ריבוי זרמים ב-QUIC

רקע

ב-Quic, אופן העברת המידע מתבצע ע"י ריבוי זרמים (Multiplexing Streams), כך שניתן יהיה להעביר נתונים במקביל וללא תלות בזרמים אחרים. העברת המידע בדרך זו מאפשרת פרודקטיביות גבוהה ושיפור בביצועים ביחס לאלגוריתם TCP.

כל זרם פועל עצמאית. כלומר, אם יש עיכוב בזרם אחד, לא תהיה לכך השפעה על שאר הזרמים. אך עם זאת, יש צורך לנהל את חלוקת המשאבים בצורה נכונה.

גודל החבילות העוברות בכל זרם אינו אחיד ולכן תהליך של ניהול כמות הזרמים הוא הכרחי ליצירת קשר יעיל בין הצדדים.
ניהול לא נכון של מספר הזרמים, עשוי לגרור בעיות שישפיעו על הביצועים ועל יעילות התקשורת. ובהם:

1. **עומס יתר על משאבי מחשוב ורשת –** הקצאה מופרזת של זרמים תוביל לשימוש מופרז במעבד ובזכרון. רוחב הפס לא ינוצל כנדרש בשל התנודות המרובות שיגרמו בסופו של דבר לקריסה בחיבור.

2. **עליה בתקורה** - כמות גדולה של הודעות המועברת בכל הזרמים יחד, עלולה לייצר מצב שנתונים רבים שאינם שימושיים לעבור ברשת (כמו ה-headers והאישורים (ACKs) הנדרשים במהלך העברת המידע), כלומר תהיה ירידה ביעילות הפרוטוקל

3. **יצירת צוואר בקבוק** – למרות ש-QUIC מתמודד עם בעיית Head-of-Line בכך שיש מספר זרמים, ריבוי זרימות מופרז עשוי כן לייצר תלויות בין זרמים שונים, מה שבסופו של דבר כן ייצר Head-of-Line בקונטקסט של זרמים, במקום הודעות.

בפרוייקט מימשנו את ריבוי הזרימות, כך שמידע מועבר מהשרת אל הלקוח. בסעיפים הבאים נפרט לגבי סקירת המימוש, ה-API, הבדיקות, הפלטים והניסויים שבוצעו בפרוייקט.

סקירת המימוש של ריבוי הזרימות

בפרוייקט זה מימשנו את הקצאת הזרמים בין צד הלקוח (**QuicClient**) לבין צד השרת (**QuicServer**). כל זרם שנשלח מיוצג ע"י המחלקה **StreamOut** (משדר זרם), וכל רכיב המקבל מידע יהיה מקושר לצד הלקוח **StreamIn** (קולט זרם).

תיאור התהליך:

<u>אתחול</u>

1. **אתחול השרת**:
השרת מאותחל עם כתובת IP, פורט, מספר זרמים וגודל כולל של הנתונים אותם יש להעביר.
הוא יוצר קבצי נתונים לכל זרם. גודל כל קובץ נקבע לפי גודל הנתונים הכולל שמחולק למספר הזרמים באופן שווה. כמו כן הוא מגריל לכל זרם את קצב הנתונים אותו יעביר (התפלגות אחידה בטווח 1,000-2,000 בתים בחבילה).
השרת יוצר סוקט מסוג UDP וממתין להודעת SYN מהלקוח.

2. **אתחול הלקוח:**
השרת מאותחל עם כתובת IP, והפורט של השרת. הוא מאתחל סוקט UDP.
הוא מייצר מבני נתונים כך שיוכל לשמור את הנתונים עבור כל הזרמים שיקבלו מידע.

<u>לחיצת יד בין הלקוח והשרת</u>

3. **הלקוח שולח SYN לשרת:**
הלקוח שולח הודעת "SYN" לשרת כדי לבקש פתיחת חיבור והתחלת תהליך התקשורת. ההודעה נשלחת לכתוב ה-IP והפורט של השרת.

4. **השרת מעביר ללקוח את פרטי הזרמים:**
לאחר קבלת הודעת ה-SYN מהלקוח, השרת מגיב על ידי שליחת מספר הזרמים שישמשו בתקשורת.
לאחר מכן השרת שולח ללקוח את מספר הפורטים המשויכים לכל זרם כדי שהלקוח יוכל להתחבר לכל זרם בנפרד.
כל פורט מיועד לזרם מסוים, והלקוח מקבל את רשימת הפורטים.

5. **הלקוח מקבל את פרטי הזרמים**:
הלקוח מקבל את כמות הזרמים ואת הפורטים ושולח ACK על כל הודעה.

<u>יצירת זרמים והקמת תקשורת בינהם (לחיצת יד בין הזרמים)</u>

6. **אתחול זרמים בצד השרת:**
השרת יגדיר כל זרם בנפרד, ייצור מופע של StreamOut(משדר זרם), יגדיר לו את מספר הפורט, את הגודל של החבילות ואת הקובץ אותו יעביר לקולט הזרם המתאים בצד הלקוח. הזרם יאזין בפורט ויחכה לפניה מהזרם המקביל בצד הלקוח.

7. **אתחול זרמים בצד הלקוח:**
הלקוח יגדיר כל זרם בנפרד, ייצור מופע של StreamIn ויעביר לו את את הפורט אליו מאזין אובייקט ה-StreamOut איתו יתקשר.
מופע ה-StreamIn ישלח הודעת SYN אל ה-StreamOut המתאים.

8. **משדר הזרם (StreamOut) שולח את גודל הזרם לקולט הזרם (StreamIn)**

9. **קולט הזרם (StreamIn) מקבל את גודל הזרימה**:
קולט הזרם מקבל את גודל הזרימה (גודל אחיד לכל חבילה) וישלח ACK.

<u>העברת נתונים</u>

10. **משדרי הזרמים שולחים את הנתונים אל הקולט הזרם המתאים:**
כל משדר זרם קורא את הבתים בגודל שנקבע ויוצר חבילה אותה ישלח לקולט הזרם המתאים. אם לא יקבל ACKבזמן
שנקבע(Time-Out), הוא ישלח מחדש את החבילה עד קבלת אישור מהלקוח.
תהליך זה נמשך עד אשר מסיים לקרוא את כל הקובץ וישלח הודעת EOF לסמן את סיום התקשורת.

11. **קולטי הזרמים מקבלים את המידע ושומרים מידע סטטיסטי:**
כל קולט זרם שמקבל את המידע מייצר הודעת ACK עבור החבילה הנוכחית ומתעד את זמן קבלת החבילה.
כאשר מקבל הודעת EOF, סוגר את החיבור.

<u>ניתוח נתונים</u>

12. **הלקוח מחשב סטטיסטיקות גלובליות, מדפיס בפלט התוכנית ומייצר גרפים.**

<u>סקירת ה-API:</u>

המודול QUIC_API:

מספק פונקציות עזר עבור המערכת כולה, המסייעות בטיפול ובניהול של נתונים ושליחת חבילות בפרוטוקול.

פונקציונאליות:

	שם הפונקציה	פעולה
1	**create_data(size_in_bytes)**	**יוצרת מחרוזת רנדומלית באורך של size_in_bytes המכיל אותיות A-Z (אותיות קטנות וגדולות).**
2	**parse_packet(packet)**	**מחלצת את המידע של חבילת נתונים (packet) , ומחזיר את מספר החבילה והמידע שהועבר.**
3	**create_file(data, index)**	**יוצרת קובץ טקסט ושומר בתוכו את המידע שסופק**
4	**create_packet(packet_id, data = '')**	**יוצרת חבילה (packet) על ידי קידוד מספר החבילה ואת המידע המועבר לתוך פורמט מתאים לשליחה.**

<u>הערה</u>: לכל חבילה יש מספר חבילה באורך 8 בתים, כך שאם הגודל שהוגרל היה X, בפועל כל חבילה תכיל X-8 בתים של מידע ו-8 בתים של מספר חבילה.

המודול QuicClient:

מממשת את צד הלקוח בתקשורת בפרוטוקול QUIC. יוצר מופעים של קולטי הזרמים(StreamIn) ומחשבת סטטיסטיקה בסיום העברת הנתונים

פונקציונאליות:

	שם הפונקציה	פעולה
1	**__init__(self, server_host, server_port, exp=False)**	**מאתחלת את המחלקה עם פרמטרים לחיבור לשרת ולניהול זרמים.**
2	**create_socket(self)**	**יוצרת ומגדירה סוקט ללקוח לתקשורת עם השרת.**
3	**handshake_with_server(self)**	**מנהלת את תהליך לחיצת היד (Handshake) מול השרת**
4	**send_syn_message(self)**	**שולחת הודעת SYN לשרת כדי ליזום את החיבור.**
5	**receive_number_of_streams(self)**	**מקבלת מהשרת את מספר הזרמים לחיבור.**
6	**send_ack_message(self, packet_num)**	**שולחת ACK לשרת כדי לאשר קבלת חבילה (packet) מסוימת**
7	**start(self)**	**מתחילה את כל תהליך החיבור לשרת והניהול של הזרמים.**
8	**initialize_streams(self, num_streams)**	**מאתחלת את הזרמים לפי מספר הזרמים שהתקבל מהשרת.**
9	**setup_stream(self, stream_index)**	**מגדירה חיבור לזרם מסוים על ידי קבלת מספר הפורט ושליחת ACK.**
10	**start_stream_thread(self, index, stream_port)**	**יוצרת תהליכון (Thread) חדש לניהול קבלת נתונים מזרם. אתחול מופע חדש של StreamIn.**
11	**wait_for_threads(self)**	**ממתינה לכל התהליכונים (Threads) שיסיימו את פעולתם.**
12	**collect_stream_stats(self, stream_in, index)**	**אוספת סטטיסטיקות מהזרמים אחרי שהעברת נתונים מסתיימת.**
13	**finalize_and_report(self)**	**מבצעת חישובים וסטטיסטיקות ומדפיסה את הנתונים.**
14	**calculate_rates(self)**	**מחשבת את שיעור העברת הנתונים ושיעור העברת החבילות הכולל לכל הזרמים בזמן ההעברה הכולל**
15	**add_stream_data(self, stream_index, time_elapsed, data_received, packets_received)**	**שומרת נתונים וחבילות שהתקבלו עבור זרם מסוים על פי מקטעי זמן שנקבעו על ידי המשתמש, לצורך מעקב מפורט.**
16	**print_stream_stats(self)**	**מדפיסה את הסטטיסטיקות של כל זרם וסטטיסטיקות כלליות של כל הזרמים, ושומרת את המידע הזה לקובץ עבור ניתוח נוסף.**

המודול QuicServer:

מחלקה המייצגת שרת QUIC המנהל חיבורים עם צד לקוח, מגדיר ומנהל מופעים של משדרי זרמים(StreamOut), השולחים מידע לצד הלקוח(דרך המחלקה StreamIn) בתהליך מקבילי.

