

פרויקט רשות תקשורת מחשבים

מגישיים: אליאן מינגוב - 325550382

שפירא ברוך 313295610

חלק 1 ניתוח תעבורת רשות

CSV חלק זה הוכן קובץ

עם הودעות בשכבה היישום CSV דרך יצירת קובץ

המכיל הودעות המציגות תקשורת בשכבה היישום. CSV במסגרת הפרויקט הוכן קובץ קובץ זה נבנה **ידנית**, בהתאם למבנה שנדרש בפרויקט ובהתבסס על הודעות תקשורת סטנדרטיות של HTTP פרוטוקול.

כל שורה בקובץ מייצגת הودעה לוגית אחת בשכבה היישום, ואינה מכילה מידע משכבות נמוכות יותר או פורטים), לאחר מכן וallow מתווספים בשלבי הארץ המאוחרים יותר IP (כגון כתובות

:שדות הקובץ

מצהה הודעה

פרוטוקול יישום

ישום מקור

ישוםיעד

תוקן ההודעה

חותמת זמן

-צילום מסך של תוכן הקובץ שהכנו

A	B	C	D	E	F	G
msg_id	app_protocol	src_app	dst_app	message	timestamp	
1	HTTP	client_browser	web_server	GET /index.html	0.005	
2	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.013	
3	HTTP	client_browser	web_server	GET /style.css	0.025	
4	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.03	
5	HTTP	client_browser	web_server	GET /script.js	0.041	
6	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.044	
7	HTTP	client_browser	web_server	GET /image.png	0.048	
8	HTTP	web_server	client_browser	HTTP/1.1 200 OK	0.054	

תיאור והסבר של תהליך אריזת המנות

TCP/IP. תהליך האrizה בוצע בהתאם למודל שכבות

message. בקובץ ה- CSV. הודעת שכבת היישום נלקחה משדה ה-

TCP. הודעה זו שימשה כנתונים שנשלחו דרך חיבור מבוסס

:بعث שליחת ההודעה

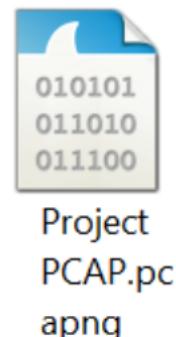
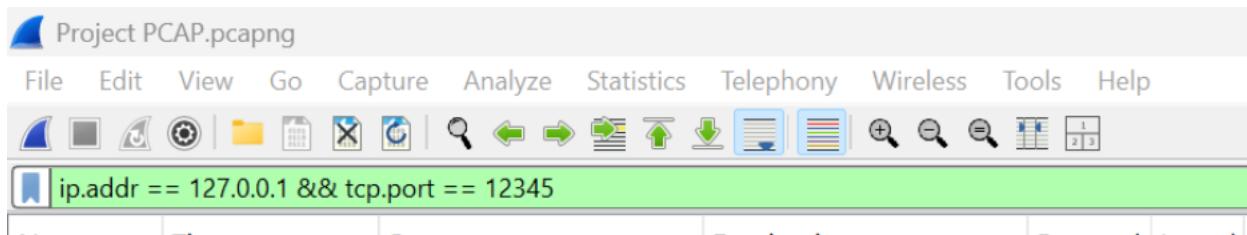
(TCP) מערכת הפעלה עטפה את תוכן ההודעה בכותרת של שכבת התעבורה

(IP) לאחר מכן נוספה כותרת של שכבת הרשות

לבסוף נוצרה מסגרת תקשורת לצורך שליחת במשק הרשות

תיאור והסבר של תהליך הילכידה

של המחשב back-to-back לcidt התעבורה הופעלת תוכנת וירשארק על מסך ה-
שנוצר על ידי המחברת. TCP בוצע סינון של התעבורה כך שייצגו רק מנות רלוונטיות לחיבור ה-
CSV. לאחר הפעלת הלכידה, המחברת הורזה ויצרה תעבורה בהתאם להודעות שהוגדרו בקובץ ה-
בסוף ההרצה הילכידה נוצרה ונשמרה בקובץ "יעודי לצורכי ניתוח".



