

רשתות תקשורת מטלה 3

מגישיים: רוני הר טוב 207199282 , אברהם שטרקברג 322530080

פרק 1 TCP

בפרק זה, מימשנו ייצרת קשר באמצעות פרוטוקול TCP בשפת C.
בפרק זה השתמשנו ב IPv4.

- פירוט חלק "השולח" (sender) ב-TCP :

שורות 22-49 : קבלת נתונים מהמשתמש - IP , PORT, ALGO

```

22 /*
23 * ***** TCP Sender main function *****
24 */
25 int main(int argc, char *argv[])
26 {
27     // Retrieve information from the user:
28     const char *algo = NULL;
29     int receiverPORT = 0;
30     const char *receiverIP = NULL;
31
32     for (int i = 1; i < argc; i += 2)
33     {
34         if (strcmp(argv[i], "-ip") == 0)
35             receiverIP = argv[i + 1];
36         else if (strcmp(argv[i], "-p") == 0)
37             receiverPORT = atoi(argv[i + 1]);
38         else if (strcmp(argv[i], "-algo") == 0)
39             algo = argv[i + 1];
40     }
41
42     // Check if the IP or port is already in use:
43     if (receiverIP == NULL || receiverPORT <= 0)
44     {
45         printf("The IP/port is already in use");
46         return -1;
47     }
48     printf("The arguments received!\n");
49

```

שורות 51-82:

יצירת סוקט עם הנתונים שקיבלנו
הגדרת האלגוריתם הרלוונטיreno או cubic (פירוט מצורף מטה)*
יצירת סטרקט שמאחסן את כתובות השרת

```

49 /**
50 * ***** Create the TCP socket: *****
51 */
52
53 int sock = -1;
54 // Create the socket
55 sock = socket(AF_INET, SOCK_STREAM, 0);
56 if (sock == -1)
57 {
58     perror("socket(2)");
59     return 1;
60 }
61 printf("The sender's socket was created!\n");
62
63 // Set Cubic\Reno algo
64 int a = setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, algo, strlen(algo));
65 if (a != 0)
66 {
67     perror("setsockopt(2)");
68     close(sock);
69     return 1;
70 }
71 printf("The algorithm was chosen!\n");
72
73 //The server's address
74 struct sockaddr_in server;
75 memset(&server, 0, sizeof(server));
76
77 if (inet_pton(AF_INET, SERVER_IP, &server.sin_addr) <= 0)
78 {
79     perror("inet_pton(3)");
80     close(sock);
81     return 1;
82 }

```

יבוא הכתובת מטיקסט לביינארית.

- Connect מתחילה לתחבר ל receiver .

לאחר קבלת התשובה מה receiver בוצעה לחיצת ידיים משולשת המUIDה על יצירת הקשר בין

receiver ל server , על פניו המקלט ועל אימות דרכי העברת המUID.

מתחילה ביצירת הקובץ הרנדומלי אותו נרצה להעביר ל receiver .

מציעים ייצרת ופתחת קובץ כהכנה להכנסת נתונים.

```

85     server.sin_family = AF_INET;
86     server.sin_port = htons(SERVER_PORT);
87
88     printf("Connecting to [%s:%d]...\n", SERVER_IP, SERVER_PORT);
89
90     // Try to connect to the server using the socket and the server structure.
91     if (connect(sock, (struct sockaddr *)&server, sizeof(server)) < 0)
92     {
93         perror("connect(2)");
94         close(sock);
95         return 1;
96     }
97     printf("Successfully connected to the server with [%s:%d] using algo %s \n", receiverIP, receiverPORT, algo)
98
99     //Generate random data to be sent to the receiver
100    char *data = util_generate_random_data(DATA_SIZE);
101    if (data == NULL)
102    {
103        printf("Failed to generate data\n");
104        return 1;
105    }
106
107    // Write it to file (if needed)
108    // Open the file to read data and send over TCP
109    FILE *file = fopen("random_data.txt", "wr");
110    if (file == NULL)
111    {
112        printf("Failed to open file for reading\n");
113        free(data);
114        close(sock);
115        return 1;
116    }

```

מציעים כתיבה לקובץ בהתאם למידע שבידינו
סוגרים את הקובץ.

מגדירים buffer בגודל המידע שאנו מעוניינים להעביר
מתחילה בתהילך קבלת המידע- לו לא חיצונית לצורך בקרה- כל עוד השולח מעוניין לשולח עוד
מידע.

.receiver קראת המידע מהקובץ- והעברתו בצורה מבוקרת ל-

```

117    fwrite(data, sizeof(char), DATA_SIZE, file);
118    fclose(file);
119    char *buffer = (char *)malloc(DATA_SIZE);
120    if (buffer == NULL)
121    {
122        perror("malloc");
123        free(data);
124        close(sock);
125        return 1;
126    }
127    while (1)
128    {
129        // Read data from file and send over TCP
130        FILE *file_read = fopen("random_data.txt", "r");
131        if (file_read == NULL)
132        {
133            printf("Failed to open file for reading\n");
134            free(data);
135            close(sock);
136            return 1;
137        }
138        int bytesSent = 0;
139        size_t bytesRead;
140        while ((bytesRead = fread(buffer, sizeof(char), DATA_SIZE, file_read)) > 0)
141        {
142            bytesSent += send(sock, buffer, bytesRead, 0);
143            if (bytesSent < 0)
144            {
145                perror("send(2)");
146                fclose(file);
147                free(data);
148                close(sock);
149                return 1;
150            }

```

שורות 152-153:

לאחר שבוצעה שליחת ה `data`, מבצעים סגירה לקובץ.
בשלב זה שואלים את המשמש האם ברצונם לשלוח עוד קבצים. במידה וכן- נכנסים שוב ללולאה.
במקרה ולא, שלוחים בקשה יציאה "FIN" ומבצעים סגירת קשיה. השולח מכחילה לאישור בקשה
היציאה ולאחר מכן מאשר בעצמו את בקשה היציאה של `receiver`.

```
152     printf("Sent %d bytes to the server!\n", bytesSent);
153     fclose(file_read);
154     printf("The data was sent!\n");
155     // Ask the user:
156     char option = 0;
157     printf("Send more data? [Y/N]: ");
158     scanf(" %c", &option);
159     if (option != 'Y' && option != 'y')
160     {
161         break;
162     }
163     //Exit
164     int exit = send(sock, "EXIT", 4, 0);
165     if (exit < 0)
166     {
167         printf("The exit message failed");
168     }
169     else
170     {
171         printf("Sender sending exit message...\n");
172     }
173     // Free allocated data and close socket
174     free(data);
175     close(sock);
176     printf("->Connection closed!<-\n");
177
178
179     return 0;
180 }
```

- פירוט חלק "המקבל" (receiver) ב-TCP (32-60):

פירוט מצורף למטרה *
קבלת הנתונים מהמשתמש - בחירת קו אלגוריתם `cubic` או `reno`

```
32 * ***** TCP Receiver main function *****
33 */
34 int main(int argc, char *argv[])
35 {
36     // first, we get the input from the user.
37     if (argc != 5) // if the input is illegal
38     {
39         return 1;
40     }
41     // define the algorithm
42     const char *algo = NULL;
43
44     // the port number
45     int port = PORT;
46
47     // gets information from the user:
48     for (int i = 1; i < argc; i += 2)
49     {
50         if (strcmp(argv[i], "-p") == 0)
51             port = atoi(argv[i + 1]);
52         else if (strcmp(argv[i], "-algo") == 0)
53             algo = argv[i + 1];
54     }
55     if (port <= 0)
56     {
57         printf("port number is invalid\n");
58         return 1;
59     }
60     printf("The Arguments received successfully.port number is: %d\n", port);
61 }
```

שורות 64-95:

פתחת סוקט, שימוש ב-IPV4
שמירת הנתונים על כתובות `receiver` ו-`server`, ותחול משתנה לאורך פרטיה השולח.

```

61  /**
62  * ***** create the TCP socket: *****
63  */
64  int sock = -1;
65  // Create the socket
66  sock = socket(AF_INET, SOCK_STREAM, 0);
67  if (sock == -1)
68  {
69      perror("socket(2)");
70      return 1;
71 }
72 printf("The receiver's socket was created!\n");
73
74 // Set Cubic\Reno algo.
75 if (algo != NULL)
76 {
77     if (setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, algo, strlen(algo)) != 0)
78     {
79         perror("Faild to set the socket\n");
80         close(sock); // close the socket
81         return 1;
82     }
83 }
84
85 // The receiver's address
86 struct sockaddr_in receiver;
87
88 // The sender's address
89 struct sockaddr_in sender;
90
91 // The sender's structure length
92 socklen_t sender_length = sizeof(sender);
93
94 memset(&receiver, 0, sizeof(receiver));
95 memset(&sender, 0, sender_length);

```

שורות 97-130:

הגדרת IPV4 והגדרת קבלה ל receiver מכל סוג כתובות 0.0.0.0 ביצוע-BIND- לפि היקו וה port שהתקבלו. ביצוע LISTEN- המקבל נמצא במצב של האזנה לקשרים חדשים. ביצוע ACCEPT- המקבל מסכים ליצור קשר עם השולח ויש קשר שמתקיים ביניהם. מקרים buffer לקראת המידע שעתיד להתקבל מהשולח מאתחלים מספר משתנים לטובת ניתוח העברת המידע

```

96
97     receiver.sin_family = AF_INET;
98     receiver.sin_addr.s_addr = INADDR_ANY;
99
100    // Set the receiver's port to the user port
101    receiver.sin_port = htons(port);
102
103    // Bind
104    if (bind(sock, (struct sockaddr *)&receiver, sizeof(receiver)) < 0)
105    {
106        perror("bind(2)");
107        close(sock);
108        return 1;
109    }
110    printf("receiver binds successfully!\n");
111
112    // The receiver listen....
113    if (listen(sock, CLIENTS_IN_QUEUE) < 0)
114    {
115        perror("listen(2)");
116        close(sock);
117        return 1;
118    }
119    printf("Sender connected. beginning to receive file....\n");
120    // Accept
121    int sender_sock = accept(sock, (struct sockaddr *)&sender, &sender_length);
122    if (sender_sock < 0)
123    {
124        perror("accept(2)");
125        close(sock);
126        return 1;
127    }
128    // using a variables in order to print the statistics after the program finished
129    char buffer[BUFSIZE];
130    int bytes_received;

```

שורות 134-168:

כל עוד יש מידע שעתיד להתקבל ועודין לא התקבל במלואו- פותחים שעון על מנת לנתח זמן ריצה ביצוע RECV- המקבל מקבל את data לתוכה הבادر. בכל פעם נודע שקיבלנו את כל המידע במלואו.

```

134 double *bandwidth = (double *)malloc(sizeof(double) * MAX_REQ);
135 double *times = (double *)malloc(sizeof(double) * MAX_REQ);
136 // while the sender continue to send a files
137 while (1)
138 {
139     double start = current_timestamp(); //start the time
140     double totalB = 0.0;
141     double msec = 0;
142     while (totalB != BUFFER_SIZE)
143     {
144         bytes_received = recv(sender_sock, buffer, BUFFER_SIZE, 0);
145         if (bytes_received < 0)
146         {
147             perror("recv(2)");
148             close(sender_sock);
149             close(sock);
150             return 1;
151         }
152         if (strncmp(buffer, "EXIT", 4) == 0)
153         {
154             break;
155         }
156         // If the amount of received bytes is 0, the client has disconnected.
157         // Close the client's socket and continue to the next iteration.
158         else if (bytes_received == 0)
159         {
160             fprintf(stdout, "Client %s:%d disconnected\n", inet_ntoa(sender.sin_addr), ntohs(sender.sin_port));
161             close(sender_sock);
162             continue;
163         }
164         totalB += bytes_received;
165     }
166     totalAllBytes += totalB;
167     double end = current_timestamp(); //stop the time

```

:167-201 שורות

ניתן לראות חישובים של רוחב פס, מהירות, כמות מידע שהתקבל וכו'.

```

167     totalAllBytes += totalB;
168     double end = current_timestamp(); //stop the time
169     msec = end - start; // Time elapsed in milliseconds
170     totaltime += msec;
171     times[numberOfRun] = msec;
172     // calculate the bandwidth for each run
173     bandwidth[numberOfRun] = ((totalB / 1024 / 1024) / (times[numberOfRun] / 1000.0));
174     numberOfRun++;
175     // if the sender finish
176     if (strncmp(buffer, "EXIT", 4) == 0)
177     {
178         break;
179     }
180     memset(buffer, 0, BUFFER_SIZE);
181 }
182 /**
183 * ***** Prints the statistics *****
184 */
185 for (int i = 0; i < numberOfRun - 1; i++) // each run
186 {
187     printf("*****\n");
188     printf("          #%d run statistics:           \n", (i + 1));
189     printf("*Time: %.3lf ms.\n", times[i]);
190     printf("*Bandwidth: %.3lf MB/s.\n", bandwidth[i]);
191 }
192
193 printf("*****\n");
194 printf("          Total statistics:           \n");
195 if (numberOfRun > 0)
196 {
197     printf("* The AVG time of sending was: %.3lf ms.\n", totaltime / numberofRun);
198     double speed = 0.0;
199     if (totaltime != 0)
200     {

```

*אלגוריתמי בקרת גודש- חיוניים לניהול אופן העברת נתונים ברשת כדי למנוע עומס בראשת, להבטיח שימושיעיל במשאבי הרשת ולשמור על העברת נתונים אמינה. המונחים Reno-Cubic לאלגוריתם שונים של בקרת גודש.

פרק 2: RUDP

בפרק זה, מימשנו פרוטוקול משתמש אמין המבוסס על הפרוטוקול UDP התומך בIPv4 .
פרוטוקול RUDP הוא פרוטוקול העברת מידע המבוסס על הפרוטוקול UDP , אך אמינותו גבוהה יותר.

ישנן מספר דרכי בהם ניתן לשפר את אמינותו של הפרוטוקול UDP, אנחנו בחרנו להוסיף את ה機能יות הבאה:

- 1) פונקציית Connect אשר מבצעת לחיצת ידיים משולשת בדומה לפרוטוקול TCP.
 - 2) שליחת ACK על כל חיבור אשר התקבלה בהצלחה , מתן אפשרות לשילוח חוזרת עבור חבילות שנכשלו בשליחתן , TIMEOUT לחבילות אשר לא הגיעו בזמן.
 - 3) בדיקת CHECKSUM בהעברת המידע לזריה שגיאות, בדיקה כי אוריך החיבור אשר הגיע שווה לאוריך החיבור שנשלח.
 - 4) סגירת הקשר עם שליחת חבילות הרלוונטיות למטרה זו.
- *לכל אורכו ורוחבו של הקוד מבוצע שימוש בפונקציות UDP עם התוספות אשר הוספנו.

מכאן, נפרט על האופן שבו בחרנו למשם את הפרוטוקול בסדר המצוין לעיל ועם צילומי המסך של קטעי הקוד הרלוונטיים:

*נציין כי רוב ההסברים על הקוד מופיעים כהערות מפורנות בקוד עצמו.

1) לחיצתידיים המשולשת:

ראשית על מנת למשם את לחיצתidiים המשולשת הוספנו את HEADER של הפרוטוקול המכיל את גודל DATA שנשלחת בחיבור,Checksum, דגלים לציין סוג המידע.

