

~~לפניהם - מילים ושורשים~~

* הלווה כבאיותה - כבאיותה השרה בְּ אַיִלָּה נָסָה.

• ס.נ.ק. ג'נ'רל צ'לט'ר פ'לו'ס'ר ו' ג'נ'רל צ'לט'ר ס.נ.ק.

Linked List.C + Linked List.h:

tcp receiver -> tcp sender -> nifed הינה הולך ורץ על הנקודות
הנוקטות על נווט הנקודות ומשדר מודולר. מודולר משלב בינה
ו-NS ו-IP ו-TCP ו-UDP ו-ICMP ו-IGMP ו-ARP ו-LLC ו-MAC ו-PHY ו-RF.
tcp receiver -> linked list הולך ורץ על הנקודות ומשדר מודולר.

* מובן הטעינה של כל אלמנט כ-node linked list

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "LinkedList.h"
4
5
6 // Node & List Data Structures
7 (1) typedef struct _node {
8     double _time;
9     int _file_size;
10    struct _node *_next;
11 } Node;
12
13 (2) typedef struct _fileList {
14     Node *head;
15     size_t size;
16 } fileList;
17
18 (3) Node *createNode(double get_time, int file_size, Node *next) {
19     Node *newNode = (Node*)malloc(sizeof(Node)); // Allocation of memory space
20     if (newNode == NULL) {
21         return NULL;
22     }
23     newNode->_time = get_time;
24     newNode->_file_size = file_size;
25     newNode->_next = next;
26     return newNode;
27 }
28
29 (4) void Node_free(Node *node) {
30     if (node!=NULL){
31         free(node);
32     }
33 }
```

1. ביצוע נארה Node
בנאי דינמי
בנאי און לינט
בנאי קבצים

הנתקן בfileList (בlist בeach בeach בeach)

: eph Node ↗ 1.3' (3)

Node *find_pos(Node *head, Node *new_node) {
 if (head == NULL) return head;
 if (head == new_node) return head;
 Node *temp = head->next;
 while (temp != head) {
 if (temp == new_node) return temp;
 temp = temp->next;
 }
 return NULL;

(Linked List גָּדוֹלָה מִכְּלֵינְדַּן) Node דֶּקָרְנִילָה גָּדוֹלָה (4)

```
34
35 (S) void fileList_free(fileList *fileList) {
36     if (fileList == NULL) {
37         return;
38     }
39     Node *current = fileList->head;
40     while (current!=NULL) {
41         Node *next = current ->_next;
42         Node_free(current);
43         current=next;
44     }
45     free(fileList);
46 }
47
48 (C) fileList *fileList_alloc() {
49     fileList *p = (fileList *)malloc(sizeof(fileList));
50     p->head = NULL;
51     p->size = 0;
52     return p;
53 }
```

סוחרים גוועה לא-טכניון.

ו' ב' ימ' ז' נ' ו' ג' מ' א' ח' ג' נ' י' ז'

הוכין תזום גענאה (כון-הנתקנות)

(۱۶۲۰-۱۷۰۰)

גַּתְתָּה יְמִינֵי בְּבִירְעֵל אֶתְנָה תְּמִימָה כְּכֹרְבָּן ⑧

סְבִּירָה מִתְּחִילָה. בְּמַה יְהִי הַעֲלָה

۱۱۰۶۰۹۰۶۰۰۰۲۷۸۷

גַּדְעָן שְׂמִינִי פֶּה וְגַם קָדוֹשׁ בָּרוּךְ הוּא.

כְּלָבִים נַיְזָרָנִים (9)

אנו ימינו

دہلی سے

(Ansatz) $\sqrt{5x-10}$

የጋብና ፈቃረስ — (C)

לענין (הנ"מ) מושג

2. מנגנון MBIs

סִירְבָּרְגִּינְסְּ

הצלה וריג'ו

מי ימיה

.MBʃ פָּרִיזׂ שְׁבֵץׂ

