
פרוייקט 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	מבוא1
3	יצירת קובץ ה-CSV2
5	תיאור והסבר אריזת הפקות3
8	ניתוח התעבורה באמצעות WIRESHARK4

13 חלק שני

13	מבוא5
14	ארQUITקטורת הפרויקט6
15	6.1 – אפליקציית השרת	
20	6.2 – אפליקציית הלקוח	
24	הוראות התקנה והרצתה7
31	דוגמאות קלט ופלט8

47ביבליוגרפיה ומקורות

חלק ראשון

אריזת נתונים ולבידת מנוט ב-Wireshark

מבוא

בחלק הראשון של הפרויקט נדרשנו להכין קובץ CSV עם עמודות שונות לניטוח בסיסי של מידע. נשתמש בקובץ בתוך מחרבת ג'ופיטר כדי ליצור תעבורת. את התעבורת נלכוד באמצעות Wireshark כדי לנתח את התעבורת. בחרתי ליבא הودעות של פרוטוקול DNS (Domain Name System DNS) כדי לנתח את התעבורת.

יצירת קובץ ה-CSV

בקובץ CSV שמצורף לדוח (liran_dns_input.csv) מופיעים שדות (עמודות) שמלמדות על הפקטות, בין היתר כלת: Request/Reply, גודל הפאקטה (פריים) וכו'.

כמובן שהקובץ מכיל גם את השדות להם נדרש בתיאור הפרויקט:

- msg_id - מספר סידורי המיחיד כל הודעה
- app_protocol - פרוטוקול ברמת האפליקציה אשר בחרנו, במקרה שלנו 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 באמצעות לבידת תעבורת בתוכנת Wireshark (אפשר למשול למצאו בקובץ Discord.pcap או בקובץ dns_input_capture.csv הודיעת משרות DNS של Google ושל Discord). תוכנות שרצו על המחשב שלי בעת ההקלטה).

את השדה message – הרכבתו על ידי שרשרת שדות:

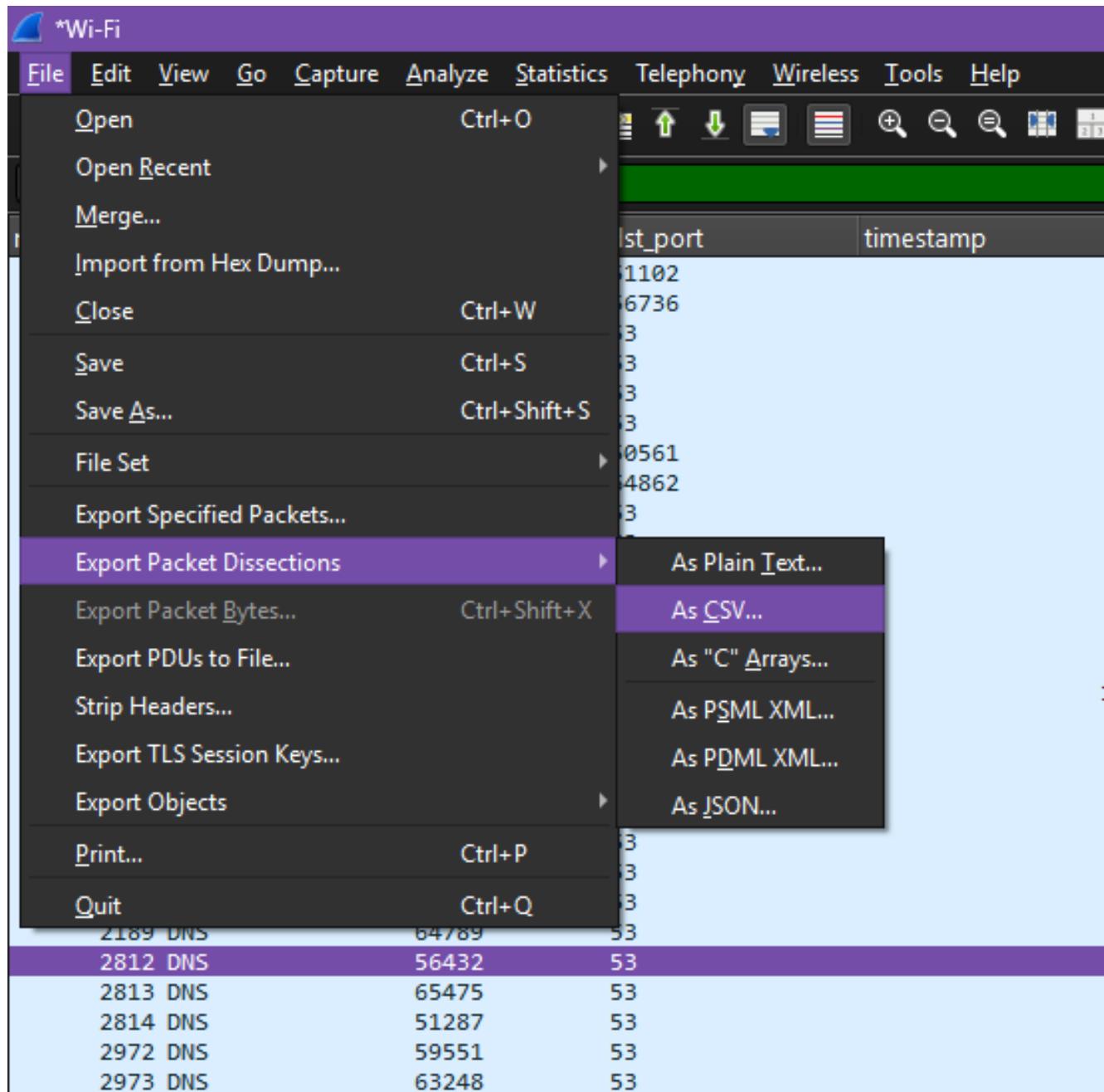
(domain name | | is Response | | Transaction ID)

כל אלו הם מאפיינים של פרוטוקול DNS. Domain name משמעו שם הכתובת של האפליקציה, is Response משמעו האם הפאקטה היא בקשה לשרת (אפס) או תגובה של השרת (אחד). Transaction ID משמעו מספר סידורי של פרוטוקול DNS עבור השאלה.

יצירת השדה message

Fields:	dns.qry.name dns.flags.response dns.id	message
destination Address	timestamp	
1.1.1	0.000108	dns.google,0,0x03d8
	0.000140	dns.google,0,0x0957
	0.000336	dns.google,0,0xb42
1.1.1	0.000071	dns.google,0,0x162f
	16.693004	dns.google,0,0x18d5
1.1.1	0.000361	dns.google,0,0x2100
	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

יצוא הפקטוט אל קובץ CSV



תיאור והסבר אריזת הפקאות

השלב השני בחלק זה הוא טיענת קובץ ה-CSV אל מחברת הג'ופיטר שמצוירת.

המטרה: להשתמש בכימוס (Encapsulation) במודל OSI כדי לאזרז את ההודעות לפי השכבות ולהעבירן. האריזות מרכיבות את הפקטה מהשכבה העליונה (האפליקציה) עד לשכבות התחתוניות. נתאר באופן כללי את תהליך האריזה ולאחר מכן נפרט.

תחילה ההודעה נוצרת בשכבת האפליקציה (לחברת הג'ופיטר טוענת את ההודעה שתישלח - payload) ולאחר מכן מועברת אל שכבת התעבורה דרך socket. במחברת - שלוחים את הודעות ה-CSV שלנו או הודעות דמה. לעומת המחברת שולחת כל payload משורות ה-CSV אל שכבת התעבורה.

שכבת התעבורה (פרוטוקול TCP) עוטפת את ה-payload ב-Segment שיפוענה על ידי שכבת התעבורה בצד מקבל. בשכבת התעבורה משתמשים לפאקטה בותרת TCP Header שמכילה:

flags source port, destination port, และ שדות נוספים נוספים לשםירה על חיבור רציף (מספר זרימה seq), בקרת זרימה (ack) ועוד. TCP הוא פרוטוקול תעבורה שמעביר מידע בצורה אמינה וקדנית. בין הערכים נוסף שדה checksum שמשמש את הצד מקבל לפענוח שגיאות. לאחר מכן שכבת התעבורה מעבירה את הסגמנט לשכבת הרשות.

שכבת הרשות עוטפת את הסגמנט ב-Datagram IP. לעומת זאת מוסיפה בותרת IP Header שמכילה: source IP, destination IP, version, TTL, ועוד. Datagram מסוגל לעبور בין רשותות שונות ולהיות מנוטבת דרך רכיבי רשות (נתבים) עד ליעד. שכבת הרשות מעבירה את ה-Datagram לשכבת הקו.

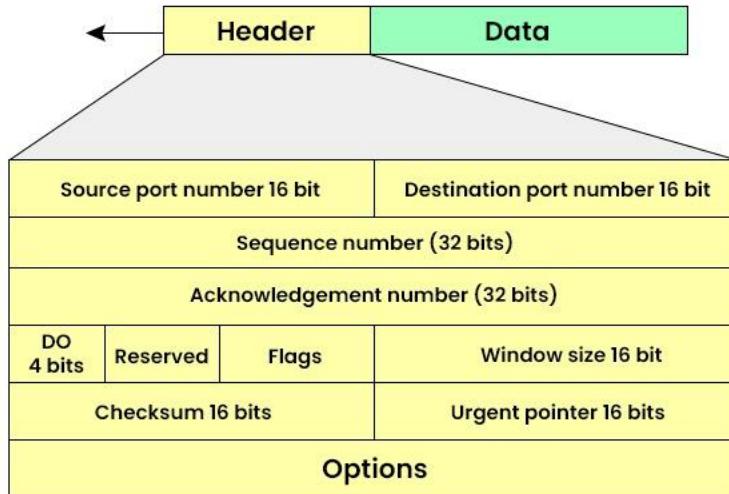
רעיניות - שכבת הקו (הליין) מוסיפה מסגרת Frame והוא השכבה האחורונה שעוטפת את ה-Datagram כדי להפוך אותה לפאקטה שתשודר פיזית (למשל דרך Ethernet, גלי רדיו, חוטי נחושת וכו'). היא מצינית בתובות MAC (מקור ויעד) ומעבירה את הפקטה לשכבה הפיזית, לעומת משדרת אותה מהמכשור החוצה.

בפועל, במקרה שלנו התהיליכים שהמחברת עושה הם מקומיים ולכן בשננתה את הפקאות לא נראה בתובות MAC אמיתיות.

תפקידו העיקרי של שכבה זו בהקשר של המחברת הוא לשדר את הפקאות אל לולאת ה-loopback, לעומת מקומית בתוך המחשב.