תיאור והסביר של התעבורה שנלכדה

במהלך הלכידה ניתן היה לזהות מנות תקשורת הכוללות:

יצירת חיבור

העברת נתוניים

סגירת חיבור

.CSV בתוכן המנות ניתן היה לראות את תוכן הודעות שכבת היישום כפי שהוגדר בקובץ ה-
לדוגמה, הודעות מסווג בקשר לשאוב ותגובה שרת הופיעו בהתאם להנחות של מנות התקשרות

צילומי מסך מצורפים לדוח ומציגים

טענות TCP על ממשק loopback

תוכן הודיעות שכבת היישום בטור המנות

מבנה המנה לפי שכבות

הניטוח מדגים כיצד הودעה לוגית פשוטה משכבה מסוימת נארחת וmovebra בפועל דרך שכבות התקשרות.

No.	Time	Source	Destination	Protocol	Length	Info
+	30121 20.686223	127.0.0.1	127.0.0.1	TCP	59	9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
30122 20.686932	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30127 20.789977	127.0.0.1	127.0.0.1	TCP	59	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15	
30128 20.789954	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30129 20.891537	127.0.0.1	127.0.0.1	TCP	58	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14	
30130 20.891607	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30131 20.994635	127.0.0.1	127.0.0.1	TCP	59	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15	
30132 20.994112	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30133 21.095766	127.0.0.1	127.0.0.1	TCP	58	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14	
30134 21.095683	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30135 21.198897	127.0.0.1	127.0.0.1	TCP	59	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15	
30136 21.199052	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30137 21.201318	127.0.0.1	127.0.0.1	TCP	58	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=14	
30138 21.301912	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0	
30139 21.403872	127.0.0.1	127.0.0.1	TCP	59	[TCP Retransmission] 9369 + 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15	
L_	30140 21.403952	127.0.0.1	127.0.0.1	TCP	44	12345 + 9369 [RST] Seq=1 Win=0 Len=0

חלק שניים, מערכת צ'אט מבוססת Sockets

לפ' דרישות הפרויקט, הוקן יישום צ'אט מבוסס רשות המדגים תקשורת ללקוח – שרת באמצעות פרוטוקול TCP תוך שימוש ישר ב-Sockets, ללא ספירות חיצונית לניהול שירותי או תקשורת. מטרת היישום היא לאפשר תקשורת דו-כיוונית בזמן אמת בין לקוחות שונים, וכן לאפשר ניתוח מעשי של התעבורה הנוצרת ברשות כתוצאה מהפעלתו.

המערכת בנויה ארכיטקטורת Client–Server, כאשר הרשות משמש כמתוך מרכזי בין הלקוחות. כל לקוח מתחבר לרשות באמצעות TCP Socket, מזדהה בשם ייחודי, ויכול ליזום פтиחת צ'אט עם לקוח אחר על ידי שליחת בקשה לרשות הכוללת את שם הלקוח המבוקש. הרשות אחראית לניהול החיבורים, לטפל במספר לקוחות בו-זמנית (לפחות חמישה), ולהעביר הודעות בין הלקוחות בהתאם לבקשתיהם.

השימוש בוצע תוך שימוש בריבוי תהליכיים (Multithreading), כך שכל חיבור לקוח מטופל בתהליך נפרד, דבר המאפשר עבודה מקבילתית ותגובה בזמן אמת להודעות כניסה. המשך של הלקוח הוא טקסטואלי, בהתאם לדרישות, ומאפשר שליחת הודעות וקבלת הודעות ללא צורך במשك גרפי.

בנוסף לפיתוח היישום, בוצעה לכידת תעבורת הרשות שנוצרה במהלך הפעלת המערכת באמצעות תוכנת Wireshark. התעבורה נوثחה עד שכבת הרשות (כולל), תוך בחינה של תהליך ייצור החיבור (TCP Handshake), העברת הנתונים בין הלקוח לרשות, ואירוע המודיע לשכבות השונות במודל התקשרות.

בהמשך חלק זה יוצגו:

מבנה המערכת והקוד

הסבר על אופן התקינה וההרצה

דוגמאות קלט ופלט

ניתוח מפורט של תעבורת הרשות שנלכדה

מבנה המערכת והקוד – Client

- אחראי להתחברות לשרת, שליחת הודעות וקבלת הודעות בזמן אמת באמצעות Client. קיבל ה- TCP פרוטוקול.