```
54     size_t fileList_size(const fileList *fileList) {
55         return fileList->size;
56     }
57
58
59     void fileList_insertLast(fileList *fileList, double get_time, int file_size) {
60         Node *curr = fileList->head;
61         Node *tmp = createNode(get_time, file_size, NULL);
62         if (tmp == NULL) {
63             return; // Memory allocation failed
64         }
65         if (curr == NULL) {
66             fileList->head = tmp;
67         }
68         else {
69             // find the last node
70             while (curr->_next != NULL) {
71                 curr = curr->_next;
72             }
73             curr->_next = tmp; // curr is the last node we found and add tmp after curr
74         }
75         ++(fileList->size);
76     }
77
78     // Function to calculate speed in megabytes per second
79     double calculateSpeed(int fileSizeBytes, double timeMilliseconds) {
80         double fileSizeMB = (double)fileSizeBytes / (1024 * 1024); // Convert bytes to megabytes
81         double timeSeconds = timeMilliseconds / 1000; // Convert milliseconds to seconds
82         return fileSizeMB / timeSeconds;
83     }
84
85     void fileList_print(const fileList *fileList) {
86         Node *p = fileList->head;
87         for (int index = 1; p != NULL; index++) {
88             printf("- Run #%d Data: Time=%fms Speed=%fMB/s\n", index, p->_time, calculateSpeed(p->_file_size, p->_time));
89             p = p->_next;
90         }
91     }
92 }
```

```
93 void fileList_AverageT_print(const fileList *fileList) { I
94     double time_combined = 0;
95     Node *p = fileList->head;
96     while (p){
97         time_combined+=p->_time;
98         p=p->_next;
99     }
100    printf("- Average time: %.2fms\n", time_combined/(double)fileList->size);
101 }
102 }
```

```
102 void fileList_AverageBT_print(const fileList *fileList) {  
103     double speed_combined = 0;  
104     Node *p = fileList->head;  
105     while (p){  
106         speed_combined+=calculateSpeed(p->_file_size, p->_time);  
107         p=p->_next;  
108     }  
109     printf("- Average bandwidth: %.2fMB/s\n", speed_combined/(double)fileList->size);  
110 }  
111 }
```

סעיף ה' – מילוי ערך ורף – גורילה בוגרין – י"ג נסן ותקדמם לאחר מכן (10)

מִתְבָּאֵן בְּזַרְחֶדֶת אֲלֹנִים וְבְצַדְקָה מִתְבָּאֵן בְּזַרְחֶדֶת אֲלֹנִים

11 סדרה של פונקציות המבוצעת במאגרים.

12 נוצר אוסף של פונקציות המבוצעת במאגרים.

13 מוחזק אוסף נתונים ופונקציית גיבוב.

Sender

LinkedList.h -

Receiver.h - מוגדרת פונקציית גיבוב.

```
1 #pragma once
2
3 #include <stdlib.h>
4
5 struct _fileList;
6 typedef struct _fileList fileList;
7
8 /*
9  * Allocates a new empty fileList.
10 * It's the user responsibility to free it with fileList_free.
11 */
12 fileList *fileList_alloc();
13
14 /*
15  * Frees the memory and resources allocated to fileList.
16  * If StrList==NULL does nothing (same as free).
17 */
18 void fileList_free(fileList *fileList);
19
20 /*
21  * Returns the number of elements (files) in the fileList.
22 */
23 size_t fileList_size(const fileList *fileList);
24
25 /*
26  * Inserts an element(new received file) in the end of the fileList with time and size
27 */
28 void fileList_insertLast(fileList *fileList, double get_time, int file_size);
29
30 /*
31  * Function to calculate speed in megabytes per second
32 */
33 double calculateSpeed(int fileSizeBytes, double timeMilliseconds);
34
35 /*
36  * Prints the fileList statistics for each file (time + speed)
37 */
38 void fileList_print(const fileList *fileList);
39
```

```
40 /*
41  * Prints the fileList average time by combine each time file received and divide by the amount of files
42 */
43 void fileList_AverageT_print(const fileList *fileList);
44
45 /*
46  * Prints the fileList average speed by combine each file speed and divide by the amount of files
47 */
48 void fileList_AverageBT_print(const fileList *fileList);
```

TCP_Sender.c:

תפקידו של סנเดרTCP הוא לשלוח נתונים בtcp. נסמן את הכתובת של הסנเดר ותפקידו של ריביירTCP.

