int port = 12345; // Our assigned port number  
27     if (!createAddress(serverAddr, ip, port))  
28     {  
29         cout << "Setting address failed " << WSAGetLastError() << endl;  
30         closesocket(serverSocket);  
31         WSACleanup();  
32         return -1;  
33     }  
34  
35     // step 3.2: Connect client to the server socket
```

לפני הרצה יש להתאים את כתובת ה-IP. ניתן להשתמש בכתובת loopback או להשתמש בכתובת ה-IP המקומית של מחשב הבודק. מומלץ להשתמש בכתובת IP אמיתית אם כי האפליקציות יעבדו אף על כן.

יש להריץ את אפליקציית השרת ראשונה, ורק לאחר מכן את אפליקציית הלוקוח.

אם הרצה תקורה בסדר הפוך (קודם לлокוח ואז שרת), השרת יעבד ברגע אבל הלוקוח יקבל את ההדפסה (שהיא חלק מהקוד): Connection failed to establish: 10061 (השרת הורץ רק אחרי הלוקוח ולכן לא קיים בזמן הרצת הלוקוח).

דוגמה לשגיאת הרצת הלוקה לפני השרת:

The screenshot shows two terminal windows side-by-side. The left window, titled 'C:\Users\liran\Desktop\' (with a red arrow pointing to it), displays the message 'Server is now listening on port 12345'. The right window, titled 'Microsoft Visual Studio' (with a red arrow pointing to it), displays an error message: 'Connection failed to establish: 10061' followed by the command-line output of a process termination: 'C:\Users\liran\Desktop\client_interface\x64\Debug\client_interface.exe (process 50404) exited with code -1 (0xffffffff). To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops. Press any key to close this window . . .' A red arrow points from the text 'Press any key to close this window . . .' towards the bottom right of the right window.

בנוסף, האפליקציות מטפלות בשגיאות (כמו זו שבדוגמה) וכך אמורות לקרות קרייסות בלתי צפויות. אם קורות שגיאות כלשהן האפליקציות לא יקרים אלא ידפיסו על המסך הודעות מתאימות. עוד יש לציין כי המסר שמקבלים מאפליקציית השרת אינם אינטראקטיביים אלא אינפורטטיביים, הוא עוזר להמחיש התהיליכים שהכוונים בזמן אמת בין ללקוחות לשרת ובין ללקוחות ללוקוחות וכן לא מקבל עצמו קלט מהמשתמש אלא בסך הכל מציג מידע שעובר דרכו. המסר שמקבלים מאפליקציית הלוקה לעומת זאת הוא מבון אינטראקטיבי וинформационטיבי.

8. דוגמאות קלט ופלט

דוגמיה לשיחה בין שני משתמשים:

```
Microsoft Visual Studio Del X + - X Microsoft Visual Studio Debug X + - X X

Connection to server established
Welcome to the chat! What is your name?
Liran
Hello Liran! Have fun chatting.
You can quit any time by typing 'QUIT'.
-----
Who do you want to message?
itzik
User Was Not Found.
-----
Who do you want to message?
yaron
User Was Not Found.
-----
Who do you want to message?
Maor
Connection created
-----
Hey Maor!!!
Maor: Whats upppppppppp Lirannnn
I'm doing great
There's a big football game today
Maor: I know today is Real Madrid vs Barcelona
Right it will be a good game for sure!!!
Do you wanna come over and watch it together? We c
an order pizza ;)
Maor: YEAH For sure
Great so be at my place in about 15 minutes
Maor: I'm on my way, see you then!
quit
C:\Users\liran\Desktop\HIT\??? ?\????? ?????? ????
??\client_interface\x64\Debug\client_interface.exe

Connection to server established
Welcome to the chat! What is your name?
Maor
Hello Maor! Have fun chatting.
You can quit any time by typing 'QUIT'.
-----
Who do you want to message?
Maor
Please Message a Different User.
-----
Who do you want to message?
Liran
Connection created
-----
Liran: Hey Maor!!!
Whats upppppppppp Lirannnn
Liran: I'm doing great
Liran: There's a big football game today
I know today is Real Madrid vs Barcelona
Liran: Right it will be a good game for sure!!!