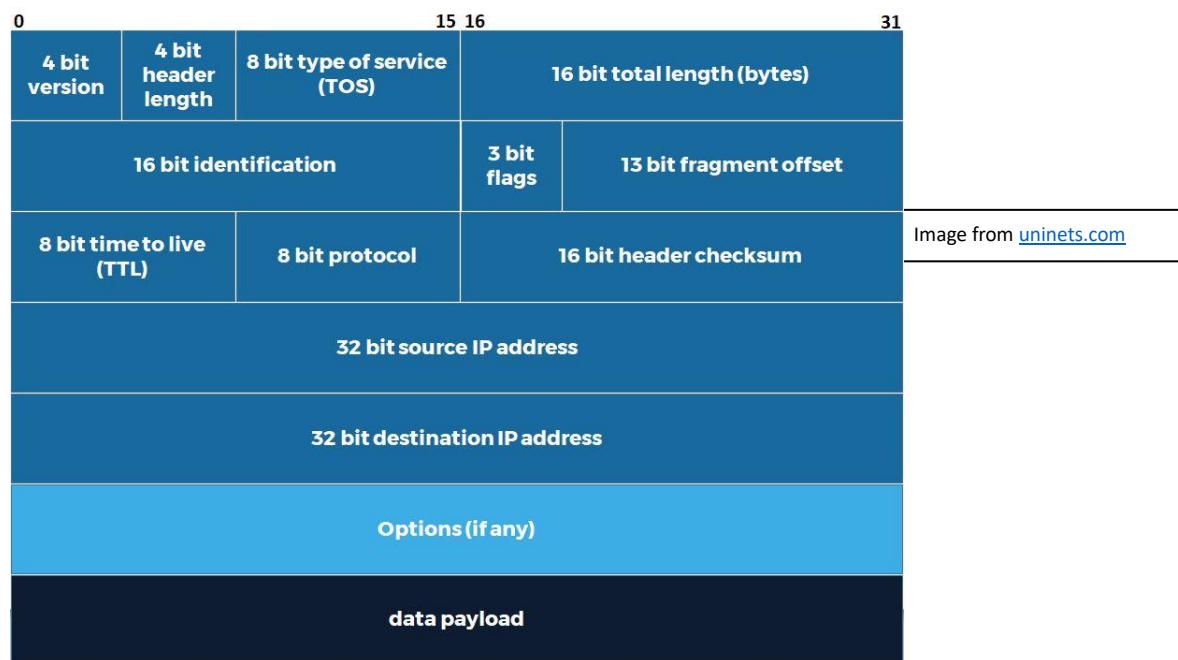
ניתן לראות איך המחברת ממליצה את האריזה לsegment ב프וטוקול TCP בהתאם לפורמט:

TCP Header Format



```
def build_tcp_header(src_ip: str, dst_ip: str, src_port: int, dst_port: int, payload: bytes=b'', seq: Optional[int]=None, ack_seq: int=0, flags: int=0x02, window: int=65535) -> bytes:  
    if seq is None: # if sequence number is not initialized  
        seq = random.randint(0, 0xFFFFFFFF)  
    doff_reserved = (5 << 4)  
    checksum_tcp = 0 #initialize checksum  
    urg_ptr = 0 # We don't have urgent data  
    tcp_header = struct.pack('!HLLBBHH', # pack an initial header before evaluating checksum  
                           src_port, dst_port, seq, ack_seq,  
                           doff_reserved, flags, window,  
                           checksum_tcp, urg_ptr)  
    placeholder = 0  
    protocol = socket.IPPROTO_TCP  
    tcp_length = len(tcp_header) + len(payload)  
    pseudo_header = struct.pack('!4s4sBBH', # pseudo_header is only used to calculate checksum, using some fields included in IP header  
                               socket.inet_aton(src_ip), socket.inet_aton(dst_ip),  
                               placeholder, protocol, tcp_length)  
    checksum = checksum(pseudo_header + tcp_header + payload) #calculate checksum  
    tcp_header = struct.pack('!HLLBBH H H', # build the complete TCP header containing all fields  
                           src_port, dst_port, seq, ack_seq,  
                           doff_reserved, flags, window,  
                           checksum, urg_ptr)  
    return tcp_header
```

ניתן לראות איך המחברת ממליצה את ארייזת שכבת הרשת בפורמט IPv4 בהתאם לשדות:



```
def build_ip_header(src_ip: str, dst_ip: str, payload_len: int, proto: int=socket.IPPROTO_TCP) -> bytes:
    version_ihl = (4 << 4) + 5 # Version = 4, Header Length = 5 words (= 20 bytes)
    tos = 0 # Type of service, not used
    total_length = 20 + payload_len # Total Length = IP header (20 bytes) + payload length
    identification = random.randint(0, 65535) # This field is used for fragmentation
    flags_fragment = 0 # don't use fragmentation
    ttl = 64 # The Maximum amount of hops the packet can traverse
    header_checksum = 0 # initialize checksum
    src = socket.inet_aton(src_ip)
    dst = socket.inet_aton(dst_ip)
    ip_header = struct.pack('!BBHHHBBH4s4s', # pack an initial header before evaluating checksum
                           version_ihl, tos, total_length, identification,
                           flags_fragment, ttl, proto, header_checksum,
                           src, dst)
    chksum = checksum(ip_header) # calculate checksum
    ip_header = struct.pack('!BBHHHBBH4s4s', # build the complete IP header containing all fields
                           version_ihl, tos, total_length, identification,
                           flags_fragment, ttl, proto, chksum,
                           src, dst)
    return ip_header
```

בכך השלמנו את האזיה של שלושת השכבות, אפליקציה (הוודאות CSV) שהמחברת טעננת ושולחת, התעבורה (הסגןט, `tcp_header`) והרשות (ה-IP). כל הוועדה שנשלחה מהמחברת תיעוף בשכבות ותועבה.

ניתוח התעבורה באמצעות WIRESHARK

בשלב הראשון ננתן תעבורה 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 הפקודות:

```
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)
```

בתמונה שבחמש, מספר הפורט 25215 הוא פורט שהוקצה למשתמש (אני, שהריץ את Wireshark) ומספר הפורט 12345 הוא מספר הפורט של האפליקציה (מחברת ה-Jupyter)

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

זו הגדרה ל-TCP segment שבו רק הדגל SYN פעול. במלחים אחרות זו הפאקטה הראשונה בתהליך 3-way-handshake.

אבל נשים לב שלא נוצר חיבור TCP מלא, כלומר לא חזרה פאקטה ack+ack, כי במחברת שלוחים פאקטות גלמיות, `listen`, `bind`, `accept`, במלחים אחרים, ברמת האפליקציה אין ביסוס של תקשורת TCP אלא שליחת פאקטות גלמיות בין כתובות פורט כלשהי במחשב ל-loopback.

לכן נצפה של לאחר הפאקטה זו, תישלח פאקטה שתבהיר על Reset, כי אין אפשרות להתחיל חיבור ובאמת ניתן לראות של לאחר כל פאקטה כזו שנשלחת למחברת (אלו שמכילות TCP payload טווח TCP payload 0-2) מקבלים 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	

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	

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	

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	

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	

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	

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	

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	

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	

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	

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	

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	

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</td					

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

זו הגדלה לפאקטת TCP שבה רק הדגלים ack+push פעילים. כמו מקודם אין באמת תקשורת TCP בין הצדדים ולבן מצפה לקבל חזרה Reset=1. הפעם נשים לב שנקלט 0. הסיבה היא שהמחברת לא קיבלת לפני כן פאקטת SYN מהמשתמש ולבן היא לא מודעת או לא חושדת בקיום קשר.

20313	25215	12345	2116.209752	TCP	0	1	1	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	0x0018	48656c6c6f205061636b65742031
20326	12345	25215	2117.214837	TCP	0	0	0	1	0	0x0004
20331	25215	12345	2118.217473	TCP	0	1	1	0	0x0018	48656c6c6f205061636b65742032
- 20332	12345	25215	2118.217552	TCP	0	0	0	1	0	0x0004

שאר הפקודות מתנהגות באופן דומה למעט האחרונה:

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

זו פאקטה שבה רק הדגל Reset פעול. בשפакטה זו נשלחת אפיו לא נקלט פאקטות בתגובה מהמחברת. הסיבה היא שפакטה מהסוג זהה לא מצפה לתגובה בחזרה אלא מבקשת מהמחברת לסגור את החיבור הקיים. מכיוון שלמעשה אין חיבור קיים בכל מקרה, לא נקלט תגובה מהמחברת.

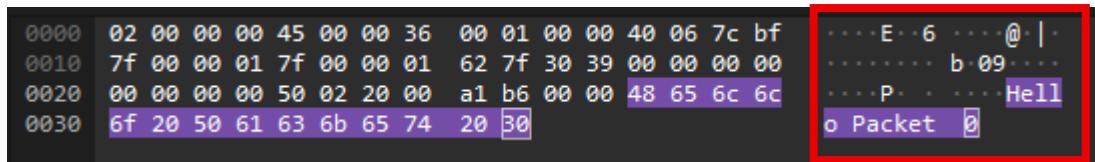
33219	25215	12345	3169.293865	TCP	0	0	0	1	0	0x0004
33432	25215	12345	3170.296178	TCP	0	0	0	1	0	0x0004
- 33440	25215	12345	3171.299444	TCP	0	0	0	1	0	0x0004

Sequence Number: 0 (relative): 0000	02 00 00 00 45 00 00 36 00 01 00 00 40 06 7c bfE...6 ...@ .
Sequence Number (raw): 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 50	o Packet []
Acknowledgment number (raw):		

התמונה הבאה מדגימה כיצד נראה ה-TCP סגמנט של הפאקטות כפי שמיוצגות ב-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]
```

ניתן לראות גם את ה-TCP Payload (ההודעה) של הפאקטות (בתמונה –



:CSV בשלב השני ננתח תעבורת ממחברת הgl'ופיטר שהתקבלת בתוצאה מהעברת הודעות-

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 שמכילות את הודעות CSV עם הדגמים ack=1,push=1

אכן ניתן לראות שכל הפקטים שמתאפשרים בעלות TCP סגמנט עם אותם דגלים :(ack+push)