```
C TCP_Sender.c > util_generate_random_data(unsigned int)
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
6 #include <time.h>
7 #include <sys/time.h>
8 #include <sys/types.h>
9 #include <sys/socket.h>
10 #include <netinet/tcp.h> // Include this header for TCP options
11 #include <netinet/in.h>
12
13 #define MAX_BUFFER_SIZE 2048
14
15
```

. Receiver -> Sender יזקם TCP מילויים עם Headers ברוחן *
ולשלוח Data בFORMAT גודל 2048bytes בפונקציית write().
. Prints 1 סנת לא מוגדר בFORMAT
וקישר לאריך MAX_BUFFER_SIZE. מוקדם מזמן במאגר הולך וגדל
על מנת לא לזרום Receiver.agaה יתאפשר לשלוח נתונים בFORMAT.
. Packet Loss -> Bad TCP גזען

```
15
16 char *util_generate_random_data(unsigned int size) {
17     char *buffer = NULL;
18     // Argument check.
19     if (size == 0)
20         return NULL;
21     buffer = (char *)calloc(size, sizeof(char));
22     // Error checking.
23     if (buffer == NULL)
24         return NULL;
25     // Randomize the seed of the random number generator.
26     srand(time(NULL));
27     for (unsigned int i = 0; i < size; i++)
28         *(buffer + i) = ((unsigned int)rand() % 256);
29     return buffer;
30 }
```

. מילויים יוצרים פונקציית write() לשלוח נתונים בFORMAT *
Data יתאפשר לשלוח בFORMAT גודל 2048bytes

```

32 int main(int argc, char *argv[]) {
33
34     // *** Pre-Parts : Get from the user the command from terminal ***
35
36     // Expecting 6 arguments in total (excluding the program name)
37     if (argc != 7) {          // ./TCP_Sender -ip 127.0.0.1 -p 12345 -algo cubic
38         fprintf(stderr, "Please provide the correct usage for the program: %s -ip <IP> -p <PORT> -algo <ALGO>\n", argv[0]);
39         exit(EXIT_FAILURE);
40     }
41
42     // Extract command-line arguments
43     const char *ip_address = NULL;
44     int port = 0;
45     const char *congestion_algo = NULL;    // congestion_algo: The congestion control algorithm to be set (e.g., "reno" or "cubic")
46
47     // Process command-line arguments
48     for (int i = 1; i < argc; i++) {
49         if (strcmp(argv[i], "-ip") == 0 && i + 1 < argc) {
50             ip_address = argv[i + 1];
51             i++;
52         } else if (strcmp(argv[i], "-p") == 0 && i + 1 < argc) {
53             port = atoi(argv[i + 1]); // Convert a string representing an integer (ASCII string) to an integer value.
54             i++;
55         } else if (strcmp(argv[i], "-algo") == 0 && i + 1 < argc) {
56             congestion_algo = argv[i + 1];
57             i++;
58         }
59     }
60
61     // Check if required arguments are provided
62     if (ip_address == NULL || port == 0 || congestion_algo == NULL) {
63         fprintf(stderr, "Invalid arguments. Please provide the correct usage.\n");
64         exit(EXIT_FAILURE);
65     }

```

הפעלה של מודול ה-**Sender** ב-**C** ב-**gcc** ב-**Terminal** *
לעומת המודול **Receiver** ב-**Python**

```
// ./TCP_Sender -ip 127.0.0.1 -p 12345 -algo cubic
```

הפעלה של מודול ה-**Receiver** ב-**Python** ב-**Terminal** *

Receiver מקבל פאץ' אם מונע ב-**Sender** ב-**Terminal** ב-
- TCP נזקם ב-**Receiver** ב-**Terminal** ב-**Python** ב-**Terminal** ב-

(**cubic** ו- **reno**)

לעומת המודול **Sender** ב-**C** ב-**gcc** ב-**Terminal** *

int **main**{ string ip = argv[1]; int port = atoi(argv[2]); char algo = argv[3][0];}

הפעלה של מודול ה-**Receiver** ב-**Python** ב-**Terminal** *

. יונצקי טריפ מס' 16 גזע

节目中 קיימת פונקציית `util_generate_random_data()` שמייצרת נתונים אקראיים. בפונקציית `Sender` מושג יוצרים קבץ נתונים אקראיים ושמור אותו בזיכרון.