Liran: Do you wanna come over and watch it together?
We can order pizza ;)
YEAH For sure
Liran: Great so be at my place in about 15 minutes
I'm on my way, see you then!
quit
C:\Users\liran\Desktop\HIT\??? ?\????? ?????? ????
??\client_interface\x64\Debug\client_interface.exe (process 5257)
```

עבור השיחה שלמעלה, פלט השירות הוא:

```
C:\Users\liran\Desktop\HIT\P X + ▾  
Server is now listening on port 12345  
New client joined: 'Liran'  
New client joined: 'Maor'  
'Liran' Tried to reach a non existent client: 'itzik'  
Maor Tried to message himself  
'Liran' Tried to reach a non existent client: 'yaron'  
'Liran' Reaches 'Maor'. Server is ready to transmit messages!  
'Maor' Reaches 'Liran'. Server is ready to transmit messages!  
'Liran' TO 'Maor': Hey Maor!!!  
'Maor' TO 'Liran': Whats upppppppppp Lirannnn  
'Liran' TO 'Maor': I'm doing great  
'Liran' TO 'Maor': There's a big football game today  
'Maor' TO 'Liran': I know today is Real Madrid vs Barcelona  
'Liran' TO 'Maor': Right it will be a good game for sure!!!  
'Liran' TO 'Maor': Do you wanna come over and watch it together? We can order pizza ;)  
'Maor' TO 'Liran': YEAH For sure  
'Liran' TO 'Maor': Great so be at my place in about 15 minutes  
'Maor' TO 'Liran': I'm on my way, see you then!  
'Liran' Disconnected  
'Maor' Disconnected
```

קצת על התקשרות בין השירות ללקוחות בשיחה זו:

המשתמשים שולחים לשרת את שמותיהם "ליין", "מאור", השירות זכר ושומר אותם. בכל פעם שהמשתמשים מנסים לפנות ללקוח שהשירות אינו מכיר, כמו "איציק" ו"ירון" הוא שולח להם פידבק "המשתמש לא נמצא". באופן דומה אם המשתמש פונה אל עצמו כמו שעשה מאור, השירות משיב: "בבקשה פנה למשתמש אחר". במקרה פונה אל מאור (שהשירות מכיר) השירות מודיע לו שנוצר חיבור וההודעות מועברות מבינן והילך אל מאור. באופן זהה השירות עשו עברו מאור כשהוא פונה אל ליין. במקרה המשתמשים משוחחים ביניהם. משתמש מתנתק מהשירות על ידי הקלט "quit", כמו שעושים מאור ולירן בהודעות האחרונות שלהם. שאלה שואלי Nasalat ha: מה היה קורה אם ליין היה שולח הודעה למאור אחרי שהוא התנתק? כמובן, מאור לא מזוהה יותר על ידי השירות ולירן בכל זאת מנסה לשלוח לו הודעה.

במקרה זה השירות לא יזהה את מאור ולכן לירן על סיום החיבור ייחזר אותו למצב ההתחלתי, כמובן יבקש ממנו לצין ל��ח חדש שהוא רוצה לתקשר איתנו.

בתמונה ניתן לראות תהליך שבו לiran מנסה לשלוח הודעה לאחר שכבר הtentak. השירות מסביר לiran שאינו יכול לשלוח את ההודעה כי השני התנטק. מיד לאחר מכן הוא מבהיר אותו לנקודת ההתחלה וambil שמן שם של משתמש אחר.

The image shows two side-by-side Microsoft Visual Studio windows. Both windows have a title bar labeled "Microsoft Visual Studio" and a close button "X".

Left Window (Liran's Session):

- Connection to server established
- Welcome to the chat! What is your name?
- Liran
- Hello Liran! Have fun chatting.
- You can quit any time by typing 'QUIT'.
- Who do you want to message?
- Maor **בקשה להתחבר למאור, בשלב זה הוא קיים**
- Connection created **ליין שלוח הודעה למאור, בשלב זה הוא מנוטק**
- hey maor
- The other party has disconnected.
- Who do you want to message?
- maor
- User Was Not Found.
- Who do you want to message?
- Quit

Right Window (Maor's Session):

- Connection to server established
- Welcome to the chat! What is your name?