```

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.... .... = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....AP...]
  Window: 8192
}
  }
```

קובץ הלכידה (מצורף בGITהאב: jupyter_send_csv_capture.pcap) לאחר שליחת הודעות ה-CSV
בתוך המחברת נראית 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	0	1	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	0	1	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	0	1	0x0004	
33	25215	12345	1.680408	TCP	0	1	1	0	0	0x0018	646973636f72642e636f6d2c302c307838306634
34	12345	25215	1.680540	TCP	0	0	0	0	0	0x0004	

וגם ניתן לראות שככל פאקטה ששלוחים נשאת את שדה message של השורה המתאימה בקובץ CSV. למשל, הפאקטה הראשונה נשאת את ההודעה:

0000	02 00 00 00 45 00 00 49	00 01 00 00 40 06 7c ac
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	de f8 00 00 69 70 76 36
0030	2e 6d 73 66 74 63 6f 6e	6e 65 63 74 74 65 73 74
0040	2e 63 6f 6d 2c 30 2c 30	78 62 62 64 30

E	I	@	.
.	.	b 09
..P.	ipv6	
.msftcon	necttest		
.com,	0,0	xbbd0	

נחזיר לקובץ CSV (מצורף בGITLAB: liran_dns_input.csv) ונשים לב שגם השורה הראשונה בקובץ נשאת את שדה message:

message
ipv6.msftconnecttest.com,0,0xbbd0

הפאקטה ושורת CSV שתיהן נשאות את אותה הודעה:

Ipv6.msftconnecttest.com,0,0xbbd0

ולכן שורה מספר 1 בקובץ CSV מתאימה לפאקטה מספר 1 בקובץ הלכידה.

דוגמה נוספת, בקובץ הלכידה (jupyter_send_csv_capture.pcap), פאקטה מספר 63 נשאות את שורה נוספת בקובץ CSV מטביעה לפאקטה מספר 1 בקובץ הלכידה:

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 67 6c 65 2c 31	2c 30 78 36 30 61 66

E	;	ו	.
.	.	b 09
..P.	..},..dns.		
google,1 ,0x60af			

נחזיר לקובץ CSV ונראה שורה מספר 32 נשאות את שדה message:

AAAA	70	dns.google,0,0x60af	0.00021	1.1.1.1	192.168.1.	53	59365	DNS	30
A	70	dns.google,0,0x0957	0.00014	1.1.1.1	192.168.1.	53	64789	DNS	31
AAAA	130	dns.google,1,0x60af	0.026362	192.168.1.1.1.1		59365	53	DNS	32
HTTPS	150	dns.google,1,0x41f8	0.001168	192.168.1.1.1.1		58331	53	DNS	33
A	106	dns.google,1,0x0957	0.008372	192.168.1.1.1.1		64789	53	DNS	34
HTTPS	90	dns.google,0,0x4261	2.948365			53	56432	DNS	35

הפאקטה ושורת CSV שתיהן נשאות את אותה הודעה:

dns.google,1,0x60af

ולכן שורה מספר 32 בקובץ CSV מטביעה לפאקטה מספר 63 בקובץ הלכידה.

באופן דומה ניתן להזות ולשייר כל הודעה בקובץ CSV גם בקובץ הלכידה.

חלק שני

פיתוח יישום רשות ללקוח/שרות ב프וטוקול TCP

מבוא

נדרכנו לבתוב פרויקט המימוש צ'אט רב משתמשים שבו קיימים לקוחות שמתקשרים אחד עם השני דרך שרת שמנהל את החלפת המסרים ביניהם.

יש לציין: דרישות הפרויקט מבקשות מהמשתמש לצין לקוח אליו רוצה לפנות כדי לשלוח אליו טקסט: "באשר השירות קיבל פניה לקוח השירות צריך לפתח צ'אט בין הלקוח ללקוח אחר. הפניה צריכה להכיל את שם הלקוח המבוקש. ניתן להניח שלכל לקוח יש שם ייחודי".

בולם, משתמש יוכל לשולח הודעה רק למשתמש אחד שהוא מצין במפורש את שמו. ניתן היה לשנות את השירות בדריכים אחרים כך שיעביר למשל הודעה לכל המשתמשים, או יעביר רק ללקוחות מסוימים. ניתן להפוך את הצ'אט למשל לקובוצתי, "כולם עם כולם". הנΚודה היא שהצ'אט עלול להיראות שונה בהתאם להעדפה. (צ'אט "אחד על אחד", צ'אט "כולם על כולם", וכו'). באנו הדרישה באהה לידי ביתוי - הפרויקט שיתווסף מכאן והילך, השירות והאפליקציה יחוורו בהמשך עוקבות בדיקות אחרי הדרישה - להעביר הודעה מלוקח אחד ורק ללקוח אחר עם שם שינוונן מראש.

לדו"ח זה מצורפים שתי אפליקציות שמרכיבות את הפרויקט, האחת היא עבר הלקוח ששולח בקשה לשרת ומתקבל מסרים מהשרת. השנייה היא עבר השירות. השירות מקבל קלטים ממספר לקוחות ויודע לנוהל אותם, יודע להעביר הודעה מלוקח פונה א' אל לקוח נמען ב' ומתקשר הוא עצמו עם הלקוחות. את היישום בחרתי לבתוב בשפת C++ ובסביבת Visual Studio 2022.

בנוסף לדו"ח מצורפים האפליקציות בתיקיות:

- Server_interface
- Client_interface

הAPPLICATION שינה בין לקוחות נוצרה לфункциית השירות,client_interface היא אפליקציית הלקוח. כדי להריץ שיחה בין לקוחות נדרש להפעיל מספר מופעים של client_interface (נפתח את האפליקציה כמה פעמים בחלונות שונים של ויזואל סטודיו). כל קבץ הקוד בשתי האפליקציות server_interface,client_interface מלאוים בתיעודים ובהסבירם.

ארכיטקטורת הפרויקט

כדי ליצור תקשורת בין יישומים בשפת ++C אנחנו משתמשים בספרייה WinSock2 שמאפשרת לנו גישה לכל פעולות ה- Sockets להן נידרש כדי ליצור את התקשרות. באשר מדובר בתקשורת מבוססת sockets אנחנו נדרשים לביצוע תהליכיים טכניים באפליקציות לפני שנתחילה להעבר הودעות.

התהליכיים השונים במעט בין הלוקה לשרת, ואלו הם:

מעבר הלוקה נבצע:

- אתחול ספריית Winsock
- יצרת socket של הלוקה (פונקציית יצירת socket)
- ציון בתובת השרת ויצירת חיבור TCP בין socket הלוקה לבין השרת (פונקציית connect)
- ניהול תקשורת עם השרת באמצעות שליחת וקבלת הודעות (recv, send)
- בסוף השימוש, קרי התنتקות של הלוקה מהשרת או יציאה מהצ'אט, נסגור את ה-socket (פונקציית closesocket)
- קריאה לפונקציית cleanup של ספריית Winsock

מעבר השרת נבצע:

- אתחול ספריית Winsock
- יצרת socket השרת שיAzin ויתקשר עמו ללקוח (פונקציית socket)
- קישור ה-socket המAzin אל כתובת הלווגית 0.0.0.0, ואל מספר הפורט שננקזה לו (פונקציית Bind)
- רק לאחר שייצרנו את ה-socket וקשרנו אותו לכתובת שלו, נתחילה בהאזנה ללקוחות נוספים.
- (פונקציית listen)
- בעת האזנה, קיבל לקוחות חדשים ונשמר אותם בראשימה שתפורט בהמשך (פונקציית accept)
- לאחר קבלת לקוח נתקשר אליו באמצעות קבלה ושליחת הודעות (recv, send).
- **חשוב לציין** שבכל לקוח פועל בצורה א-סינכרונית. כלומר הפעולות של לקוח אחד אין תלויות בזמן הפעולה של לקוח שני. התהליכיים של כל לקוח קורים במקביל, שכן גם השרת מטפל בכל לקוח במקביל.
- באופן עקרוני השרת אמר לעובוד תמיד (ריצה אינסופית) כי בלבדו הצ'אט לא קיים, בשונה מה לקוח שבן אמר להפסיק מהצ'אט בשלב מסוים (לא רץ אינסופית). עם זאת, אנחנו מתעסקים עם sockets ולבן למען הסדר הטוב נציין שיש לסגור את ה-socket בסוף השימוש. (פונקציית closesocket).
- בהמשך לסייף הקודם, קריאה לפונקציית cleanup של ספריית Winsock

כאמור הפרויקט מורכב ממשתי אפליקציות

נפרט על אפליקציית השרת ולאחר מכן על אפליקציית הלוקה.

– אפליקציית השרת – Server interface

תיקיה זו מורכבת מ-4 קבצי header שם מוחזים הפונקציות עם תיעודים כלליים ומ-4 קבצי cpp גם הם עם תיעודים מפורטים. פירוט רחוב נמצא בתיעוד שבפרויקט, כאן נתאר את ה-headers:

– client_info.h

בקובץ זה מוגדר class עם פרטיהם על הלוקו. אנחנו שומרים את השם של הלוקו (השם ייחודי לכל לוקו – דרישות הפרויקט אומרות זאת), ואת ה-socket של הלוקו.

נגידר גם פונקציה למציאת לוקו בודד בהתאם לקוביות בהתבסס על השם שלו. כפי שניתן להבין, בסיס הנתונים שבו נשמר את הלוקות יהיה רשיימה.

נגידר גם פונקציה למחיקת הלוקו מהרשימה (במקרה של יציאה מהצ'אט).

ונגידר פונקציה לקבלת הודעות מהשרת, כמו למשל Socket השמור של הלוקו באמצעות הפונקציה הזו.

```
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 class ClientInfo {
12     public:
13         string name;
14         SOCKET socket;
15
16     public:
17         ClientInfo();
18         ClientInfo(string& name, SOCKET socket);
19         bool operator==(const ClientInfo&);
20
21     // A method to find a specific client in a list, We are given that all client names are unique
22     // This is used when a client specifies the name of the target they wish to message.
23     static ClientInfo* findByName(const string&, list<ClientInfo>&);
24
25     // Given a message, deliver it to this client via the socket with "send" function. true for a successful send, false for a failed send
26     bool getMessage(const string& message);
27
28     // Given a list containing this client, we remove the client from the list.
29     void deleteFromList(list<ClientInfo>& all_clients);
30
31 };
```

– socket_setups.h

בקובץ זה מוגדר class עם פעולות מקידימות שהשרות מבצע לפני שהוא מתחילה לקבל לקוחות.
כלומר בשתמש בפונקציות שבו כדי לאתחל את ספריית Winsock, לאתחל socket עבור השירות
ולקשרו אותו לבתוות והפורט שלו.

```
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 to 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 We would close the socket and clean up too, but the server never stops running anyway so it's not crucial to mention
22
23 The goal: Use 'recv' to get inputs from the users, Use 'send' to instruct users and
24 forward their messages onwards
25 */
26 class SocketSetups {
27 public:
28     // step 1: Initialize WinSock version 2.2
29     static bool initialize();
30
31     // step 2: Create a listening TCP socket for the server to accept clients
32     static SOCKET createTCPsocket();
33
34     // step 3.1: Create the server's Address details, ip and port
35     static 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     static bool assignAddress(const SOCKET& listenSocket, sockaddr_in& serverAddr);
39 };

```

– interact_with_client.h

בקובץ זה הפונקציה `interact` נפעיל את הפונקציה זו באמצעות `thread` כדי לנהל את התקשרות עם כל הלקוחות א-סינכראונית. הפונקציה נעזרת בפונקציות אחרות וממשת לוגיקה והעברת הודעה בין השירות ללקוח.

פונקציות העזר הן:

`equalStrings`: משווה מחרוזות בלי חשיבות לאותיות קטנות וגדלות. היא לא מהותית בלוגיקה, היא נעודה לאפשר למשתמש להקליט קלט נוח יותר (למשל "Tזעפ" במקום "QUIT")

`isQuit`: בודקת אם ההודעה של הלקוח היא "Quit" ואם כן מנטתקת אותו

`validateTarget`: בודקת אם המשתמש אליו רוצים לשלוח חוקי ומחזירה פידבק למשתמש.

`getTarget`: אם המשתמש הרצוי מצוהה על ידי השירות תחזיר אותו