פונקציונאליות:

	שם הפונקציה	פעולה
1	**__init__(self, host, port, streams, total_size=0.1, ack_timeout=5, is_test=False,**	**מאתחלת שרת QUIC עם כתובת IP, פורט, מספר זרמים ופרמטרים נוספים, ויוצרת קבצי נתונים עבור כל זרם.**
2	**create_server_socket(self)**	**יוצרת ומגדירה את הסוקט של השרת לשימוש בתקשורת עם הלקוח.**
3	**create_stream_data_files(self, total_size)**	**יוצרת קבצי נתונים עבור כל זרם בהתאם לגודל כולל מחולק בין הזרמים.**
4	**wait_for_client_syn(self)**	**ממתינה להודעת SYN מהלקוח כדי להתחיל את תהליך החיבור.**
5	**send_number_of_streams(self)**	**שולחת ללקוח את מספר הזרמים המוגדרים עבור החיבור.**
6	**wait_for_ack(self, expected_packet_num)**	**ממתינה ומקבלת הודעת ACK מהלקוח על מנת לאשר קבלת חבילה עם מספר מסוים.**
7	**start_stream_threads(self)**	**מפעילה תהליכונים (threads) נפרדים עבור כל זרם לצורך העברת המידע לצד הלקוח.**
8	**start_stream_threads(self)**	**יוצר ומתחיל תהליכון (threads) עבור כל זרם, ומוודא שכל זרם מעביר מידע לצד הלקוח.**
9	**send_stream_port_to_client(self, stream_index, stream_port)**	**שולח ללקוח את מספר הפורט של זרם ספציפי**
10	**wait_for_all_threads(self)**	**ממתין שכל תהליכוני משדרי הזרמים יסתיימו, ומוודא שהם הסתיימו כנדרש.**
11	**start(self)**	**מתניע את כל תהליך השרת: קבלת SYN מהלקוח, שליחת מספר הזרמים ללקוח, הפעלת תהליכי הזרם, והמתנה לסיום כל התהליכים.**

המודול StreamIn:

מחלקה המייצגת קולט זרמים (Stream Receiver) בתקשורת המנהלת את תהליך הקליטה של נתונים משרת QUIC.

המחלקה אחראית לקלוט מידע, לתעד באמצעות logger את העברת הנתונים, ולחשב סטטיסטיקות עבור כל זרם בודד בתהליך התקשורת.

פונקציונאליות:

	שם הפונקציה	פעולה
1	**__init__(self, server_host, server_port, index, global_start_time, add_stream_data_callback)**	**אתחול המחלקה, יצירת הסוקט, הגדרת פרמטרים של הזרם, ופתיחת קובץ log לתיעוד תהליכים.**
2	**create_socket(self)**	**יוצר ומגדיר סוקט עבור הזרם.**
3	**log(self, message)**	**פונקציה לרישום ותיעוד ב-log (נעשית כתיבה לקובץ)**
4	**close_resources(self)**	**סוגר את הסוקט וקובץ ה-log כאשר כל הנתונים התקבלו.**
5	**initialize_connection(self)**	**אתחול החיבור עם מופע StreamOut על ידי שליחת הודעת SYN וקבלת גודל הזרם.**
6	**send_syn(self)**	**שולח הודעת SYN לשרת כדי להתחיל את החיבור.**
7	**receive_stream_size(self)**	**מקבל את גודל הזרם מהשרת ושולח הודעת ACK.**
8	**send_ack(self, packet_num)**	**שולח הודעת ACK עבור חבילה שהתקבלה.**
9	**receive_data(self)**	**מקבל נתונים מהשרת בהודעות ושולח ACK לכל הודעה**
10	**handle_received_data(self, packet_num, data)**	**מנהל נתונים שהתקבלו: רושם אותם ב-log, שולח ACK ושומר את הסטטיסטיקות.**
11	**get_stats(self)**	**מחזיר את הסטטיסטיקות עבור הזרם הנוכחי, כולל משך הזמן, כמות נתונים, מהירות קבלת נתונים (גודל/חבילות)**
12	**start(self)**	**הנקודה המרכזית להתחלת תהליך קבלת הנתונים. מתניע את החיבור ומתחיל את קבלת הנתונים.**

המודול StreamOut:

מחלקה זו מייצגת את משדר הזרם בתקשורת QUIC. אחראית לניהול תהליך השידור של נתונים משרת ה-QUIC ללקוח.

פונקציונאליות:

	שם הפונקציה	פעולה
1	**__init__(self, host, port, data_size, index)**	**מאתחלת את המופע של משדר הזרם עם הגדרות רשת, גודל נתונים, ומיקום בקובץ.**
2	**create_socket(self)**	**יוצר ומגדיר סוקט עבור הזרם.**
3	**log(self, message)**	**פונקציה לרישום ותיעוד ב-log (נעשית כתיבה לקובץ)**
4	**close_resources(self)**	**סוגרת את הסוקט וקובץ ה-log כאשר כל הנתונים נשלחו.**
5	**start(self)**	**מנהלת את התקשורת ההתחלתית ומתחילה את תהליך שידור הנתונים**
6	**wait_for_client_syn(self)**	**מאזינה להודעת SYN מהלקוח לצורך הקמת החיבור.**
7	**send_stream_size(self)**	**שולחת את גודל הזרם ללקוח.**
8	**wait_for_ack(self)**	**ממתינה להודעת ACK מהלקוח לפני המשך השידור.**
9	**send_data_from_file(self)**	**שולחת נתונים מהקובץ ללקוח.**
10	**send_packet(self, packet_data)**	**שולחת חבילת נתונים ללקוח.**
11	**wait_for_ack_with_retry(self, packet_data)**	**ממתינה להודעת ACK מהלקוח עם ניסיונות חוזרים במקרה של time-out.**
12	**send_eof(self)**	**שולחת הודעת 'EOF' לסימון סיום השידור.**

טסטים ובדיקות

לכל אחד מהמודולים יצרנו קובץ טסטים. נציג כאן טבלה ובה הטסטים מחולקים על פי המודול.

מודול	#	טסט	בדיקה
QuicServer	1	test_server_initialization_and_socket_setup	בודק את אתחול השרת והגדרת שקע ה- UDP (socket) והחיבור שלו לכתובת והפורט.
	2	test_wait_for_client_syn	בודק את קבלת הודעת SYN מהלקוח להתחלת חיבור.
	3	test_send_number_of_streams	בודק את שליחת מספר הזרמים (streams) ללקוח.
	4	test_wait_for_ack	בודק את המתנה לאישור ACK מהלקוח.
	5	test_start_stream_threads	בודק את התחלת התהליכונים (threads) של הזרמים להעברת נתונים.
	6	test_invalid_syn_message	בודק טיפול בהודעת SYN לא חוקית.
	7	test_timeout_waiting_for_ack	בודק טיפול בזמן קצוב (timeout) במהלך המתנה לאישור ACK.
	8	test_send_stream_port_to_client	בודק את שליחת מספר הפורט של הזרם ללקוח.
	9	test_wait_for_all_threads	בודק המתנה לסיום כל התהליכונים(threads) שמייצגים משדרי זרמים.
	10	test_server_start	בודק את התהליך השלם של התחלת השרת, כולל כל השלבים הנדרשים לתקשורת עם הלקוח.
QuicClient	11	test_initialization	בודק את תהליך האתחול של QuicClient ואת הגדרת הנתונים ההתחלתיים.
	12	test_socket_creation	בודק יצירת socket של הלקוח והגדרות התצורה שלו.
	13	test_handshake_with_server	בודק את תהליך לחיצת היד (handshake) עם השרת וקבלת מספר הזרמים.
	14	test_receive_number_of_streams	בודק את תהליך קבלת מספר הזרמים מהשרת.