- Maor
- Hello Maor! Have fun chatting.
- You can quit any time by typing 'QUIT'.
- Who do you want to message?
- quit **מאור מתנטק תור כדי שיחה**
- C:\Users\liran\Desktop\HIT\??? ?\????? ?????
code 0 (0x0).
- To automatically close the console when debug
le when debugging stops.
- Press any key to close this window . . .

בר השירות מתמודד עם ניטוקים יזומים תור כדי שיחות.

יש לציין גם כי המשתמשים לא חייבים בהכרח להתהבר זה אל זה בו זמן, כלומר משתמש אחד יכול לקבל הודעה ממשתמש שני ללא תלות בפעולות אחרות. (איסינכרוניות), לדוגמה:

```
C:\Users\liran\Desktop\ X + - □ X C:\Users\liran\Desktop\HIT\p X + -  
Connection to server established  
Welcome to the chat! What is your name?  
Liran  
Hello Liran! Have fun chatting.  
You can quit any time by typing 'QUIT'.  
-----  
Who do you want to message?  
Maor  
Connection created  
-----  
hello →  
|  
Maor: HI!  
|  
Connection to server established  
Welcome to the chat! What is your name?  
Maor  
Hello Maor! Have fun chatting.  
You can quit any time by typing 'QUIT'.  
-----  
Who do you want to message?  
Liran: hello ← ההודעה של לiran  
Who do you want to message?  
xxx  
User Was Not Found.  
-----  
Who do you want to message?  
la la la  
User Was Not Found.  
-----  
Who do you want to message?  
Liran רק בשלב הזה מאור מתהבר  
Connection created  
-----  
HI!
```

גנחת את השיכחה הראשונה בין לירן למארו ב-Wireshark :

No.	Time	Source Address	Destination Address	Source Port	Destination Port	Protocol	TCP payload	Syn	Acknowledgment	Fin	Push	Sequence Number	Acknowledgment Number	ip version	Time to Live
277	9.771315	192.168.1.15	192.168.1.15	57918	12345	TCP		1	0	0	0	0	0	4	128
278	9.771436	192.168.1.15	192.168.1.15	12345	57918	TCP		1	1	0	0	0	1	4	128
279	9.771475	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1	0	0	1	1	4	128
542	17.205985	192.168.1.15	192.168.1.15	52906	12345	TCP		1	0	0	0	0	0	4	128
543	17.206091	192.168.1.15	192.168.1.15	12345	52906	TCP		1	1	0	0	0	1	4	128
544	17.206126	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1	0	0	1	1	4	128
796	45.658828	192.168.1.15	192.168.1.15	57918	12345	TCP	4c6972616e	0	1	0	1	1	1	4	128
797	45.658868	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	1	6	4	128
816	52.423497	192.168.1.15	192.168.1.15	52906	12345	TCP	4d616f72	0	1	0	1	1	1	4	128
817	52.423528	192.168.1.15	192.168.1.15	12345	52906	TCP		0	1	0	0	1	5	4	128
827	63.691099	192.168.1.15	192.168.1.15	57918	12345	TCP	69747a696b	0	1	0	1	6	1	4	128
828	63.691131	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	1	11	4	128
829	63.691492	192.168.1.15	192.168.1.15	12345	57918	TCP	533a55736572...	0	1	0	1	1	11	4	128
830	63.691512	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1	0	0	11	15	4	128
831	72.934822	192.168.1.15	192.168.1.15	52906	12345	TCP	4d616f72	0	1	0	1	5	1	4	128
832	72.934855	192.168.1.15	192.168.1.15	12345	52906	TCP		0	1	0	0	1	9	4	128
833	72.935063	192.168.1.15	192.168.1.15	12345	52906	TCP	533a53616d65...	0	1	0	1	1	9	4	128
834	72.935084	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1	0	0	9	11	4	128
874	87.834920	192.168.1.15	192.168.1.15	57918	12345	TCP	7961726f6e	0	1	0	1	11	15	4	128
875	87.834957	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	15	16	4	128
876	87.835487	192.168.1.15	192.168.1.15	12345	57918	TCP	533a55736572...	0	1	0	1	15	16	4	128
877	87.835514	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1	0	0	16	29	4	128
894	89.980732	192.168.1.15	192.168.1.15	57918	12345	TCP	4d616f72	0	1	0	1	16	29	4	128
