

פרוייקט רשתות תקשורת מחשבים

מגשים: אליאן מינגוב - 325550382

שפירא ברוך 313295610

חלק 1 ניתוח תעבורת רשת

CSV חלק זה הוכן קובץ

עם הודעות בשכבת היישום CSV דרך יצירת קובץ

המכיל הודעות המייצגות תקשורת בשכבת 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 הודעה לוגית אחת בשכבת היישום, ואינה מכילה מידע משכבות נמוכות יותר (או פורטים), מאחר ואלו מתווספים בשלבי האריזה המאוחרים יותר IP (כגון כתובות

שדות הקובץ:

מזהה הודעה

פרוטוקול יישום

יישום מקור

יישום יעד

תוכן 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 האריזה בוצע בהתאם למודל שכבות

CSV בקובץ ה־message תחילה, הודעת שכבת היישום נלקחה משדה ה־
TCP הודעה זו שימשה כנתונים שנשלחו דרך חיבור מבוסס

בעת שליחת ההודעה

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

(IP) לאחר מכן נוספה כותרת של שכבת הרשת

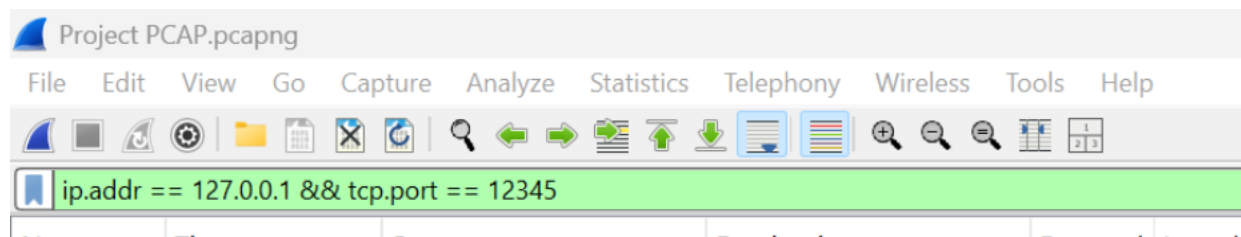
לבסוף נוצרה מסגרת תקשורת לצורך שליחה בממשק הרשת

תיאור והסבר של תהליך הלכידה

של המחשב loopback לצורך לכידת התעבורה הופעלה תוכנת ווירשארק על ממשק ה־

שנוצר על ידי המחברת. TCP בוצע סינון של התעבורה כך שיוצגו רק מנות רלוונטיות לחיבור ה־ CSV. לאחר הפעלת הלכידה, המחברת הורצה ויצרה תעבורה בהתאם להודעות שהוגדרו בקובץ ה־

בסיום ההרצה הלכידה נעצרה ונשמרה בקובץ ייעודי לצורך ניתוח.



Project
PCAP.pc
apng

תיאור והסבר של התעבורה שנלכדה

במהלך הלכידה ניתן היה לזהות מנות תקשורת הכוללות

יצירת חיבור

העברת נתונים

סגירת חיבור

CSV. בתוכן המנות ניתן היה לראות את תוכן הודעות שכבת היישום כפי שהוגדרו בקובץ ה-
לדוגמה, הודעות מסוג בקשת משאב ותגובת שרת הופיעו בתוך שדה הנתונים של מנות התקשורת.

צילומי מסך מצורפים לדוח ומציגים

loopback על ממשק TCP תעבורת

תוכן הודעות שכבת היישום בתוך המנות

מבנה המנה לפי שכבות

הניתוח מדגים כיצד הודעה לוגית פשוטה משכבת היישום נארזת ומועברת בפועל דרך שכבות
התקשורת.

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	20.686392	127.0.0.1	127.0.0.1	TCP	44	12345 → 9369 [RST] Seq=1 Win=0 Len=0
30127	20.789077	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	20.789164	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	20.994035	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	21.006766	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	21.006843	127.0.0.1	127.0.0.1	TCP	44	12345 → 9369 [RST] Seq=1 Win=0 Len=0
30135	21.108087	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	21.109052	127.0.0.1	127.0.0.1	TCP	44	12345 → 9369 [RST] Seq=1 Win=0 Len=0
30137	21.301318	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	21.301392	127.0.0.1	127.0.0.1	TCP	44	12345 → 9369 [RST] Seq=1 Win=0 Len=0
30139	21.401872	127.0.0.1	127.0.0.1	TCP	58	[TCP Retransmission] 9369 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=15
30140	21.401952	127.0.0.1	127.0.0.1	TCP	44	12345 → 9369 [RST] Seq=1 Win=0 Len=0