QuicClient	15	test_send_ack_message	בודק את תהליך שליחת הודעת אישור (ACK) לשרת.
	16	test_initialize_streams	בודק את אתחול הזרמים ויצירת תהליכים (threads) עבור כל זרם.
	17	test_setup_stream	בודק את הגדרת הזרם על ידי קבלת מספר פורט ושליחת ACK
	18	test_start_stream_thread	בודק התחלת תהליכון קולט זרם (thread) עבור מופע של StreamIn.
	19	test_wait_for_threads	בודק המתנה לסיום כל התהליכים ואיסוף סטטיסטיקות שלהם.
	20	test_collect_stream_stats	בודק איסוף ושמירת סטטיסטיקות עבור זרם.
	21	test_calculate_rates	בודק חישוב קצב קבלת הנתונים והחבילות הכולל עבור כל הזרמים.
	22	test_add_stream_data	בודק הוספת נתונים ומנות לכל זרם במקטעי זמן מוגדרים.
	23	test_finalize_and_report	בודק את תהליך סיום הדיווח של QuicClient.
	24	test_print_stream_stats	בודק הדפסת ושמירת סטטיסטיקות של זרמים ונתונים כלליים לקובץ.
	25	test_handle_invalid_packet_structure	בודק טיפול במבנה חבילה לא חוקי בזמן קבלת נתונים.
	26	test_acknowledgement_reception_handling	בודק טיפול בקבלת ACK מהשרת.
	27	test_handle_incorrect_data_type_for_stream_ports	בודק טיפול בקבלת סוג נתונים שגוי עבור פורטים של זרמים.
StreamIn	28	test_initialize_connection	בודק את תהליך האתחול והקמת החיבור של המחלקה.
	29	test_receive_data_and_ack	בודק את תהליך קבלת הנתונים ושליחת הודעות ACK לשרת.
	30	test_idle_connection_handling	בודק טיפול במצב של חיבור ריק כאשר אין נתונים שמתקבלים במשך זמן מה.
	31	test_receive_multiple_packets	בודק את קליטת הנתונים כאשר מתקבלים מספר חבילות (packets).
	32	test_invalid_packet_handling	בודק טיפול בחבילות לא חוקיות או פגומות.
	33	test_large_data_reception	בודק את תהליך קבלת נתונים בגודל גדול.
	34	test_error_logging	בודק תיעוד של שגיאות במהלך קליטת נתונים.

	35	test_ack_resending_on_timeout	בודק שליחת הודעת ACK מחדש במקרה של פסק זמן (Time-out)
	36	test_connection_closure_behavior	בודק את ההתנהגות הנכונה של המחלקה כאשר החיבור נסגר באופן בלתי צפוי.
StreamOut	37	test_wait_for_client_syn	בודק שהמתודה wait_for_client_syn מזהה ומטפלת בהודעת SYN מהלקוח ומגדירה את הכתובת של הלקוח כראוי.
	38	test_send_data_flow	בודק את תהליך שליחת חבילות הנתונים והמתנה לאישורי ACK מהלקוח, כולל ניסיונות חוזרים במקרה של פסק זמן.
	39	test_multiple_timeouts_then_success	בודק שהלקוח מבצע ניסיונות חוזרים במקרה של מספר פסקי זמן (Time-Out) ומצליח לבסוף לקבל ACK.
	40	test_send_data_without_syn	בודק שמתרחשת שגיאה כאשר מנסים לשלוח נתונים לפני שמתקבלת הודעת SYN מהלקוח.
	41	test_proper_closure_after_sending	בודק שהמשאבים (סוקט וקבצי לוג) נסגרים כראוי לאחר שליחת הנתונים.
	42	test_incorrect_ack_handling	בודק טיפול במצבים שבהם מתקבלים אישורי ACK לא נכונים.
	43	test_empty_data_packet	בודק טיפול במצב של שליחת חבילת נתונים ריקה.
	44	test_receive_eof_signal	בודק שהמתודה send_data_from_file מטפלת נכון בהודעת סוף קובץ (EOF) כאשר נגמרים הנתונים בקובץ.
	45	test_proper_logging	בודק שהאירועים מתועדים כראוי בקובץ הלוג במהלך תהליך שליחת הנתונים והמתנה ל- ACK

פלטים וניסויים

תיקיה וקבצי הפלטים

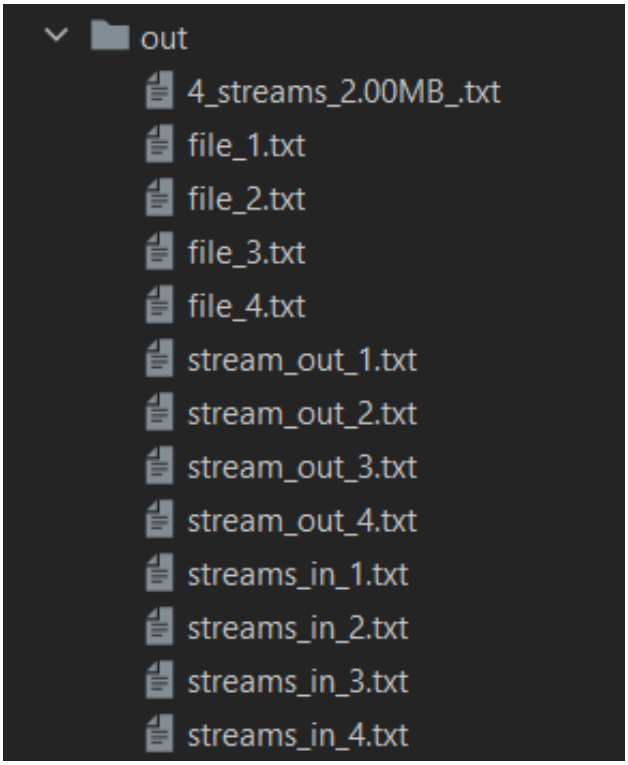
הפלטים של הריצות נשמרים בתיקיה בשם "out" בתוך הפרויקט.
במהלך ההרצות אנחנו מייצרים תיעוד להעברת הזרמים, עבור כל זרם בנפרד.
תחילת הריצה צד השרת מייצר כמות קבצים בהתאם לכמות הזרמים, על פי המידע שנקבע. כל זרם יעביר את החלק היחסי מגודל המידע הכולל. כלומר אם נניח המידע שיועבר בכל הזרמים יחד יהיה 10 מ"ב, ויש 4 זרמים, אז כל אחד מהם יעביר 2.5 מ"ב. כל קובץ כזה ייקרא file_X.txt כך ש-X זה יהיה אינדקס הזרם.

כל משדר זרם ייצר קובץ לוג משלו, קובץ בשם Stream_out_X.txt (X זה אינדקס המשדר זרם), בו יתעד את העברת הנתונים וקבלת חיווי הגעה (ACK).

כל קולט זרם ייצר קובץ לוג משלו, קובץ בשם Stream_in_X.txt (X זה אינדקס הקולט זרם), בו יתעד את קבלת הנתונים ושליחת חיווי הגעה (ACK).

בסיום הריצה, יישמר קובץ נוסף בשם X_streams_YMB_.txt, כך ש-X זה כמות הזרמים, Y זה גודל המידע שהועבר.

למשל, עבור הרצה ובה הועברו 2 מ"ב באמצעות 4 זרמים, התיקייה תראה כך:



נציג לדוגמה את הקובץ stream_out_1.txt:

```
Stream 1: Waiting for connection on 127.0.0.1:8001
Stream 1: Open connection with ('127.0.0.1', 62471)
Stream 1: Sent data size: 1689 bytes to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #0
Stream 1: Sent packet #1 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #1
Stream 1: Sent packet #2 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #2
Stream 1: Sent packet #3 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #3
Stream 1: Sent packet #4 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #4
Stream 1: Sent packet #5 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #5
Stream 1: Sent packet #6 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #6
Stream 1: Sent packet #7 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #7
Stream 1: Sent packet #8 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #8
Stream 1: Sent packet #9 (1689 bytes) to ('127.0.0.1', 62471)
```

ניתן לראות שגודל המידע המועבר בזרם זה הוא בגודל 1,689 בתים.

כך נראה החלק האחרון של הקובץ, בו מתבצעת שליחה של החבילה שאומרת שהשליחה הגיעה לסיומה:

```
Stream 1: Sent packet #302 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #302
Stream 1: Sent packet #303 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #303
Stream 1: Sent packet #304 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #304
Stream 1: Sent packet #305 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #305
Stream 1: Sent packet #306 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #306
Stream 1: Sent packet #307 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #307
Stream 1: Sent packet #308 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #308
Stream 1: Sent packet #309 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #309
Stream 1: Sent packet #310 (1689 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #310
Stream 1: Sent packet #311 (698 bytes) to ('127.0.0.1', 62471)
Stream 1: Received ACK for packet #311
Stream 1: Completed sending data. Sent EOF.
```

וכך למשל נראות השורות הראשונות בקובץ stream_in_1.txt באותה ההרצה:

```
Stream 1: Connecting to 127.0.0.1:8001
Stream 1: Sent SYN to 127.0.0.1:8001
Stream 1: Data size is 1697 bytes
Stream 1: Sent ACK for packet #0
Stream 1: Sent ACK for packet #1
Stream 1: Received packet #1 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #2
Stream 1: Received packet #2 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #3
Stream 1: Received packet #3 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #4
Stream 1: Received packet #4 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #5
Stream 1: Received packet #5 with 1689 bytes and sent ACK
```

כך נראה החלק האחרון של הקובץ, בו מתבצעת קבלה של החבילה שאומרת שהשליחה הגיעה לסיומה:

```
Stream 1: Received packet #306 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #307
Stream 1: Received packet #307 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #308
Stream 1: Received packet #308 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #309
Stream 1: Received packet #309 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #310
Stream 1: Received packet #310 with 1689 bytes and sent ACK
Stream 1: Sent ACK for packet #311
Stream 1: Received packet #311 with 698 bytes and sent ACK
Stream 1: Received EOF from server.
Stream 1: Completed receiving data.
```

כמו כן נוצר קובץ מסכם (שגם מודפס למשתמש בהרצה של צד הלקוח) ובו נתונים גלובליים לגבי הריצה וגם לגבי כל זרם. הקובץ של אותה הרצה בדוגמה נראה כך:

נתונים גלובליים:

```
Global Statistics:
  Number of Streams: 4
  Total Data Received: 2097152 bytes (2.00 MB)
  Total Packets Received: 1327
  Data Rate: 43.41 MB/sec
  Packet Rate: 28801.85 packets/sec

```

מידע על הזרם הראשון:

```
Stream 1:
  Total Data Received: 524288 bytes
  Data Size: 1697 bytes
  Bytes per Second: 13219653.46
  Packets per Second: 7841.71
  Total Packets Received: 311
  Total Time: 0.04 seconds
```

מידע על הזרם השני:

```
Stream 2:
  Total Data Received: 524288 bytes
  Data Size: 1209 bytes
  Bytes per Second: 11379398.57
  Packets per Second: 9484.86
  Total Packets Received: 437
  Total Time: 0.05 seconds
```

מידע על הזרם השלישי:

```
Stream 3:
  Total Data Received: 524288 bytes
  Data Size: 1827 bytes
  Bytes per Second: 12772098.32
  Packets per Second: 7040.28
  Total Packets Received: 289
  Total Time: 0.04 seconds
```

מידע על הזרם הרביעי:

```
Stream 4:
  Total Data Received: 524288 bytes
  Data Size: 1816 bytes
  Bytes per Second: 12464139.79
  Packets per Second: 6894.30
  Total Packets Received: 290
  Total Time: 0.04 seconds
```

תיעוד משורת הפקודה

כאשר נפעיל את התוכנית מתוך שורת הפקודה, אנחנו נריץ בחלון אחד את צד השרת ובחלון נפרד את צד הלקוח.