"יעוד לארזנה להודעות מהשרת, ובמקביל מאפשר Thread לאחר יצירת החיבור, הליקוח מפעיל למשתמש להזין הודעות ולשלוח אותן לשרת בפורמט
@username:message.

הליקוח תומך בתקשות דיסקונט מלאה באמצעות ריבוי תהליכיון, ומוסים את פועלתו בצורה מסודרת עם סגירת החיבור.

- Client קוד

```
import socket
import threading

host = "127.0.0.1"
port = 12345

def receive_messages(sock):
    while True:
        try:
            data = sock.recv(1024)
            if not data:
                break
            print(data.decode())
        except:
            break
```

המשר קוד - Client

```
def start_client():
    username = input("enter username: ")

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))

    sock.send(username.encode())

    thread = threading.Thread(
        target=receive_messages, args=(sock,), daemon=True
    )
    thread.start()

    print("connected to server")
    print("message format: @user:message")

    while True:
        msg = input()
        if msg == "exit":
            break
        sock.send(msg.encode())
    sock.close()

if __name__ == "__main__":
    start_client()
```

מבנה המערכת והקוד – Server

רכיב ה-Server אחראי לניהול חיבורים של לקוחות ולניהוב הודעות ביניהם. השירות מאמין לחיבורים נוכנים באמצעות TCP Socket, ומטפל בכל חיבור ללקוח בתהילך נפרד (Thread), מה שמאפשר עבודה עם מספר לקוחות בו-זמנית.

בעת התחברות, כל לקוח שלוח שם משתמש, והשרת שומר מייפוי בין שמות משתמש ל-Sockets פעילים. כאשר מקבלת הודעה בפורמט `username:message@`, השרת מנtab את הודעה ללקוח היעד. אם הלקוח המבוקש אינו מחובר, נשלחת הודעה שגיאה לשולח.

השרת כולל טיפול בניהוק לקוחות, הסרתם מבנה הנתונים וסגירת החיבור בצורה מסודרת.

- **קוד Server**

```
import socket
import threading

host = "127.0.0.1"
port = 12345

clients = {}
lock = threading.Lock()

def handle_client(client_socket):
    username = ""
    try:
        username = client_socket.recv(1024).decode().strip()

        with lock:
            clients[username] = client_socket
            print(username + " connected")
```

המשר קוד - Server

```
while True:  
    data = client_socket.recv(1024)  
    if not data:  
        break  
  
    message = data.decode().strip()  
  
    if message.startswith("@") and ":" in message:  
        target, text = message.split(":", 1)  
        target = target.replace("@", "").strip()  
        text = text.strip()  
  
        with lock:  
            if target in clients:  
                clients[target].send(  
                    ("from " + username + ": " + text).encode()  
                )  
            else:  
                client_socket.send(  
                    ("error: user not connected").encode()  
                )  
  
    except:  
        pass  
  
finally:  
    with lock:  
        if username in clients:  
            del clients[username]  
  
    client_socket.close()  
    print(username + " disconnected")
```

המשר קוד - Server

```
def start_server():
    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen()

    print("server listening on " + host + ":" + str(port))

while True:
    client_socket, addr = server_socket.accept()
    thread = threading.Thread(
        target=handle_client, args=(client_socket,))
    )
    thread.start()

if __name__ == "__main__":
    start_server()
```

הוראות התקנה והריצה

דרישות מקדימות -

מערכת הפעלה התומכת ב-**Python 3** ומעלה

התקנה

יש לשמר את קבצי הקוד בתיקייה אחת:

server.py

client.py

הרכבת השרת

לפתוח חלון טרמינל.

לנוט לתיקיית הפרויקט.

להרץ את הפקודה:

python server.py

השרת יתחל להאזין לחיוברים בכתובת 127.0.0.1 ובפורט 12345.

הרכבת לקוחות

לפתוח חלון טרמינל נוסף.

לנוט לאוֹתָה תיִקְיַת פְּרוֹיָקֶט.

להרץ את הפקודה: **python client.py**

להזין שם משתמש ייחודי.

שלוחה הודעה בפורמט:

username:message@

ניתן להרץ מספר מופעים של הלקוּח במקביל לצורך בדיקת תקשורת בין לקוחות שונים.