895	89.980763	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	29	20	4	128
896	89.981272	192.168.1.15	192.168.1.15	12345	57918	TCP	533a55736572...	0	1	0	1	29	20	4	128
897	89.981302	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1	0	0	20	40	4	128
898	99.156198	192.168.1.15	192.168.1.15	52906	12345	TCP	4c6972616e	0	1	0	1	9	11	4	128
899	99.156240	192.168.1.15	192.168.1.15	12345	52906	TCP		0	1	0	0	11	14	4	128
900	99.156652	192.168.1.15	192.168.1.15	12345	52906	TCP	533a55736572...	0	1	0	1	11	14	4	128
901	99.156682	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1	0	0	14	22	4	128
1014	125.662005	192.168.1.15	192.168.1.15	57918	12345	TCP	486579204d61...	0	1	0	1	20	40	4	128
1015	125.662041	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	40	31	4	128
1016	125.662259	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	22	14	4	128
1017	125.662292	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1	0	0	14	42	4	128
1038	137.925910	192.168.1.15	192.168.1.15	52906	12345	TCP	576861747320...	0	1	0	1	14	42	4	128
1039	137.925961	192.168.1.15	192.168.1.15	12345	52906	TCP		0	1	0	0	42	40	4	128
1040	137.926124	192.168.1.15	192.168.1.15	12345	57918	TCP	433a4d616f72...	0	1	0	1	40	31	4	128
1041	137.926163	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1	0	0	31	74	4	128
1114	145.462912	192.168.1.15	192.168.1.15	57918	12345	TCP	49276d20646f...	0	1	0	1	31	74	4	128
1115	145.463054	192.168.1.15	192.168.1.15	12345	57918	TCP		0	1	0	0	74	46	4	128
1116	145.463245	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	42	40	4	128
1117	145.463273	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1	0	0	40	66	4	128

1205	166.677188	192.168.1.15	192.168.1.15	57918	12345	TCP	546865726527...	0	1	0	1	46	74	4	128
1206	166.677220	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	74	79	4	128	
1207	166.677264	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	66	40	4	128
1208	166.677283	192.168.1.15	192.168.1.15	52906	12345	TCP	49206b6e6f77...	0	1	0	1	40	108	4	128
1299	202.731631	192.168.1.15	192.168.1.15	52906	12345	TCP	49206b6e6f77...	0	1	0	1	40	108	4	128
1300	202.731665	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	0	0	108	80	4	128	
1301	202.731747	192.168.1.15	192.168.1.15	12345	57918	TCP	433a4d616f72...	0	1	0	1	74	79	4	128
1302	202.731783	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0	0	79	122	4	128	
1379	231.741231	192.168.1.15	192.168.1.15	57918	12345	TCP	526967687420...	0	1	0	1	79	122	4	128
1380	231.741277	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	122	119	4	128	
1381	231.741335	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	108	80	4	128
1382	231.741363	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	0	0	80	157	4	128	
1464	279.675940	192.168.1.15	192.168.1.15	57918	12345	TCP	446f20796f75...	0	1	0	1	119	122	4	128
1465	279.675974	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	122	186	4	128	
1466	279.676059	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	157	80	4	128
1467	279.676083	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	0	0	80	233	4	128	
1822	405.954706	192.168.1.15	192.168.1.15	52906	12345	TCP	594541482046...	0	1	0	1	80	233	4	128
1823	405.954744	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	0	0	233	93	4	128	
1824	405.954825	192.168.1.15	192.168.1.15	12345	57918	TCP	433a4d616f72...	0	1	0	1	122	186	4	128
1825	405.954866	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0	0	186	143	4	128	
1993	449.486308	192.168.1.15	192.168.1.15	57918	12345	TCP	477265617420...	0	1	0	1	186	143	4	128