```
68 // *** Part A: Create file and read it ***
69
70 // Create File
71 const char *file_path = "random_data.txt"; // Adjust file path as needed
72 unsigned int min_size = 2*1024*1024; // At least file with 2MB size
73 unsigned int size;
74
75 // Seed the random number generator with the current time
76 srand(time(NULL));
77 // Generate a random size greater than or equal to the minimum size
78 size = min_size + (rand() % (5 * 1024 * 1024)); // Generate between 2MB to 7MB to make sure the file is at least 2MB
79
80 // Step 1: Generate random data
81 char *data = util_generate_random_data(size);
82
83 // Step 2: Check if random data generation failed
84 if (data == NULL) {
85     perror("Random data generation failed");
86     exit(EXIT_FAILURE);
87 }
88
89 // Step 3: Create and write data to file
90 FILE *file = fopen(file_path, "wb"); // wb - write binary
91 if (!file) {
92     perror("File creation failed");
93     free(data);
94     exit(EXIT_FAILURE);
95 }
96 fwrite(data, 1, size, file);
97 fclose(file);
```

הפעלה של פונקציית `util_generate_random_data()` מוציאה קבץ נתונים אקראיים. בפונקציית `Sender` מושג יוצרים קבץ נתונים אקראיים ושמור אותו בזיכרון. בפונקציית `Receiver` מושג קובץ נתונים אקראיים ששמור בזיכרון נטען בפונקציית `fread()`.

```
99 // Read the file
100 // Step 1: open data from file for reading
101 fopen(file_path, "rb"); // rb - read binary
102 if (!file) {
103     perror("File open for reading failed");
104     free(data);
105     exit(EXIT_FAILURE);
106 }
107
108 // Step 2: Read the entire file
109 if (fread(data, 1, size, file) != size) {
110     perror("Error reading file");
111     fclose(file);
112     free(data);
113     exit(EXIT_FAILURE);
114 }
```

בפונקציית `Receiver` מושג קובץ נתונים אקראיים ששמור בזיכרון נטען בפונקציית `fread()`. בפונקציית `Sender` מושג יוצרים קבץ נתונים אקראיים ושמור אותו בזיכרון. בפונקציית `Receiver` מושג קובץ נתונים אקראיים ששמור בזיכרון נטען בפונקציית `fread()`.

Receiver δ Sender ⇒ ↗ TCP-socket ↗ ביז'ן גוף

```

116 // *** Part B: Create TCP socket bet Sender - Receiver ***
117
118 //creat socket
119 int _sockfd = socket(AF_INET, SOCK_STREAM, 0); // Create a TCP socket for IPv4
120 if (_sockfd == -1) {
121     perror("Socket creation failed");
122     free(data);
123     exit(EXIT_FAILURE);
124 }
125

```

הו Socket ביז'ן

.IPv4 נאר לו Af_inet-

TCP Sc 6700 נאר לו SOCK_STREAM -

.TCP δ send ↗ מילו לו 0 -

.הו מילו ומי רבד איז'ן ה. ה כל הולך ומשהו יתאפס ב Socket נזלו לו מילו

```

126 // Set congestion control algorithm
127 // setsockopt system call to configure the TCP_CONGESTION option.
128 // After that the specified congestion control algorithm is applied to the TCP connection.
129 if (setsockopt(_sockfd, IPPROTO_TCP, TCP_CONGESTION, congestion_algo, strlen(congestion_algo)) != 0) {
130     perror("Error setting congestion control algorithm");
131     close(_sockfd);
132     free(data);
133     exit(EXIT_FAILURE);
134 }
135

```

.הו מילו ומי רבד איז'ן ה. ה כל הולך ומשהו יתאפס ב Socket נזלו לו מילו

Socket δ IPv4 נאר לו TCP δ IPv4 נאר לו מילו ומי רבד איז'ן ה. ה כל הולך ומשהו יתאפס ב Socket נזלו לו מילו

(setsockopt(_sockfd, IPPROTO_TCP, TCP_CONGESTION, congestion_algo, strlen(congestion_algo)))