הדגלים אשר יכולים להתווסף בHEADER של החיבור:

- 'S' - FOR SENDING SYN PACKET.
- 'A' - FOR SENDING ACK PACKET.
- 'Y' - FOR SENDING THE SYN-ACK PACKET.
- 'L' - FOR SENDING THE LAST ACK IN THE 3WAY HANDSHAKE.
- 'D' - FOR FOR SENDING A DATA PACKET.
- 'F' - FOR SENDING THE FIN PACKET.

FLAGS:

```
RUDP.API.h > #include <HEADER_SIZE>
1 #define SYN 'S'
2 #define ACK 'A'
3 #define SYNACK 'Y'
4 #define LASTACKIN3 'L'
5 #define FIN 'F'
```

THE HEADER STRUCT:

```
22 <#include <Header.h>>
23 <#include <types.h>>
24 <#include <sys/types.h>>
25 <#include <sys/socket.h>>
26 <#include <arpa/inet.h>>
27 <#include <netdb.h>>
```

לאחר מכן, מימשו את הפונקציה CONNECT:

```
/*
int rudp_connect(int sock, struct sockaddr_in *addr)
{
    Header syn = {0};
    syn.flags = SYN;
    syn.checksum = calculate_checksum(&syn, sizeof(syn));
    if (sendto(sock, &syn, sizeof(syn), 0, (struct sockaddr *)addr, sizeof(*addr)) < 0)
    {
        printf("Syn message wasn't sent!\n");
        return -1;
    }
    // chatgpt was used in lines 51-58.
    fd_set read_fds;
    struct timeval timeout = {3, 0};
    FD_ZERO(&read_fds);
    FD_SET(sock, &read_fds);
    struct sockaddr_in syn_ack_add;
    struct sockaddr_in syn_ack_from;
    socklen_t from_len = sizeof(syn_ack_from);
    int s = select(sock + 1, &read_fds, NULL, NULL, &timeout);
    printf("SYN message sent, waiting for SYN-ACK\n");
    Header syn_ack;
    if (s > 0)
    {
        if (recvfrom(sock, &syn_ack, sizeof(syn_ack), 0, (struct sockaddr *)&syn_ack_add, &from_len) < 0)
        {
            printf("The SYN-ACK wasn't sent.");
            return -1;
        }
        if (syn_ack.flags != SYNACK)
        {
            printf("received not a SYN-ACK packet");
            return -1;
        }
        printf("The SYN-ACK answer was received!\n");
        Header ack = {0};
        ack.flags = LASTACKIN3;
        ack.checksum = calculate_checksum(&ack, sizeof(ack));
        if (sendto(sock, &ack, sizeof(ack), 0, (struct sockaddr *)addr, sizeof(*addr)) < 0)
        {
            printf("The ACK message wasn't sent\n");
            return -1;
        }
        printf("The ACK was sent!\n");
    }
}
```

פונקציה זו אוחראית על שליחת SYN בצד של SENDER, לאחר מכן מצפה לקבל SYN-ACK מהצד השני, לאחר קבלת SYN-ACK הפונקציה תשלוח ACK בחזרה ל RECEIVER וכן תושלם לחיצת הידיים המשולשת.

במידה והתהליך נכשל בדרך תוחזר השגיאה הרלוונטית.

2) שליחת המידע:

הוספנו את פונקציית SEND_RUDP על מנת להוסיף בדיקה של קבלת ACK על חבילות שהתקבלו, TIMEOUT עבור חבילות שלא הגיעו עליון ACK בזמן, כמו כן איפשרנו שליחת חוזרת של חבילות אשר נכשלו בשילוחן.

```
int rudp_send(int sock, Header *packet, size_t packet_size, const struct sockaddr_in *dest_addr)
{
    int atp = 0;
    Header ack = {0};
    socklen_t from_len = sizeof(struct sockaddr_in);
    struct sockaddr_in from_addr;
    while (atp < MAX_ATTEMPTS)
    {
        int send_bty = sendto(sock, packet, packet_size, 0, (const struct sockaddr *)dest_addr, sizeof(*dest_addr));
        if (send_bty < 0)
        {
            printf("Failed to send the packet!\n");
            return -1;
        }
        // chatgpt was used in lines 108-112.
        fd_set read_fds;
        struct timeval timeout = {4, 0};
        FD_ZERO(&read_fds);
        FD_SET(sock, &read_fds);
        int s = select(sock + 1, &read_fds, NULL, NULL, &timeout);
        if (s > 0 && recvfrom(sock, &ack, sizeof(ack), 0, (struct sockaddr *)&from_addr, &from_len) >= 0 && ack.flags == ACK)
        {
            return send_bty;
        }
        atp++;
    }
    printf("The send function failed, reached to max attempts or timeout!\n");
    return -1;
}
```

הוספנו את הפונקציה `RUDP_RECV` אשר אחראית על קבלת המידע בצד השני לאחר ביצוע `SEND`. כמו כן, פונקציה זו אחראית לשינון החבילות לפי הדגל שלו ושליחת התשובה הרלוונטית לכל סוג של חבילה. בין היתר פונקציה זו תזودא כי `CHECKSUM` והאוורך של החבילה הגיעו בצורה שלא ציפינו, במידה ומזהו הגע בצורה לא תקינה המוכנית תזרוק שגיאה(מה שלא קורה UDP רגיל). פונקציה זו בין היתר תהיה אחראית גם על שליחת `ACK` כאשר חבילות מתיקבלות בהצלחה בעזרת הפונקציה `SENDACK`.

```

int rudp_recv(int sock, void *buffer, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen)
{
    int recv_bytes = recvfrom(sock, buffer, len, flags, src_addr, addrlen);
    if (recv_bytes < 0)
    {
        printf("Receive 0 bytes-error!\n");
        return -1;
    }
    Header *packet = (Header *)buffer;
    unsigned short int senders_side_checksum = packet->checksum;
    int senders_side_length = packet->length;
    packet->checksum = 0;
    unsigned short int received_side_checksum = calculate_checksum(packet, recv_bytes);
    if (packet->flags == 'D')
    {
        received_side_checksum = calculate_checksum(packet + 1, recv_bytes - HEADER_SIZE);
    }
    if (senders_side_checksum != received_side_checksum || senders_side_length != recv_bytes - HEADER_SIZE)
    {
        printf("Checksum error or length error");
        return -1;
    }
    switch (packet->flags)
    {
    case 'D': // data
        rudp_sendack(sock, (struct sockaddr_in *)src_addr);
        break;

    case 'S': // syn
        printf("SYN packet received from the sender!\n");
        Header syn_ack_packet = {0};
        syn_ack_packet.flags = SYNACK;
        syn_ack_packet.checksum = 0;
        syn_ack_packet.checksum = calculate_checksum(&syn_ack_packet, sizeof(syn_ack_packet));
        sendto(sock, &syn_ack_packet, sizeof(syn_ack_packet), 0, src_addr, sizeof(struct sockaddr_in));
        printf("SYN-ACK sent!\n");
        break;

    case 'F': // fin
        printf("FIN packet received from the sender!\n");
        printf("Sending Ack to fin!\n");
        rudp_sendack(sock, (struct sockaddr_in *)src_addr);
        break;

    case 'A': // ack
        printf("ACK packet received!\n");
        break;

    case 'L': // last ack for 3 way handshake
        printf("ack for syn-ack received\n");
        printf("The 3 handshake was completed, waiting for data!\n");
        break;

    default: // if none of the above
        printf("Received an unknown type!\n");
        return -1;
    }
    return recv_bytes;
}

```

:RUDP_CLOSE מtbody בעזרת הפונקציה

```

int rudp_close(int sock, const struct sockaddr_in *server_addr)
{
    Header fin = {0};
    fin.flags = FIN;
    fin.checksum = 0;
    fin.checksum = calculate_checksum(&fin, sizeof(fin));
    if (sendto(sock, &fin, sizeof(fin), 0, (const struct sockaddr *)server_addr, sizeof(*server_addr)) < 0)
    {
        printf("Failed to send fin packet");
        return -1;
    }
    printf("FIN packet sent!\n");
    Header ack = {0};
    if (recvfrom(sock, &ack, sizeof(ack), 0, NULL, NULL) < 0)
    {
        printf("faild to receive the ack packet (fin)\n");
        return -1;
    }
    else
    {
        if (ack.flags != ACK)
        {
            printf("not an ack packet\n");
            return -1;
        }
        printf("ack for FIN received!\n");
        return close(sock);
    }
}

```

בפונקציה זו כאשר בוחרים לסגור את הקשר, ראשית נשלח FIN מהצד הסוגר לצד השני, לאחר מכן מחכים לקבל ACK, במידה והתקבל ACK הקשר נסגר.

הסברים רלוונטיים נוספים:

5) אופן שליחת חבילות המידע הצד של הSENDER

```

//to read from the file.
while ((bytes_read = fread(buffer2, sizeof(char), sizeof(buffer2), file_read)) > 0)
{
    int size = sizeof(Header) + bytes_read;
    // Will set a dynamic size of packet
    // We don't know how much data will be sent and that's the reason for
    // malloc in this part.(not always same size of data).
    Header *packet = (Header *)malloc(size);
    if (packet == NULL)
    {
        printf("malloc for the packet failed.\n");
        break;
    }
    //Will set the arguments of our header:
    packet->length = bytes_read;
    packet->flags = DATA;
    packet->checksum = calculate_checksum(buffer2, bytes_read);
    // The next cell in the memory will be the data that
    // we want to send (from the file):
    memcpy(packet + 1, buffer2, bytes_read);
    //The sending using rudp packet as we declared.
    bytes_sent += rudp_send(sock, packet, size, &receiver);
    if (bytes_sent < 0)
    {
        perror("send(2)");
        fclose(file);
        free(data);
        close(sock);
        return 1;
    }
}
printf("Sent %d bytes to the server!\n", bytes_sent);

```

ראשית נקבעים השדות של HEADER עם החבילה אשר עומדת להשלחה. כל החבילות נשלחות עם HEADER בתא הראשון בזיכרון ולאחר מכן בתא הבא בזיכרון האחרון HEADER נשלחת חבילת המידע כפי שניתן לראות בפונקציה MEMCOPY. בסוף שליחה של הקובץ אנו שואלים האם לשЛОח את הקובץ מחדש. במידה והמשתמש בחר לא לשЛОח את הקובץ שובי התוכנית תסתיימ.

```

char option = 0;
printf("Send more data? (Y/N): ");
scanf(" %c", &option);
if (option != 'Y' && option != 'y')
{
    break;
}

```

שליחת המידע תבוצע דרך BUFFER אליו נקרא המידע מהקובץ הרנדומי. גודל ה BUFFER אצטנו הוא 65400 כיון שהמקסימום שהפונקציה SENDTO בפרוטוקול UDP מאפשר לשלוח הוא 65507 בתים. לכן, אפשר גודל מעט יותר מרוחק למקרים חריגים.

6) אופן קבל המידע מצד של RECEIVER:

BUFFER מקבל בעזרה הפונקציה RUDP_RECV כפי שמפורט לעיל. מצד של מקבל המידע יהיה HEADER בוגול של הקובץ + מכפלה של כמות HEADERSLNG בגודל של HEADER, את החישוב ניתן לראות בקוד המצורף מטה.

```
double totalB = 0;
// the header that will receive to.
Header recv_packet;
// the buffer that will receive the data to.
char buffer[DATA_SIZE + (16 * (DATA_SIZE / 65400))];
memset(buffer, 0, DATA_SIZE + (16 * (DATA_SIZE / 65400)));
double timeTakeninSec = 0;
// this loop will run until we get all the data from the file
// that the sender is trying to send.
double startTime = current_timestamp();
while (totalB < DATA_SIZE + (16 * (DATA_SIZE / 65400)))
{
    // time start.
    // will use the rudp_recv function to receive the data to the buffer.
    int this_run_bytes = rudp_recv(sock, buffer, sizeof(buffer), 0, (struct sockaddr *)&sender, &sender_len);
    if (this_run_bytes == -1)
    {
        perror("recv(2)");
        close(sock);
        break;
    }
}
```

במידה וקיים חבית FIN נבצע עצירה של הלולאות אשר מקבלות מידע, אך בעצם תסתitem התוכנית מצד של RECEIVER:

```
Header *recv = (Header *)buffer;
// printf("the flag is: %d\n", recv->flags);
// if the header that we receive is Fin we want to send
// ack(in the api file) and close the connection.
if (recv->flags == 'F')
{
    close(sock);
    isRunning = 0;
    break;
}
```

ביצוע וחישוב הסטטיסטיות מצד של המקלט באמצעות דומה להשהה TCP:

```
for (int i = 1; i < numberofRun; i++)
{
    printf("*****\n");
    printf("          #%d run statistics:           \n", (i));
    printf("*Time: %.3lf ms.\n", times[i]);
    printf("*Bandwidth: %.3lf MB/s.\n", bandwidth[i]);
    totaltime += times[i];
}
printf("*****\n");
printf("          Total statistics:           \n");
if (numberofRun > 0)
{
    printf("* The AVG time of sending was: %.3lf ms.\n", totaltime / (numberofRun - 1));
    double speed = 0.0;
    if (totaltime != 0)
    {
        double TotalDataMB = totalAllBytes / (1024.0 * 1024.0); // Convert bytes to MB
        speed = TotalDataMB / (totaltime / 1000.0); // Convert totaltime to seconds
    }
    printf("* The AVG bandwidth was: %.3lf MB/s.\n", speed);
}
```

פרק 3: מחקר על TCP

1. מחקר על TCP

הרצה עם אובדן פקודות:

- עברו שימוש בסוגה:

הרוצות עברו 0% אובדן פקודות

הרוצות עברו 2% אובדן פקודות

הרוצות עברו 5% אובדן פקודות

הרוצות עברו 10% אובדן פקודות

- עברו שימוש בcubic:

הרוצות עברו 0% אובדן פקודות

הרוצות עברו 2% אובדן פקודות

הרוצות עברו 5% אובדן פקודות

הרוצות עברו 10% אובדן פקודות

שימוש ב reno:

הרצה עברו 0% אובדן פקודות:

אתחולן עברו 0% אובדן פקודות:

```
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 0%
[sudo] password for roni:
roni@roni-ThinkPad-X1-Carbon-7th:~$
```

הרצה התוכנה בטרמינל:

:receivern חלון ה

TCP_Receiver -p 8888 -algo reno/.