```
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  enum Session {
10    REGISTER,
11    SEARCH_TARGET,
12    DELIVER_TO_TARGET
13  };
14  /* Given 2 strings, compare them case insensitively(for example "liRaN" == "LIRan") */
15  bool equalStrings(const string& str1, const string& str2);
16
17  /* Given any client's message to the server we first have to check if he entered 'quit'
18   * If he did, then we notify the client on his disconnection and return true
19   * If he didn't, we return false */
20  bool isQuit(string& message, ClientInfo& client);
21
22  /* Given a client's input of a target he wants to message, the server checks if the target
23   * is valid and sends feedback to the client. Errors to be detected are:
24   * 1. target is the client himself (Logical error, client is trying to message himself).
25   * 2. target doesn't exist (technical error, the target can not be found)
26   * If there is no error we notify the client that we found the target, else we
27   * send the error to the client. */
28  bool validateTarget(string& targetName, ClientInfo& client, list<ClientInfo>& all_clients);
29
30  /* Once the target is validated, we need to make sure they're still active on send.
31   * If the target has disconnected mid conversation we notify the client that the connection is closed
32   * and ask him to pick a different target */
33  ClientInfo* getTarget(string& targetName, ClientInfo& client, list<ClientInfo>& all_clients);
34
35  /* 'Interact' is the main function that manages the interactions between the server and the client.
36   * It Handles the server's actions with a single client and forwards messages from the client
37   * to a target client.
38   * If needed, the server sends feedback messages regarding errors.
39   * Interact utilizes all the functions from above.
40
41   * In short, we do the following:
42   * 0. We got a message from the client
43   * 1. Check for 'quit' (isQuit), if the client didn't send quit we proceed to 2., else we stop running
44   * and remove him (deleteClient).
45   * 2. Use a 'mode' variable that gets 0, 1 or 2 to distinguish between actions as followed:
46   * mode = REGISTER -> We receive client's name and save his details (createClient)
47   * mode = SEARCH_TARGET -> we receive the target client and check for errors (isError)
48   * mode = DELIVER_TO_TARGET -> We got a message to deliver and we send it to the target client by finding his socket in the clients list
49   * 3. return to 0.
50
51  */
52  void Interact(SOCKET clientSocket, list<ClientInfo>& all_clients);
53
```

- main.h

בקובץ זה פונקציית `main`, בה אנחנו מקיים את השירות עם כל פעולות ה-socket המקידימות. ככלומר אנחנו משתמשים בקלאס `socketSetups` כדי להקים את השירות ובפונקציות `listen`, `accept`, `interact`.

כדי לקבל לקוחות. כל לקוח שמתබל ינהל בנפרד באמצעות `thread` ובפונקציית `interact`.
כדי לנוול את כל הלקוחות במקביל אנחנו צריכים לשמר את כולם. ככלומר אנחנו צריכים מבנה נתונים
בלשונו כדי לנוול את הלקוחות, ובאמת כפי שנאמר אנחנו משתמשים ברשימה. בפרויקט נדרש
למבנה נתונים שיאפשר להוסיף, למחוק ולחפש לקוחות בתוך מגאר.

אפשר לבצע זאת עם המון שיטות וסוגים של מבנים. למשל אפשר לשמר עץ AVL מאוזן שמסדר את
הלקוחות לפי סדר האלף-בית, או להשתמש בדרכים מתחכבות יותר ויעילות יותר. עם זאת,
התמיקדות היא בתקשורת בין השירות לבין לקוח לנפеш מעט את המימוש ואת ההסבירים. נשמר את
הלקוחות בתוך רשימה מקוושתת (`ClientInfo`) מסוג `list`.

נתחיל את הרשימה בבקשתה, ממש כל לקוח שמתחבר לשירות יתווסף לרשימה, כל לקוח שיתנתק
מהשירות ימחק מהרשימה ואם נרצה להעביר הודעה אל לקוח מסוים אפשר פשוט לחפש אותו ברשימה
ולקבל פוינטර לאובייקט שבו מאוחסנים הפרטים שלו.

זכיר שככל הפעולות קורות במקביל בין השירות לכל הלקוחות באמצעות פיצולים של `thread`.

```
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
13    accepting clients. We check for errors in each procedure. We then create an empty
14    list of type ClientInfo in which we will store all the active clients.
15    Once a client comes in and we accept him, we send his socket to the 'Interact'
16    function from interact_with_client.h by a thread to isolate the server and him, along with a
17    reference to the list of clients we created.
18    By using threads we split the server's attention to each client in particular creating desynchronization
19    which is essential for chatting between one and another */
20 int main();
21
```

מתוך main.cpp - קבלת הלקוחות ופיזול הקשբ של השרת לבולם במקביל:

```
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 manageClient(Interact, clientSocket, ref(clients)); // Each client gets his own treatment simultaneously with threads
        manageClient.detach();
    }
}
closesocket(listenSocket);
WSACleanup();
return 0;
```

– אפליקציית הלקוח – Client interface

תיקיה זו מורכבת מ-3 קבצי header עם תיעודים כלליים ומ-3 קבצי קפ גם הם עם תיעודים מפורטים ובהם בפועל מיושמים התהילכים. פירוט רחוב נמצא בתיעוד שבספרויקט, באן נציג את קבצי .header.

– socket_setups.h

בדומה לאפליקציית השרת, גם באפליקציית הלקוח נגדי class עם פעולות מקדיימות שאנו צריכים לקיים כדי לאפשר תקשורת מבוססת sockets. בקובץ מופיע מימוש בסיסי של אתחול ספריית Winsock, יצירה socket הלקוח 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 communicating 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  class SocketSetups
24  {
25  public:
26      // step 1: Initialize WinSock version 2.2
27      static bool initialize();
28
29      // step 2: Create a TCP socket to contact the server with
30      static SOCKET createTCPsocket();
31
32      // step 3.1: Create the address details of the server the client connects - ip and port
33      static bool createAddress(sockaddr_in& serverAddr, const string& ip, const int portNumber);
34
35      // step 3.2: Connect to the server using the 'connect' function
36      static bool connectToServer(SOCKET& serverSocket, sockaddr_in& serverAddr);
37  };
```

- interact_with_server.h

בקובץ זה הפונקציות .sendMessage, receiveMessage

פונקציות אלה מרכזות את הלוגיקה של הלוקה בתקשורת עם השירות. פונקציה אחת אחראית על קבלת הודעות מהשירות והשניה על שליחת הודעות לשירות. הפעולות של הפונקציות משפיעות זו על זו והן מתואמות ביניהן. הן משתמשות בפונקציות אחרות כדי לנהל את התהליכים מול השירות.

פונקציות העזר הן:

: כמו קודם, משווה מחרוזות בלי חשיבות לסוג האות. כדי להקל על קלט המשתמש.

: בשלב ההתחברות לשירות, הפונקציה שולחת לשירות את שם הלוקה.

: כזכור הלוקה צריך לציין את שם הלוקה אליו רוצה לשלוח הודעות, פונקציה זו שולחת לשירות שם זה.

```
4 #include <iostream>
5 #include <thread>
6 #include <string>
7
8 using namespace std;
9
10 enum Session {
11     REGISTER,
12     SEARCH_TARGET,
13     CHAT
14 };
15 /* Given 2 strings, compare them case insensitively(for example "LiRaN" == "LIRan") */
16 bool equalStrings(const string&, const string&);
17
18 // Send client's name to the server to be known. Return success (true) or fail (false)
19 bool registerClient(SOCKET& server);
20
21
22 /* Send name of another client we want to message. Server will respond with a feedback message.
23    Return success (true) or fail (false) */
24 bool reachTarget(SOCKET& server);
```

```

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

```

- main.h

בקובץ זה פונקציית ה-main, בה אנחנו מבצעים את כל פעולות ה-socket המקדים. בלומר אנחנו משתמשים בפונקציות של קלאס SocketSetups. חשוב לציין – אנחנו יוצרים שני thread כאשר פונקציות thread-interact_with_server.h: sendMessage ,receiveMessage הן.

שילוב שני threads מאפשר לנו לשלוח הודעות ולקבל הודעות בו זמנית.

```
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
```

מתוך main.cpp – חיבור שני הפונקציות כדי לתקשר עם השרת:

```
// Connection is created, we're ready to communicate with the server
cout << "Connection to server established" << endl;

// We split the application's attention to both send messages to the server and receive messages from the server, simultaneously
thread senderthread(sendMessage, clientSocket);
thread receiverthread(receiveMessage, clientSocket);
senderthread.join();
receiverthread.join();

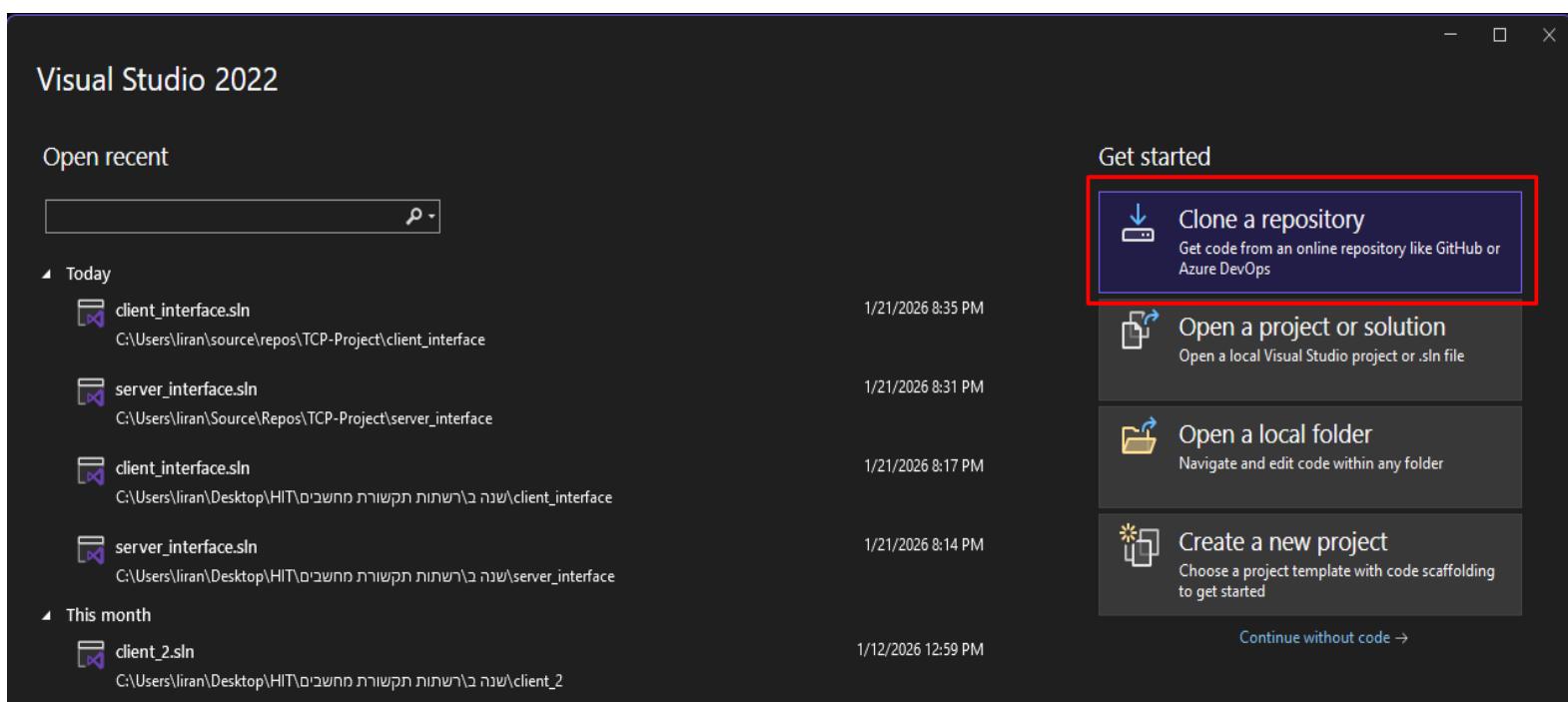
}

// When both threads finish their run (by typing 'Quit' or by server connection error)
// in other words when the connection to the server stops, the program ends.
closesocket(clientSocket);
WSACleanup();
return 0;
```