<div><div>Acknowledgment Number: 1 (relative ack number)</div><div>Acknowledgment number (raw): 0</div><div>0101 ... = Header Length: 20 bytes (5)</div><div>> Flags: 0x018 (PSH, ACK)</div><div>Window: 8192</div><div>[Calculated window size: 8192]</div><div>[Window size scaling factor: -1 (unknown)]</div><div>Checksum: 0x8730 [unverified]</div><div>[Checksum Status: Unverified]</div><div>Urgent Pointer: 0</div><div>> [Timestamps]</div><div>> [SEQ/ACK analysis]</div><div>> [TCP Analysis Flags]</div><div>> [Expert Info (Note/Sequence): This frame is a (suspected) retransmission]</div><div>> [This frame is a (suspected) retransmission]</div><div>> [Severity level: Note]</div><div>> [Group: Sequence]</div><div>[The RTO for this segment was: 717.649000 milliseconds]</div><div><small>TOTV based on delta from previous: 301393</small></div></div> <div><div>0000 02 00 00 00 45 00 00 37 00 01 00 00 40 06 7c baE-7...@ .</div><div>0010 7f 00 00 01 7f 00 00 01 24 99 30 39 00 00 00 00S-09----</div><div>0020 00 00 00 00 50 18 20 00 87 30 00 00 48 54 54 50P...-8..HTTP</div><div>0030 2f 31 2e 31 20 32 30 30 20 4f 4b /1.1.200.OK</div></div>

חלק שתיים, מערכת צ'אט מבוססת Sockets

לפי דרישות הפרויקט, הוכן יישום צ'אט מבוסס רשת המדגים תקשורת לקוח-שרת באמצעות פרוטוקול TCP תוך שימוש ישיר ב-Sockets, ללא ספריות חיצוניות לניהול שרתים או תקשורת. מטרת היישום היא לאפשר תקשורת דו-כיוונית בזמן אמת בין לקוחות שונים, וכן לאפשר ניתוח מעשי של התעבורה הנוצרת ברשת כתוצאה מהפעלתו.

המערכת בנויה מארכיטקטורת Client-Server, כאשר השרת משמש כמתווך מרכזי בין הלקוחות. כל לקוח מתחבר לשרת באמצעות Socket TCP, מזדהה בשם ייחודי, ויכול ליזום פתיחת צ'אט עם לקוח אחר על ידי שליחת בקשה לשרת הכוללת את שם הלקוח המבוקש. השרת אחראי לנהל את החיבורים, לטפל במספר לקוחות בו-זמנית (לפחות חמישה), ולהעביר הודעות בין הלקוחות בהתאם לבקשותיהם.

המימוש בוצע תוך שימוש בריבוי תהליכונים (Multithreading), כך שכל חיבור לקוח מטופל בתהליך נפרד, דבר המאפשר עבודה מקבילית ותגובה בזמן אמת להודעות נכנסות. הממשק של הלקוח הוא טקסטואלי, בהתאם לדרישות, ומאפשר שליחת הודעות וקבלת הודעות ללא צורך בממשק גרפי.

בנוסף לפיתוח היישום, בוצעה לכידת תעבורת הרשת שנוצרה במהלך הפעלת המערכת באמצעות תוכנת Wireshark. התעבורה נותחה עד שכבת הרשת (כולל), תוך בחינה של תהליך יצירת החיבור (TCP Handshake), העברת הנתונים בין הלקוח לשרת, ואריזת המידע לשכבות השונות במודל התקשורת.

בהמשך חלק זה יוצגו:

מבנה המערכת והקוד

הסבר על אופן ההתקנה וההרצה

דוגמאות קלט ופלט

ניתוח מפורט של תעבורת הרשת שנלכדה

מבנה המערכת והקוד – Client

אחראי להתחברות לשרת, שליחת הודעות וקבלת הודעות בזמן אמת באמצעות Client רכיב ה- עם הפעלת הלקוח, המשתמש מזין שם משתמש, הנשלח לשרת לצורך זיהוי. TCP פרוטוקול

ייעודי להאזנה להודעות מהשרת, ובמקביל מאפשר Thread לאחר יצירת החיבור, הלקוח מפעיל למשתמש להזין הודעות ולשלוח אותן לשרת בפורמט @username:message.