התקשורת מהכיוון של צד השרת:

```
Server listening on 127.0.0.1:8000
QuicServer starts
File 1 created with size 2621440 bytes
File 2 created with size 2621440 bytes
File 3 created with size 2621440 bytes
File 4 created with size 2621440 bytes
The client ('127.0.0.1', 49010) sent a SYN message
Sent number of streams: 4
Received ACK for packet #0
Sent stream 1 port: 8001
Received ACK for packet #1
Sent stream 2 port: 8002
Received ACK for packet #2
Sent stream 3 port: 8003
Received ACK for packet #3
Sent stream 4 port: 8004
Received ACK for packet #4
All streams have completed.
```

פלטים:

1. צד השרת מודיע כי מאזין בכתובת 127.0.0.1 בפורט 8000.
2. השרת מייצר 4 קבצים במשקל של 2621440 בתים (2.5 מ"ב)
3. הלקוח מהכתובת 127.0.0.1 בפורט 49010 שלח הודעת SYN
4. השרת שלח ללקוח את כמות הזרמים: 4
5. השרת שלח לו את הפורטים 8001-8004 וקיבל עליהם ACK מהלקוח.
6. מודפסת בתום תהליך השליחה הודעה על כך שתהליך העברת המידע הושלם בכל הזרמים.

התקשורת מהכיוון של צד הלקוח:

```
Sent SYN to 127.0.0.1:8000
Received number of streams: 4
Sent ACK for packet #0
Received stream 1 port: 8001
Sent ACK for packet #1
Received stream 2 port: 8002
Sent ACK for packet #2
Received stream 3 port: 8003
Sent ACK for packet #3
Received stream 4 port: 8004
Sent ACK for packet #4

Data Rate (Bytes/Second): 64102681.43
Packet Rate (Packets/Second): 32180.45
Global Statistics:
  Number of Streams: 4
  Total Data Received: 10485760 bytes (10.00 MB)
  Total Packets Received: 5264
  Data Rate: 61.13 MB/sec
  Packet Rate: 32180.45 packets/sec
```

פלטים:

1. הלקוח שולח הודעת SYN לשרת בכתובת 127.0.0.1 בפורט 8000.
2. מתקבלת הודעה על כמות הזרמים: 4
3. נשלח ACK על כמות הזרמים.
4. התקבלו מהשרת הפורטים 8001-8004 ונשלחו על חבילות המידע האלו ACK.
5. בתום התהליך, הלקוח מדפיס למסך נתונים סטטיסטיים גלובליים לגבי כל הזרמים יחד ולחוד (ראה תמונה מצורפת בסעיף הבא)

בהמשך הפלט של הלקוח, מופיעים נתונים לגבי כל זרם בנפרד:

```
Stream 1:
  Total Data Received: 2621440 bytes
  Data Size: 2000 bytes
  Bytes per Second: 16241443.65
  Packets per Second: 8153.43
  Total Packets Received: 1316
  Total Time: 0.16 seconds


Stream 2:
  Total Data Received: 2621440 bytes
  Data Size: 2000 bytes
  Bytes per Second: 17346011.45
  Packets per Second: 8707.94
  Total Packets Received: 1316
  Total Time: 0.15 seconds
```

כאן ניתן לראות את הפלט עבור זרמים 1 ו-2.

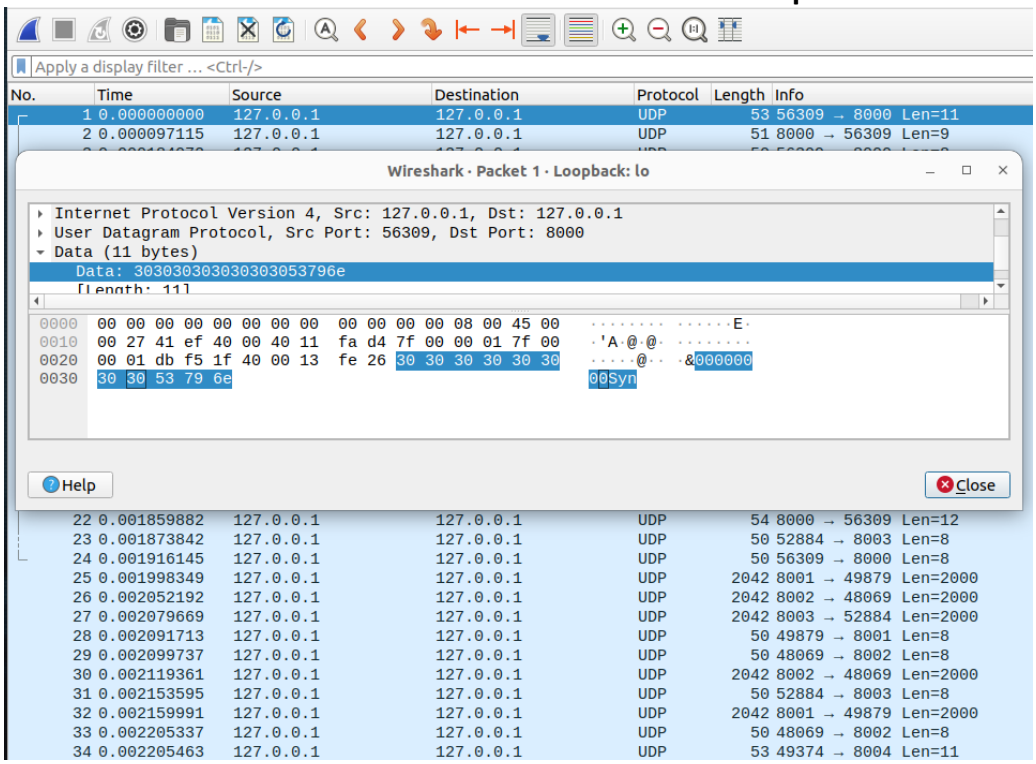
```
Stream 3:
  Total Data Received: 2621440 bytes
  Data Size: 2000 bytes
  Bytes per Second: 18289803.03
  Packets per Second: 9181.74
  Total Packets Received: 1316
  Total Time: 0.14 seconds


Stream 4:
  Total Data Received: 2621440 bytes
  Data Size: 2000 bytes
  Bytes per Second: 16025670.36
  Packets per Second: 8045.11
  Total Packets Received: 1316
  Total Time: 0.16 seconds
```

כאן ניתן לראות את הפלט עבור זרמים 3 ו-4.