דוגמאות קלט ופלט

בצלום המוצג ניתן לראות הריצה של שרת אחד ומספר מופעי ל��וח הפעלים במקביל. השרת מופעל תחילה ומזמן לחברים נכנסים בכתובת מקומית (127.0.0.1) ובפורט שנקבע. לאחר מכן מתחברים מספר לקוחות שונים, כאשר כל אחד מזדהה בשם משתמש ייחודי (ClientA, ClientB, ClientC וכו').

לאחר ההתחברות, כל לקוח יכול לשЛОוח הודעות בפורמט
`username:message@`
כאשר הודעה נשלחת לשרת ומנותבת ללקוח היעד בלבד. בצלום ניתן לראות שליחת הודעות בין לקוחות שונים, קבלת הודעות בזמן אמת, וכן פלט מצד השרת המציג התוצאות והויתוק של לקוחות.

הפלט מדגים תקשורת דו-כיוונית מלאה, ניתוב נכון של הודעות בין לקוחות, ותמייהה במספר לקוחות הפעילים בו-זמנית.

```
Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python server.py
server listening on 127.0.0.1:12345
ClientA connected
ClientB connected
ClientC connected
ClientD connected
ClientE connected

Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python client.py
enter username: ClientA
connected to server
message format: @user:message
@ClientB:hey
from ClientB: hello
from ClientE: have a good day
@ClientE:thank you

Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python client.py
enter username: ClientB
connected to server
message format: @user:message
From ClientA: hey
@ClientA:hello

Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python client.py
enter username: ClientC
connected to server
message format: @user:message
@ClientD:have a great day
from ClientD: thanks

Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

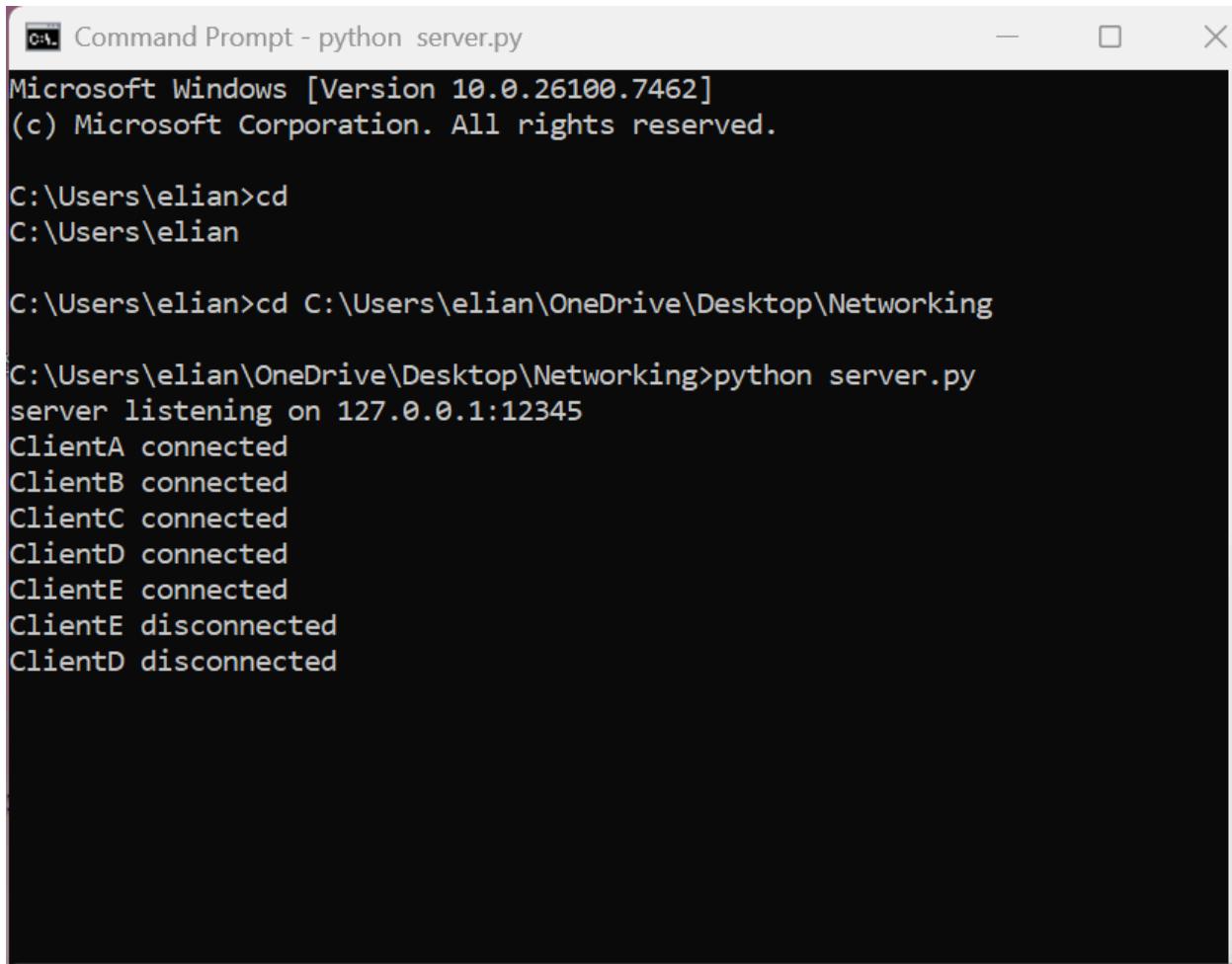
C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python client.py
enter username: ClientD
connected to server
message format: @user:message
from ClientC: have a great day
@ClientC:thanks

Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking
C:\Users\elian>python client.py
enter username: ClientE
connected to server
message format: @user:message
@ClientA:have a good day
from ClientA: thank you
```