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo reno
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 53.646 ms.
*Bandwidth: 37.281 MB/s.
*****
#2 run statistics:
*Time: 3355.135 ms.
*Bandwidth: 0.596 MB/s.
*****
#3 run statistics:
*Time: 1057.714 ms.
*Bandwidth: 1.891 MB/s.
*****
#4 run statistics:
*Time: 787.646 ms.
*Bandwidth: 2.539 MB/s.
*****
#5 run statistics:
*Time: 801.932 ms.
*Bandwidth: 2.494 MB/s.
*****
Total statistics:
* The AVG time of sending was: 1187.667 ms.
* The AVG bandwidth was: 1.403 MB/s.
*****
->the connection closed .program finished<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 1187.667

רוחב פס ממוצע: MB\S 1.403

חילון הsender

TCP_Sender -ip 127.0.0.1 -p 8888 -algo reno/.

```

ip 127.0.0.1 -p 8888 -algo reno
The arguments received!
The sender's socket was created!
The algorithm was chosen!
Connecting to [127.0.0.1:8888]...
Successfully connected to the server with [127.0.0.1:8888] using algo reno
Sent 2097152 bytes to the server!
The data was sent!
Send more data? [Y/N]: y
Sent 2097152 bytes to the server!
The data was sent!
Send more data? [Y/N]: y
Sent 2097152 bytes to the server!
The data was sent!
Send more data? [Y/N]: y
Sent 2097152 bytes to the server!
The data was sent!
Send more data? [Y/N]: y
Sent 2097152 bytes to the server!
The data was sent!
Send more data? [Y/N]: n
Sender sending exit message...
->Connection closed!<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot projects/ex3$
```

צלום מסך מתוך הקלטה: TCP_reno_0%: מצורפת הקבלה

No.	Time	Source	Destination	Protocol	Length	Info
35	22.175549195	127.0.0.1	127.0.0.1	TCP	74	35628 - 8888 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSecr=1805514384 TSecr=0 WS=128
36	22.175578293	127.0.0.1	127.0.0.1	TCP	74	8888 - 35628 [SYN, ACK] Seq=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSecr=1805514384 TSecr=1805514384
37	22.175580595	127.0.0.1	127.0.0.1	TCP	66	35628 - 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSecr=1805514384 TSecr=1805514384
38	22.227705225	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514384
39	22.227706141	127.0.0.1	127.0.0.1	TCP	66	35628 - 8888 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514384
40	22.227715196	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514384
41	22.227719483	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=65483 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
42	22.227275162	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=65483 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
43	22.227391967	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [PSH, ACK] Seq=98224 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
44	22.227311593	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=139896 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
45	22.227338381	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=139895 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
46	22.227345202	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=139895 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
47	22.227346216	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=1 Ack=139895 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
48	22.227372763	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=190447 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
49	22.227393557	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=190447 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
50	22.227492115	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=229188 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
51	22.227492163	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [PSH, ACK] Seq=1 Ack=229188 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
52	22.227500127	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=229188 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
53	22.227504897	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=229200 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
54	22.227475048	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=229200 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
55	22.227474984	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=934679 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
56	22.227482757	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=327411 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
57	22.227512038	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
58	22.227512039	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=327411 Ack=1 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
59	22.227514918	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
60	22.227549536	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=392983 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
61	22.227568485	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=392983 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
62	22.227576919	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=425632 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
63	22.227596781	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=425634 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
64	22.227606374	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=425634 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
65	22.227606375	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [ACK] Seq=1 Ack=425634 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
66	22.227634468	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=491116 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
67	22.227661374	127.0.0.1	127.0.0.1	TCP	32867	35628 - 8888 [PSH, ACK] Seq=491116 Ack=1 Win=65536 Len=32741 TSecr=1805514436 TSecr=1805514436
68	22.227678841	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=523857 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
69	22.227720733	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=523857 Ack=1 Win=65536 Len=65483 TSecr=1805514436 TSecr=1805514436
70	22.227730838	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=589340 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
71	22.227730839	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=1 Ack=589340 Win=65536 Len=65483 TSecr=1805514436 TSecr=1805514436
72	22.227787166	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=654823 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
73	22.227839393	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=654823 Ack=1 Win=65536 Len=65483 TSecr=1805514436 TSecr=1805514436
74	22.227849578	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=728936 Win=65536 Len=0 TSecr=1805514436 TSecr=1805514436
75	22.227992888	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=728936 Ack=1 Win=65536 Len=65483 TSecr=1805514436 TSecr=1805514436

בתמונה זו ניתן לראות את הנתונים שהכניםנו - הפורט הוא 8888, והו IP 127.0.0.1, והוא מוחזק ל-SYN ACK, המתקבל שולח שולח על ذات ACK. כמו כן ניתן לראות את תהליך לחיצת הידיים ב-3 השורות הראשונות. כשאר השולח שולח SYN ACK, המתקבל שולח שולח על ذات ACK. ניתן לראות את המושברות פקטו.

No.	Time	Source	Destination	Protocol	Length	Info
265	28.230949668	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=8977956 Win=0 TSecr=1805520440 TSecr=1805520440
266	28.230975348	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=8977956 Ack=1 Win=65536 TSecr=1805520440 TSecr=1805520440
267	28.230956569	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=3043439 Win=3114016 Len=99 TSecr=1805520440 TSecr=1805520440
268	28.231000424	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=1 Ack=3043439 Win=3114016 Len=99 TSecr=1805520440 TSecr=1805520440
269	28.231121938	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=3108922 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
270	28.231167532	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=108922 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
271	28.231216573	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9174465 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
272	28.231241656	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9174465 Win=3112574 Len=99 TSecr=1805520440 TSecr=1805520440
273	28.231243444	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9239888 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
274	28.231243445	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9239888 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
275	28.231243446	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9305371 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
276	28.231243448	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=1 Ack=9378854 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
277	28.231349966	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9378854 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
278	28.231392228	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=1 Ack=9378854 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
279	28.231459763	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9436337 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
280	28.231509242	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9436337 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
281	28.231510242	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9436337 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
282	28.231510243	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9567393 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
283	28.231689284	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9567383 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
284	28.231689308	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9632789 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
285	28.231689309	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9698269 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
286	28.231741841	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9698269 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
287	28.231741842	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9698269 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
288	28.231920891	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=98924713 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
289	28.231955232	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=98924735 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
290	28.231981465	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=1 Ack=9962081 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
291	28.231993928	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=9962081 Win=3112576 Len=99 TSecr=1805520440 TSecr=1805520440
292	28.232098722	127.0.0.1	127.0.0.1	TCP	65549	35628 - 8888 [ACK] Seq=9962081 Ack=1 Win=65536 Len=65483 TSecr=1805520440 TSecr=1805520440
293	28.232205118	127.0.0.1	127.0.0.1	TCP	66	8888 - 35628 [ACK] Seq=1 Ack=10485761 Win=3144448 Len=99 TSecr=1805521511 TSecr=1805521511
294</td						

לבסוף, ניתן לראות שנשלחות בקשות FIN שמעודת על סגירת הקשר מ2 הצדדים.
nicr שאינו כל "עיכוב" ממשועוטי בדרך אכן כי הגדרנו שלא משתמש באובדן פקודות.

הרצות עברו 2% אובדן פקודות:
אתה חול עבו 0% אובדן פקודות

```
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc del dev lo root netem
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 2%
roni@roni-ThinkPad-X1-Carbon-7th:~$
```

חלון הרץ :receivern
TCP_Receiver -p 8888 -algo reno/.

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo reno
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 1.968 ms.
*Bandwidth: 1016.377 MB/s.
*****
#2 run statistics:
*Time: 2797.237 ms.
*Bandwidth: 0.715 MB/s.
*****
#3 run statistics:
*Time: 1417.345 ms.
*Bandwidth: 1.411 MB/s.
*****
#4 run statistics:
*Time: 1085.351 ms.
*Bandwidth: 1.843 MB/s.
*****
#5 run statistics:
*Time: 1289.240 ms.
*Bandwidth: 1.551 MB/s.
*****
Total statistics:
* The AVG time of sending was: 1251.549 ms.
* The AVG bandwidth was: 1.332 MB/s.
*****
->the connection closed .program finished<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 1251.549
רוחב פס ממוצע: MB\S 1.322

(ה sender מופיע בטרמינל בדיק אותו דבר لكن תמונה המסר שצולמה ב 0% תהיה רלוונטית מכאן והלאה עברו כל ריצה).

צילום מסך מתוך Wireshark: מצורפת הקלטה: **TCP_reno_2%**:
בתמונה זו ניתן לראות את הנתונים שהכנסנו- הפורט הוא 8888, והו IP 127.0.0.1.
ניתן לראות את תהליך לחיצת הידיים ב 3 השורות הראשונות.
כשאר השולח שולח SYN , המקבל מחזר לו SYN ACK והשולח שולח על זאת ACK.
במהרש מועברות פקודות :

הרצות עברו 5% אובדן פקודות.

```
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc del dev lo root netem
[sudo] password for roni:
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 5%
```

חלון הרץ: receiver

TCP_Receiver -p 8888 -algo reno/.

```
Total statistics:
* The AVG time of sending was: 943.417 ms.
* The AVG bandwidth was: 1.767 MB/s.
*****
->the connection closed .program finished<-
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo reno
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 30.850 ms.
*Bandwidth: 64.830 MB/s.
*****
#2 run statistics:
*Time: 3607.898 ms.
*Bandwidth: 0.554 MB/s.
*****
#3 run statistics:
*Time: 725.948 ms.
*Bandwidth: 2.755 MB/s.
*****
#4 run statistics:
*Time: 1996.037 ms.
*Bandwidth: 1.002 MB/s.
*****
#5 run statistics:
*Time: 1601.555 ms.
*Bandwidth: 1.249 MB/s.
*****
Total statistics:
* The AVG time of sending was: 1607.144 ms.
* The AVG bandwidth was: 1.037 MB/s.
*****
->the connection closed .program finished<-
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 1251.549

רוחב פס ממוצע: MB\S 1.322

צילום מסך מתוך WIRESHARK: מציגת הקלטה: TCP_reno_5%

בתמונה הבאה ניתן לראות שבמהלך הדריך נתקלנו ב"שורות שחורות" רבות יותר מאשר היו ב-2% אובדן פקודות(ኒיכר בהקלטה). כיוון שהגדכנו מצב זה של 5% אובדן פקודות, אפשר בקלות שזה היה קושי למכוד אותו בWIRESHARK ולכן ישנו מספר לא מועט של פקודות שנכשלו בהעברה.

No.	Time	Source	Destination	Protocol	Length	Info
157	11. 8816698914	127.0.0.1	127.0.0.1	TCP	65540	33694 - 8888 [ACK] Seq=5242983 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
158	11. 8817155667	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5307516 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
159	11. 881724275	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=5372989 Win=3112576 Len=0 Tsvl=1807746148 Tscr=1807746148
160	11. 881761394	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5372999 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
161	11. 881801514	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5438482 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
162	11. 881801719	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=5438483 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
163	11. 881801724	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5509440 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
164	11. 881997735	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=55094401 Ack=1 Win=65536 Len=65483 Tsvl=1807746148 Tscr=1807746148 [TCP segment of a reassembled segment captured]
165	11. 881997736	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=5634931 Win=3112576 Len=0 Tsvl=1807746148 Tscr=1807746148
166	11. 882018382	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 33694 - 8888 [PSH, ACK] Seq=5705897 Ack=1 Win=65536 Len=65483 Tsvl=1807746149 Tscr=1807746149
167	11. 882048455	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 33694 [ACK] Seq=1 Ack=5831388 Win=65483 Tsvl=1807746149 Tscr=1807746149
168	11. 882196153	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5831388 Ack=1 Win=65536 Len=65483 Tsvl=1807746149 Tscr=1807746149 [TCP segment of a reassembled segment captured]
169	11. 882296179	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=5896845 Ack=1 Win=65536 Len=65483 Tsvl=1807746149 Tscr=1807746149 [TCP segment of a reassembled segment captured]
170	11. 882296763	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 33694 [ACK] Seq=1 Ack=60291457 Win=3145728 Len=0 Tsvl=1807746149 Tscr=1807746149
171	11. 882296719	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 33694 [ACK] Seq=1 Ack=60291457 Win=3145728 Len=0 Tsvl=1807746149 Tscr=1807746149
172	12. 546694130	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6422423 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 [TCP segment of a reassembled segment captured]
173	12. 546694297	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 17#] 33694 [ACK] Seq=1 Ack=6291457 Win=3145728 Len=0 Tsvl=1807746013 Tscr=1807746013 SLE=6356940
174	12. 546738768	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6422423 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 [TCP segment of a reassembled segment captured]
175	12. 546738773	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6422423 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 [TCP segment of a reassembled segment captured]
176	12. 546738774	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 33694 - 8888 [ACK] Seq=6484355 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013
177	12. 546915054	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6749838 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
178	12. 546915059	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6815321 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
179	12. 546915064	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6815321 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
180	12. 546915069	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6815321 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
181	12. 546915074	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6815321 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
182	12. 546915079	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6815321 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
183	12. 546915084	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 33694 - 8888 [ACK] Seq=684355 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013
184	12. 546915089	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=684355 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
185	12. 546915094	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=684355 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
186	12. 547099134	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6880994 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 [TCP segment of a reassembled segment captured]
187	12. 547109884	127.0.0.1	127.0.0.1	TCP	65549	[TCP Dup ACK 18#] 33694 [ACK] Seq=6946287 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 SLE=6684355
188	12. 547133218	127.0.0.1	127.0.0.1	TCP	78	8888 - 33694 [ACK] Seq=1 Ack=6553389 Win=2883848 Len=0 Tsvl=1807746013 Tscr=1807746013 SLE=6084355
189	12. 547162514	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6946287 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 [TCP segment of a reassembled segment captured]
190	12. 547162519	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6946287 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013 [TCP segment of a reassembled segment captured]
191	12. 547167448	127.0.0.1	127.0.0.1	TCP	65549	[TCP Dup ACK 18#] 33694 [ACK] Seq=1 Ack=6553389 Win=2883848 Len=0 Tsvl=1807746013 Tscr=1807746013 SLE=6684355
192	12. 547204431	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 33694 - 8888 [ACK] Seq=6553389 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013
193	12. 547204436	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 33694 - 8888 [ACK] Seq=6553389 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746013
194	12. 547204441	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=6553389 Ack=1 Win=65536 Len=65483 Tsvl=1807746013 Tscr=1807746014 SLE=6084355
195	12. 547259128	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=7011778 Ack=1 Win=65536 Len=65483 Tsvl=1807746014 Tscr=1807746014
196	12. 547269114	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=7011778 Win=2884889 Len=0 Tsvl=1807746014 Tscr=1807746014
197	12. 547294656	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=7077253 Ack=1 Win=65536 Len=65483 Tsvl=1807746014 Tscr=1807746014 [TCP segment of a reassembled segment captured]
198	12. 547329554	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=7077253 Ack=1 Win=65536 Len=65483 Tsvl=1807746014 Tscr=1807746014 [TCP segment of a reassembled segment captured]
199	12. 547335224	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=7077253 Ack=1 Win=65536 Len=65483 Tsvl=1807746014 Tscr=1807746014
200	12. 547335272	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=7142738 Win=3922992 Len=0 Tsvl=1807746014 Tscr=1807746014
201	12. 547336737	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=7208219 Win=3922992 Len=0 Tsvl=1807746014 Tscr=1807746014
202	12. 547369246	127.0.0.1	127.0.0.1	TCP	65549	33694 - 8888 [ACK] Seq=7208219 Ack=1 Win=65536 Len=65483 Tsvl=1807746014 Tscr=1807746014 [TCP segment of a reassembled segment captured]
203	12. 547369250	127.0.0.1	127.0.0.1	TCP	66	8888 - 33694 [ACK] Seq=1 Ack=7273762 Win=3112576 Len=0 Tsvl=1807746014 Tscr=1807746014

הרשות עבר % אובן פקטוט:

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3/rec_ex3$ sudo tc qdisc add dev lo root netem loss 10%
```

:receivern חולן

TCP_Receiver -p 8888 -algo reno/.