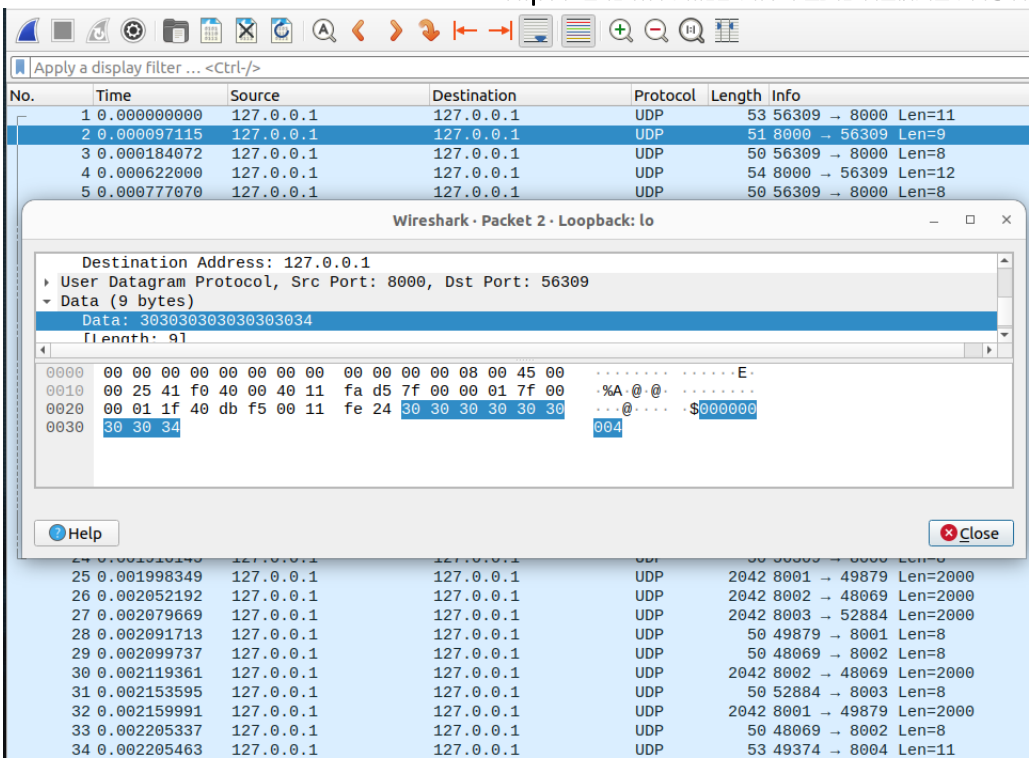
תיעוד בהקלטות Wireshark

הודעת ה-SYN מהלקוח

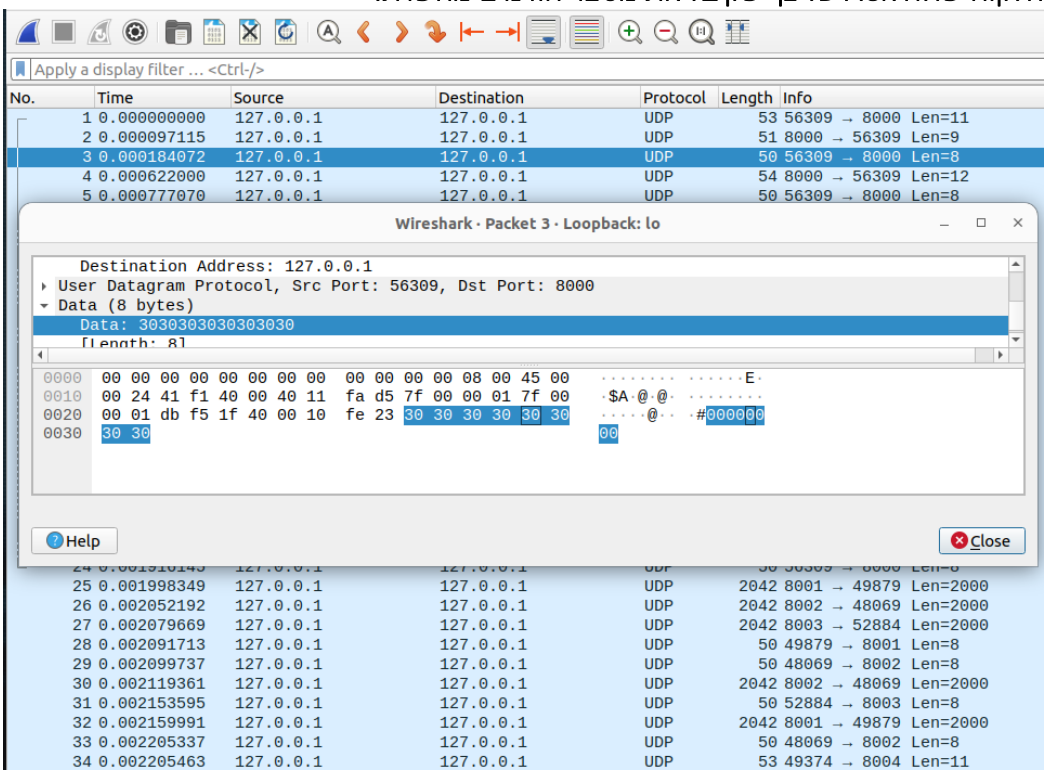


ההודעה הראשונה שנשלחת מהלקוח היא Syn בה הוא פונה לשרת. ניתן לראות את תוכן הודעה זו בתמונה המצורפת בצבע כחול.

השרת בתגובה מעביר את כמות הזרמים ללקוח

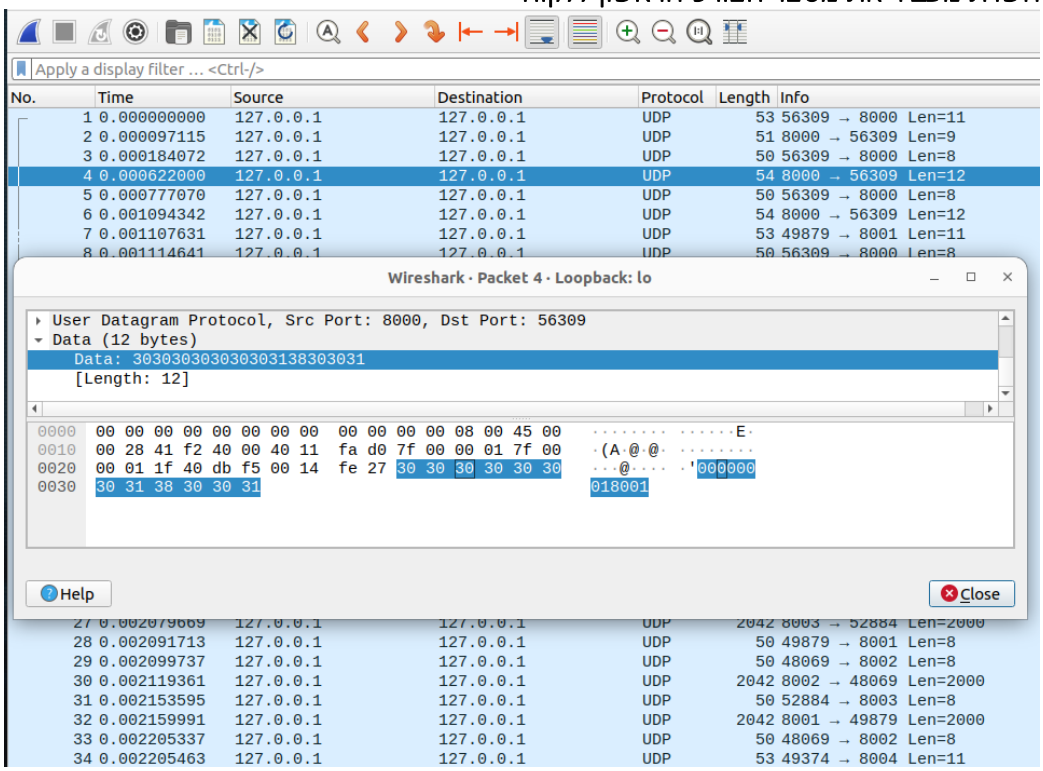


הלקוח שולח ACK על כך שקיבל את מספר הזרמים מהשרת.



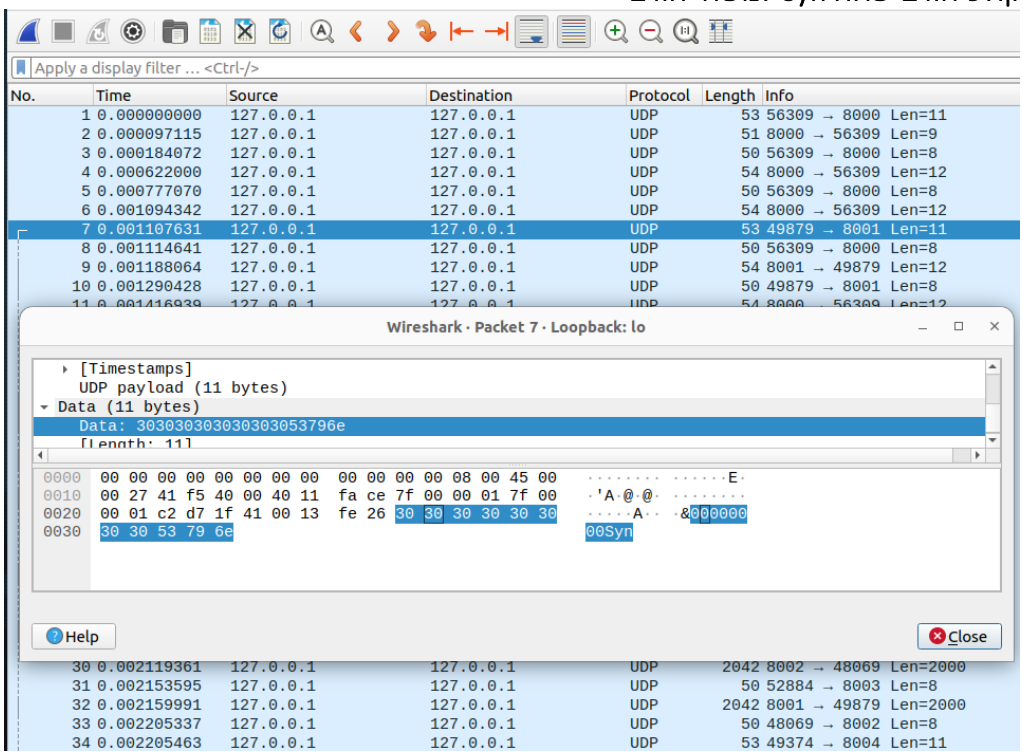
מדובר בהודעה עם מספר ההודעה ששלח השרת, ללא תוכן.