```

136 //create receiver address struct
137 struct sockaddr_in server_address; // Struct sockaddr_in is defined in the <netinet/in.h> header file.
138 memset(&server_address, 0, sizeof(server_address));
139
140 server_address.sin_family = AF_INET;
141 server_address.sin_port = htons(port); // htons() function ensures that the port number is properly formatted for network
142 int rval = inet_pton(AF_INET, ip_address, &server_address.sin_addr); // inet_pton() is a function that converts an IPv4 a
143 if (rval <= 0)
144 {
145     printf("inet_pton() failed");
146     return -1;
147 }
148

```

Struct sockaddr_in ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

.31ג'ן δ IPv4 נאר לו מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

.לע"ז מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

.לע"ז מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

(inet_pton() מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

הו inet_pton() מילו ומי רבד איז'ן ה. ה Receiver δ Fe-receivser מילו ומי רבד איז'ן ה.

הו inet_pton() מילו ומי רבד איז'ן ה.

```
149 // Connect to the Receiver
150 printf("Connecting to Receiver...\n");
151
152 if (connect(_sockfd, (struct sockaddr *)&server_address, sizeof(server_address)) == -1) {
153     perror("Connection failed");
154     close(_sockfd);
155     free(data);
156     exit(EXIT_FAILURE);
157 }
158
159 printf("Connected to Receiver. Beginning file transfer...\n");
160
```

לנורא נסח רכזת אמצעים Receiver שמיינטן נורא נסח. מושג הנקרא Receiver Se הינו מושג שמיינטן נסח. מושג הנקרא Receiver Se. מושג הנקרא Receiver Se. מושג הנקרא Receiver Se.

• >10 miles per second) $\sqrt{6 \ln(1 - T_{\text{dip}}/\tau_{\text{diff}})} = 2 + 2 \sqrt{\ln}$

```
161 // *** Part C + D: Send the file + user's decision
162
163 // Send the size of the file to the receiver
164 ssize_t size_sent = send(_sockfd, &size, sizeof(size), 0);
165 if (size_sent < 0) {
166     perror("Error sending file size");
167     close(_sockfd);
168     free(data);
169     exit(EXIT_FAILURE);
170 }
171 int send_again = 1; // Flag to control the loop
172 while (send_again>0) {
173     // Send the file in chunks to minimize packet loss and bad TCP
174     printf("Send the file...\n");
175     unsigned int sent_total = 0;
176     while (sent_total<size){
177         unsigned int remaining = size - sent_total;
178         unsigned int chunk_size = remaining < MAX_BUFFER_SIZE ? remaining : MAX_BUFFER_SIZE;
179         ssize_t sent = send(_sockfd, data+sent_total, chunk_size, 0); // ssize_t val for negative num for errors
180         // Check for errors during sending
181         if (sent < 0) {
182             perror("Error sending random data");
183             close(_sockfd);
184             free(data);
185             exit(EXIT_FAILURE);
186         }
187         sent_total += sent;
188     }
189     printf("a file with %u bytes has been sent successfully\n", sent_total);
190 }
```

הנחיות משלב ה- n -הו' מתקבלות על ידי סכום כל הנחיות משלב ה- $n-1$ -הו' מילויים.

.rec() — "צורה" :

רְשָׁוָה וְעַמְּדָה בְּרִית מִקְּדוֹשָׁה יְהוָה

בגנום של *Aec* ישנו גן אחד, *EF1α*, שמייצרת RNA-polimeraza. גן זה מודפס ב-*EF1α* ו-*EF1β*. *EF1α* מודפס ב-*EF1α* ו-*EF1β*. *EF1α* מודפס ב-*EF1α* ו-*EF1β*.

.ephn

היקף זה אוסף נתונים מוגדרים בפונקציית `df` (הנמצאת בקובץ `iris`).

הפרטיר-סינס מס' ה-1000 הוחזק כמי שהשאלה גאהם.

የኢትዮ ከደራል ቤትና ገዢና በፍጥነት የተዘረዘሩ ተከራክር ያለንበት ነው

Infected bytes \rightarrow viral nodes