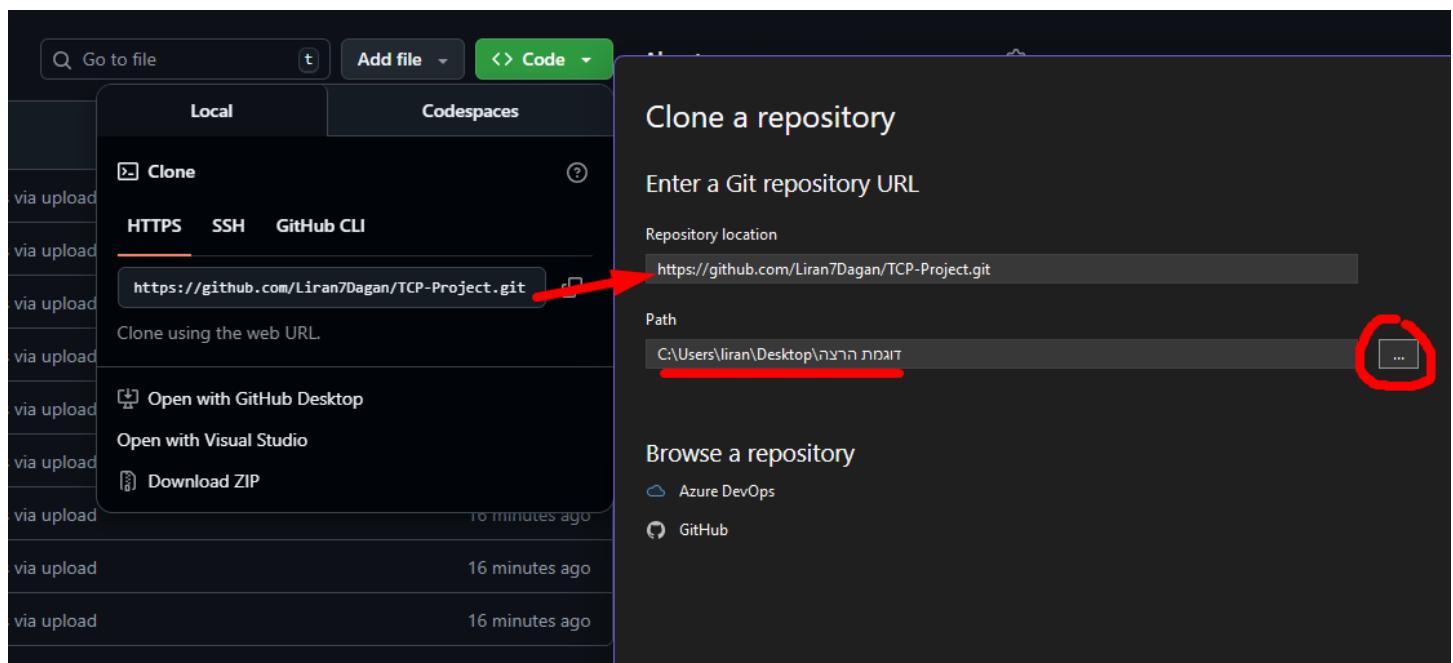
הוראות התקנה והרכבה

LAGIETHEAB מצורפים התייחסות שצינו client_interface, server_interface קודם לכך. כאמור שתי האפליקציות נכתבו בשפת C++ ולכן להריץ אותן על המחשב יש לפתח את קבצי ה-solution שליהן בזיאול סטודיו.

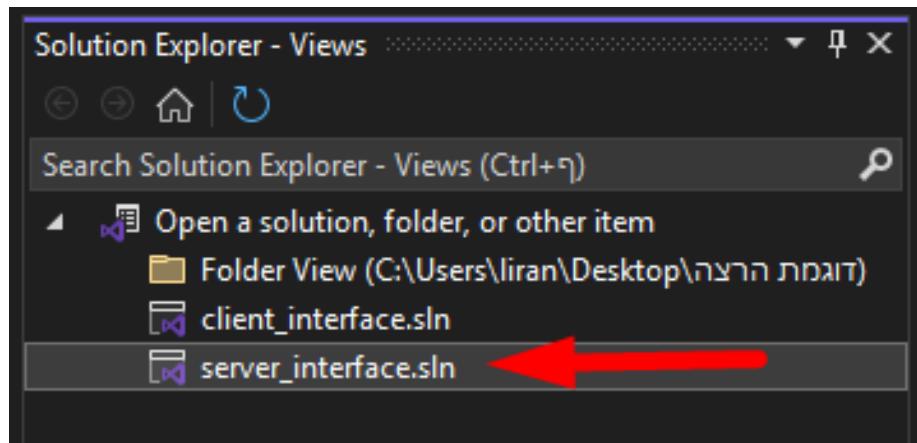
:clone a repository פותחים את ייזאל סטודיו ולוחצים



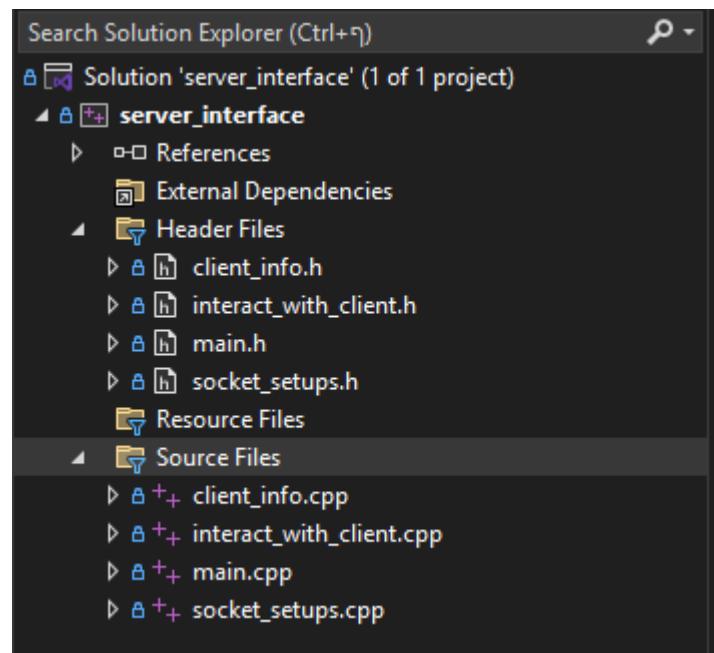
מעתיקים את הlienק של ספריית הגיטהאב, חשוב לשמוך בנתיב משמעותי על המחשב כי נשתמש בתיקייה הזאת שוב



נלחץ **clone** ומכאן אפשר לפתח את אפליקציית השירות:



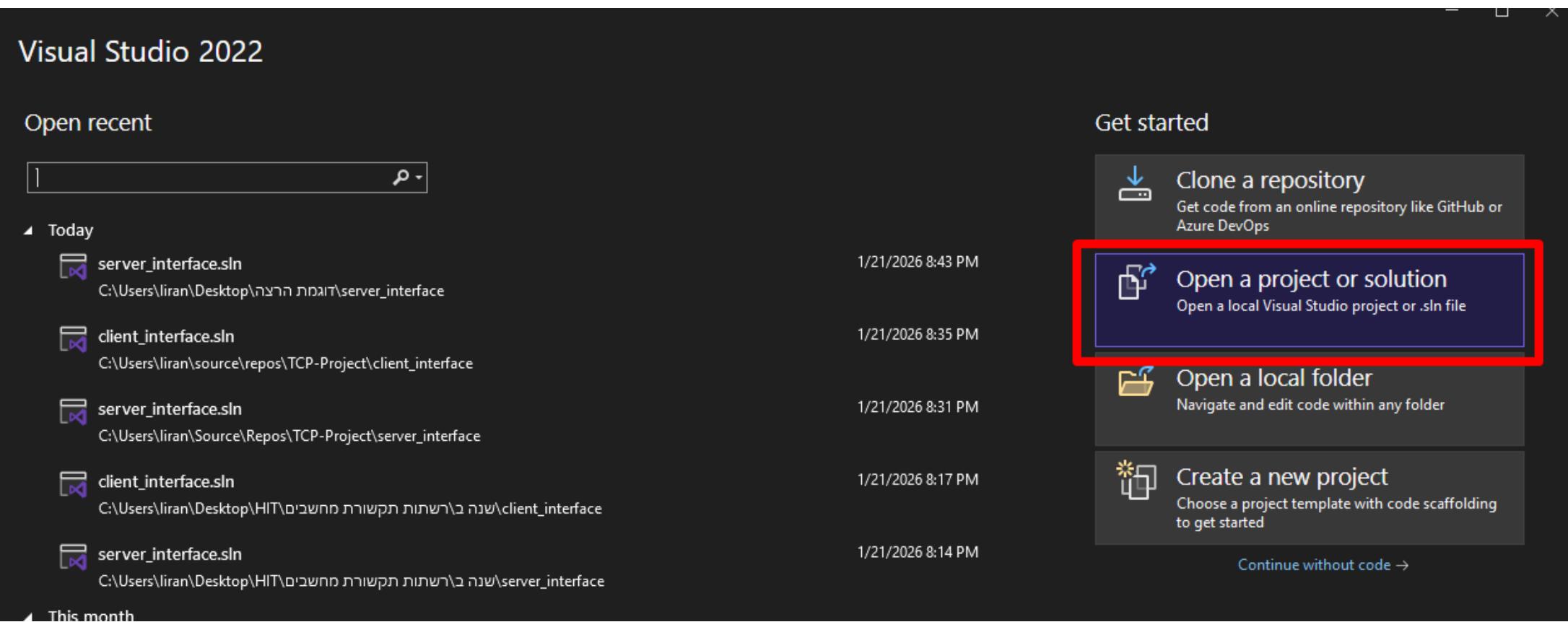
בשלב זה ניתן לפתח ולראות את כל קבצי הקוד: ה-.h ו- .cpp של השירות בתוך ויזואל סטודיו:



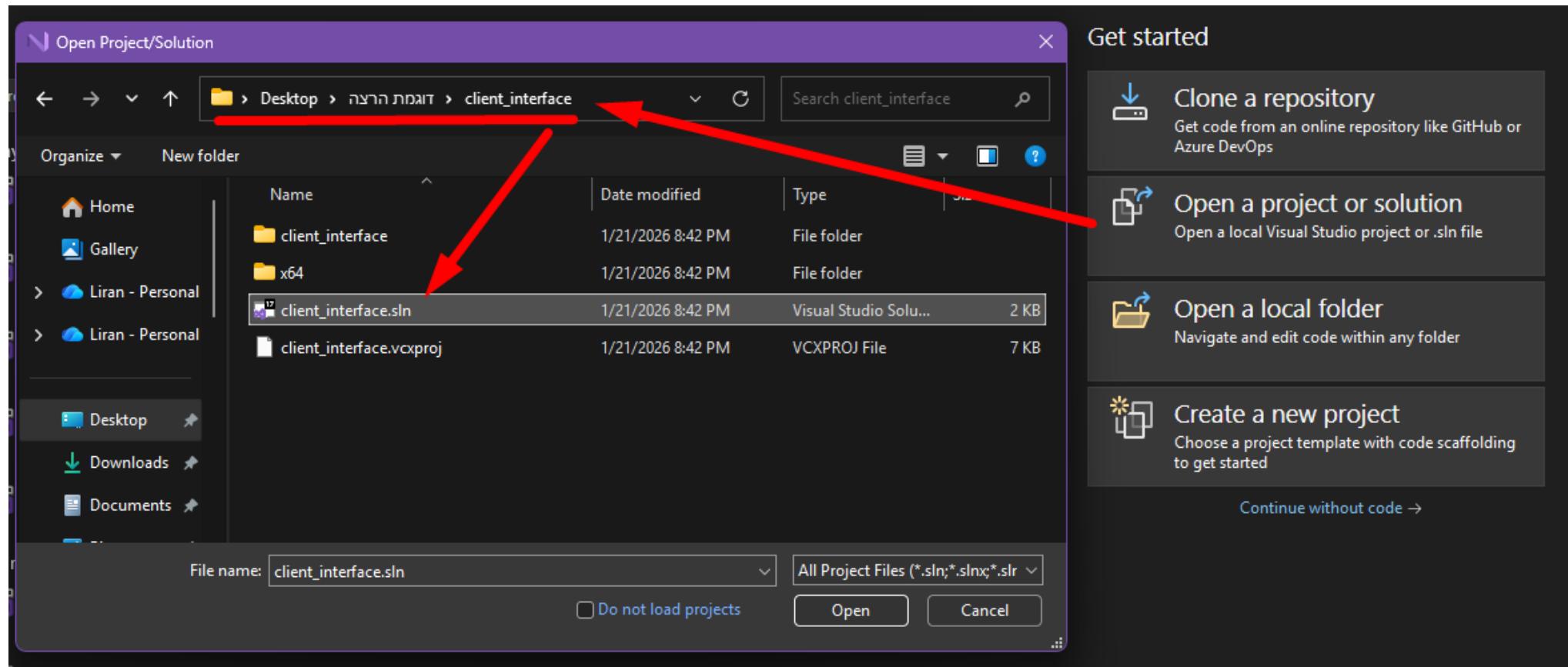
עכשו צריך לפתח את אפליקציית הלוקוח, בשבייל כך צריך לפתח חלון נוסף של ויזואל סטודיו:



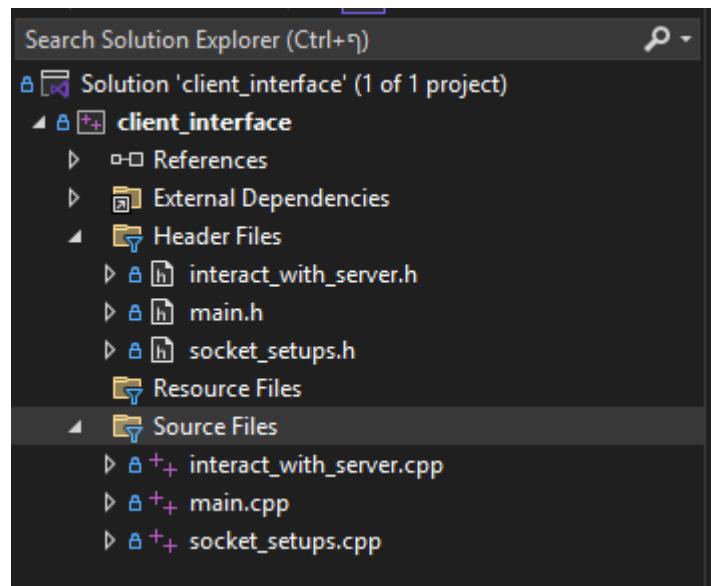
יש לנו חלון חדש של ויזואל סטודיו - נפתח את קובץ ה-solution של אפליקציית הלקווט בנתיב ששמרנו:



פתיחת אפליקציית הלוקוט:

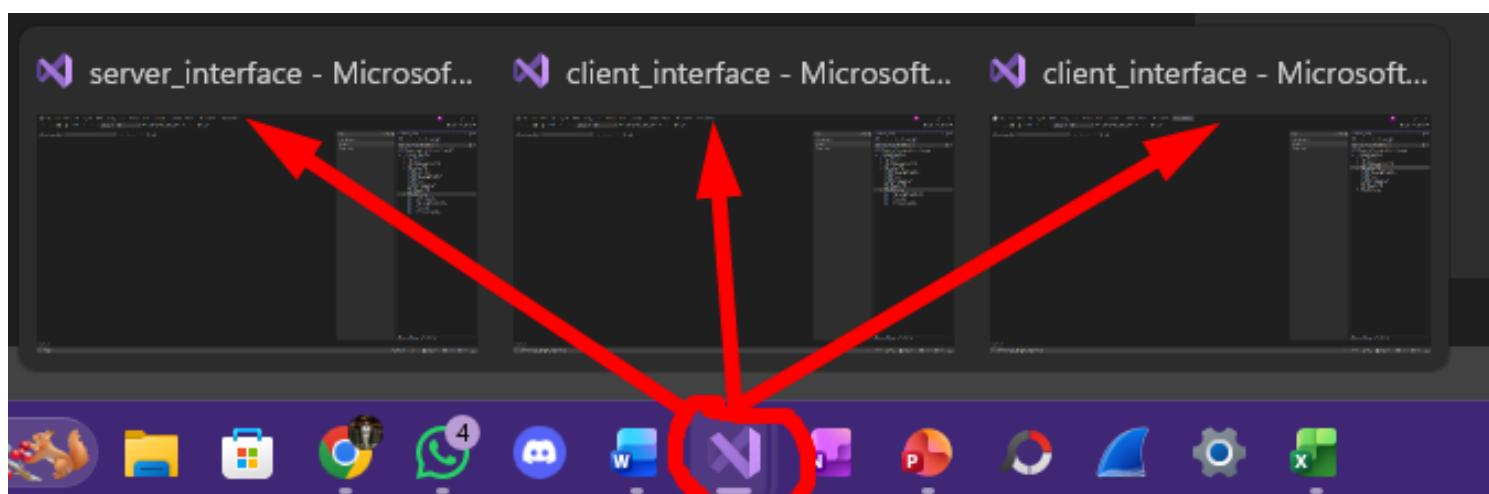


בשלב זה ניתן לפתח ולראות את כל קבצי הקוד: ה-.h ו-ה-.cpp של הלוקוט בתוך ויזואל סטודיו.



אם נרצה להריץ צ'אט יש לפתח לפחות עד לקו אחד באתנו אופן שתואר בשני העמודים האחרונים.

עד לקבלת שרת אחד ומספר בלחשו לבחירתנו של לקוחות, ככה:



באן מסתיים שלב ההתקנה, יש לנו שרת אחד ולפחות שני לקוחות שמוכנים להריצה.

שგיאה צפוייה היא שגייה מהסוג:

```
C:\Users\liran\Desktop\HIT\> Server is now listening on port 12345
```

```
Microsoft Visual Studio Debug> 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 . . .|

השגיאה תופיע לאחר זמן מה שהל庫ום לא הצליח להתחבר לשרת.
הסיבה לשגיאה: באפליקציית הלוקום, בקובץ main.cpp, שורה 23:

```
1      /* Made by Liran Dagan 215609397 */
2      #include "main.h"
3
4      int main()
5      {
6          // step 1: Start winsock
7          if (!SocketSetups::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 clientSocket = SocketSetups::createTCPsocket();
14         if (clientSocket == INVALID_SOCKET)
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 or change if needed
25
26         const int port = 12345; // Our assigned port number
27         if (!SocketSetups::createAddress(serverAddr, ip, port))
28         {
29             cout << "Setting address failed " << WSAGetLastError() << endl;
30             closesocket(clientSocket);
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 באפליקציית הלוקום לזה של מחשב הבודק (או כתובת ה-IP של המחשב עליו רץ השירות).

ישנה גם אפשרות להשתמש בכתובת ה-loopback במקומ, אם רצים. אני השתמשתי בכתובת IP אמיתית, מומלץ להשתמש בכתובת אמיתית אבל הוסףתי גם את הלוופבאק בשבייל הנוחות. יש להריץ את אפליקציית השרת ראשונה, ורק לאחר מכן את אפליקציות הלוקום.

אם הרצה תקלה בסדר הפור (קודם לקובץ ואז שירות), השירות יעבוד כרגע אבל הלוקום יקבל את ההדפסה (שהיא חלק מהקובץ): Connection failed to establish: 10061 (השרות הורץ רק אחרי הלוקום ולבן בזמן הרצת הלוקום, השירות לא נמצא). וכן לא ניתן להתחבר אליו. (השרות הורץ רק אחרי הלוקום ולבן בזמן הרצת הלוקום, השירות לא קיים).

דוגמיה:

The screenshot shows two terminal windows side-by-side. The left window, titled 'C:\Users\liran\Desktop\' (though the path is partially cut off), displays the message 'Server is now listening on port 12345'. The right window, titled 'Microsoft Visual Studio' (though the path is partially cut off), displays an error message: 'Connection failed to establish: 10061' followed by the text '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 in the right window towards the explanatory text below.

בנוסף, האפליקציות מטפלות בשגיאות (כמו זו שבדוגמה) ולכן קוראות קריסות בלתי צפויות. אם קוראות שגיאות כלשהן כמו התנתקויות למשל, האפליקציות ידפיסו על המסר הודעות מתאימות.

המסר שמקבלים מאפליקציית השרת הוא נטו אינפורטטיבי, והוא לא מקבל קלט מהמשתמש.
המסר שמקבלים מאפליקציית הלקוּח לעומת זאת הוא כמפורט גם אינטראקטיבי.

דוגמאות קלט ופלט

דוגמה לשיחה בין שני משתמשים, לiran ומאור:

The image shows two side-by-side Microsoft Visual Studio windows. Both windows have a title bar labeled 'Microsoft Visual Studio Debug' and a close button 'X'. The left window contains messages from 'Liran' and the right window contains messages from 'Maor'. Both users are connected to a server and introduce themselves. They then attempt to message each other but receive a 'User Was Not Found' response. They eventually succeed in connecting and start a conversation about a football game between Real Madrid and Barcelona. Maor invites Liran to watch the game together, and they agree to meet at his place in 15 minutes. Both users then type 'quit'.