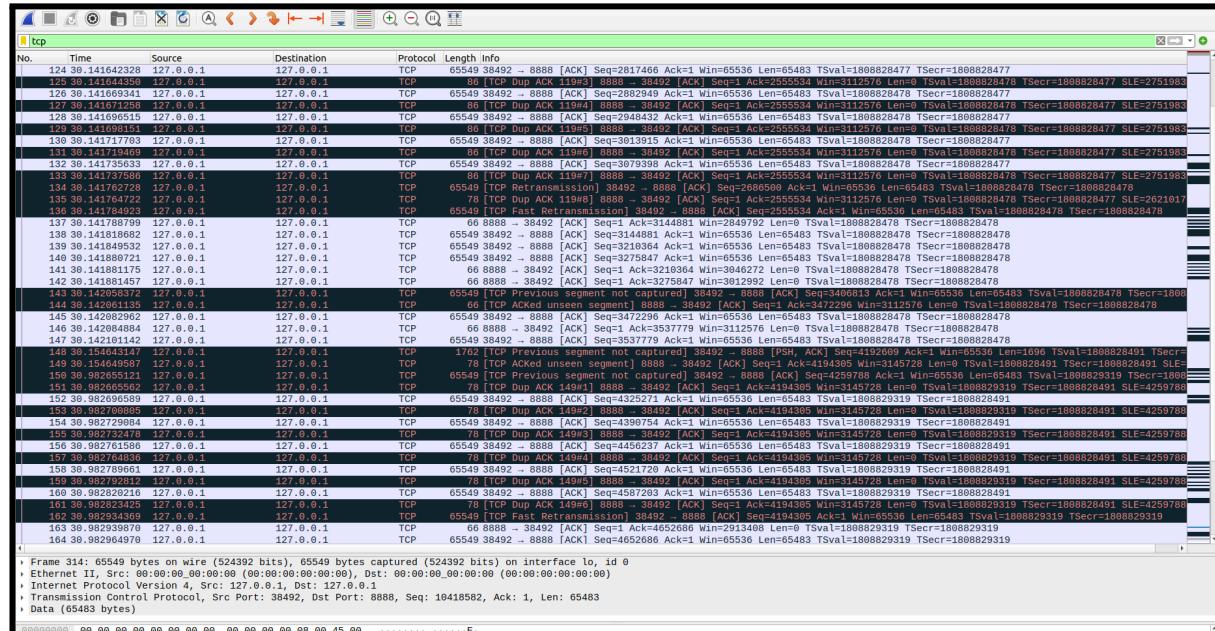
```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver -p 8888 -algo reno
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****# run statistics:
*Time: 55.857 ms.
*Bandwidth: 35.806 MB/s.
*****# run statistics:
*Time: 5139.512 ms.
*Bandwidth: 0.389 MB/s.
*****# run statistics:
*Time: 5749.703 ms.
*Bandwidth: 0.348 MB/s.
*****# run statistics:
*Time: 1256.701 ms.
*Bandwidth: 1.591 MB/s.
*****# run statistics:
*Time: 1646.191 ms.
*Bandwidth: 1.215 MB/s.
*****# run statistics:
* The AVG time of sending was: 2544.440 ms.
* The AVG bandwidth was: 0.655 MB/s.
*****->the connection closed .program finished<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: 2544.440

רוחב פס ממוצע: 0.655 MB/S

צלום מסך מתוך Wireshark: מיצירת הקלטה: TCP_reno_10%: Wireshark

בתמונה הבאה ניתן לראות שבמהלך הדרך נתקלנו ב"שורות שחורות" רבות. הרבה יותר מאשר היו ב-5% אובדן פקודות(nicer בהקלטה). כיוון שהגדרכנו מצב זה של 10% אובדן פקודות, אפשר בקלות שזה היה קושי למכוד אותו בWireshark וכן ישנו מספר לא מעט של פקודות שנכשלו בהערכה.



טבלת סיכום סטטיסטיות עבור אובדן פקודות ב프וטוקול TCP עם אלגוריתם reno:

BANDWIDTH\MB.S	TIME\MS	%LOSS PACKETS
1.403	1187.667	0%
1.322	1251.549	2%
1.037	1607.144	5%
0.655	2544.440	10%

שימוש ב cubic

אתחול עברו 0% אובדן פקודות

```
[sudo] password for roni:  
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 0%  
:receivern חלון TCP_Receiver -p 8888 -algo cubic/.
```

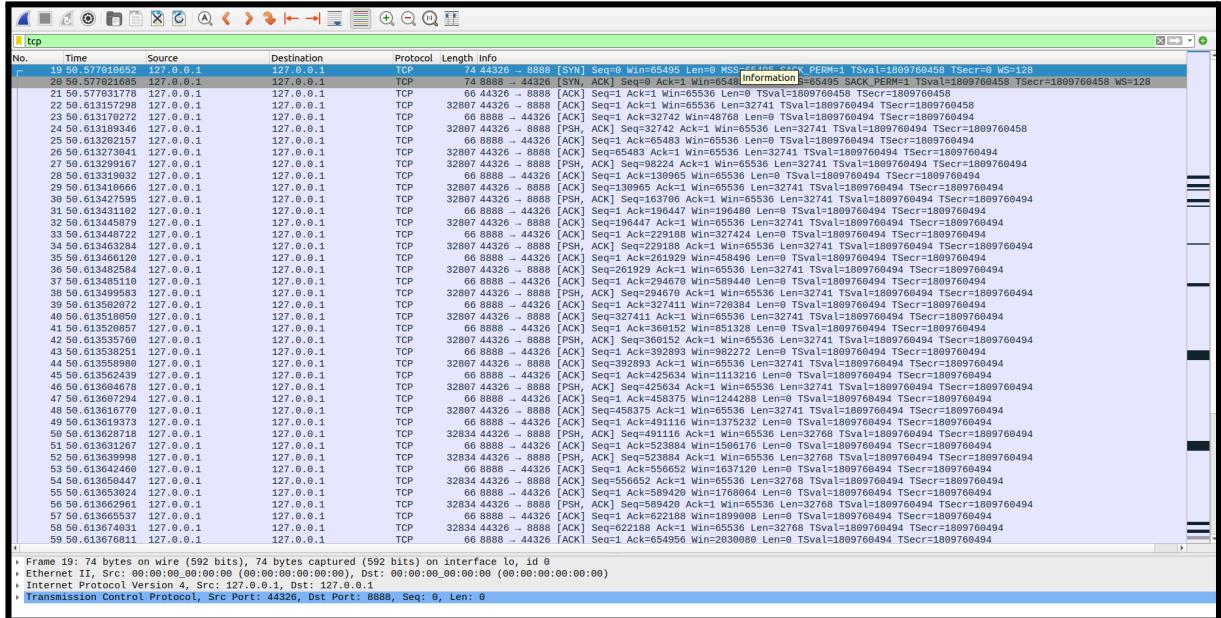
```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver  
-p 8888 -algo cubic  
The Arguments received successfully.port number is: 8888  
The receiver's socket was created!  
receiver binds successfully!  
Sender connected. beginning to receive file....  
*****  
#1 run statistics:  
*Time: 41.164 ms.  
*Bandwidth: 48.586 MB/s.  
*****  
#2 run statistics:  
*Time: 2761.737 ms.  
*Bandwidth: 0.724 MB/s.  
*****  
#3 run statistics:  
*Time: 553.024 ms.  
*Bandwidth: 3.616 MB/s.  
*****  
#4 run statistics:  
*Time: 532.509 ms.  
*Bandwidth: 3.756 MB/s.  
*****  
#5 run statistics:  
*Time: 523.688 ms.  
*Bandwidth: 3.819 MB/s.  
*****  
Total statistics:  
* The AVG time of sending was: 974.895 ms.  
* The AVG bandwidth was: 1.710 MB/s.  
*****  
->the connection closed .program finished<-  
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 974.895
רוחב פס ממוצע: MB/S 1.710

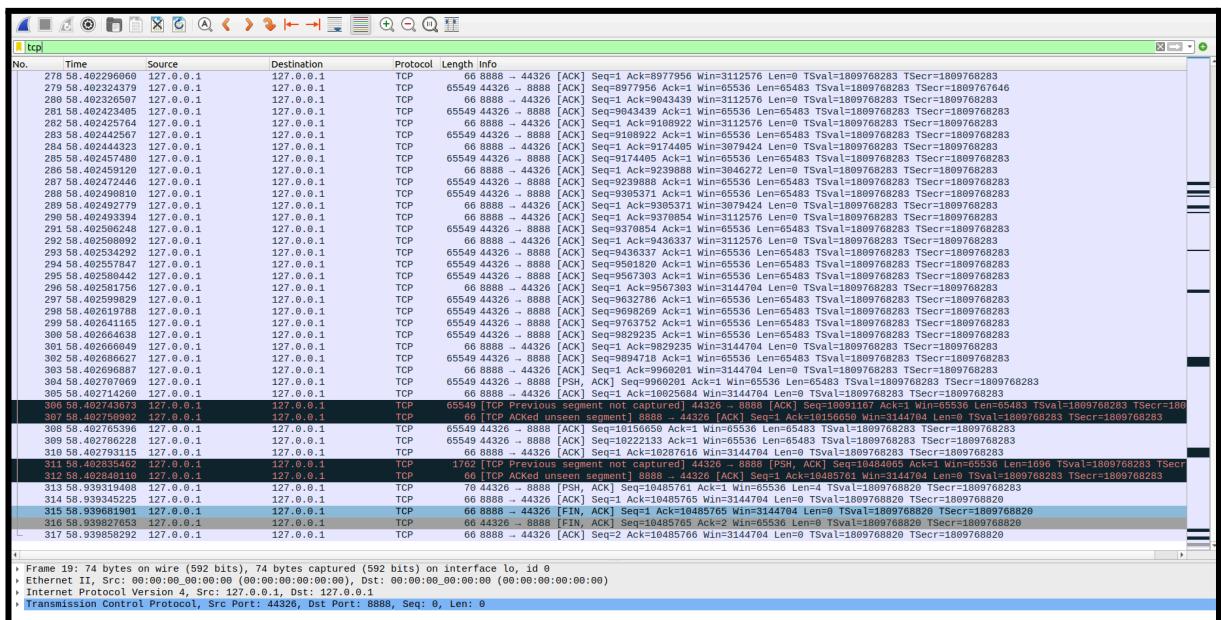
חלון sendern TCP_Sender -ip 127.0.0.1 -p 8888 -algo cubic/.

```
bind(2). Address already in use  
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Sender -  
ip 127.0.0.1 -p 8888 -algo cubic  
The arguments received!  
The sender's socket was created!  
The algorithm was chosen!  
Connecting to [127.0.0.1:8888]...  
Successfully connected to the server with [127.0.0.1:8888] using algo cubic  
Sent 2097152 bytes to the server!  
The data was sent!  
Send more data? [Y/N]: y  
Sent 2097152 bytes to the server!  
The data was sent!  
Send more data? [Y/N]: y  
Sent 2097152 bytes to the server!  
The data was sent!  
Send more data? [Y/N]: y  
Sent 2097152 bytes to the server!  
The data was sent!  
Send more data? [Y/N]: y  
Sent 2097152 bytes to the server!  
The data was sent!  
Send more data? [Y/N]: n  
Sender sending exit message...  
->Connection closed!<-  
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

צלום מסך מתוך Wireshark: TCP_cubic_0% מציגות הקלטה:
 בתמונה זו ניתן לראות את הנתונים שהכנסנו - הפורט הוא 8888, וה IP 127.0.0.1.
 ניתן לראות את תהליך לחיצת הידיים ב 3 השורות הראשונות.
 כאשר השולח שולח SYN, המתקבל מהדריך לו SYN ACK והשולח שולח על זאת ACK.
 בהמשך מועברות פקודות:



לבסוף, ניתן לראות שנשלחות בקשות FIN שמיעידות על סגירת הקשר מ 2 הצדדים.
 ניכר שאין כל "עיכוב" משמעותי בדרך אקן כי הגדרנו שלא השתמש באובדן פקודות.



אתחול עבור 2% אובדן פקודות:

```
[root@roni ~]# sudo password for root.
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 2%
```

:chanonTCP_Receiver -p 8888 -algo cubic/.

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo cubic
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 45.416 ms.
*Bandwidth: 44.037 MB/s.
*****
#2 run statistics:
*Time: 1839.988 ms.
*Bandwidth: 1.087 MB/s.
*****
#3 run statistics:
*Time: 476.859 ms.
*Bandwidth: 4.194 MB/s.
*****
#4 run statistics:
*Time: 479.226 ms.
*Bandwidth: 4.173 MB/s.
*****
#5 run statistics:
*Time: 1147.826 ms.
*Bandwidth: 1.742 MB/s.
*****
Total statistics:
* The AVG time of sending was: 755.984 ms.
* The AVG bandwidth was: 2.205 MB/s.
*****
->the connection closed .program finished<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ █
```

זמן ריצה ממוצע: MS 755.984
רוחב פס ממוצע: MB\S 2.205
צילום מסך מתרן WIRESHARK: מצורפת הקלטה: TCP_cubic_2%: בתוכונה זו ניתן לראות את הנתונים שהכניםנו- הפורט הוא 8888, והא IP 127.0.0.1. ניתן לראות את תהליך לחיצת הידיים ב 3 השורות הראשונות. כשאר השולח שולח SYN , המתקבל מחזר לו SYN ACK והשלוח שולח על זאת ACK. בהמשך מועברות פקודות :

No.	Time	Source	Destination	Protocol	Length	info
9	16.8674338898	127.0.0.1	127.0.0.1	TCP	74	51988 → 8888 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TsvaI=1810447534 TSecr=8 WS=128
10	16.8674558841	127.0.0.1	127.0.0.1	TCP	74	8888 → 51988 [SYN, ACK] Seq=1 Ack=1 Win=65483 Len=0 TSvaI=1810447534 TSecr=1810447534 WS=128
11	16.867478391	127.0.0.1	127.0.0.1	TCP	66	51988 → 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSvaI=1810447534 TSecr=1810447534
12	16.855988592	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
13	16.856010694	127.0.0.1	127.0.0.1	TCP	66	51988 → 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSvaI=1810447583 TSecr=1810447534
14	16.8560128236	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=32742 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
15	16.8560686877	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSvaI=1810447583 TSecr=1810447534
16	16.8560692907	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=65483 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
17	16.8560692907	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=98229 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
18	16.856103679	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=138965 Win=196480 Len=0 TSvaI=1810447583 TSecr=1810447534
19	16.8561036792	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=138965 Ack=1 Win=196480 Len=0 TSvaI=1810447583 TSecr=1810447534
20	16.856117702	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=137766 Win=327424 Len=0 TSvaI=1810447583 TSecr=1810447534
21	16.8561275549	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=137706 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
22	16.856130283	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=164447 Win=458496 Len=0 TSvaI=1810447583 TSecr=1810447534
23	16.856139991	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=196447 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
24	16.856142015	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=229181 Win=589448 Len=0 TSvaI=1810447583 TSecr=1810447534
25	16.8561539018	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=138965 Ack=1 Win=196480 Len=0 TSvaI=1810447583 TSecr=1810447534
26	16.8561539018	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=138965 Ack=1 Win=196480 Len=0 TSvaI=1810447583 TSecr=1810447534
27	16.856165789	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=261929 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
28	16.856168412	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=294670 Win=851328 Len=0 TSvaI=1810447583 TSecr=1810447534
29	16.856177740	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=294670 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
30	16.856180167	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=32741 Win=982272 Len=0 TSvaI=1810447583 TSecr=1810447534
31	16.856193616	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
32	16.856202021	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
33	16.856205225	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=360152 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
34	16.8562057681	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=392893 Win=1244288 Len=0 TSvaI=1810447583 TSecr=1810447534
35	16.856216841	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=392893 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
36	16.856219262	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=425632 Win=1375232 Len=0 TSvaI=1810447583 TSecr=1810447534
37	16.856219262	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=425634 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
38	16.856224667	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=425634 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
39	16.856242667	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [ACK] Seq=485375 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534
40	16.856245156	127.0.0.1	127.0.0.1	TCP	66	8888 → 51988 [ACK] Seq=1 Ack=491116 Win=1637120 Len=0 TSvaI=1810447583 TSecr=1810447534
41	16.856257751	127.0.0.1	127.0.0.1	TCP	32807	51988 → 8888 [PSH, ACK] Seq=491116 Ack=1 Win=65536 Len=32741 TSvaI=1810447583 TSecr=1810447534

Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 51988, Dst Port: 8888, Seq: 0, Len: 0

בתמונה הבאה ניתן לראות שבמהלך הדרך נתקלנו ב"שורות שחורות", הכוונה לפקוטות שהלכו לאיבוד בדרך. כיוון שהגדרכנו מצב זה של 2% אובדן פקוטות, אפשר בקבילות שזה היעוה קושי לילכוד אותן ב WIRESHARK ולכן ישם פקוטות שנכשלו בהערכה.