הלקוח תומך בתקשורת דו-כיוונית מלאה באמצעות ריבוי תהליכונים, ומסיים את פעולתו בצורה מסודרת עם סגירת החיבור.

קוד Client -

```
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
```

המשך 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 ולניתוב הודעות ביניהם. השרת מאזין לחיבורים נכנסים באמצעות Socket TCP, ומטפל בכל חיבור לקוח בתהליך נפרד (Thread), מה שמאפשר עבודה עם מספר לקוחות בו-זמנית.

בעת התחברות, כל לקוח שולח שם משתמש, והשרת שומר מיפוי בין שמות משתמשים ל-Sockets פעילים. כאשר מתקבלת הודעה בפורמט @username:message, השרת מנתב את ההודעה ללקוח היעד. אם הלקוח המבוקש אינו מחובר, נשלחת הודעת שגיאה לשולח.

השרת כולל טיפול בניתוק לקוחות, הסרתם ממבנה 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")
```


- המשך 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")
```

המשך 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 והרצה

דרישות מקדימות -

מערכת הפעלה התומכת ב-Python
Python גרסה 3 ומעלה

התקנה

יש לשמור את קבצי הקוד בתיקייה אחת:

server.py

client.py

הרצת השרת

לפתוח חלון טרמינל.

לנווט לתיקיית הפרויקט.

להריץ את הפקודה:

python server.py

השרת יתחיל להאזין לחיבורים בכתובת 127.0.0.1 ובפורט 12345.

הרצת לקוח

לפתוח חלון טרמינל נוסף.

לנווט לאותה תיקיית פרויקט.

להריץ את הפקודה: python client.py

להזין שם משתמש ייחודי.

לשלוח הודעות בפורמט:

username:message@

ניתן להריץ מספר מופעים של הלקוח במקביל לצורך בדיקת תקשורת בין לקוחות שונים.