```
191 // Ask user for decision
192 printf("Do you want to send the file again? (yes/no): ");
193
194 // Get only valid answers - yes or no
195 int valid_char = 0;
196 while (valid_char==0) {
197     char decision[4];
198     scanf("%s",decision);
199     if(strcmp(decision, "no")==0 || strcmp(decision, "yes")==0) {
200         ssize_t send_decision = send(_sockfd,&decision,strlen(decision),0);
201         if(send_decision < 0) {
202             perror("Error sending decision");
203             close(_sockfd);
204             free(data);
205             exit(EXIT_FAILURE);
206         }
207         if (strcmp(decision, "no")==0) {
208             send_again = 0; // If decision is not "yes", exit loop
209         }
210         valid_char=1;
211     }
212     else {
213         printf("not valid answer, try again\n");
214     }
215 }
216 }
```

```
217
218 // *** Part E: Send exit message to the receiver ***
219
220 char exit_message[] = "Exit";
221 ssize_t sendX= send(_sockfd, &exit_message, strlen(exit_message), 0);
222 if(sendX < 0) {
223     perror("Error sending exit_message");
224     close(_sockfd);
225     free(data);
226     exit(EXIT_FAILURE);
227 }
228 printf("Exit\n");
```

• A δ ሱዣንግ፣ ማኅበና አቀፍናን በተደረገው ጥሩ

```
229  
230     // *** Part F: Close the TCP connection ***  
231     close(_sockfd);  
232     free(data);  
233     remove(file_path); // Remove temporary file  
234  
235     // *** Part G: Exit ***  
236     return 0;  
237 }
```

TcpReceiver.c ;

```
C TCP_Receiver.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
6 #include <sys/time.h>
7 #include <sys/types.h>
8 #include <sys/socket.h>
9 #include <netinet/tcp.h> // Include this header for TCP options
10 #include <netinet/in.h>
11 #include "LinkedList.h"
12
13 #define MAX_BUFFER_SIZE 2048
14
15 // a function to calculate milliseconds
16 double get_time_in_milliseconds(struct timeval start, struct timeval end) {
17     return (double)(end.tv_sec - start.tv_sec) * 1000.0 + (double)(end.tv_usec - start.tv_usec) / 1000.0;
18 }
19
20 int main(int argc, char *argv[]) {
21
22     // *** Pre-Parts : Get from the user the command from terminal ***
23
24     if (argc != 5) {           // ./TCP_Receiver -p 12345 -algo cubic
25         fprintf(stderr, "Please provide the correct usage for the program: %s -ip IP -p PORT -algo ALGO <FILE_PATH>\n", argv[0]);
26         exit(EXIT_FAILURE);
27     }
28
29     // Extract command-line arguments
30     int port = 0;
31     const char *congestion_algo = NULL; // congestion_algo: The congestion control algorithm to be set (e.g., "reno" or "cubic")
32 }
```

טראנסFORMERS
 millis ↴

מיליס נס

कृष्ण

```
33 // Process command-line arguments
34 for (int i = 1; i < argc; i++) {
35     if (strcmp(argv[i], "-p") == 0 && i + 1 < argc) {
36         port = atoi(argv[i + 1]); // Convert a string representing an integer (ASCII string) to an integer value.
37         i++;
38     } else if (strcmp(argv[i], "-algo") == 0 && i + 1 < argc) {
39         congestion_algo = argv[i + 1];
40         i++;
41     }
42 }
43
44 // Check if required arguments are provided
45 if (port == 0 || congestion_algo == NULL) {
46     fprintf(stderr, "Invalid arguments. Please provide the correct usage.\n");
47     exit(EXIT_FAILURE);
48 }
49
50 // *** Part A: Create a TCP socket ***
51
52 int listeningSocket = socket(AF_INET, SOCK_STREAM, 0); // Create a TCP socket for IPv4
53 if (listeningSocket == -1) {
54     perror("Socket creation failed");
55     exit(EXIT_FAILURE);
56 }
57
58 // Set congestion control algorithm
59 // setsockopt system call to configure the TCP_CONGESTION option.
60 // After that the specified congestion control algorithm is applied to the TCP connection.
61 if (setsockopt(listeningSocket, IPPROTO_TCP, TCP_CONGESTION, congestion_algo, strlen(congestion_algo)) != 0) {
62     perror("Error setting congestion control algorithm");
63     close(listeningSocket);
64     exit(EXIT_FAILURE);
65 }
66
67 // Define Receiver's values
68 printf("Starting Receiver...\n");
69 struct sockaddr_in server_address; // Struct sockaddr_in is defined in the <netinet/in.h> header file.
70 memset(&server_address, 0, sizeof(server_address));
71
72 server_address.sin_family = AF_INET;
73 server_address.sin_addr.s_addr = INADDR_ANY; // The server will listen on all available network interfaces.
74 server_address.sin_port = htons(port); // htons() function ensures that the port number is properly formatted for network
75
76 // Bind the socket to the specified address and port
```

Gpio_raj
psice Received
inse won't

8 נסח, נסח היה נושא של סעיפים

וכן מוכיח:

```
75 // Bind the socket to the specified address and port
76 if (bind(listeningSocket, (struct sockaddr *)&server_address, sizeof(server_address)) == -1) {
77     perror("Bind failed");
78     close(listeningSocket);
79     exit(EXIT_FAILURE);
80 }
81 
```

לכונת הלקוח ותפקידו בBIND()

(A) מטרת הקוד היא לארוך חיבור הסocket, אם כתובתו לא תואם IP הכתובת בדרכו. נורא אם אפליקציית קידום יתאפשרה.

במהותו bihdt() מודול הלקוח

```
82 // Listen for incoming connections
83 if (listen(listeningSocket, 1) == -1) {
84     perror("listen() failed");
85     close(listeningSocket);
86     exit(EXIT_FAILURE);
87 }
88 // listen to just one TCP connection
```

במהותו bihdt() מודול הלקוח ותפקידו בBIND()

. Sender מאריך חיבור הסocket ותפקידו בBIND()

. Sender מאריך חיבור הסocket ותפקידו בBIND()

```
90 // *** Part B: Get a connection from the sender ***
91
92 // Define Sender
93 struct sockaddr_in client_address;
94 socklen_t client_address_len = sizeof(client_address);
95 memset(&client_address, 0, sizeof(client_address));
96
97 // Accept a connection
98 printf("Waiting for TCP connection...\n");
99
100 int clientSocket = accept(listeningSocket, (struct sockaddr *)&client_address, &client_address_len);
101 if (clientSocket == -1) {
102     perror("Accept failed");
103     close(listeningSocket);
104     exit(EXIT_FAILURE);
105 }
106 printf("Sender connected, beginning to receive file...\n");
107 
```

מזהה קידום הלקוח

Sender מאריך חיבור הסocket ותפקידו בBIND()

תפקידו בBIND() מאריך חיבור הסocket ותפקידו בBIND()

(A) se ondress מאריך חיבור הסocket ותפקידו בBIND()

אך לא מאריך חיבור הסocket ותפקידו בBIND()

לינוקס הקודן וריבוי הרים

```
108 // *** Part C: Receive the file, measure the time it took to save ***
109
110 struct timeval start_time, end_time;
111 fileList* files = fileList_alloc(); // Start a new null list of files
112 char buffer[MAX_BUFFER_SIZE];
113
```

headers ו- `struct timeval` נאשנה ב-`recv` על מנת-

.`timeval` יתאפשר לרשום וארון. מחרה `recv` יתאפשר

לצורך שדרוג מ-`recv` ל-`gettimeofday` כנ"ט פה, לאן כל ערך -

לפיה `start_time`, נציגו רצף של נתונים עבור `Data` ו-`Time` יתאפשר

לפיה `end_time` (באותה ערך ה-`Time` מ-`start_time`, ה-`Time` יתאפשר

.`struct timeval` Linkedlist

מ-`recv` מ-`recl()` מ-`buf` נתונים ב-`file_size` ו-`buf` נזק -
הנ"ט מ-`recv` מ-`buf` מ-`recl()` מ-`buf` נזק

```
114 // Receive the size of the file from the sender
115 unsigned int file_size;
116 ssize_t size_received = recv(clientSocket, &file_size, sizeof(file_size), 0);
117 if (size_received < 0) {
118     perror("Error receiving file size");
119     close(clientSocket);
120     close(listeningSocket);
121     exit(EXIT_FAILURE);
122 }
```

שידור מ-`recv` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recl()` מ-`buf` מ-`recv`.

הנ"ט מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recv`.

לעתה מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`recv`.

הנ"ט מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recv`.