השרת מעביר את מספר הפורט הראשון ללקוח



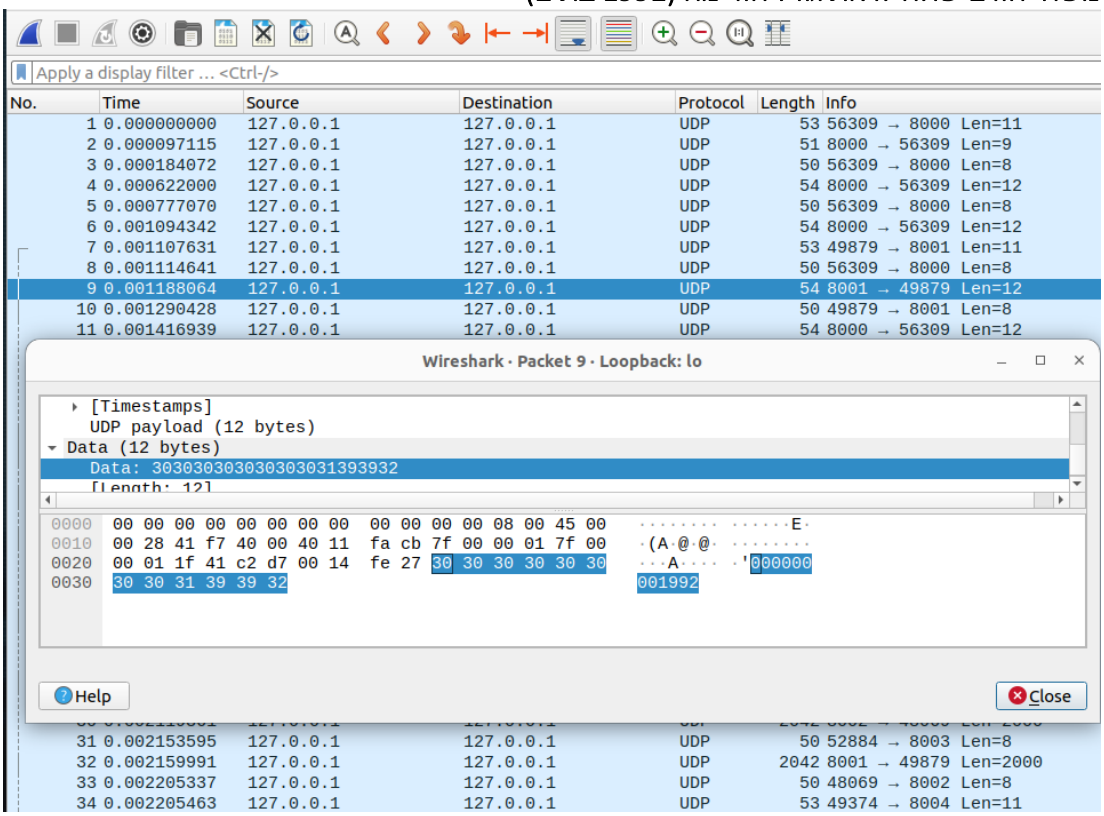
ניתן לראות שהשרת מעביר ללקוח את החבילה מספר 1 (00000001) ואת מספר הפורט 8001

קולט הזרם שולח Syn למשדר הזרם

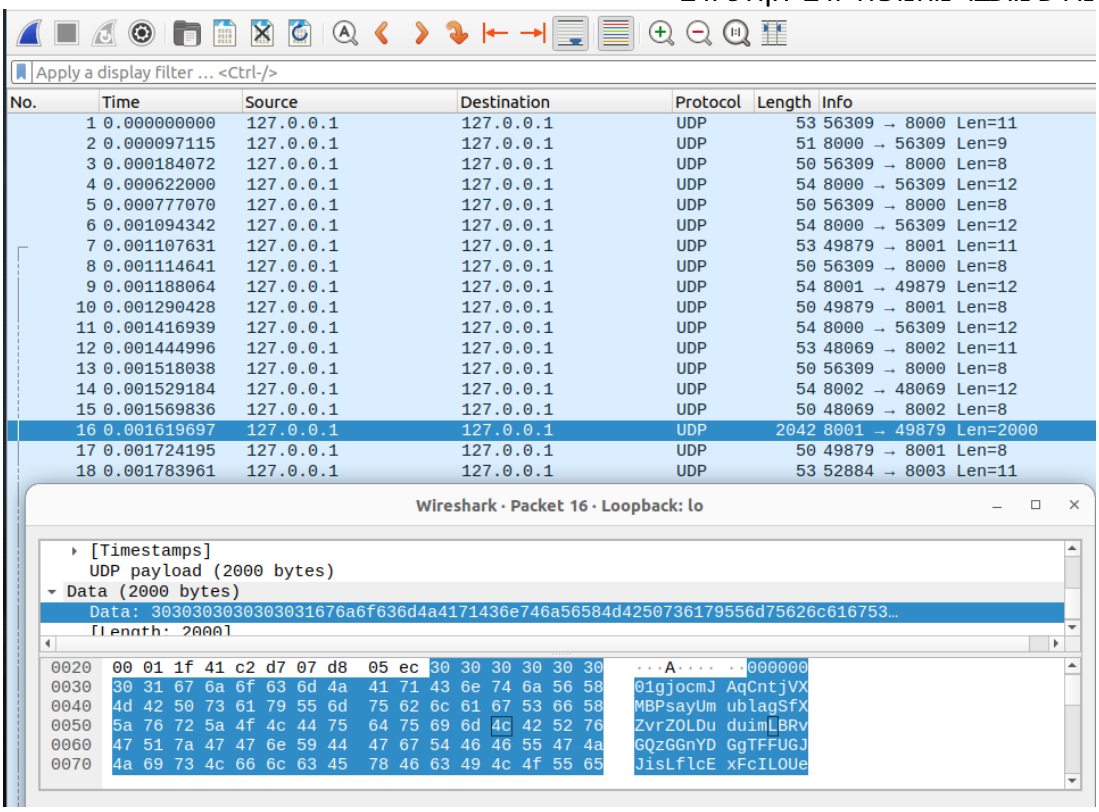


ניתן לראות שהזרם שנוצר בלקוח (מופע של streamIn) שלח Syn לפורט 8001 (בו המופע של StreamOut) מאזין.

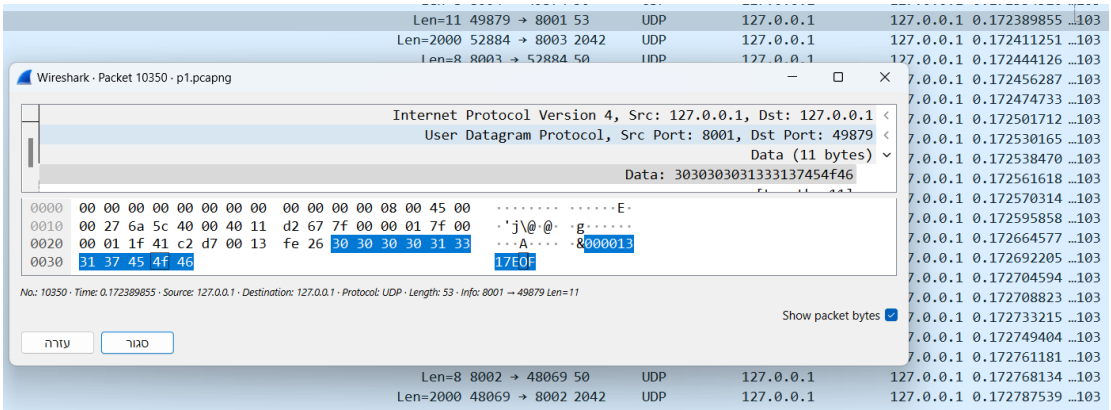
משדר הזרם שולח לו את גודל הזרימה (1992 בתים)



מידע מועבר מהמשדר זרם לקולט זרם



משדר הזרם שולח הודעת EOF – ומבשר על סיום העברת המידע



ניסויים

מטרת העל שלנו בניסויים היתה לבדוק את כמות הזרמים האידיאלית.
אנחנו מצפים כי התוצאה תהיה תלויה במערכת עליה הניסוי מורץ, שכן יש הבדלים בין מחשבים מבחינת מעבדים, כמות ליבות (במיוחד כשיש שימוש בתהליכונים), וגם מערכת הפעלה.

את הניסוי אנחנו נריץ 50 פעמים לכל כמות של זרמים בטווח [1,10]. כלומר 50 הרצות לזרם בודד, 50 הרצות ל-2 זרמים, וכך גם לגבי שאר הגדלים עד 10 זרמים. הסיבה לכך היא שראינו שיש שונות גבוהה בין ההרצות לכל ניסוי. לכן אנחנו נייצר תרשים קופסה (Box plot או Whisker plot) שמראה לכל גודל זרם את הפרמטרים הסטטיסטיים (חציון, רבעון, רבעון עליון), טווח, וחריגים.

סביבות הניסוי

את ההרצות ביצענו בשתי סביבות שונות של מערכת הפעלה Windows 11.

הסביבה הראשונה עם מעבד Intel Core i7-1255U Processor, בעל10 ליבות פיזיות (מהן 2 ליבות ביצועים) ו-8 יעילות. המעבד יכול לטפל בעד 12 משימות בו זמנית.

הסביבה השניה עם חומרה חלשה, עם מעבד Intel Core i5-7200U, בעל 2 ליבות פיזיות ו-4 ליבות לוגיות. המעבד יכול לטפל בעד 4 משימות בו זמנית.

גודל המידע שיועבר בניסויים הוא 20MB, כך שכל פעם יתחלקו שווה בשווה בין כמות הזרמים.

בניסוי הגדרנו כי לכל הזרמים גודל אחיד של חבילות. כל הזרמים יעבירו 2,000 בתים בחבילה, כמות שגם בהנתן עשרה זרמים לא אמורה לייצר עומס ברוחב הפס. החלטנו לייצר גודל אחיד בניסוי על מנת להשוות את התנאים של הריצות ושל כל הזרמים.