User	Message	
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? 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	----- 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!!! ----- 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

עבור השיחה שלמעלה, פلت השירות הוא:

```
C:\Users\liran\Desktop\HIT\o 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
```

מנקודת המבט שלי – הכותב:) למעלה פלט השרת, משמאלי הקלט/פלט של לiran ומימין הקלט/פלט של Maor בתמונה שני מחשבים – המסך העליון (השרת) והמסך הימני (הלקוח מאור) שייכים לאוטו מחשב. המסך השמאלי (לiran) שייר למחשב אחר באוותה רשות.

```

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 uppppppppp 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

```



```

Microsoft Visual Studio Debug
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
Please Message a Different User.

Who do you want to message?
Liran
Connection created. You can message a different user by typing 'BACK'

Liran: Hey Maor!!!
Whats uppppppppp 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

```

קצת על התקשרות בין השירות ללקוחות בשיחה זו (כנרת תעבורה בהמשך, כאן אסביר על האפליקציה):

המשתמשים שלחיהם לשרת את שמותיהם "לiran", "מאור", השירות זכר ושמור אותם. בכל פעם שהמשתמשים מנסים לפנות ללקוח שהשרת 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 (שהשרת מכיר) השירות מודיע לו שנוצר חיבור וההודעות מועברות מכאן והילך אל מאור. באופן זהה השירות עשו עבור מאור כשהוא פונה אל לiran. מכאן המשתמשים משוחחים ביניהם. משתמש מתנקת מהשרת על ידי הקלט "quit", כמו שעושים מאור ולiran בהודעות האחרונות שלהם. שאלת שואלי נשאלת היא: מה היה קורה אם לiran היה שולח הודעה למאור אחרי שהוא התנקק? כמובן, מאור לא מזוהה יותר על ידי השירות ולiran בכל זאת מנסה לשלוח לו הודעות.

במקרה זהה השירות לא יזהה את מאור ולכן לiran על סיום החיבור ימחזר אותו למצב ההתחלתי, ככלומר יבקש ממנו לצין ללקוח חדש שהוא רוצה לתקשר אליו.

בתמונה הבאה ניתן לראות תהליך שבו לiran מנסה לשלוח הודעה למאור לאחר שכבר התנקק. השירות מספר לiran שאינו אפשר לשלוח את ההודעה כי השני התנקק. מיד לאחר מכן הוא ממחזר אותו לנקודת ההתחלת ומבקש ממנו שם של משתמש אחר.

```

Microsoft Visual Studio  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?
Maor
Connection created
-----  

בקשה להתחבר למאור, בשלב זה הוא קיים  

hey maor
-----  

ליין שלוח הודעה למאור, בשלב זה הוא מונתק  

The other party has disconnected.
-----  

Who do you want to message?
maor
User Was Not Found.
-----  

Who do you want to message?
Quit
-----  

Microsoft Visual Studio Debug  X + ▾
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 מתנקק תור כד שיחה  

C:\Users\liran\Desktop\HIT\??? ?\????? ????
code 0 (0x0).
To automatically close the console when debugle when debugging stops.
Press any key to close this window . . .

```

בר השירות מתמודד עם ניתוקים יזומים תור כד שיחות.

יש לציין גם כי המשתמשים לא חייבים בהכרח להתחבר זה אל זה בו זמנית, כלומר משתמש אחד יכול לקבל הודעות ממשתמש שניי ללא תלות בהתחברות שלו אליו. הוזכר מספר פעמים שהלקוחות מתנהלים אסינכראונית. היתרון שהדבר נותן הוא יכולת גמישה למשתמש לשלוח הודעות רק למי שהוא רוצה בכל רגע. דוגמה:

```
C:\Users\liran\Desktop\ X + - □ X C:\Users\liran\Desktop\HIT\ 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?
Maor
Connection created
-----
hello
Maor: HI! ההודעה של מאור

C:\Users\liran\Desktop\ X + - □ X C:\Users\liran\Desktop\HIT\ X + - □ X

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?
 רק בשלב זהה מאור מתחבר Connection created
-----
HI!
```

ננתן את השicha הראשונה (עמוד 31) בין לירן למאר ב-Wireshark. ננתן את הצד של לירן (שער במחשב אחר ברשות המשרת ומאר):

קובץ הלכידה גם מצורף בשם 2_clients_chat.pcap

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
245	21.223307	192.168.1.15	192.168.1.56	57760	12345	TCP		1	0	0	0	0	0	4	128
249	21.341685	192.168.1.56	192.168.1.15	12345	57760	TCP		1	1	0	0	0	1	4	128
250	21.341881	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	1	1	4	128
561	43.966002	192.168.1.15	192.168.1.56	57760	12345	TCP	4c6972616e	0	1	0	1	1	1	4	128
562	44.349885	192.168.1.15	192.168.1.56	57760	12345	TCP	4c6972616e	0	1	0	1	1	1	4	128
563	44.387799	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	1	6	4	128
657	63.896643	192.168.1.15	192.168.1.56	57760	12345	TCP	69747a696b	0	1	0	1	6	1	4	128
658	64.278017	192.168.1.15	192.168.1.56	57760	12345	TCP	69747a696b	0	1	0	1	6	1	4	128
659	64.286456	192.168.1.56	192.168.1.15	12345	57760	TCP	533a55736572...	0	1	0	1	1	11	4	128
660	64.339513	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	11	15	4	128
661	64.461313	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	15	11	4	128
738	78.110474	192.168.1.15	192.168.1.56	57760	12345	TCP	7961726f6e	0	1	0	1	11	15	4	128
744	78.347603	192.168.1.56	192.168.1.15	12345	57760	TCP	533a55736572...	0	1	0	1	15	16	4	128
745	78.390215	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	16	29	4	128
746	78.742892	192.168.1.56	192.168.1.15	12345	57760	TCP	533a55736572...	0	1	0	1	15	16	4	128
747	78.742974	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	16	29	4	128
844	83.144811	192.168.1.15	192.168.1.56	57760	12345	TCP	4d616f72	0	1	0	1	16	29	4	128
845	83.188005	192.168.1.56	192.168.1.15	12345	57760	TCP	533a55736572...	0	1	0	1	29	20	4	128
846	83.239717	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	20	40	4	128
918	108.988693	192.168.1.15	192.168.1.56	57760	12345	TCP	486579204d61...	0	1	0	1	20	40	4	128
919	109.157486	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	40	31	4	128
1018	126.671707	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	40	31	4	128
1019	126.717885	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	31	74	4	128
1161	136.825199	192.168.1.15	192.168.1.56	57760	12345	TCP	49276d20646f...	0	1	0	1	31	74	4	128
1163	137.108872	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	74	46	4	128
1266	157.734175	192.168.1.15	192.168.1.56	12345	57760	TCP	546865726527...	0	1	0	1	46	74	4	128
1267	158.000740	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	74	79	4	128
1386	181.944050	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	74	79	4	128
1387	181.985822	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	79	122	4	128
1510	211.550325	192.168.1.15	192.168.1.56	57760	12345	TCP	526967687420...	0	1	0	1	79	122	4	128
1511	211.757499	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	122	119	4	128
1674	235.291860	192.168.1.15	192.168.1.56	57760	12345	TCP	446f20796f75...	0	1	0	1	119	122	4	128
1675	235.392300	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	122	186	4	128
1706	245.037435	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	122	186	4	128
1707	245.079946	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	186	143	4	128
1708	245.651650	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	122	186	4	128
1709	245.651729	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	186	143	4	128
1710	245.652483	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	122	186	4	128
1711	245.652526	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	186	143	4	128
1825	269.606406	192.168.1.15	192.168.1.56	57760	12345	TCP	477265617420...	0	1	0	1	186	143	4	128
1826	269.819964	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	143	229	4	128
1950	287.024120	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72...	0	1	0	1	143	229	4	128
1951	287.069915	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	229	179	4	128
1977	291.217116	192.168.1.15	192.168.1.56	57760	12345	TCP	71756974	0	1	0	1	229	179	4	128
1979	291.260262	192.168.1.56	192.168.1.15	12345	57760	TCP	533a71756974	0	1	0	1	179	233	4	128
1981	291.285594	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	1	0	185	233	4	128
1982	291.285649	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	0	0	233	186	4	128
1984	291.727135	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1	1	0	233	186	4	128
1986	291.793995	192.168.1.56	192.168.1.15	12345	57760	TCP		0	1	0	0	186	234	4	128

הדבר הראשון שנראה לעין הוא מבון סוג התעבורה – TCP בכל הפקטות. ובאמת זה המצב מכיוון שהגדכנו את השרת והלקחות ברמת האפליקציה לבעוד עם חיבור TCP.

מזהים גם שקיימות שתי כתובות P:

– כתובת המחשב שבו רץ המשתמש לירן

– כתובת המחשב שבו רץ השירות 192.168.1.56

מזהים גם שקיימים שני מספרי פורט:

57760 – מס' הפורט של לין

12345 – מספר הפורט של השירות (הגדנו זאת בקוד)

נתיחה מכאן והילך בניתוח למספר הפורט 57760 ו-12345 במאדים, בהרASON מזהה עبور לiran והשני מזהה עبور השרת. במובן שאפשר להתייחס גם לבתוות-h-IP במאדים, 192.168.1.15 עبور לiran ו-192.168.1.56 עbor השרת.

נשים לב לחיבור של המשתמש לירן לשרת. בחבילות הראשונות נוכל בבירור לראות את ה-way handshake של פרוטוקול TCP בין המשתמש לשרת:

No.	Time	Source Address	Destination Address	Source Port	Destination Port	Protocol	TCP payload	Syn	Acknowledgment
245	21.223307	192.168.1.15	192.168.1.56	57760	12345	TCP		1	0
249	21.341685	192.168.1.56	192.168.1.15	12345	57760	TCP		1	1
250	21.341881	192.168.1.15	192.168.1.56	57760	12345	TCP		0	1

בחבילה מספר 245 קורה 245, בלוור זו חבילה שהשתתת קיבל מהמשתמש "ליין" והוא מהוויה בקשה של הלוקה לתחבר לשרת.

בכדילה מס' 249 קורה 12345 כו, source port = syn=1, ack=1, בולם זו חביבה שהשתתף שלח למשתמש "ליין" והוא מהו אישור לביקשת החיבור של הלוקו

ניתן לראות את כל ההודעות שלIRON שולח לשרת ואת כל ההודעות שהשרת שולח לIRON, למשל:

לירן שלוח את שמו לשרת כדיו להירשם לצ'אט, ההודעה הראשונה:

שים לב שהפקטה נושאת את ה-payload: "Liran", זו ההודעה הראשונה לגמרי ששלחת לשרת, וזה כי הדבר הראשון שמשתמש עשויה בשהוא פותח את האפליקציה זה להקליד את השם שלו לשרת.