דוגמאות ניתוקים

ניתן לראות ניתוק של לקוחות מהשרת. השרת מזהה ניתוק ללקוח, מסיר אותו מראשיתם הלקוחות הפעילים, וסגור את החיבור בצורה מסודרת. פلت זה מדגים את יכולת השרת לניהל חיבורים וניתוקים של מספר לקוחות במקביל ולטפל בשגיאות וניתוקים לא צפויים.



```
Command Prompt - python server.py
Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\elian>cd
C:\Users\elian

C:\Users\elian>cd C:\Users\elian\OneDrive\Desktop\Networking

C:\Users\elian\OneDrive\Desktop\Networking>python server.py
server listening on 127.0.0.1:12345
ClientA connected
ClientB connected
ClientC connected
ClientD connected
ClientE connected
ClientE disconnected
ClientD disconnected
```

ניתוח תעבורת של היפר-טב (עד שכבת הרשת)

לצורך ניתוח התעבורת, בוצעה לכידת מנוט של היפר-טב באמצעות תוכנת Wireshark בזמן הפעלת השרת ומספר ליקוחות במקביל. הלכידה בוצעה על משך ה-Loopback, מאחר והתקשרות מתבצעת מקומית בין ליקוחות לשרת (127.0.0.1), ונשמרה לקובץ PCAP.

יצירת חיבור TCP
במהלך הלכידה ניתן לראות את תהליך יצירת החיבור בין המהו לשרת באמצעות TCP Three-Way Handshake:

חביבת SYN נשלחת מהלהו לשרת (פורט יעד 12345)
השרת משב בחביבת SYN, ACK
הלהו משלים את החיבור באמצעות ACK
שלב זה מאשר הקמת חיבור אמין לפני העברת נתונים היפר-טב.

העברת נתונים בשכבות התעborות

לאחר יצירת החיבור, מועברות מנוט TCP הכוללות נתונים היפר-טב. מנוט אלו מסומנות Wireshark כ-ACK, PSH, ACK, C-ACK, ועוד. ומצביעות על ההודעות שנשלחו מהלהו לשרת ומהשרת להו. ניתן לראות כי כל הודעה נשלחת כמען נתונים (Payload) בתוך מקטע TCP.

שכבות הרשת (IP)

ברמת שכבת הרשת, כל המנות משתמשות בפרוטוקול IPv4, כאשר כתובות המקור והיעד הן 127.0.0.1, בהתאם להרצה מקומית של היפר-טב. ניתוח זה מדגים את עטיפות נתונים היפר-טב בתוך TCP, אשר עטופ בתוך חבילות IP.