1994	449.486336	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	143	229	4	128	
1995	449.486501	192.168.1.15	192.168.1.15	12345	52906	TCP	433a4c697261...	0	1	0	1	233	93	4	128
1996	449.486526	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	0	0	93	285	4	128	
2094	504.798385	192.168.1.15	192.168.1.15	52906	12345	TCP	49276d206f6e...	0	1	0	1	93	285	4	128
2095	504.798429	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	0	0	285	121	4	128	
2096	504.798503	192.168.1.15	192.168.1.15	12345	57918	TCP	433a4d616f72...	0	1	0	1	143	229	4	128
2097	504.798533	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0	0	229	179	4	128	
2154	522.033410	192.168.1.15	192.168.1.15	57918	12345	TCP	71756974	0	1	0	1	229	179	4	128
2155	522.033453	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	179	233	4	128	
2156	522.033573	192.168.1.15	192.168.1.15	12345	57918	TCP	533a71756974	0	1	0	1	179	233	4	128
2157	522.033599	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0	0	233	185	4	128	
2158	522.033969	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	1	0	185	233	4	128	
2159	522.033992	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0	0	233	186	4	128	
2160	522.034187	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	1	0	233	186	4	128	
2161	522.034229	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0	0	186	234	4	128	
2241	533.052152	192.168.1.15	192.168.1.15	52906	12345	TCP	71756974	0	1	0	1	121	285	4	128
2242	533.052188	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	0	0	285	125	4	128	
2243	533.052354	192.168.1.15	192.168.1.15	12345	52906	TCP	533a71756974	0	1	0	1	285	125	4	128
2244	533.052380	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	0	0	125	291	4	128	
2245	533.052682	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	1	0	291	125	4	128	
2246	533.052700	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	0	0	125	292	4	128	
2247	533.052915	192.168.1.15	192.168.1.15	52906	12345	TCP	0	1	1	0	125	292	4	128	
2248	533.052956	192.168.1.15	192.168.1.15	12345	52906	TCP	0	1	0	0	292	126	4	128	

הדבר הראשון שנראה לעין הוא במובן סוג התעבורה – TCP בכל הפקטות. ובאמת זה המצב מכיוון שהגדנו את השרת ברמת האפליקציה לבודع עם חיבור TCP.

מזהים גם שכתובות ה-*קוֹן* זהה בין הלקחות, במקרה זה מכיוון ששניהם פועלים על אותו המחשב. דבר נספ שרואים הוא בשכבת התעבורה, מספר הפורט 12345 חוזר על עצמו בכל הפקטות, ולא ב כדי מכיוון שהוא מספר הפורט שהקצינו לשרת. لكن כל פאקטה שה-*port* source שלה הוא 12345, משמעות הדבר שהשרת שלח את הפאקטה וכל פאקטה שה-*port* destination שלה הוא 12345, משמעות הדבר שהשרת קיבל את הפאקטה מלוקח שלח אותה.

כמו כן במספרי הפורט 57918 ו- 52906 חוזרים על עצמם בכל הפקטות, לעיתים מספר אחד ולעתים המספר השני. זה בגל שallow במספרי הפורט מהם מתקשרים משתמשים לירן ומאור וכן הם קבועים וחוזרים על עצמם.

מספר הפורט 57918 שייך לירן ומספר הפורט 52906 שייך למאור, נסביר בהמשך מדוע. נשים לב לחיבור של המשתמשים "ליין" ו"מאור" לשרת. בחבילות הראשונות נוכל בבירור לראות את ה-*handshake 3 way* של כל אחד מהמעורבים עם השירות. נראה את התחרבות של לירן מפורט 57918 לשרת:

No.	Time	Source Address	Destination Address	Source Port	Destination Port	Protocol	TCP payload	Syn	Acknowledgment
277	9.771315	192.168.1.15	192.168.1.15	57918	12345	TCP		1	0
278	9.771436	192.168.1.15	192.168.1.15	12345	57918	TCP		1	1
279	9.771475	192.168.1.15	192.168.1.15	57918	12345	TCP		0	1
542	17.205985	192.168.1.15	192.168.1.15	52906	12345	TCP			

בחבילה מס' 277 קורה 277 קורה 12345 *syn=1, ack=0, destination port = 12345*, כלומר זו חvíלה שהשרת קיבל מהמשתמש "ליין" והוא מהוות בקשה של הלוקוח להתחבר לשרת.