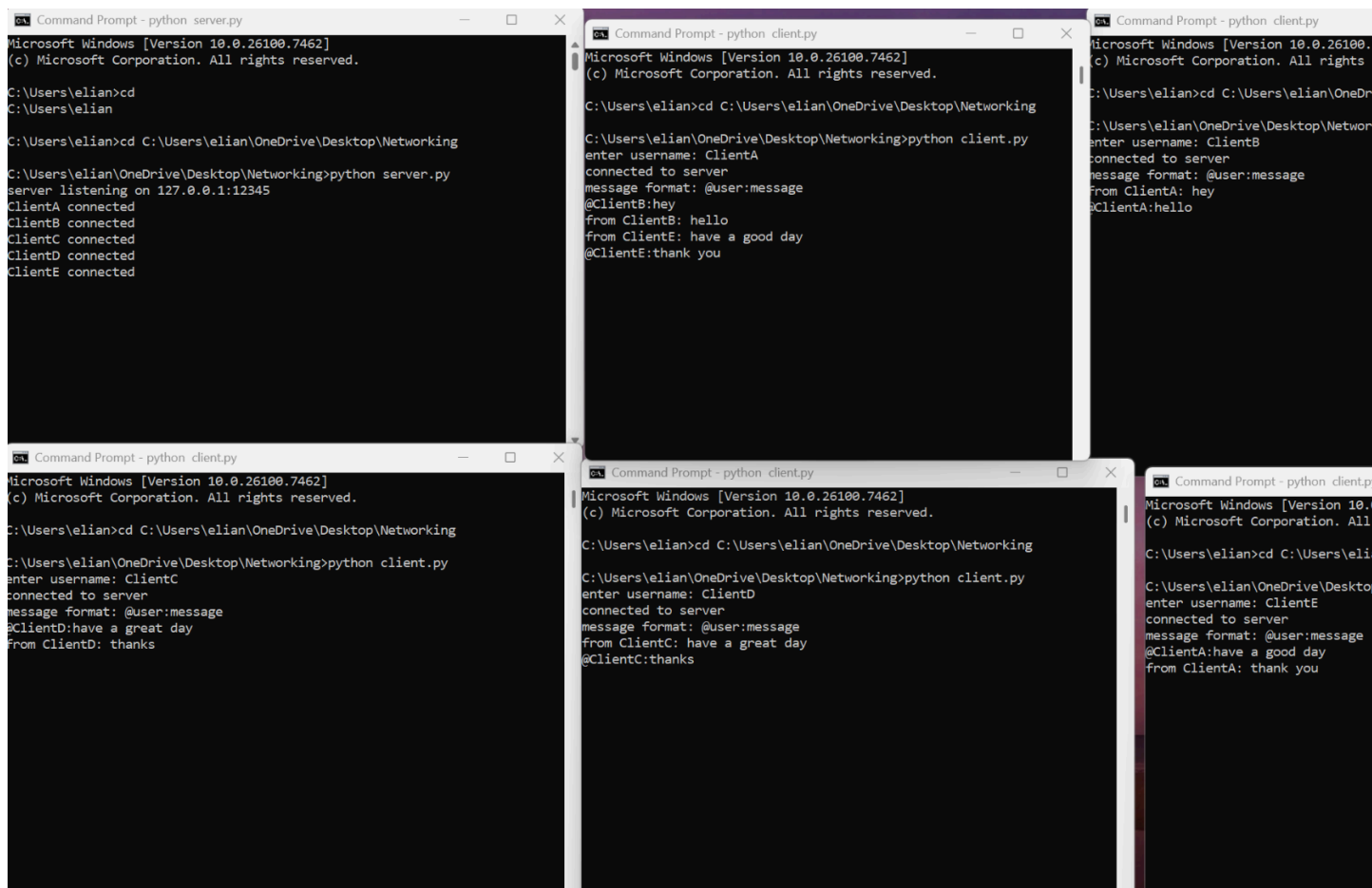
דוגמאות קלט ופלט

בצילום המסך המוצג ניתן לראות הרצה של שרת אחד ומספר מופעי לקוח הפועלים במקביל. השרת מופעל תחילה ומאזין לחיבורים 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 לשלוח הודעות בפורמט
username:message@

כאשר ההודעה נשלחת לשרת ומנותבת ללקוח היעד בלבד. בצילום ניתן לראות שליחת הודעות בין לקוחות שונים, קבלת הודעות בזמן אמת, וכן פלט בצד השרת המציג התחברות וניתוק של לקוחות.