No.	Time	Source	Destination	Protocol	Length	Info
57	16. 856493843	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=1113204 Win=0 TSval=1810447583 TSecr=1810447583	
58	16. 856515156	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1113204 Ack=1 Win=0 TSval=1810447583 Len=65536 Len=65483 TSecr=1810447583	
59	16. 856539138	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1178687 Ack=1 Win=0 TSval=1810447583 Len=65536 Len=65483 TSecr=1810447583	
60	16. 856542416	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=1244170 Win=0 TSval=1810447583 TSecr=1810447583	
61	16. 856542417	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1 Ack=1244170 Win=0 TSval=1810447583 Len=65536 Len=65483 TSecr=1810447583	
62	16. 856610659	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [PSH, ACK] Seq=173156 Ack=1 Win=0 TSval=1810447583 Len=65483 TSecr=1810447583	
63	16. 856622385	127.0.0.1	127.0.0.1	TCP	66 [TCP Acked unseen segment] 51988 - 8888 [ACK] Seq=1 Ack=1406191 Len=0 TSval=1810447583 TSecr=1810447583	
64	16. 856654489	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=1715196 Ack=1 Win=0 TSval=1810447584 TSecr=1810447584	
65	16. 856657555	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 8888 - 51988 [ACK] Seq=1 Ack=1715198 Win=0 TSval=1810447584 TSecr=1810447584	
66	16. 8566597338	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 8888 - 51988 [ACK] Seq=1 Ack=1702551 Win=0 TSval=1810447584 TSecr=1810447584	
67	16. 856739121	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=1766934 Ack=1 Win=0 TSval=1810447584 Len=65483 TSecr=1810447584	
68	16. 856739121	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1766934 Ack=1 Win=0 TSval=1810447584 TSecr=1810447584	
69	16. 856742219	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 8888 - 51988 [ACK] Seq=1833517 Ack=1 Win=0 TSval=1810447584 TSecr=1810447584	
70	16. 856760362	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1833517 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
71	16. 856760584	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=2029966 Ack=1 Win=0 TSval=1810447584 TSecr=1810447584	
72	16. 856808011	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 8888 - 51988 [ACK] Seq=1 Ack=2095449 Win=0 TSval=1810447584 TSecr=1810447584	
73	16. 856808011	127.0.0.1	127.0.0.1	TCP	1766934 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=1766934 Ack=1 Win=0 TSval=1810447584 Len=65483 TSecr=1810447584	
74	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2097153 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
75	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2097153 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
76	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2126236 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
77	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=212636 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
78	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
79	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=222819 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
80	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
81	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
82	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
83	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
84	16. 856817646	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=222802 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
85	16. 856817646	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=1 Ack=2233602 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
86	16. 856817781	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2359985 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
87	16. 856817781	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2359985 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
88	16. 856817781	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2424568 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
89	16. 856817781	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2424568 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
90	16. 856817781	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=222819 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
91	16. 856817781	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2496051 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
92	16. 856817781	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2555334 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
93	16. 856817781	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2555334 Ack=1 Win=0 TSval=1810447584 Len=65536 Len=65483 TSecr=1810447584	
Frame 9: 74 bytes wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0						
Ethernet II, Src: Ralink RT2571 [00:0c:29:b4:4d:01], Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)						
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1						
Transmission Control Protocol, Src Port: 51988, Dst Port: 8888, Seq: 0, Len: 0						

לבסוף, ניתן לראות שנשלחות בקשות FIN שמעדינות על סיגרת הקשר מ2 הצדדים.

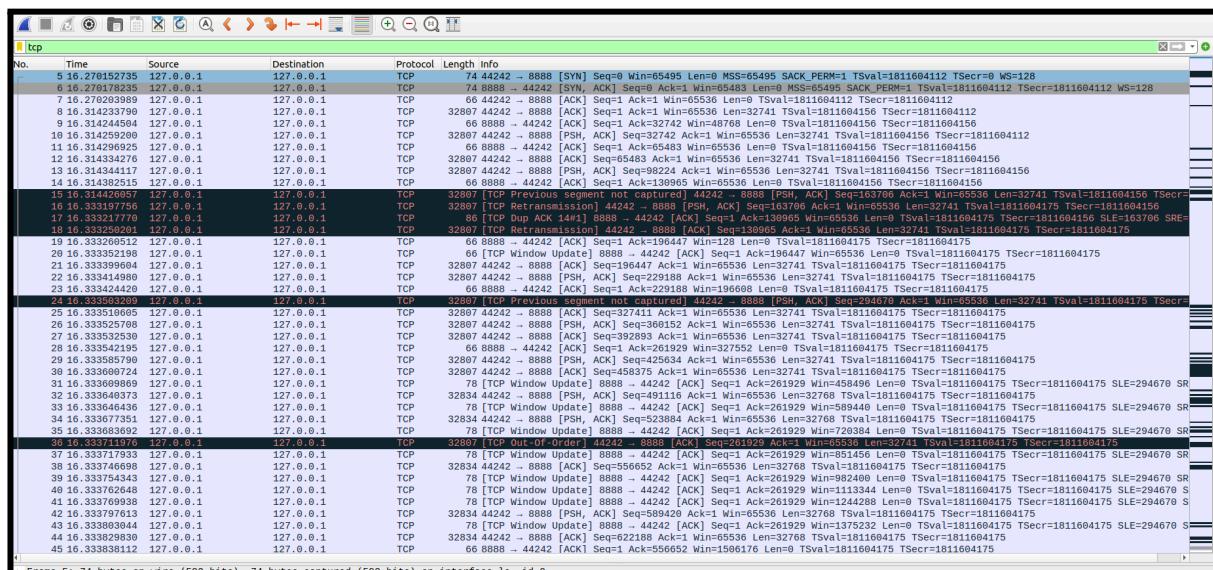
No.	Time	Source	Destination	Protocol	Length	Info
219	21. 620891612	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=312576 Len=0 TSval=1810452347 TSecr=1810452347	
220	21. 620891612	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9174469 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
221	21. 620891612	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=2239888 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
222	21. 620891612	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9239888 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
223	21. 620891612	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=9395371 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
224	21. 620891612	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9395371 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
225	21. 620891612	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=9395371 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
226	21. 620891612	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9395371 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
227	21. 620891612	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=9395371 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
228	21. 620891612	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9436337 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
229	21. 620451181	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=9667399 Ack=1 Win=0 TSval=1810452347 TSecr=1810452347	
230	21. 620451181	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=9561820 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347 SRE=9632786	
231	21. 620451181	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9632786 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
232	21. 620451181	127.0.0.1	127.0.0.1	TCP	73 [TCP Dup ACK 2398.0] 51988 - 8888 [ACK] Seq=9632786 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347 SLE=95673	
233	21. 620451181	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9632786 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
234	21. 620518028	127.0.0.1	127.0.0.1	TCP	73 [TCP Dup ACK 2398.1] 51988 - 8888 [ACK] Seq=9632786 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
235	21. 620518028	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9763752 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
236	21. 620518028	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=9829235 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
237	21. 620606521	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [PSH, ACK] Seq=9960240 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
238	21. 620606521	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [PSH, ACK] Seq=9960240 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
239	21. 620606521	127.0.0.1	127.0.0.1	TCP	73 [TCP Dup ACK 2098.0] 51988 - 8888 [PSH, ACK] Seq=9960240 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
240	21. 620746074	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=10330999 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
241	21. 620753483	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=10418582 Ack=1 Win=0 TSval=1810452347 Len=65536 Len=65483 TSecr=1810452347	
242	21. 620753483	127.0.0.1	127.0.0.1	TCP	65549 [TCP Past Retransmission] 51988 - 8888 [ACK] Seq=95016280 Ack=1 Win=0 TSval=1810452348 TSecr=1810452348	
243	21. 620796064	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=1 Ack=10484065 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
244	21. 620796064	127.0.0.1	127.0.0.1	TCP	1766934 [TCP Previous segment not captured] 51988 - 8888 [ACK] Seq=10484065 Ack=1 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
245	21. 620796064	127.0.0.1	127.0.0.1	TCP	65 8888 - 51988 [ACK] Seq=10484065 Ack=1 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
246	21. 620796064	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [PSH, ACK] Seq=10485761 Ack=1 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
247	21. 620796064	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [ACK] Seq=10485761 Ack=1 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
248	21. 620796064	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [FIN, ACK] Seq=1 Ack=10485765 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
249	21. 620796064	127.0.0.1	127.0.0.1	TCP	66 8888 - 51988 [FIN, ACK] Seq=1 Ack=10485765 Ack=2 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
250	22. 646922717	127.0.0.1	127.0.0.1	TCP	65549 51988 - 8888 [ACK] Seq=2 Ack=10485766 Win=0 TSval=1810452348 Len=65536 Len=65483 TSecr=1810452348	
Frame 9: 94 bytes wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0						
Ethernet II, Src: roni@roni-ThinkPad-X1-Carbon-7th:~ (00:0c:29:b4:4d:01), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)						
Internet Protocol Version 4, Src: 127.0.0.1						

```

roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo cubic
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 83.230 ms.
*Bandwidth: 24.030 MB/s.
*****
#2 run statistics:
*Time: 2371.626 ms.
*Bandwidth: 0.843 MB/s.
*****
#3 run statistics:
*Time: 567.704 ms.
*Bandwidth: 3.523 MB/s.
*****
#4 run statistics:
*Time: 853.225 ms.
*Bandwidth: 2.344 MB/s.
*****
#5 run statistics:
*Time: 335.465 ms.
*Bandwidth: 5.962 MB/s.
*****
Total statistics:
* The AVG time of sending was: 805.119 ms.
* The AVG bandwidth was: 2.070 MB/s.
*****
->the connection closed .program finished-
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 805.119
רוחב פס ממוצע: MB\ S 2.070

TCP_cubic_5%: מצורפת הקלטה: WIRESHARK
בתמונה זו ניתן לראות את הנתונים שהכניםנו- הפורט הוא 8888, והו IP 127.0.0.1.
ניתן לראות את תהליך לחיצת הידיים ב 3 השורות הראשונות.
כשאר השולח שולח SYN , המתקבל מחריז לו SYN ACK והשלוח שולח על זאת ACK .
בהמשך מועברות פקוטות :



בתמונה הבאה ניתן לראות שבמהלך הדרך נתקלנו ב"שורות שחורות" רבות יותר מאשר היו ב% אובדן פקודות (nicer בהקלטה). כיוון שהגדרכנו מצב זה של 5% אובדן פקודות, אפשר בקלות שזה היונה קושי ללקוד אותן בWIRESHARK וכן ישים מספר לא מועט של פקודות שנכשלו בעברה.

No.	Time	Source	Destination	Protocol	Length	Info
188	19. 273693079	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=51765590 Ack=1 Win=65536 Len=65483 Tsvl=1811607115 Tscr=1811607115
189	19. 273688889	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=51765590 Win=3079424 Len=0 Tsvl=1811607115 Tscr=1811607115	
190	19. 273703254	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=Ack=520233 Win=3079424 Len=0 Tsvl=1811607115 Tscr=1811607115	
191	19. 273836258	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 44242 - 8888 [ACK] Seq=507510 Ack=1 Win=65536 Len=65483 Tsvl=1811607115 Tscr=1811607115
192	19. 273836259	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 8888 - 44242 [ACK] Seq=1 Ack=5242833 Win=3112576 Len=0 Tsvl=1811607115 Tscr=1811607115
193	19. 273836259	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=1 Ack=5242833 Win=3112576 Len=0 Tsvl=1811607115 Tscr=1811607115
194	19. 273836259	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 44242 - 8888 [ACK] Seq=507510 Ack=1 Win=65536 Len=65483 Tsvl=1811607115 Tscr=1811607115
195	19. 274808395	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=53438482 Ack=1 Win=65536 Len=65483 Tsvl=1811607115 Tscr=1811607115
196	19. 274808397	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 19@2] 8888 - 44242 [ACK] Seq=1 Ack=5242833 Win=3112576 Len=0 Tsvl=1811607115 Tscr=1811607115
197	19. 274808601	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=5003965 Ack=1 Win=65483 Tsvl=1811607115 Tscr=1811607115
198	19. 274808601	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 19@3] 8888 - 44242 [ACK] Seq=1 Ack=5242833 Win=3112576 Len=0 Tsvl=1811607115 Tscr=1811607115
199	19. 274808601	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 44242 - 8888 [ACK] Seq=5003965 Ack=1 Win=65483 Tsvl=1811607115 Tscr=1811607115
200	19. 274808601	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=5242833 Win=3112576 Len=0 Tsvl=1811607115 Tscr=1811607115	
201	19. 274733397	127.0.0.1	127.0.0.1	TCP	78	[TCP ACKed unseen segment] 8888 - 44242 [ACK] Seq=1 Ack=5313389 Win=3846272 Len=0 Tsvl=1811607116 Tscr=1811607116
202	19. 293108134	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 8888 - 44242 [ACK] Seq=1 Ack=5313389 Win=3846272 Len=0 Tsvl=1811607116 Tscr=1811607116
203	19. 293108134	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 8888 - 44242 [ACK] Seq=1 Ack=5291457 Win=3111296 Len=0 Tsvl=1811607135 Tscr=1811607135
204	19. 837856499	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 8888 - 44242 [ACK] Seq=1 Ack=6356949 Win=3112576 Len=0 Tsvl=1811607678 Tscr=1811607678
205	19. 837120409	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 8888 - 44242 [ACK] Seq=1 Ack=6356949 Win=3112576 Len=0 Tsvl=1811607678 Tscr=1811607678
206	19. 837247149	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=6422423 Win=3112576 Len=0 Tsvl=1811607679 Tscr=1811607679	
208	19. 837806695	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=6422423 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679
209	19. 837806695	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=6422423 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679	
210	19. 837889570	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=6487998 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679
211	19. 837945237	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=6553389 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679
212	19. 837945237	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] 8888 - 44242 [ACK] Seq=1 Ack=6553389 Win=3970242 Len=0 Tsvl=1811607679 Tscr=1811607679
213	19. 837952950	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=6618872 Win=39394916 Len=0 Tsvl=1811607679 Tscr=1811607679	
214	19. 838021543	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 8888 - 44242 [ACK] Seq=6843595 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679
215	19. 838033515	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 8888 - 44242 [ACK] Seq=1 Ack=6618872 Win=3112576 Len=0 Tsvl=1811607679 Tscr=1811607679
216	19. 838033515	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=67498383 Ack=1 Win=65536 Len=65483 Tsvl=1811607679 Tscr=1811607679
217	19. 838108085	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 21@3] 8888 - 44242 [ACK] Seq=1 Ack=6618872 Win=3112576 Len=0 Tsvl=1811607688 Tscr=1811607688
218	19. 838108085	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=68151553 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
219	19. 838251391	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 21@4] 8888 - 44242 [ACK] Seq=68151553 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
220	19. 838251391	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 44242 - 8888 [ACK] Seq=6943287 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
221	19. 838251394	127.0.0.1	127.0.0.1	TCP	86	[TCP Dup ACK 21@3] 8888 - 44242 [ACK] Seq=1 Ack=6618872 Win=3112576 Len=0 Tsvl=1811607688 Tscr=1811607688
222	19. 838383020	127.0.0.1	127.0.0.1	TCP	65549	[TCP Fast Retransmission] 44242 - 8888 [ACK] Seq=6618872 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
223	19. 838383081	127.0.0.1	127.0.0.1	TCP	78	8888 - 44242 [ACK] Seq=6980804 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 SLE=6942087 SRE=7811770
224	19. 838412972	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 44242 - 8888 [ACK] Seq=6888984 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
225	19. 838412972	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=6988984 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688	
226	19. 838412973	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=7011770 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
227	19. 838567914	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=7077253 Ack=1 Win=65536 Len=65483 Tsvl=1811607688 Tscr=1811607688
228	19. 838587993	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=7077253 Len=0 Tsvl=1811607688 Tscr=1811607688	

ויום הקשר בבקשות FIN הנשלחות מ2 הצדדים:

No.	Time	Source	Destination	Protocol	Length	Info
275	20. 479784892	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=8716024 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
276	20. 479796037	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=8716024 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321	
277	20. 479808653	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=8715087 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
278	20. 479808653	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=8715087 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321	
279	20. 479808653	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=8846989 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
280	20. 479808653	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=8912473 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321	
281	20. 480912818	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=8912473 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
282	20. 480928106	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=8977956 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321	
283	20. 480984793	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=89844549 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
284	20. 480984562	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=89844549 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321	
285	20. 480984562	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=89844549 Ack=1 Win=65536 Len=65483 Tsvl=1811608321 Tscr=1811608321
286	20. 480916745	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=2108922 Win=3112576 Len=0 Tsvl=1811608322 Tscr=1811608322	
287	20. 480923888	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9108922 Ack=1 Win=65536 Len=65483 Tsvl=1811608322 Tscr=1811608322
288	20. 4809241755	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=914495 Win=3979424 Len=0 Tsvl=1811608322 Tscr=1811608322	
289	20. 480934764	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9174405 Ack=1 Win=65536 Len=65483 Tsvl=1811608322 Tscr=1811608322
290	20. 480934773	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9239860 Ack=1 Win=3112576 Len=0 Tsvl=1811608322 Tscr=1811608322	
291	20. 480934773	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9239860 Ack=1 Win=3112576 Len=0 Tsvl=1811608322 Tscr=1811608322
292	20. 480936556	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9305371 Ack=1 Win=3112576 Len=0 Tsvl=1811608322 Tscr=1811608322	
293	20. 480944556	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9305371 Ack=1 Win=65536 Len=65483 Tsvl=1811608322 Tscr=1811608322
294	20. 480951688	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9370854 Ack=1 Win=65536 Len=65483 Tsvl=1811608322 Tscr=1811608322
295	20. 480951483	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9370854 Ack=1 Win=65536 Len=65483 Tsvl=1811608322 Tscr=1811608322	
296	20. 480951483	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=9430337 Win=3112576 Len=0 Tsvl=1811608322 Tscr=1811608322	
297	20. 481235801	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 44242 - 8888 [ACK] Seq=9053694 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323
298	20. 481361459	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9068269 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323	
299	20. 481361459	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9698269 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323
300	20. 481421625	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9763752 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323
302	20. 481433162	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=9829235 Win=3112576 Len=0 Tsvl=1811608323 Tscr=1811608323	
304	20. 481433668	127.0.0.1	127.0.0.1	TCP	65549	44242 - 8888 [ACK] Seq=9829235 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323
305	20. 481433668	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9829235 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323	
306	20. 481575785	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=9962001 Win=3112576 Len=0 Tsvl=1811608323 Tscr=1811608323	
307	20. 481575785	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=1 Ack=10485826 Ack=1 Win=65536 Len=65483 Tsvl=1811608323 Tscr=1811608323	
309	20. 481575785	127.0.0.1	127.0.0.1	TCP	66 8888 - 44242 [ACK] Seq=10485826 Ack=1 Win=65536 Len=65483 Tsvl=181	