בחבילה מס' 278 קורה 278 קורה 12345 *syn=1, ack=1, source port = 12345*, כלומר זו חvíלה שהשרת שלח למשתמש "ליין" והוא מהוות אישור לבקשת החיבור של הלוקוח

בחבילה מס' 279 קורה 279 קורה 12345 *syn=0, ack=1, destination port = 12345*, כלומר זו חvíלה שהשרת קיבל מהמשתמש "ליין" שמהוות את אישור התחרבות הלוקוח לשרת, תהליך *handshake 3 way* הסתיים ומכאן והילך כל הפקטות שלוחה המשמש "ליין" הן מהצורה זו.

נשים לב שבפקטות 542,543,544 קורה אותו התהליך בדיקת עבור המשתמש "מאור" מפורט 52906 מופיע שגם הוא המתחבר לשרת.

542	17.205985	192.168.1.15	192.168.1.15	52906	12345	TCP		1	0
543	17.206091	192.168.1.15	192.168.1.15	12345	52906	TCP		1	1
544	17.206126	192.168.1.15	192.168.1.15	52906	12345	TCP		0	1

שאלה שנשאלת היא כיצד ידעתו לשער את הפורט 57918 דואק לאחיבור של "ליין" ואת הפורט 52906 דואק לאחיבור של "מאור"? הרעיון יכול להיות הפורט? תשובה אחת לשאלה היא שאני, בתור מי שהריץ את הדיאלוג מן הסתם ידוע מי המתחבר קודם...

אבל זו לא התשובה שאנו מחפשים; התשובה האמיתית היא, נשים לב לפאקטה מס' 796:

796	45.658828	192.168.1.15	192.168.1.15	57918	12345	TCP	4c6972616e	0
0000	02 00 00 00 45 00 00 2d ff 4f 40 00 80 06 00 00		E... 0@.....				
0010	c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 94		>0931..				
0020	34 3f cd 3d 50 18 00 ff 06 99 00 00 4c 69 72 61			4? =P..... Liran				
0030	6e			n				

שמות לב שהפקטה נשאת את payload: "Liran", זו הודעה הראשונה לגמרי שנשלחת לשרת, זה כי הדבר הראשון שמשתמש עושה כשהוא פותח את האפליקציה זה להקליד את השם שלו לשרת. נשים לב שפורט השולח הוא 57918 והמקבל הוא מבון השירות. אם כן בעת אנחנו יכולים בוודאות לשער את מספר הפורט 57918 למשתמש "לiran" ואילו את מספר הפורט الآخر 52906 לשער ל" מאור" למשתמש " מאור". ובאמת לפאקטה מס' 816 רואים שהפורט 52906 שייר ל" מאור" לפי אותו ניתוח:

816	52.423497	192.168.1.15	192.168.1.15	52906	12345	TCP	4d616f72	0
0000	02 00 00 00 45 00 00 2c ff 51 40 00 80 06 00 00		E... , 0@.....				
0010	c0 a8 01 0f c0 a8 01 0f ce aa 30 39 3c 9d 6c cc		09<1.				
0020	8d 0d 6b 6d 50 18 00 ff cd be 00 00 4d 61 6f 72		kmp..... Maor				

באוטו אופן אפשר לבדוק את ניתוק המשתמשים מהשרת, נשים לב לפאקטות 2158-2161. נזכיר שבתמונה הבאה, הבית הימני הוא השדה Fin, משמאלו Ack ומשמאלו Sy שמאופס בכל השירות.

2158	522.033969	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	1
2159	522.033992	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	0
2160	522.034187	192.168.1.15	192.168.1.15	57918	12345	TCP	0	1	1
2161	522.034229	192.168.1.15	192.168.1.15	12345	57918	TCP	0	1	0

אלו הפאקטות האחרונות שהוקלטו עבור המשתמש "לiran" שברפורט 57918.
נשים לב לתהליך 4 way handshake של לiran מהשרת:
בחבילה מס' 2158 קורה fin=1, ack=1, וזו חvíלה שהשרת שולח לירן. בגין השרת שולח בקשה לניתוק החיבור. (בתגובה לפאקטה קודמת שבה לירן מבון שלח הודעה quit לשרת).