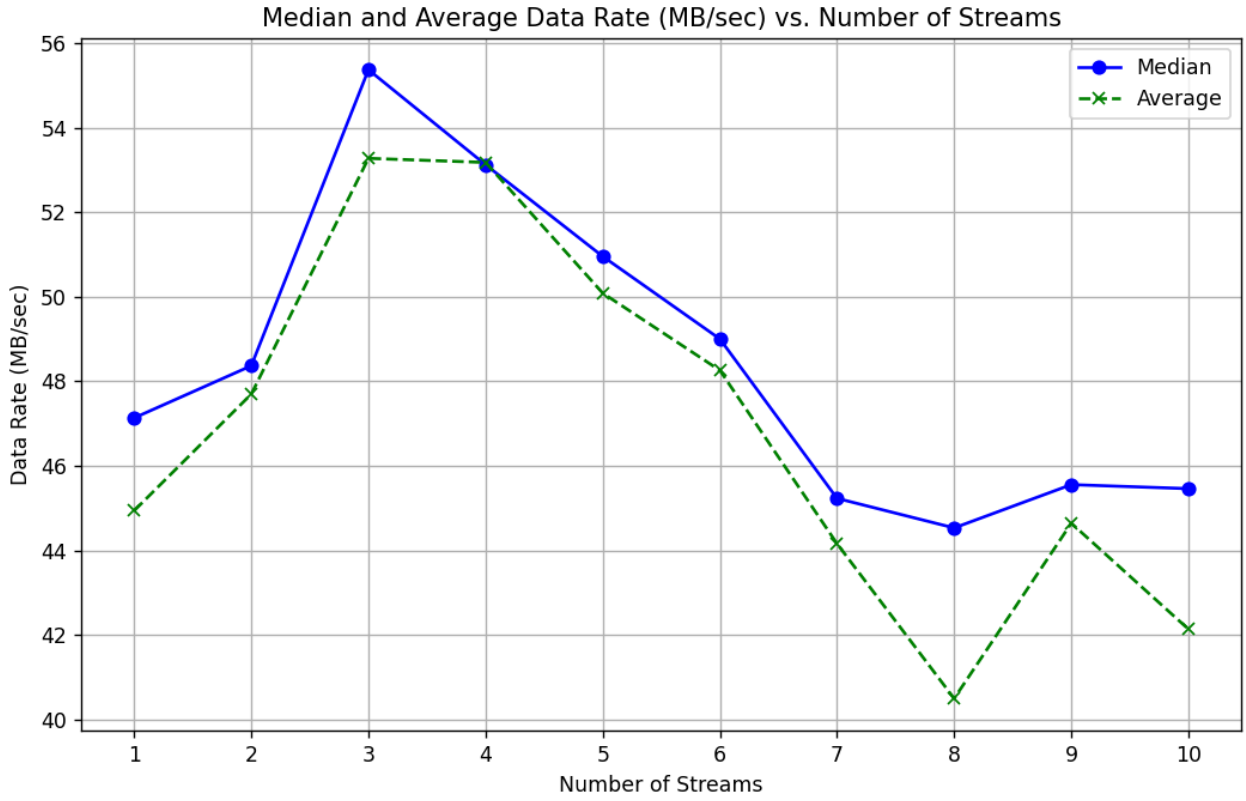
בסביבה הראשונה אנחנו מצפים שעד 6 זרמים (המצריך 12 תהליכונים) הביצועים יהיו טובים, לאחר מכן צפויה להיות ירידה מוסיימת ככל שיהיה גידול במספר הזרמים. לאחר מכן עשויה להיות עליה קטנה בביצועים כתוצאה מכך שתזמון אקראי של שליחה וקבלת חבילות עשוי לנצל טוב יותר את המערכת.

בסביבה השניה אנחנו נצפה לעליה בין זרם אחד לשניים, ובין 2 ל-3 אולי נראה אף ירידה בביצועים, מכיוון שכל זרם מצריך 2 תהליכונים. גם כאן עשויה להיות לאחר הירידה עליה קלה אם המערכת מצליחה לנצל משאבים אחרים ביעילות גבוהה יותר.

תוצאות ומסקנות

את ההרצות שנשמרו ניתחנו באמצעות המודול exp.py, המכיל פונקציונאליות של קריאת הנתונים מהקבצים, שמירת המידע והצגתו בגרפים.

בסביבה הראשונה התקבלו התוצאות:



הקו הכחול מייצג את ערך החציון של ההרצות, הערך הירוק את הממוצע. נראה כי הם בקורלציה אחד לשני ללא סתירות, עם כי ישנם מקומות שבהם הפער הוא יחסית גבוה.

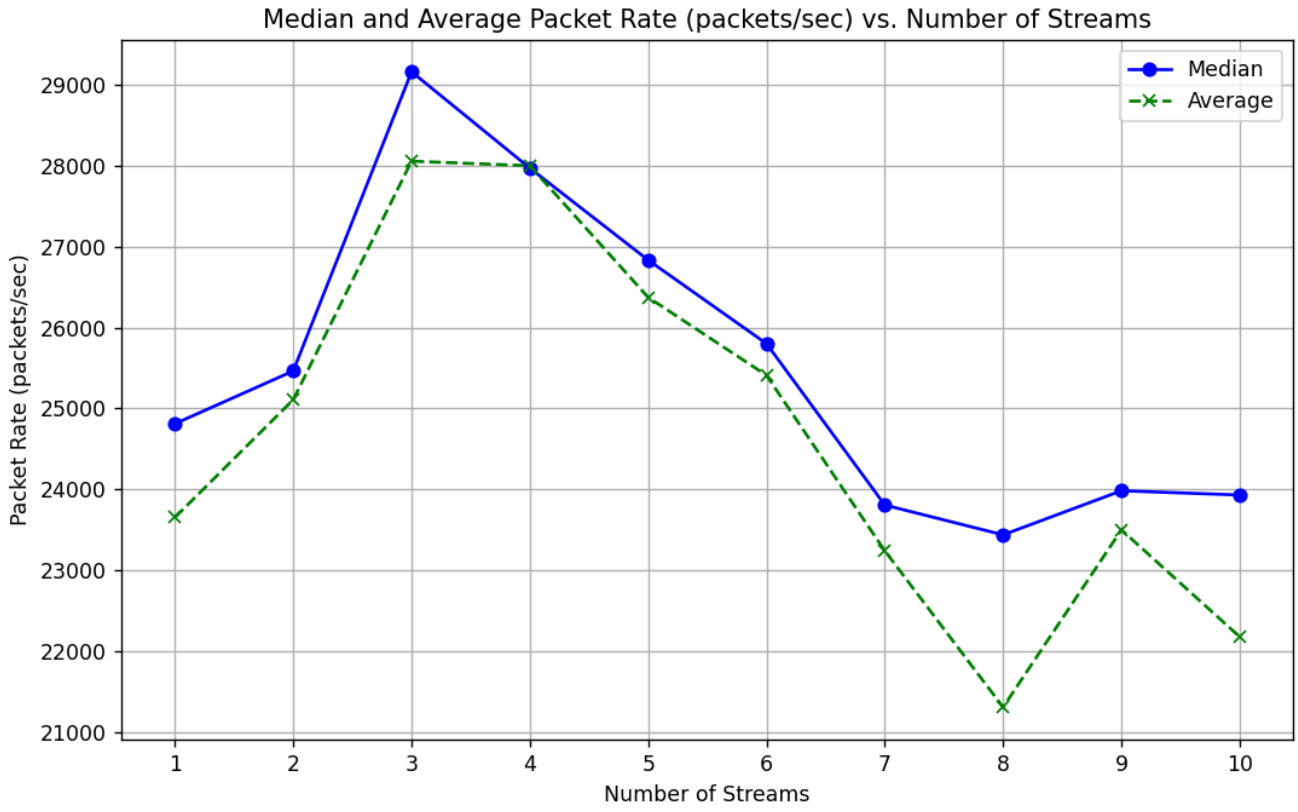
בגרף ניתן להבחין בעליה משמעותית בין זרם בודד ל-3 זרמים. וזה תואם את הציפיות שהיו לנו, מכיוון שזה מראה כי המעבד מנצל את הליבות והתליכונים הזמינים בצורה יעילה.

לאחר שקצב הנתונים מגיע לשיא ב-3 זרמים, חלה ירידה הדרגתית. זה מצביע על כך שהמעבד מגיע למגבלה במשאבים ככל שיש עליה בכמות הזרמים, מה שמצריך כמות גבוהה יותר של תהליכונים.

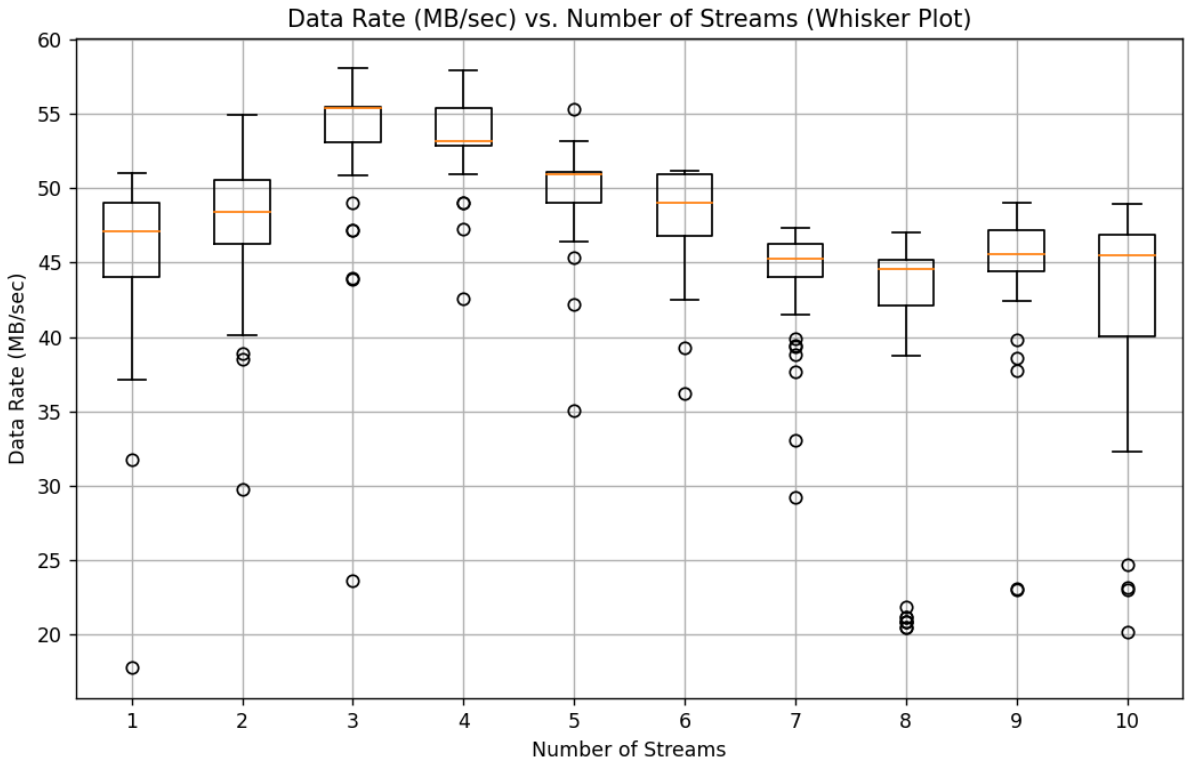
נראה שקצב העברת הנתונים יורד באופן חד כשעוברים את 7 הזרמים. ירידה זו עשויה להצביע על כך שהמעבד מגיע לנקודה בה העומס על הליבות עולה על היכולת של המעבד לנהל את כל התהליכים בצורה אפקטיבית.