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



```
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

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

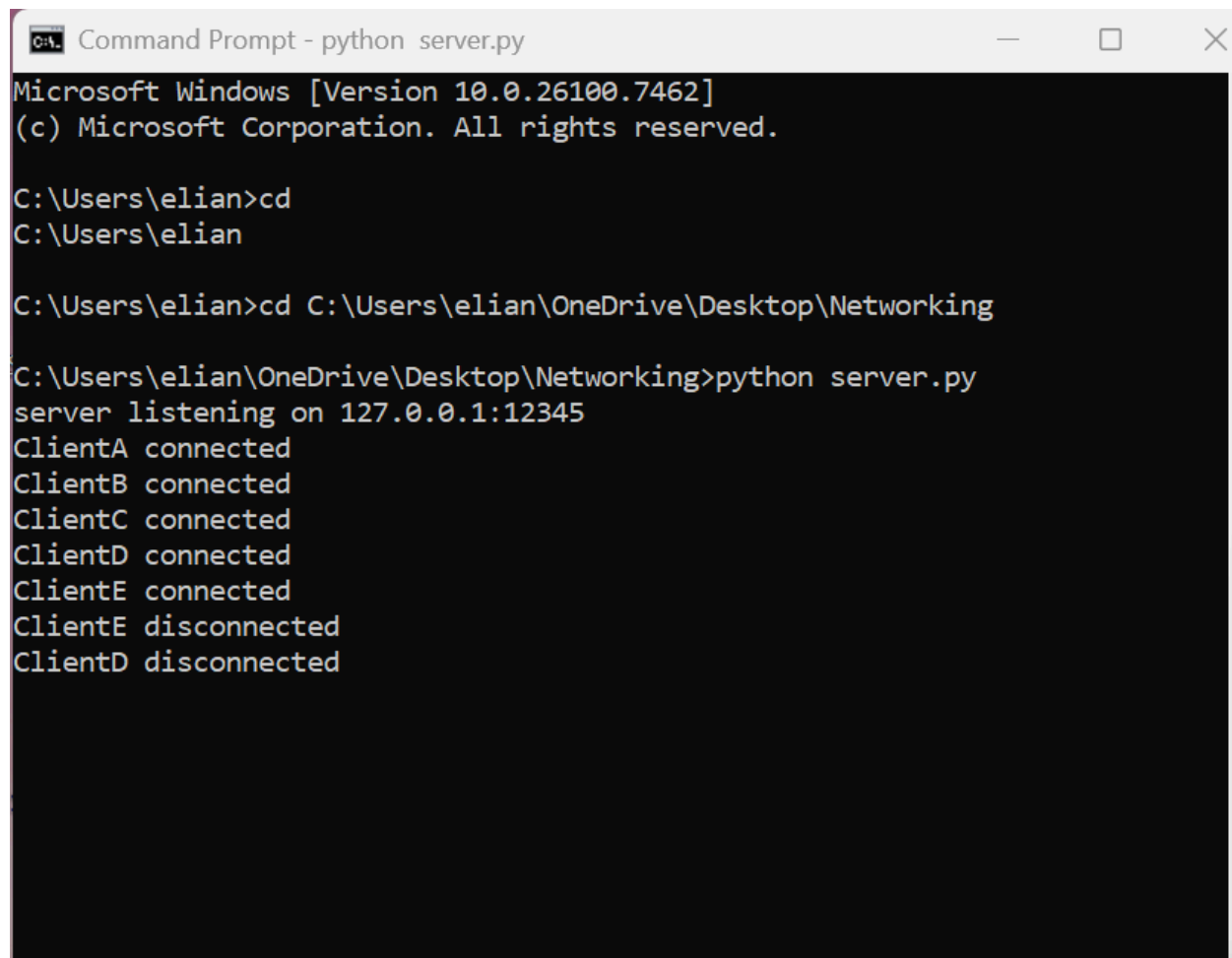
Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

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

C:\Users\elian\OneDrive\Desktop\Networking>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 לקוח, מסיר אותו מרשימת הלקוחות הפעילים, וסוגר את החיבור בצורה מסודרת. פלט זה מדגים את יכולת השרת לנהל 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 תעבורה של היישום (עד שכבת הרשת)

לצורך ניתוח התעבורה, בוצעה לכידת מנות של היישום באמצעות תוכנת Wireshark בזמן הפעלת השרת ומספר לקוחות במקביל. הלכידה בוצעה על ממשק ה-Loopback, מאחר והתקשורת מתבצעת מקומית בין לקוחות לשרת (127.0.0.1), ונשמרה לקובץ PCAP.

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

חבילת SYN נשלחת מהלקוח לשרת (פורט יעד 12345)
השרת משיב בחבילת SYN, ACK
הלקוח משלים את החיבור באמצעות ACK
שלב זה מאשר הקמת חיבור אמין לפני העברת נתוני היישום.

העברת נתונים בשכבת התעבורה

לאחר יצירת החיבור, מועברות מנות TCP הכוללות נתוני יישום. מנות אלו מסומנות ב-Wireshark כ-PSH, ACK, ומכילות את ההודעות שנשלחות מהלקוח לשרת ומהשרת ללקוח. ניתן לראות כי כל הודעה נשלחת כמטען נתונים (Payload) בתוך מקטע TCP.

שכבת הרשת (IP)

ברמת שכבת הרשת, כל המנות משתמשות בפרוטוקול IPv4, כאשר כתובת המקור והיעד הן 127.0.0.1, בהתאם להרצה מקומית של היישום. ניתוח זה מדגים את עטיפת נתוני היישום בתוך TCP, אשר עטוף בתוך חבילות IP.

ניתוקים ותחזוקת חיבור

בנוסף, ניתן לראות מנות TCP Keep-Alive המעידות על תחזוקת החיבור הפעיל, וכן סיום חיבורים כאשר לקוחות מתנתקים מהשרת.

דוגמאות מקובץ ה-PCAP שנשמר

בצילום המסך מוצגת חבילת TCP מסוג PSH, ACK שנשלחה במהלך העברת הודעה בין לקוחות דרך השרת. דגל PSH מצוין העברת נתוני יישום, ו-ACK מאשר קבלת נתונים קודמים.

החבילה נשלחת על גבי חיבור TCP קיים, מפורט מקור זמני לפורט השרת 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 בפענוח הנתונים.

No.	Time	Source	Destination	Protocol	Length	Info
64	111.486917	127.0.0.1	127.0.0.1	TCP	51	62385 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=7
65	111.486935	127.0.0.1	127.0.0.1	TCP	44	12345 → 62385 [ACK] Seq=1 Ack=8 Win=65280 Len=0
68	131.065479	127.0.0.1	127.0.0.1	TCP	56	62379 → 12345 [PSH, ACK] Seq=8 Ack=1 Win=65280 Len=12
69	131.065516	127.0.0.1	127.0.0.1	TCP	44	12345 → 62379 [ACK] Seq=1 Ack=20 Win=65280 Len=0
70	131.065740	127.0.0.1	127.0.0.1	TCP	61	12345 → 62380 [PSH, ACK] Seq=1 Ack=8 Win=65280 Len=17