```

roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./TCP_Receiver
-p 8888 -algo cubic
The Arguments received successfully.port number is: 8888
The receiver's socket was created!
receiver binds successfully!
Sender connected. beginning to receive file....
*****
#1 run statistics:
*Time: 256.616 ms.
*Bandwidth: 7.794 MB/s.
*****
#2 run statistics:
*Time: 1606.014 ms.
*Bandwidth: 1.245 MB/s.
*****
#3 run statistics:
*Time: 643.675 ms.
*Bandwidth: 3.107 MB/s.
*****
#4 run statistics:
*Time: 606.373 ms.
*Bandwidth: 3.298 MB/s.
*****
#5 run statistics:
*Time: 973.627 ms.
*Bandwidth: 2.054 MB/s.
*****
Total statistics:
* The AVG time of sending was: 709.784 ms.
* The AVG bandwidth was: 2.348 MB/s.
*****
->the connection closed .program finished<
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

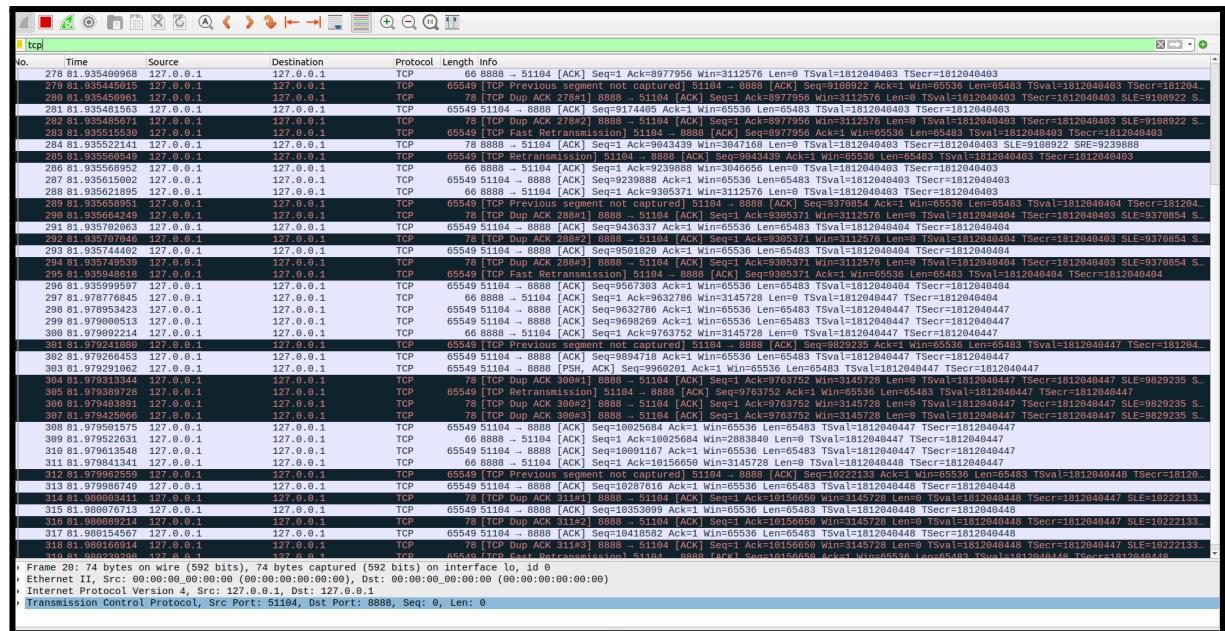
זמן ריצה ממוצע: MS 709.784

רוחב פס ממוצע: MB\S 2.348

TCP_cubic_10%: מצורפת הקלטה: WIRESHARK
 צילום מסך מתוך ה-WIRESHARK בתרמונה זו ניתן לראות את הנתונים שהכנסנו- הפורט הוא 8888, והא-IP 127.0.0.1, והו-השורת הראשונית.
 ניתן לראות את תהליך לחיצת הידיים ב 3 השורות הראשונות.
 כאשר השולח שולח SYN , המתקבל מ-ACK , השולח שולח על זאת ACK .
 בהמשך מועברות פקודות :

No.	Time	Source	Destination	Protocol	Length	Info
28	78.35978181	127.0.0.1	127.0.0.1	TCP	74	51164 -> 8888 [SYN] Seq=0 Win=65456 SACK PERM=1 Tsvl=1812036572 Tscr=0 WS=128
29	78.35980024	127.0.0.1	127.0.0.1	TCP	66	51164 -> 8888 [PSH, ACK] Seq=1 Win=65456 Len=9 Tsvl=1812036572 Tscr=1812036572 WS=128
22	78.183997361	127.0.0.1	127.0.0.1	TCP	66	51164 -> 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=9 Tsvl=18120365692 Tscr=1812036572 WS=128
23	78.134420626	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036572
24	78.134430223	127.0.0.1	127.0.0.1	TCP	66	8888 -> 51164 [ACK] Seq=1 Ack=32742 Tsvl=1812036602 Tscr=18120365692
25	78.134445048	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=32742 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036572
26	78.134451459	127.0.0.1	127.0.0.1	TCP	66	8888 -> 51164 [ACK] Seq=32742 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=18120365692
27	78.134463997	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=32742 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=18120365692
28	78.134529509	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=38224 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
29	78.134538048	127.0.0.1	127.0.0.1	TCP	66	8888 -> 51164 [ACK] Seq=1 Ack=38065 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
30	78.134619998	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=139065 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
31	78.134630077	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=1637069 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
32	78.154674694	127.0.0.1	127.0.0.1	TCP	32897	[TCP Retransmission] 51164 -> 8888 [PSH, ACK] Seq=1637069 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
33	78.154689726	127.0.0.1	127.0.0.1	TCP	32897	[TCP Retransmission] 51164 -> 8888 [PSH, ACK] Seq=1637069 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
34	78.154690093	127.0.0.1	127.0.0.1	TCP	76	8888 -> 51164 [ACK] Seq=1637069 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602 SRE=103706
35	78.359490888	127.0.0.1	127.0.0.1	TCP	32897	[TCP Previous segment not captured] 51164 -> 8888 [PSH, ACK] Seq=729108 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
36	78.3596747478	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=2619249 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
37	78.359787896	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=294670 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
38	78.359995841	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=327411 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
39	78.360000001	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 51164 -> 8888 [ACK] Seq=1 Ack=165447 Win=327424 Len=9 Tsvl=1812036602 Tscr=1812036602 SRE=-
40	78.360004548	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=165447 Win=327424 Len=9 Tsvl=1812036602 Tscr=1812036602 SRE=-
41	78.359994947	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=166447 Win=89440 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=229188 SRE=-
42	78.359167994	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=166447 Win=89440 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=229188 SRE=-
43	78.359137224	127.0.0.1	127.0.0.1	TCP	32897	[TCP Previous segment not captured] 51164 -> 8888 [ACK] Seq=392893 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
44	78.359155315	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=425634 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
45	78.359171207	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=458374 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
46	78.359180319	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=491111 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
47	78.359202616	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=523884 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
48	78.359213705	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=5566522 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
49	78.359224243	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=5894249 Ack=1 Win=65536 Len=32741 Tsvl=1812036602 Tscr=1812036602
50	78.359229107	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=165447 Win=851328 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=392893 SRE=-
51	78.359232381	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=165447 Win=932722 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=392893 SRE=-
52	78.359235643	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=165447 Win=1019872 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=392893 SRE=-
53	78.359271867	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=166447 Win=2244288 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=32993 SRE=-
54	78.359288336	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=166447 Win=3752328 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=32993 SRE=-
55	78.359307006	127.0.0.1	127.0.0.1	TCP	86	[TCP Window Update] 8888 -> 51164 [ACK] Seq=1 Ack=166447 Win=637121 Len=9 Tsvl=1812036602 Tscr=1812036602 SLE=32993 SRE=-
56	78.359373486	127.0.0.1	127.0.0.1	TCP	32897	[TCP Retransmission] 51164 -> 8888 [PSH, ACK] Seq=369159 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
57	78.359391801	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=622189 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
58	78.359404380	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=656989 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
59	78.359422834	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [ACK] Seq=687724 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
60	78.359435539	127.0.0.1	127.0.0.1	TCP	32897	[TCP OUT-OF-ORDER] 51164 -> 8888 [TICK] Seq=196447 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
61	78.359453744	127.0.0.1	127.0.0.1	TCP	32897	51164 -> 8888 [PSH, ACK] Seq=7764192 Ack=1 Win=65536 Len=32741 Tsvl=1812036827 Tscr=1812036827
Frame 20: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface lo, id 0 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00) Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 Transmission Control Protocol, Src Port: 51104, Dst Port: 8888, Seq: 0, Len: 0						

בתמונה הבאה ניתן לראות שבמהלך הדרך נתקלנו ב"שורות שחורות" רבות וברות. הרבה יותר מאשר הינו 5% אובדן פקודות (nicer בהקלטה). כיוון שהגדרנו מצב זה של 10% אובדן פקודות, אפשר בנסיבות שזה היהו קושי למכוד אותן בWIRESHARK וכן ישנו מספר לא מועט של פקודות שנכשלו בעברה.



טבלה סיכום סטטיסטיות עברו אובדן פקודות בפרוטוקול TCP עם אלגוריתם cubic

BANDWIDTH\MB.S	TIME\MS	%LOSS PACKETS
1.710	974.895	0%
2.205	755.984	2%
2.070	805.119	5%
2.348	709.784	10%

RENO

CUBIC

BANDWIDTH\MB.S	TIME\MS	%LOSS PACKETS
1.403	1187.667	0%
1.322	1251.549	2%
1.037	1607.144	5%
0.655	2544.440	10%

BANDWIDTH\MB.S	TIME\MS	%LOSS PACKETS
1.710	974.895	0%
2.205	755.984	2%
2.070	805.119	5%
2.348	709.784	10%

נתח נטוניים:

- בRENO ניתן לראות שהמהירות הממוצעת עולה ככל שהתקדמנו בעליית אחוז איבוד הפקטות.
- בRENO ניתן לראות שרוחב הפס ירד ככל שהתקדמנו בעליית אחוז איבוד הפקטות.
- בCUIC ניתן לראות שאין שינוי עקבי במהירות הממוצעת ככל שהתקדמנו בעליית אחוז איבוד הפקטות.
- בCUIC ניתן לראות שאין שינוי עקבי ברוחב הפס הממוצע ככל שהתקדmeno בעליית אחוז איבוד הפקטות.
- ניתן להסיק כי באופן כללי שימוש באלגוריתם CUIC לברכית גודש מSIG תוצאות טובות יותר. גם ללא אובדן פקטות וגם כשןפעיל אובדן פקטות.
- ניתן להסיק שכמובן באיבוד פקטות באחוזים יחסית גבוהים, היעילות של אלגוריתם המטפל בברכית גודש תלולה ביכולתו לשחרר במהירות את קצב השילחה לאחר אובדן מידע.