בחבילה מס' 2159 קורה fin=0, ack=1, וזו חvíלה ששולח לירן לשרת. בגין לiran מאשר את בקשה השרת לניתוק החיבור.
בחבילה מס' 2160 קורה fin=1, ack=1, וזו חvíלה ששולח לירן לשרת. בגין לiran שולח בקשה לניתוק החיבור.
בחבילה מס' 2161 קורה fin=0, ack=1, וזו חvíלה ששולח השרת לירן. בגין השרת מאשר את הבקשה של לiran והחיבור נסגר סופית. זו הפאקטה الأخيرة שעברה בין לiran לשרת או להיפר.
התהליך חוזר על עצמו עבור המשתמש " מאור" שברפורט 52906 ובאמת 4 הפאקטות האחרונות של ההקלטה הן הניתוק של מאור מהשרת:

1921	244.105287	12345	63992	TCP	0	1	1
1922	244.105308	63992	12345	TCP	0	1	0
1923	244.107007	63992	12345	TCP	0	1	1
1924	244.107064	12345	63992	TCP	0	1	0

בשכבה הרשת, כאמור כתובות המקור ובכתובת היעד זהות, מכיוון שהלכה והשרות רצים על אותו מחשב.

יש לציין שבשונה מהריצה על כתובות ה-loopback, השתמשנו בכתובת IP מקומית אמיתית של המחשב.

בתוצאה מבר הפאקטוֹת נארזות בכתובת IP וניתן לראות את הארץיה של הפאקטוֹת ולנתח אותן ב-Wireshark.

את השרת הגדרנו בכתובת 0.0.0.0, מה שמאפשר לו להאזין לכל כתובות ה-IP של המחשב ולקבל לקוחות לפי כתובות ה-IP שלהם.

הכתובת 0.0.0.0 היא בתובת לוגית לצורך binding שאונה נשלחת ברשות, וכן הוא לא מופיעה בתוֹר כתובות ה-IP האמיתית של השרת בקובץ הלכידה.

בוחרות IPv4 של כל הפאקטוֹת הן מהצורה:

```
Raw / Loopback
▼ Internet Protocol Version 4, Src: 192.168.1.15, Dst: 192.168.1.15
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0xffff54 (65364)
  ▼ 010. .... = Flags: 0x2, Don't fragment
    0.... .... = Reserved bit: Not set
    .1... .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.15
    Destination Address: 192.168.1.15
    [Stream index: 1]
```

עם שינויים מסוימים בשדות ה-Header Length, Total Length בהתאם לגודל הסגמנט וההודעה ובשדה ה-Identification. נשים לב לסוג הпрוטוקול, 6 עבור חיבור TCP. נשים לב גם לשדה

Time to Live, שדה זה תמיד עם ערך 128, ערך שהוא קבוע וגודלו יחסית.

הסיבה לכך היא שchipor הרשת הוא בין המחשב לעצמו (כתובות IP זהות). לעומת זאת ראים או רכיבי רשת אחרים בדרך כלל הפאקטוֹת עם ערך קבוע וגדול שקבועת מערכת הפעלה.

מקודם ציינה הפקטה 796 שנשאה את ה-payload .“Liran” :payload
במובן שבקובץ הלכידה שמצורף אפשר לעיין בכל ההודעות ברמת האפליקציה, כמו:
ליין שולח לשרת "itzik" בפקטה 827 (משתמש אליו רצה לשלוח הודעה)

No.	Time	Source Address	Destination Address	Source Port	Destination Port	Protocol	TCP payload
277	9.771315	192.168.1.15	192.168.1.15	57918	12345	TCP	
278	9.771436	192.168.1.15	192.168.1.15	12345	57918	TCP	
279	9.771475	192.168.1.15	192.168.1.15	57918	12345	TCP	
542	17.205985	192.168.1.15	192.168.1.15	52906	12345	TCP	
543	17.206091	192.168.1.15	192.168.1.15	12345	52906	TCP	
544	17.206126	192.168.1.15	192.168.1.15	52906	12345	TCP	
796	45.658828	192.168.1.15	192.168.1.15	57918	12345	TCP	4c6972616e
797	45.658868	192.168.1.15	192.168.1.15	12345	57918	TCP	
816	52.423497	192.168.1.15	192.168.1.15	52906	12345	TCP	4d616f72
817	52.423528	192.168.1.15	192.168.1.15	12345	52906	TCP	
827	63.691099	192.168.1.15	192.168.1.15	57918	12345	TCP	69747a696b
828	63.691131	192.168.1.15	192.168.1.15	12345	57918	TCP	
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version: 0020 34 3f cd 3d 50 18 00 ff e4 80 00 00 69 74 7a 69							
0100 = Header Length: 0030 6b							
► Frame 827: Packet, 49 0000 02 00 00 45 00 00 2d ff 53 40 00 80 06 00 00							
► Null/Loopback 0010 c0 a8 01 0f c0 a8 01 0f e2 3e 30 39 33 6c b0 99							
▼ Internet Protocol Version 0100 = Version:							

נציג דוגמה לשיחה בין חמישה(!) משתמשים. נציג בהתחלה את פלט השרת שמסכם את השיחה. הודעתות בין משתמשים מסומנות בקו אדום, הפלט נראה כרך:

הפלט של המשתמש "Liran":

```
Connection to server established
Welcome to the chat! What is your name?