71	131.065777	127.0.0.1	127.0.0.1	TCP	44	62380 → 12345 [ACK] Seq=8 Ack=18 Win=65280 Len=0
72	137.385485	127.0.0.1	127.0.0.1	TCP	58	62380 → 12345 [PSH, ACK] Seq=8 Ack=18 Win=65280 Len=14
73	137.385516	127.0.0.1	127.0.0.1	TCP	44	12345 → 62380 [ACK] Seq=18 Ack=22 Win=65280 Len=0
74	137.385648	127.0.0.1	127.0.0.1	TCP	63	12345 → 62379 [PSH, ACK] Seq=1 Ack=20 Win=65280 Len=19
75	137.385697	127.0.0.1	127.0.0.1	TCP	44	62379 → 12345 [ACK] Seq=20 Ack=20 Win=65280 Len=0
82	151.077944	127.0.0.1	127.0.0.1	TCP	69	62381 → 12345 [PSH, ACK] Seq=8 Ack=1 Win=65280 Len=25
83	151.077994	127.0.0.1	127.0.0.1	TCP	44	12345 → 62381 [ACK] Seq=1 Ack=33 Win=65280 Len=0
84	151.078170	127.0.0.1	127.0.0.1	TCP	74	12345 → 62384 [PSH, ACK] Seq=1 Ack=8 Win=65280 Len=30
85	151.078241	127.0.0.1	127.0.0.1	TCP	44	62384 → 12345 [ACK] Seq=8 Ack=31 Win=65280 Len=0
88	164.128907	127.0.0.1	127.0.0.1	TCP	59	62384 → 12345 [PSH, ACK] Seq=8 Ack=31 Win=65280 Len=15
89	164.128958	127.0.0.1	127.0.0.1	TCP	44	12345 → 62384 [ACK] Seq=31 Ack=23 Win=65280 Len=0
90	164.129069	127.0.0.1	127.0.0.1	TCP	64	12345 → 62381 [PSH, ACK] Seq=1 Ack=33 Win=65280 Len=20
91	164.129112	127.0.0.1	127.0.0.1	TCP	44	62381 → 12345 [ACK] Seq=33 Ack=21 Win=65280 Len=0
92	174.860938	127.0.0.1	127.0.0.1	TCP	68	62385 → 12345 [PSH, ACK] Seq=8 Ack=1 Win=65280 Len=24
93	174.860982	127.0.0.1	127.0.0.1	TCP	44	12345 → 62385 [ACK] Seq=1 Ack=32 Win=65280 Len=0
94	174.861113	127.0.0.1	127.0.0.1	TCP	73	12345 → 62379 [PSH, ACK] Seq=20 Ack=20 Win=65280 Len=29
95	174.861145	127.0.0.1	127.0.0.1	TCP	44	62379 → 12345 [ACK] Seq=20 Ack=49 Win=65280 Len=0

.....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...]

0000 02 00 00 00 45 00 00 3b d1 a6 40 00 80 06 00 00E...; @.....
0010 7f 00 00 01 7f 00 00 01 30 39 f3 ab 64 4a 01 cd09...dJ..
0020 d3 3e 4c 54 50 18 00 ff 4c f9 00 00 66 72 6f 6d ->LTP... L...from
0030 20 43 6c 69 65 6e 74 42 3a 20 68 65 6c 6c 6f ClientB : hello

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

0000	02 00 00 00 45 00 00 3b d1 a6 40 00 80 06 00 00E...; @.....
0010	7f 00 00 01 7f 00 00 01 30 39 f3 ab 64 4a 01 cd09...dJ.. ->LTP... L...from
0020	d3 3e 4c 54 50 18 00 ff 4c f9 00 00 66 72 6f 6d	ClientB : hello
0030	20 43 6c 69 65 6e 74 42 3a 20 68 65 6c 6c 6f	

שימוש בבינה מלאכותית

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

פרומפט - "איך כדאי לעצב פורמט הודעות פשוט וברור לצ'אט מבוסס Sockets עם מספר לקוחות?"