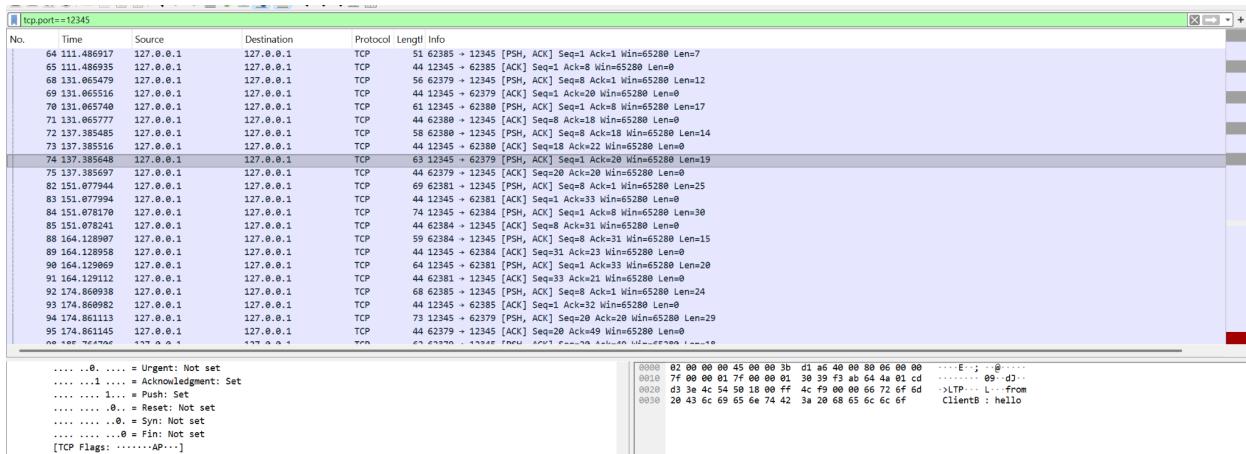
ניתוקים ותחזוקת חיבור

בנוסף, ניתן לראות מנות TCP המעידות על תחזוקת החיבור הפעיל, וכן סיום חיבורים כאשר ליקוחות מתנתקים מהשרת.

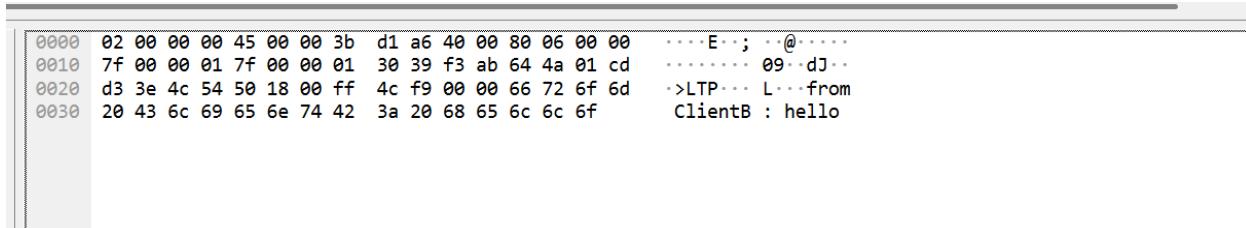
דוגמאות מקובץ PCAP שנשמר

בצילום המסר מוצגת חבילת TCP מסווג ACK שנשלחה במהלך העברת הודעה בין לקוחות דרך השירות. דגל PSH מצין העברת נתונים יישום, ו-ACK מאשר קבלת נתונים קודמים.

החבילה נשלחת על גבי חיבור TCP קיימ, מפורט מקור זמני לפורט השירות 12345. ברמת שכבת הרשת ניתן לראות שימוש ב-IPv4 עם כתובות מקור ויעד 127.0.0.1. מטען הנתונים (Payload) מכיל את תוכן ההודעה שנשלחה, כפי שניתן לראות בפונCTIONם.



נוכל גם לראות את תוכן ההודעה, למשל בדוגמה זו שלח ClientB hello



שימוש במבנה מלאכותית

במהלך פיתוח המערכת נעשה שימוש במבנה מלאכותית לצורך סיוע בתכנון לוגיקת היישום ובפרט בהגדרת פורמט ברור ונוח לשילוח הודעות בין לköחות בסביבה מרובת משתמשים. הבינה המלאכותית סייעה בגיבוש פורמט אחיד להודעות (username:message@message@), אשר מאפשר זיהוי פשוט של יעד ההודעה וניתוב יעיל מצד הרשת.

פורומט - “איך כדאי לעצב פורמט הודעות פשוט וברור לצ’אט מבוסס Sockets עם מספר לköחות ?”