ישנה תנודתיות לא יציבה בין 8 ל-9 זרמים, מה שמראה כי המעבד מנסה לאזן את העומסים בין הליבות, אך זה לא מצליח לשחזר את הביצועים של כמות פחותה יותר של זרמים.

מבחינת כמות החבילות העוברות בשניה, התקבלה התוצאה הבאה:



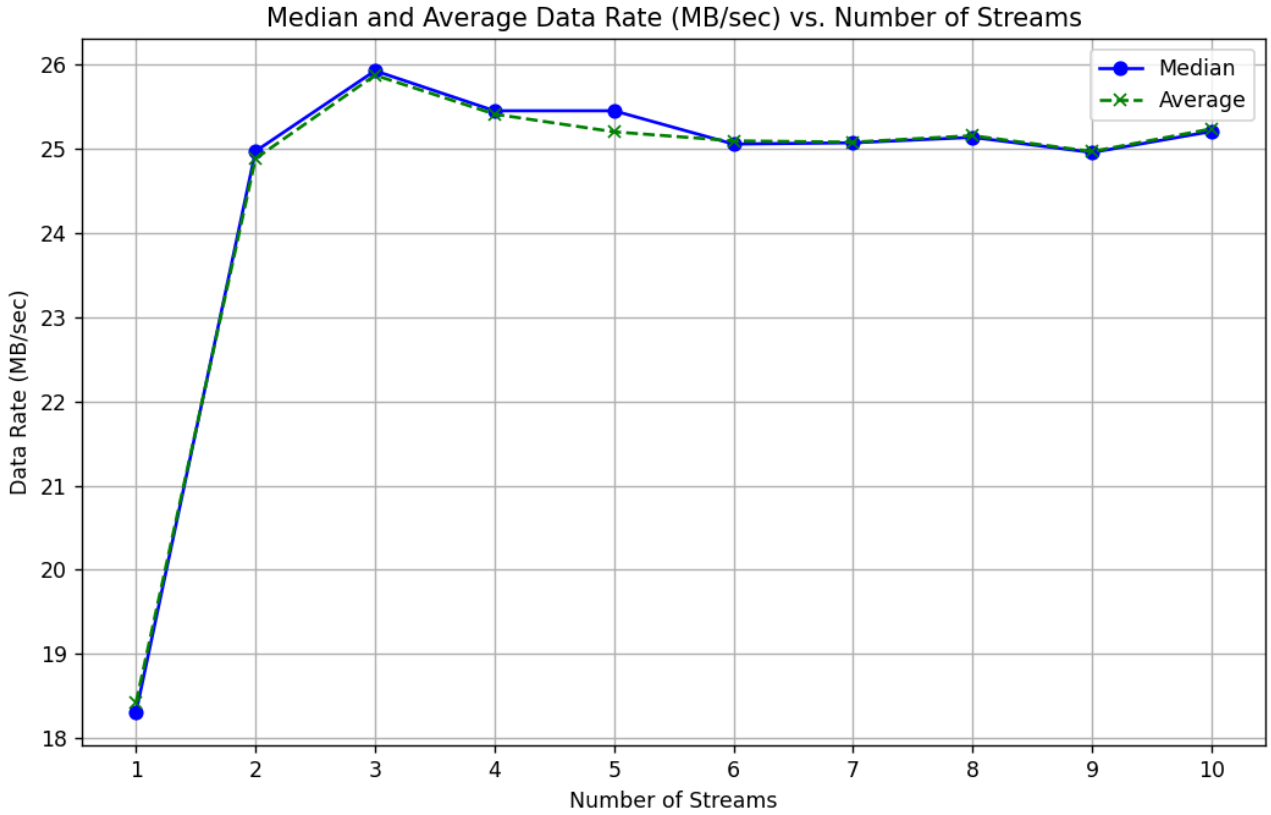
כיוון שגודל החבילות זהה, יש קורלציה מלאה לכמות החבילות שנקבל בכל זרם, כתמונת מראה לכמות המידע העובר.



מבחינה התפלגות התוצאות על פני הזרמים נראה שעבור כמעט כל הזרמים ישנן נקודות חורגות, המציינות מצבים בהם קצב העברת הנתונים היה נמוך בהרבה מהממוצע.
ניתן לראות כי עבור מספר זרמים נמוך (1 עד 4), יש פיזור צר יחסית של תוצאות, מה שמעיד על יציבות גבוהה יותר בקצב העברת הנתונים במצבים אלה. לעומת זאת, עבור מספר זרמים גבוה יותר (כמו 7 עד 10), הפיזור רחב יותר ויש הרבה יותר נקודות חורגות, מה שמעיד על חוסר יציבות בביצועים.

כאשר מספר הזרמים גדול מ-6, התרשים הנוכחי מראה עלייה במספר החריגות כלפי מטה, מה שמעיד על מצבים בהם המערכת לא מצליחה להתמודד עם העומס בצורה טובה.

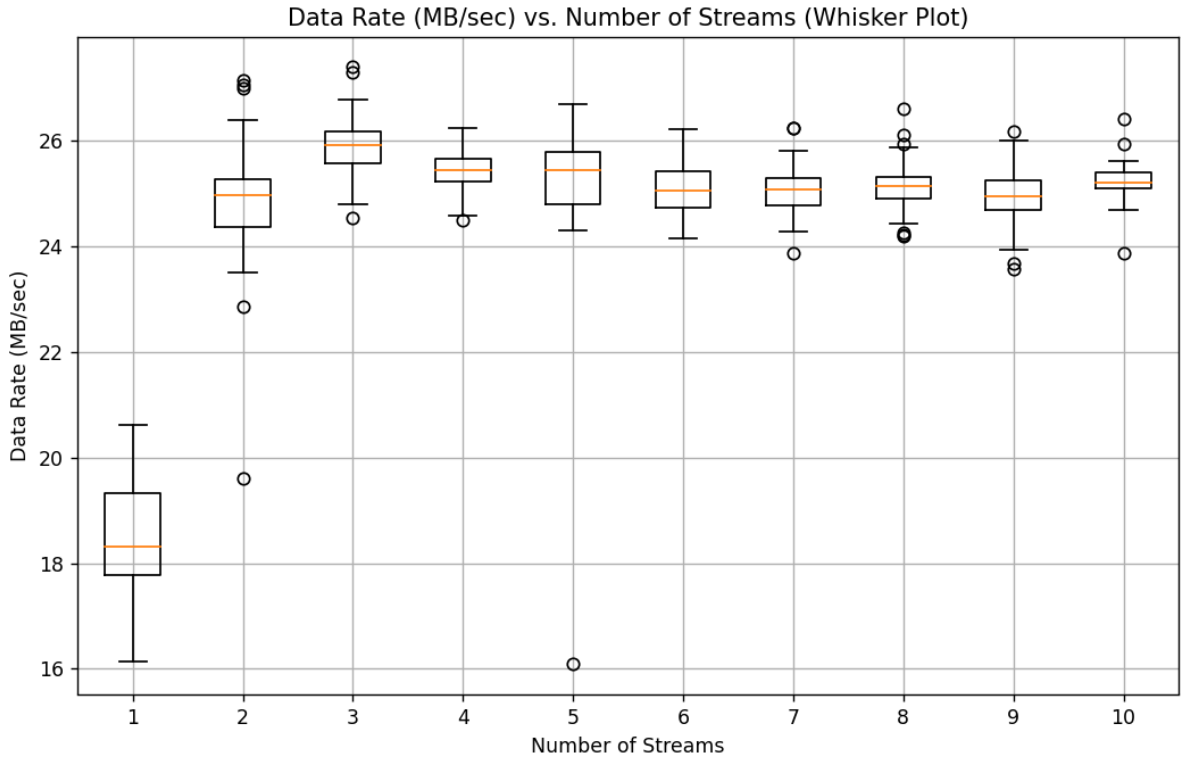
בסביבה השניה התקבלה התמונה הבאה למדד הממוצע והחציון של קצב העברת הנתונים עבור מספר זרימות שונות:



אכן יש עליה בין 1 ל-3 זרמים, מה שאומר שקיימת עליה בקצב העברת הנתונים כאשר אנחנו נעלה מ-1 ל-3 זרמים. לאחר מכן קיימת דעיכה קלה שמתייצבת. זה עשוי להצביע על כך שהמערכת מתחילה להגיע לגבול השימוש היעיל במשאבי חישוב שלה.

בסביבה זו, שמכיל 2 ליבות פיזיות ו-4 ליבות לוגיות, ישנה מגבלה על מספר התהליכונים שהמעבד יכול לנהל במקביל ביעילות. במקרים בהם יש מספר זרמים גבוה, המעבד עלול להגיע לנקודת רוויה, במיוחד כאשר מספר הזרמים גבוה יותר מ-4 (שווה ערך למספר הליבות הלוגיות).

מבחינה התפלגות התוצאות על פני הזרמים נראה שיש יציבות. אנחנו מקבלים "קופסאות" מכווצות, המראה על עקביות במהירות שנמדדה, ללא הרבה מדידות חריגות.



גם כאן, כיוון שגודל החבילות זהה, יש קורלציה מלאה לכמות החבילות שנקבל בכל זרם, כתמונת מראה לכמות המידע העובר.