בצלום מסך הבא ניתן לראות את ביצוע לחיצת היד המשולשת שמיימשנו בפונקציית `.connect`.

13	15.682473224	127.0.0.1	127.0.0.1	UDP	58	35552 → 8888	Len=16
14	15.682536145	127.0.0.1	127.0.0.1	UDP	58	8888 → 35552	Len=16
15	15.682579185	127.0.0.1	127.0.0.1	UDP	58	35552 → 8888	Len=16
16	15.725464856	127.0.0.1	127.0.0.1	UDP	65458	35552 → 8888	Len=65416

השלוח שלוח SYN ל receiver שזו תהיה בגודל 16. (LEN=16)
ה receiver שלוח SYN ACK בחזרה שזו תהיה בגודל 16.
ה sender שלוח ACK שזו תהיה בגודל 16.
ה פרוטוקול הוא UDP.

ה HEADER של RUDP הוא בגודל 16 בתים ולכן גודל הפקטוטה.
ניתן לראות את הפורט שבחרנו (8888) ואת כתובת ה IP (127.0.0.1)

▶ Frame 14139: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface lo, id 0x00000000000000000000000000000000
▼ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Destination: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ User Datagram Protocol, Src Port: 34077, Dst Port: 8888
▶ Data (16 bytes)

בנוסף בתור HEADER הגדרנו שדה LENGTH והוא בגודל 2 בתים:

Destination Port: 8888
Length: 24
Checksum: 0xfe2b [unverified]
[Checksum Status: Unverified]
[Stream index: 68]
▶ [Timestamps]
▶ UDP payload (16 bytes)
▼ Data (16 bytes)
Data: 0000000000000000acff530000000000
[Length: 16]
.....
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 2c 55 32 40 00 40 11 e7 8c 7f 00 00 01 7f 00 ..,U2@.@.....
0020 00 01 85 1d 22 b8 00 18 fe 2b 00 00 00 00 00 00"
0030 00 00 ac ff 53 00 00 00 00 00 00 00 00 00 00 S.....

הגדרנו שדה CHECKSUM והוא בגודל 2 בתים:

Destination Port: 8888
Length: 24
Checksum: 0xfe2b [unverified]
[Checksum Status: Unverified]
[Stream index: 68]
▶ [Timestamps]
▶ UDP payload (16 bytes)
▼ Data (16 bytes)
Data: 0000000000000000acff530000000000
[Length: 16]
.....
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 2c 55 32 40 00 40 11 e7 8c 7f 00 00 01 7f 00 ..,U2@.@.....
0020 00 01 85 1d 22 b8 00 18 fe 2b 00 00 00 00 00 00"
0030 00 00 ac ff 53 00 00 00 00 00 00 00 00 00 00 S.....

בנוסף, על סמך הדגלים שקבענו לכל סוג פקטה ניתן לראות:

עבור NYSE התו 'S' מופעל:

```
▶ Frame 142: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface lo, id 0
  Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  User Datagram Protocol, Src Port: 34077, Dst Port: 8888
  ▶ Data (16 bytes)
    Data: 0000000000000000acff530000000000
    [Length: 16]
```

0000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00	E
0010	00 2c 55 32 40 00 40 11 e7 8c 7f 00 00 00 01 7f 00	, U2@-@
0020	00 01 85 1d 2d b8 00 18 fe 2b 00 00 00 00 00 00 00 00	" +
0030	00 00 ac ff 53 00 00 00 00 00 00	S ..

עבור SYN ACK הטו 'ז' מופעל:

```
▶ Frame 143: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface lo, id 0
  Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  User Datagram Protocol, Src Port: 8888, Dst Port: 34077
  ▶ Data (16 bytes)
    Data: 0000000000000000a6ff590000000000
    [Length: 16]
```

0000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00		E
0010	00 2c 55 33 40 00 00 40 11 e7 8b 7f 00 00 00 01 7f 00	, U3@ @	
0020	00 01 22 b8 85 1d 00 18 fe 2b 00 00 00 00 00 00 00 00	"	+ Y
0030	00 00 a6 ff 59 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.	.

ובור הCK שסוגרת את לחיצת הידיים התו 'L' מופעל:

```
▶ Frame 144: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface lo, id 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ User Datagram Protocol, Src Port: 34077, Dst Port: 8888
▶ Data (16 bytes)
    Data: 0000000000000000b3ff4c0000000000
    [Length: 16]
```

לאחר ייצור הקשר, מתחילה מידע של העברת פרטיות:

13 15.682473224	127.0.0.1	127.0.0.1	UDP	58 35552 → 8888 Len=16
14 15.682536145	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
15 15.682579185	127.0.0.1	127.0.0.1	UDP	58 35552 → 8888 Len=16
16 15.725464856	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
17 15.725646724	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
18 15.725786524	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
19 15.725943583	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
20 15.726089309	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
21 15.726255024	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
22 15.726390618	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
23 15.726576156	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
24 15.726699942	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
25 15.726850954	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
26 15.726969539	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
27 15.727116049	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
28 15.727236481	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
29 15.727383987	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
30 15.727516361	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
31 15.727663381	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
32 15.727776184	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
33 15.727920519	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
34 15.728034187	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
35 15.728177262	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
36 15.728293276	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
37 15.728438597	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
38 15.728559672	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416

מכיוון שהגדכנו ה BUFFER להיות 65400, ניתן לראות שגודל החבילה סה"כ 65416 שזה גודל המידע + גודל HEADER

▼ User Datagram Protocol, Src Port: 35552, Dst Port: 8888
Source Port: 35552
Destination Port: 8888
Length: 65424
Checksum: 0xfdफ [unverified]
[Checksum Status: Unverified]
[Stream index: 6]
[timestamps]
UDP payload (65416 bytes)
▶ Data (65416 bytes)
0020 00 01 8a e0 22 b8 ff 90 fd a4 78 ff 00 00 00 00 . . . " x
0030 00 00 da 5b 44 00 00 00 00 00 02 df e6 5e 46 49 . . . [D ^FI
0040 77 53 81 bf 6e c7 37 28 83 f9 1a 9b e6 a3 8b 87 wS . n . 7(.
0050 0b c7 78 df 89 02 09 e8 d4 0b c8 ba 6a 0e 04 e1 . . x j
0060 61 85 a0 cf 4c d7 f8 d0 d1 12 6b b7 b5 f6 3e c0 a . . L k . . . > .
0070 be b6 a0 47 b9 a9 30 8d b5 f8 48 1f 06 4c 00 67 . . G . 0 . . H . . . g .
0080 d1 a0 36 1d 78 2e ed 49 40 59 00 f6 4f 3e b6 0d . . 6 . x . I @Y . 0 > . .
0090 f5 56 55 ae 00 85 3b b5 7d 83 d4 83 cf d4 ea a0 . . VU . . ; . . }
00a0 74 20 be ec 4f ab 35 8f 04 35 85 54 74 3c 61 69 t . . 0 . 5 . . 5 . Tt <ai
00b0 92 b6 17 92 3b 52 47 b8 d6 1b 3b a5 ef 25 46 64 . . . ; RG . . . ; . %Fd
00c0 10 01 50 05 5 00 01 b1 b1 b1 b1 b1 b1 b1 b1 b1

לבסוף, כשהקשר נסגר והתוכנית נגמרה:

341 19.742703802	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
342 19.742905472	127.0.0.1	127.0.0.1	UDP	65458 35552 → 8888 Len=65416
343 19.743150194	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
344 19.743207627	127.0.0.1	127.0.0.1	UDP	4410 35552 → 8888 Len=4368
345 19.743271673	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16
346 20.195678500	127.0.0.1	127.0.0.1	UDP	58 35552 → 8888 Len=16
347 20.195770805	127.0.0.1	127.0.0.1	UDP	58 8888 → 35552 Len=16

dag F' (לאחר שעולה A עבור ACK) עולה ומסמל FINISH שכאן נסגר הקשר:

[Checksum Status: Unverified]
[Stream index: 6]
↳ [Timestamps]
UDP payload (16 bytes)
↳ Data (16 bytes)
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 ..E..
0010 00 2c f7 7d 40 00 40 11 45 41 7f 00 00 01 7f 00 ..}@@ EA..
0020 00 01 8a e0 22 b8 00 18 fe 2b 00 00 00 00 00 00 .."....+....
0030 00 00 b9 ff 46 00 00 00 00 00 ..F....

הרצה עם אובדן פקודות:

- הרצתה עברו 0% אובדן פקודות:

(צילומי המסר עברו אתחול אוחזי האובדן פקודות זהה לחلك של TCP ולכן לא סופק אותם גם כאן)
חלון הTerminal, במציאות פקודת הרצתה ./RUDP_Receiver -p 8888/. receivern

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./RUDP_Receiver -p 8888
Arguments received successfully. port number is 8888
socket was opened!
receiver binds successfully!
SYN packet received from the sender!
SYN-ACK sent!
ack for syn-ack received
The 3 handshake was completed, waiting for data!
FIN packet received from the sender!
Sending Ack to fin!
*****
#1 run statistics:
*Time: 2234.922 ms.
*Bandwidth: 0.939 MB/s.
*****
#2 run statistics:
*Time: 529.081 ms.
*Bandwidth: 3.965 MB/s.
*****
#3 run statistics:
*Time: 631.602 ms.
*Bandwidth: 3.321 MB/s.
*****
#4 run statistics:
*Time: 613.250 ms.
*Bandwidth: 3.421 MB/s.
*****
#5 run statistics:
*Time: 452.444 ms.
*Bandwidth: 4.635 MB/s.
*****
Total statistics:
* The AVG time of sending was: 892.260 ms.
* The AVG bandwidth was: 2.242 MB/s.
Connection closed!
```

זמן ריצה ממוצע: MS 892.260
רוחב פס ממוצע: MB\S 2.242

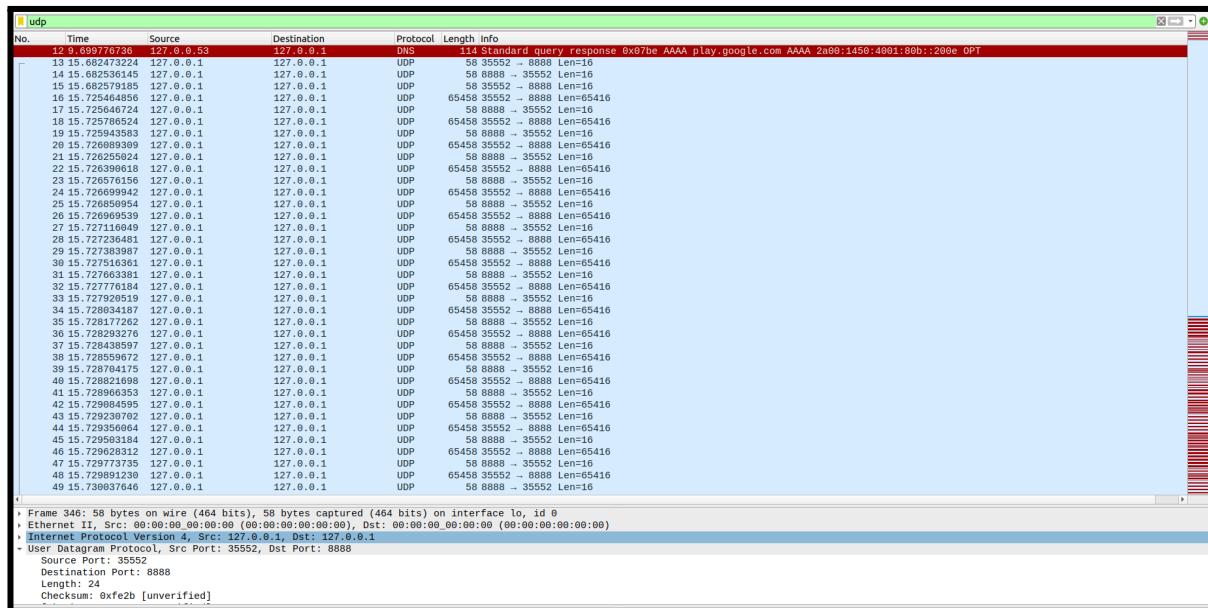
חלון ה :sender

```

roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./RUDP_Sender
-ip 127.0.0.1 -p 8888
The arguments received!
RUDP socket created successfully!
SYN messege sent, wating for SYN-ACK
The SYN-ACK answer was received!
The ACK was sent!
Sent 2097680 bytes to the server!
Data sent successfully!
Send more data? (Y/N): y
Sent 2097680 bytes to the server!
Data sent successfully!
Send more data? (Y/N): y
Sent 2097680 bytes to the server!
Data sent successfully!
Send more data? (Y/N): y
Sent 2097680 bytes to the server!
Data sent successfully!
Send more data? (Y/N): y
Sent 2097680 bytes to the server!
Data sent successfully!
Send more data? (Y/N): n
FIN packet sent!
ack for FIN received!
Connection closed!

```

:RUDP_0%. הפקלה שומרה בתקיית REC תחת השם
נישוח ממופרט בתחילת חלק 2.



• הריצות עברו 2% אובדן פקודות:

חולון הינו receiver , באמצעות פקודת הריצה ./RUDP_Receiver -p 8888/.

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./RUDP_Receiver -p 8888
Arguments received successfully. port number is 8888
socket was opened!
receiver binds successfully!
SYN packet received from the sender!
SYN-ACK sent!
ack for syn-ack received
The 3 handshake was completed, waiting for data!
FIN packet received from the sender!
Sending Ack to fin!
*****
#1 run statistics:
*Time: 8499.551 ms.
*Bandwidth: 0.247 MB/s.
*****
#2 run statistics:
*Time: 6197.631 ms.
*Bandwidth: 0.348 MB/s.
*****
#3 run statistics:
*Time: 10251.950 ms.
*Bandwidth: 0.205 MB/s.
*****
#4 run statistics:
*Time: 6513.027 ms.
*Bandwidth: 0.322 MB/s.
*****
#5 run statistics:
*Time: 1719.616 ms.
*Bandwidth: 1.220 MB/s.
*****
Total statistics:
* The AVG time of sending was: 6636.355 ms.
* The AVG bandwidth was: 0.305 MB/s.
Connection closed!
```

זמן ריצה ממוצע: MS 6636.355

רוחב פס ממוצע: MB\S 0.305

RUDP_2% REC שמורה בתקיית WIRESHARK תחת השם הקלטת ה

```
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 5%
./RUDP_Receiver -p 8888/
הלוון הינה מקבלת הרצתה. receiver
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./RUDP_Receiver
-p 8888
Arguments received successfully. port number is 8888
socket was opened!
receiver binds successfully!
SYN packet received from the sender!
SYN-ACK sent!
ack for syn-ack received
The 3 handshake was completed, waiting for data!
FIN packet received from the sender!
Sending Ack to fin!
*****
#1 run statistics:
*Time: 4346.541 ms.
*Bandwidth: 0.483 MB/s.
*****
#2 run statistics:
*Time: 26049.037 ms.
*Bandwidth: 0.081 MB/s.
*****
#3 run statistics:
*Time: 14156.087 ms.
*Bandwidth: 0.148 MB/s.
*****
#4 run statistics:
*Time: 29744.493 ms.
*Bandwidth: 0.071 MB/s.
*****
#5 run statistics:
*Time: 2698.099 ms.
*Bandwidth: 0.777 MB/s.
*****
Total statistics:
* The AVG time of sending was: 15398.851 ms.
* The AVG bandwidth was: 0.136 MB/s.
Connection closed!
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$
```