Liran
Hello Liran! Have fun chatting.
You can quit any time by typing 'QUIT'.
```

Who do you want to message?

Osher

Connection created

Hey, want to go see a movie tonight?

Osher: I was just talking with Eden about it, join us

ok I can drive us

Osher: ok I'll tell eden

back

Who do you want to message?

Noa

Connection created

WANNA COME WITH US?

QUIT

הפלט של המשתמש "Osher":

```
Connection to server established
Welcome to the chat! What is your name?
Osher
Hello Osher! Have fun chatting.
You can quit any time by typing 'QUIT'.
```

Who do you want to message?

Eden

Connection created

Want to go to the cinema later?

Liran: Hey, want to go see a movie tonight?

Back

Who do you want to message?

Liran

Connection created

I was just talking with Eden about it, join us

Liran: ok I can drive us

ok I'll tell eden

back

Who do you want to message?

eden

User Was Not Found

Who do you want to message?

Eden

Connection created

Liran can pick us up.

Eden: Ok - also Gal comes too

back

Who do you want to message?

WHO
No 3

Noa
Connection created

WANNA COME WTTH US?

quit

הפלט של המשתמשת "Eden":

```
Connection to server established
Welcome to the chat! What is your name?
Eden
Hello Eden! Have fun chatting.
You can quit any time by typing 'QUIT'.
```

Who do you want to message?
Osher: Want to go to the cinema later?

Who do you want to message?

Gal

Connection created

Are you free for a movie tonight?

Gal: Yeah sure

Osher: Liran can pick us up

back

Who do you want to message?

Osher

Connection created

Ok, also Gal comes too

back

卷之三

Liran

connection created

back

DATA

Noa

Who do you want to message:
Noa

Noa
Copi

Connection created

WANNA COME WITH US?

No.: YEAR

quit

הפלט של המשתמש "Gal":

```
Connection to server established
Welcome to the chat! What is your name?
Gal
Hello Gal! Have fun chatting.
You can quit any time by typing 'QUIT'.
```

Who do you want to message?
Eden: Are you free for a movie tonight?

Who do you want to message?
Eden
Connection created

Yeah sure

back

Who do you want to message?
Noa
Connection created

WANNA COME WITH US?

הפלט של המשתמשת "Noa":

```
Connection to server established
Welcome to the chat! What is your name?
Noa
Hello Noa! Have fun chatting.
You can quit any time by typing 'QUIT'.
```

Who do you want to message?
Liran: WANNA COME WITH US?

Who do you want to message?
Osher: WANNA COME WITH US?

Who do you want to message?
Eden: WANNA COME WITH US?

Who do you want to message?
Gal: WANNA COME WITH US?

Who do you want to message?
Liran
Connection created

back

Who do you want to message?
Osher
Connection created

back

Who do you want to message?
Eden
Connection created

back

Who do you want to message?
Gal
Connection created

quit