השרות מעביר לליין את ההודעות של מאור, לדוגמה:

1018	126.671707	192.168.1.56	192.168.1.15	12345	57760	TCP	433a4d616f72... 0						
Identification: 0x2cff (11!	010. = Flags: 0x2, Do	0000 08 5b d6 f4 fe 1b ea 94 f6 21 f7 c8 08 00 45 00	[. . . . ! E	0010 00 4a 2c ff 40 00 80 06 4a 17 c0 a8 01 38 c0 a8 J, @ . J . . 8	...0 0000 0000 0000 = Frag	0020 01 0f 30 39 e1 a0 8a 31 d7 d6 00 b7 65 10 50 18 . 09 . 1 . . e P	Time to Live: 128	0030 10 04 99 3f 00 00 43 3a 4d 61 6f 72 3a 20 57 68 . . ? . C: Maor: Wh	Protocol: TCP (6)	0040 61 74 73 20 75 70 70 70 70 70 70 70 70 70 70 20 ats uppp ppppppp	Header Checksum: 0x4a17 [v	0050 4c 69 72 61 6e 6e 6e 6e Lirannnn

באופן דומה להתחברות, אפשר לבדוק את ניתוק המשתמש מהשרת, נשים לב לפאקטות: Ack 1981, 1984, 1982, 1986. נזכיר שבתמונה הבאה, הבית הימני הוא השדה Fin, משמאלו מושםallo Syn שמאופס בכל השורות.

1981	291.285594	192.168.1.56	192.168.1.15	12345	57760	TCP	0	1	1
1982	291.285649	192.168.1.15	192.168.1.56	57760	12345	TCP	0	1	0
1984	291.727135	192.168.1.15	192.168.1.56	57760	12345	TCP	0	1	1
1986	291.793995	192.168.1.56	192.168.1.15	12345	57760	TCP	0	1	0

אלו הפאקטות האחרונות שהוקלטו עבור המשתמש "ליין" שבפורט 57760.
 נשים לב לתהילך handshake 4 עבור הניתוק של ליין מהשרת:
 בחבילה מס' 1981 קורה fin=1, ack=1, וזה חביבה שהשרת שולח לליין. כאן השרת שולח בקשה לניתוק החיבור. (בתגובה לפאקטה קודמת שבה ליין מבונן שלח הודעה quit לשרת).
 בחבילה מס' 1982 קורה fin=0, ack=1, וזה חביבה ששלוח ליין לשרת. כאן ליין מאשר את בקשת השרת לניתוק החיבור.
 בחבילה מס' 1984 קורה fin=1, ack=1, וזה חביבה ששלוח ליין לשרת. כאן ליין שלוח בקשה לניתוק החיבור.
 בחבילה מס' 1986 קורה fin=0, ack=1, וזה חביבה ששלוח השרת לליין. כאן השרת מאשר את הבקשה של ליין והחיבור נסגר סופית. זו הפאקטה الأخيرة שעבירה.
 בשכבות הרשת, כאמור כתובות ה-IP שונות עבור ליין והשרת כי הם רצים במכשירים שונים על אותה רשת.
 יש לציין שבשונה מהריצה של כל האפליקציות על אותו המחשב ועל כתובות ה-loopback, השתמשנו כתובות IP אמיתיות של מחשבים שונים כדי לוודא את ההריצה עבור מכשירים שונים.
 את השרת הגדרנו כתובות 0.0.0.0, כלומר אפשרנו לו להאזין לכל כתובות ה-IP של השרת המקומיית ולקלל לקוחות לפי כתובות ה-IP שליהם.
 הכתובת 0.0.0.0 היא כתובות לוגית לצורכי binding לשינה נשלחת ברשות, ולכן היא לא מופיעה בתווך כתובות ה-IP האמיתית של השרת בקובץ הלכידה. (בזכור כתובת השרת היא 192.168.1.56, הכתובת של המחשב עצמו).

בוחרות ה-IPv4 של כל הפאקטות הן מהצורה:

```

▼ Internet Protocol Version 4, Src: 192.168.1.15, Dst: 192.168.1.56
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
▼ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 40
    Identification: 0xe8f9 (59641)
▼ 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: 192.168.1.56
    [Stream index: 11]

```

למעט שינויים מעט לנעט בשדות ה-Total Length בהתאם לגודל הסגמנט וההודהה, ובשדה ה-Identification. בנוסף שמאפעם לפעם בתובות ה-IP מקור ויעד מתחלפים ביניהם (ליין והשרת מחליפים תפקדים, אחד שולח והשני מקבל). נשים לב לסוג הпрוטוקול, 6 עבור חיבור TCP שהגדנו. נשים לב גם לשדה

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

הסיבה לשדה זה לא משתנה היא שחברו הרשות הוא בין מכשירים באותו רשת מקומית. על אף שהמכשירים נפרדים ובעלי בתובות IP שונות, אין רכיבי רשות אחרים בדרך, הפאקטות לא עוזבות את הרשות המקומית ולבן תמיד נשארות עם ערך קבוע וגודלו.

בשימוש לתמונות האחוריות, מבון שבקובץ הלכידה שמצורף (2_clients_chat.pcap) אפשר לעיין בכל ההודעות ברמת האפליקציה, כמו:

lien שולח לשרת "itzik" בפקטה 658 (משתמש אליו רוצה לשלוח הודעה)

	657	63.896643	192.168.1.15	192.168.1.56	57760	12345	TCP	69747a696b
	658	64.278017	192.168.1.15	192.168.1.56	57760	12345	TCP	69747a696b
►	Ethernet II, Src: Intel_	0000 ea 94 f6 21 f7 c8 08 5b d6 f4 fe 1b 08 00 45 00		...!...[..... E				
▼	Internet Protocol Version 4, Src: 192.168.1.15, Dst: 192.168.1.56	0010 00 2d e8 e5 40 00 80 06 00 00 c0 a8 01 0f c0 a8		...@.....				
0100 = Version:		0020 01 38 e1 a0 30 39 00 b7 64 f7 8a 31 d7 af 50 18		.8.09...d..1..P..				
.... 0101 = Header Length:		0030 00 ff 83 b7 00 00 69 74 7a 69 6b	it zik				
0000 00.. = Differentiated Services Codepoint:								
..... ..00 = Explicit Congestion Notification:								
Not ECN-Capable Transport (0)								

השרת משיב לירן: UserNotFound בפקטה 659 כי איזיק הוא לא משתמש קיים:

659	64.286456	192.168.1.56	192.168.1.15	12345	57760	TCP	533a55736572...
► Ethernet II, Src: ea:94: ▼ Internet Protocol Version 4.0000 = Version: 0100 = Version: 0101 = Header Length: 0100 bytes (16) ▼ Differentiated Services 0000 00.. = Differentiated Service 0000 00.. = Differentiated Service	0000 08 5b d6 f4 fe 1b ea 94 f6 21 f7 c8 08 00 45 00 0010 00 36 2c f9 40 00 80 06 4a 31 c0 a8 01 38 c0 a8 0020 01 0f 30 39 e1 a0 8a 31 d7 af 00 b7 64 fc 50 18 0030 10 04 94 04 00 00 53 3a 55 73 65 72 4e 6f 74 46 0040 6f 75 6e 64	[..... ! E 6, @ J1 .. 8 09 .. 1 d .. P S: UserNotF ound					

הודעה שלין שלח לאחר דרכו השרת בפקטה 1674 :1674

1674	235.291860	192.168.1.15	192.168.1.56	57760	12345	TCP	446f20796f75...
.... ..00 = Explicitly Routed Total Length: 107 Identification: 0xe8f ▼ 010. = Flags: 0x0 0... = Reserved .1... = Don't Fragment ..0. = More Fragments ...0 0000 0000 0000 = Time to Live: 128 Protocol: TCP (6)	0000 ea 94 f6 21 f7 c8 08 5b d6 f4 fe 1b 08 00 45 00 0010 00 6b e8 f2 40 00 80 06 00 00 c0 a8 01 0f c0 a8 0020 01 38 e1 a0 30 39 00 b7 65 68 8a 31 d8 28 50 18 0030 00 ff 83 f5 00 00 44 6f 20 79 6f 75 20 77 61 6e 0040 6e 61 20 63 6f 6d 65 20 6f 76 65 72 20 61 6e 64 0050 20 77 61 74 63 68 20 69 74 20 74 6f 67 65 74 68 0060 65 72 3f 20 57 65 20 63 61 6e 20 6f 72 64 65 72 0070 20 70 69 7a 7a 61 20 3b 29! .. L E .. k .. @ 8 .. 09 .. eh .. 1 .. (P Do you wan na come over and watch it together? We c an order pizza ;)					

מלבד הדוגמאות האלו ניתן למצוא את כל שאר ההודעות והפקטות בקובץ הליבריה המצורף.

נציג דוגמה לשיחה בין חמישה משתמשים. כזכור, לפי הדרישות - כל משתמש צריך לפנות למשתמש אחד. יש לו כموון את היכולת לפנות למשתמש אחר ולהחליף בין המשתמשים בכל רגע, אבל בכל פעם ההודעה תישלח רק למשתמש שנבחר. הקלטה של השיחה מצורפת בשם pcap **5_clients_chat.pcap**.
ההודעות בין "ליון", "אושר", "נועה", "עדן" ו"גל" מסומנות בקו אדום, הפלט של השרת נראה כך:

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?

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

1000

Liran

Conn

[View Details](#)

BACK

Who do you want to message:

No
Com

Connection created

WANNA COME WITH US?

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

ביבליוגרפיה ומקורות

- Geeks For Geeks on TCP analysis using wireshark –
[/https://www.geeksforgeeks.org/ethical-hacking/tcp-analysis-using-wireshark](https://www.geeksforgeeks.org/ethical-hacking/tcp-analysis-using-wireshark)
- Jim Kurose on Wireshark –
<https://youtu.be/kCwd2YoJcvg?si=Tpu3bIBNL6yWyGfe>
- Nicolas Day on C++ socket programming (part 1) -
<https://youtu.be/gntyAFoZp-E?si=59aVkJLMTQO1X15>
- Nicolas Day on C++ socket programming (part 2)
https://youtu.be/sXW_sNGvqcU?si=ZmPkmQ1Q2F5KMAow
- Jim Kurose on socket programming –
https://youtu.be/_iHMMo7SDfQ?si=q-hwG4Nf6ODydAAQ
- stack overflow Q&A's
<https://stackoverflow.com/questions>

As well as other public Computer Science and networking forums, Youtube tutorials regarding C++ libraries, wireshark, TCP and IP properties

לא נעשה שימוש בכל AI בשום שלב של הפרויקט. הוא נכתב לאורך כל הסמסטר.
כל פסקה ותמונה בדו"ח זה וכל שאר הקבצים שמצורפים:
כולל קבצי לכידה ב-Wireshark, קבצי קוד בשפת C++, כולל הלוגיקה והתייעודים שלהם נכתבו
על ידי. הרוחחת מהפרויקט הבנה פרקטית ועמוקה יותר, של המונחים מושגים ומוסאים בסילוב.

Fin