זמן ריצה ממוצע: MS 15398.851

רוחב פס ממוצע: MB\S 0.136

RUDP_5% REC תחת השם Wireshark התקין

No.	Time	Source	Destination	Protocol	Length	Info
10	15.211751681	127.0.0.53	127.0.0.1	DNS	118	Standard query response 0xe202 A signaler-pa.clients6.google.com A 172.217.16.202 OPT
15	211751680	127.0.0.1	127.0.0.53	DNS	102	Standard query response 0x951b AAAA signaler-pa.clients6.google.com AAAA 2a00:1450:4028:803::20a OPT
12	15.527972629	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
13	15.527972629	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
14	15.528978240	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
15	15.528215697	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
16	15.570891418	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
18	15.5710180367	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
19	15.571661765	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
20	19.576168698	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
21	19.576858567	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
22	19.584507222	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
23	19.585767875	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
24	19.585767875	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
25	19.586453461	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
26	19.587124743	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
27	19.587782133	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
28	19.588309651	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
29	19.589499126	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
30	19.589499126	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
31	19.590318673	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
32	19.590851369	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
33	19.591515397	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
34	19.592173936	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
35	19.593060259	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
36	19.593360259	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
37	19.594048418	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
38	19.594799081	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
39	19.595365265	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
40	19.595887259	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
41	19.596484466	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
42	19.597248486	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
43	19.597857450	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
44	19.598409937	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
45	19.599142941	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16
46	19.599796695	127.0.0.1	127.0.0.1	UDP	65458	36369 - 8888 Len=65416
47	19.600455534	127.0.0.1	127.0.0.1	UDP	58	8888 - 36369 Len=16

ניתן לראות בתמונה לעיל מצד ימין את ריכוז הפקודות האדומות שנוצרו עקב בחירה באחיזה איבוד 5%.

• הרצאות עברו 10% אובדן פקודות

```
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc del dev lo root netem
[sudo] password for roni:
roni@roni-ThinkPad-X1-Carbon-7th:~$ sudo tc qdisc add dev lo root netem loss 10%
roni@roni-ThinkPad-X1-Carbon-7th:~$
```

חלון ./RUDP_Receiver -p 8888/. receiversה באמצעות פקודת הריצה

```
roni@roni-ThinkPad-X1-Carbon-7th:~/Desktop/reshatot_projects/ex3$ ./RUDP_Receiver -p 8888
Arguments received successfully. port number is 8888
socket was opened!
receiver binds successfully!
SYN packet received from the sender!
SYN-ACK sent!
ack for syn-ack received
The 3 handshake was completed, waiting for data!
FIN packet received from the sender!
Sending Ack to fin!
*****
#1 run statistics:
*Time: 27252.128 ms.
*Bandwidth: 0.077 MB/s.
*****
#2 run statistics:
*Time: 76739.382 ms.
*Bandwidth: 0.027 MB/s.
*****
#3 run statistics:
*Time: 108568.675 ms.
*Bandwidth: 0.019 MB/s.
*****
#4 run statistics:
*Time: 57145.213 ms.
*Bandwidth: 0.037 MB/s.
*****
#5 run statistics:
*Time: 24577.371 ms.
*Bandwidth: 0.085 MB/s.
*****
Total statistics:
* The AVG time of sending was: 58856.554 ms.
* The AVG bandwidth was: 0.038 MB/s.
Connection closed!
```

זמן ריצה ממוצע: MS 58856.554

רוחב פס ממוצע: MB/S 0.038

הקלטת ה RUDP_10% REC שמורה בתקיית Wireshark

No.	udp	Source	Destination	Protocol	Length	Info
1	udpcap	00000 127.0.0.1	127.0.0.53	DNS	86	Standard query 0x7458 A play.google.com OPT
2	udpcap	17697 127.0.0.1	127.0.0.1	DNS	102	Standard query response 0x7458 A play.google.com A 172.217.22.14 OPT
3	www.84795	127.0.0.1	127.0.0.53	DNS	40	Standard query 0x9564 AAAA play.google.com OPT
4	0.000406150	127.0.0.53	127.0.0.1	DNS	114	Standard query response 0x9564 AAAA play.google.com AAAA 2a00:1450:4001:830::200e OPT
5	11.897321493	127.0.0.1	127.0.0.53	DNS	86	Standard query 0x5a18 A play.google.com OPT
6	11.897439988	127.0.0.53	127.0.0.1	DNS	102	Standard query response 0x5a18 A play.google.com A 172.217.22.14 OPT
7	www.84795	127.0.0.1	127.0.0.53	DNS	40	Standard query 0x9564 AAAA play.google.com OPT
8	11.89739377	127.0.0.53	127.0.0.1	DNS	114	Standard query response 0x48e6 AAAA play.google.com AAAA 2a00:1450:4001:830::200e OPT
9	18.669545638	127.0.0.1	127.0.0.1	UDP	58	55867 - 8888 Len=16
10	18.66971369	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
11	18.669865479	127.0.0.1	127.0.0.1	UDP	58	55867 - 8888 Len=16
12	18.670000000	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
13	18.708753899	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
14	18.709347415	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
15	18.707242174	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
16	18.707489370	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
17	18.707778867	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
18	18.708361197	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
19	18.708361197	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
20	18.708655724	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
21	18.708994948	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
22	22.487875178	127.0.0.1	127.0.0.53	DNS	86	Standard query 0x1cb7 A play.google.com OPT
23	22.488179561	127.0.0.53	127.0.0.1	DNS	102	Standard query response 0x1cb7 A play.google.com A 172.217.22.14 OPT
24	22.488243276	127.0.0.1	127.0.0.53	DNS	86	Standard query 0x64b3 AAAA play.google.com OPT
25	22.488243276	127.0.0.53	127.0.0.1	DNS	102	Standard query response 0x64b3 AAAA play.google.com AAAA 2a00:1450:4001:830::200e OPT
26	22.488243276	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
27	24.838728955	127.0.0.1	127.0.0.53	DNS	86	Standard query 0xcbef A mail.google.com OPT
28	24.838728955	127.0.0.53	127.0.0.1	DNS	86	Standard query 0x681d A mail.google.com OPT
29	24.902928368	127.0.0.53	127.0.0.1	DNS	102	Standard query response 0xcbef A mail.google.com A 172.217.22.37 OPT
30	24.902977095	127.0.0.53	127.0.0.1	DNS	102	Standard query response 0x681d A mail.google.com A 172.217.22.37 OPT
31	24.903077645	127.0.0.1	127.0.0.53	DNS	86	Standard query 0x5ce6 AAAA mail.google.com OPT
32	24.903077645	127.0.0.53	127.0.0.1	DNS	114	Standard query response 0x5ce6 AAAA mail.google.com AAAA 2a00:1450:4028:801::2005 OPT
33	26.734203804	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
34	26.744573575	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
35	26.721871922	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
36	26.722381897	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16
37	26.722664593	127.0.0.1	127.0.0.1	UDP	65458	55867 - 8888 Len=65416
38	26.723073379	127.0.0.1	127.0.0.1	UDP	58	58888 - 55867 Len=16

Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface lo, id 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
User Datagram Protocol, Src Port: 46130, Dst Port: 53
Domain Name System (query)

ניתן לראות בתמונה לעיל מצד ימין את ריכוז הפקטוות האדומות שנוצרו עקב בחירה באחוז איבוד פקטות 10% . הזמן בו ליקח לתוכנית לזרע היה איטי מרגיל באופן חריג. בסכם את זמני הריצה וערכי הרוחב פס הממצאים.

טבלת סיכום סטטיסטיות עבור אובדן פקודות בפרוטוקול RUDP :

BANDWIDTH\MB.S	TIME\MS	%LOSS PACKETS
2.242	892.260	0%
0.305	6636.355	2%
0.136	15398.851	5%
0.038	58856.554	10%

מתוצאות הניסוי על RUDP ניתן להסיק שימוש רגיל של RUDP שלא כולל בתוכו בקרת גודש וכל מיין מנגןונים מוביל לכך ש A. זמן שליחת חבילה גדול באופן קיצוני ככל שמעלים את אחוז האיבוד. B. WIDTH bandwidth קטן באופן משמעותי ככל שמעלים את אחוז האיבוד.

מענה על שאלה [2]

(השאלה- השוו בין הביצועים של UDP RELIABLE לשל TCP, כתבו מי מהם עדיף לשימוש באובדן פקודות גבוההה.)

תשובה:

לפי הtablאות והנתונים המודפסים בניסויים שערכנו אפשר לומר באופן חד וחילק כי ביצוע TCP הינו טובים משל UDP.

מכיוון שTCP משתמש באלגוריתמים האחראים על בקרת הגודש , יש לפרוטוקול זה יכולת לנוהל את גודל החולון באופן גמיש.

כשביצענו אובדן פקודות באחוזים גבוהים- הביצוע של RUDP עם השימוש של STOP&WAIT נראה שהוא שההתמודדות שלו עם המצב לא הייתה אידיאלית, וביצוע TCP גם כאן היה טובים יותר. כמו שציינו בסעיף הקודם, TCP באופן כללי פועל בצורה טובה יותר מ RUDP וראינו זאת בניסויים שביצענו.

אר, יתכן ובמצב של אובדן פקודות גבוההה- TCP לא "יתפרק" עקב האלגוריתמים שלו לבקרת גודש.

ניתן להטמע מנגנונים ובקורות גם לתוכה פרוטוקול UDP על מנת לשפר את הביצועים שלו בתנאים אלה.
אם לא נוסיף אותם, נראה UDP בצורה פשוטה שלו לא יהיה עדיף על TCP.

מענה על שאלה [3]

(השאלה- באיזה תרחישים נעדיף להשתמש ב-UDP ובאיו תרחישים ב-TCP?)

תשובות:

בהתנאי אפליקציה מסוימת, נרצה לדעת איזה פרוטוקול עדיף בשימוש.
על מנת לקבל את ההחלטה נדרש לדעת מה הדרישות מהאפליקציה במגוון קטגוריות כגון הצפנה,
אמינות, תנאים וכו'.

-> נעדיף להשתמש ב-UDP עבור מתן שירות לאפליקציות שמתפקידן ב"זמן אמת".
למדנו בהרצאות ותרגולים למשל שאפליקציית ZOOM משתמשת UDP כיוון שאובדן החבילות
לא תמיד ישפייע באופן משמעותי על איכות השימוש והחויה, יותר חשוב הרציפות. וזה דוגמא טובה
למצבים בהם נעדיף להשתמש בפרוטוקול זה.

השימוש ב-UDP מקנה יתרונות בהירות התגובה והמענה, אך עשוי להוביל לאיבוד חלק של נתונים
ולא תמיכה בתיקונים אוטומטיים של שגיאות תקשורת.

-> נעדיף להשתמש ב-TCP עבור מtan שירות לאפליקציות שדורשות הצפנה מסוימת. מכיוון שהזהו
פרוטוקול אמין הוא יספק לנו הבטחה שהמידע שנביבס נשלח יהיה שלם ובטוח.
נעדיף להשתמש בו במצבים שאובדן פקודות עלול להוביל לצרות רבות ופגיעה במהימנות של
האפליקציה.

למדנו בהרצאות ותרגולים למשל, שימוש באינטרנט (גלאישה) מתבצע באמצעות פרוטוקול TCP
ושהערת קבצים גם כן, כדי להבטיח שנוביל את הקובץ בשלמותו.

מענה על שאלות פתוחות:

שאלת 1

תשובות:

ראשית נסביר מה הוא SSThreshold - הנקודה בה אנחנו עוברים מגידול אקספוננציאלי לגידול
לינארי בברכית עומסים ביחס לחילון CWND הכלול את הפקודות שנשלחו וטרם קיבלנו ACK ואת
הפקודות שמכילות מידע אך טרם נשלחו.
נסביר מה הוא RTT - הזמן שהוקצץ מתחילה שליחת המידע עד לקבלת ACK על המידע שנשלח.
על פי הגדרת השאלה:

- 1) רשות אמינה הינה רשת שבה מעט מאוד חbillות חולכות לאיבוד.
- 2) קשר ארוך הואTCP שיש בו הרבה מידע לשולח.

כעת, מבין המקרים הבאים נבחר את המקרים בהם הגדלת ה-SSThreshold תועיל במידה
המירבית:

1. קשר ארוך על גבי רשות אמינה עם RTT גדול.
2. קשר קצר על גבי רשות לא אמינה עם RTT גדול.
3. קשר ארוך על גבי רשות לא אמינה עם RTT גדול.

4. בקשר קצר על גבי רשות אמינה עם RTT קטן.
5. בקשר ארוך על גבי רשות אמינה עם RTT קטן.
6. בקשר קצר על גבי רשות לא אמינה עם RTT קטן.
7. בקשר ארוך על גבי רשות לא אמינה עם RTT קטן.
8. בקשר קצר על גבי רשות אמינה עם RTT גדול.

בביצוע הגדלת `SSThreshold` אנחנואפשרים דחיה של זמן בקרת העומסים ולכן אנחנו מסתכנים בכך שחבריות ילכו לאיבוד ותהיה העמסת מידע. לכן ראשית נשים לב כי קשר אשר אינו אמין אינו מתאים לתרחיש זה, כמו כן בקשר קצר(קשר ללא הרבה מידע לשילחה) הגדלת `SSThreshold` תביא לידי מצב בו כמות גדולה מיותר המידע תועבר ללא בקרת עומס טוביה. מה שימוש באופציות 1 ו 5, מביניהם אפשרות 1 היא העדיפה שכן RTT גדול יותר מאפשר לנו לשלוח את החבריות בקרה טובה יותר עד ל `SSThreshold` ולנקודות בקרת העומסים. **לסיכום, אופציה 1 היא העדיפה ביותר ואופציה 5 אחרת.**

שאלה 2

נתון בשאלת:

- (1) גודל החלון הינו MSS(גודל הסגמנט המקורי).
- (2) החיבור מתנתך כאשר הוא מגע לSHRESH(Sיום הגידול האקספוננציאלי), מסומנים בקיצור $S^*MSS = SHRESH$.
- (3) גודל החלון בקצת החיבור
- (3) לא נאבדו חברות במהלך השילחה.
- (4) כמו כן נתון כי $chwt > MSS * S$ (כאשר RWND הוא גודל החלון של מקבל).
- (5) מסעיף 4 נובע כי השולח יכול לנצל באופן מלא את גודל החלון של מקבל.
- (6) לכן נוכל לחשב קירוב לתפוקה באופן הבא: $RTT/CWND$ ונקבל כי התפוקה בערך $L / MSS * RTT$.

לסיכום קיבלנו את תשובה 3.

שאלה 3

ספרת הביקורת הקטנה מבין הת"ז- 0

ולכן מכיוון שגודל החבילה הוא 0
גודל החלון גם יהיה 0 כי לא צריך אותו בשביל להעביר חבילה בגודל 0.