```
124 // Receive the data of the file from the sender
125 while (1)
126 {
127     int total_received = 0;
128     while(total_received<file_size) {
129         gettimeofday(&start_time,NULL);
130         ssize_t bytes_received = recv(clientSocket, buffer, MAX_BUFFER_SIZE, 0);
131         if (bytes_received < 0) {
132             perror("Error receiving file");
133             close(clientSocket);
134             close(listeningSocket);
135             exit(EXIT_FAILURE);
136         }
137         // Connection closed
138         if (bytes received == 0) {
139             printf("Connection closed by peer.\n");
140             close(clientSocket);
141             close(listeningSocket);
142             exit(EXIT_SUCCESS);
143         }
144         total_received+=bytes_received;
145     }
146 }
```

הנ"ט מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recv`.

הנ"ט מ-`recv` מ-`buf` מ-`recl()` מ-`buf` מ-`file_size` מ-`buf` מ-`recv`.

פְּרָקִים 1-2 Data שוכן מ-Rec1 ו-Rec2, כלומר ה-Rec1 וה-Rec2 הם פרטיים.

הנוסף ל-`total_rpc` מוגדר ב-`total_rpc_size`. מטרת הערך הוא לרשום את גודל כל RPC שמבצעים. מטרת הערך היא לרשום את גודל כל RPC שמבצעים.

```
146     gettimeofday(&end_time, NULL); // Set the end time before measuring elapsed time
147     double elapsed_time = get_time_in_milliseconds(start_time, end_time);
148     fileList_insertLast (files,elapsed_time,file_size); // Create new file
149     printf("File transfer completed (%d bytes).\n",total_received);
150     printf("Waiting for Sender response...\n");
151     memset(buffer, 0, strlen(buffer));
152 }
```

לפניהם נתקל בטרכז (טרם-טרכז) וטרכז (טרכז-טרטרכז).
טרכז מושך אליו טרכזים וטרכזים מושכים אליו טרכזים.

. Sender - an המודפסים יוצרים:

```
154     // ** Part D: Wait for Sender Response **
155
156     ssize_t decision = recv(clientSocket, buffer, 1, 0);
157     if (decision < 0) {
158         perror("Error receiving file");
159         close(clientSocket);
160         close(listeningSocket);
161         exit(EXIT_FAILURE);
162     }
163     // if user don't want to send the file again, the next time in the loop won't measure time and will get exit message
164     if(buffer[0]=='\n') {
165         memset(buffer, 0, strlen(buffer));
166         break;
167     }
168     if (decision == 0) {
169         printf("Connection closed by peer.\n");
170         close(clientSocket);
171         close(listeningSocket);
172         exit(EXIT_SUCCESS);
173     }
174 }
```

$\therefore 3 + 1 + 7 = 11$

```
176 // Get Exit message from Sender
177 recv(clientSocket, buffer, 5, 0);
178 if(buffer[1]=='E') {
179     printf("Sender sent exit message.\n");
180 }
181 memset(buffer, 0, MAX_BUFFER_SIZE);
182
183 // ** Part E: Print out statistics **
184
185 printf("-----\n");
186 printf(" * Statistics * -\n");
187 fileList_print(files); //function that prints time and speed for each file
188
189 // ** Part F: Calculate the average time and the total average bandwidth **
190
191 fileList_AverageT_print(files);
192 fileList_AverageBT_print(files);
193 printf("-----\n");
194 close(clientSocket);
195 close(listeningSocket);
196 fileList_free(files);
197
198 // ** Part G: Exit **
199
200 printf("Receiver end.\n");
201
202 }
```

15. נס่ง `send()` מ-`N` ה-`מ`רחבת `func`, `func` מ-`func`

Sendet in die Zeit

הנישות הדרומית והרומית.

• תְּהִלָּה רַבָּה נְגֻדָּלָה

Makefile:

```
M Makefile
1 CC=gcc
2 FLAGS=-Wall -g
3
4 all: TCP_Sender TCP_Receiver
5
6 TCP_Sender: TCP_Sender.o
7 | $(CC) $(FLAGS) -o TCP_Sender TCP_Sender.o
8
9 TCP_Sender.o: TCP_Sender.c
10 | $(CC) $(FLAGS) -c TCP_Sender.c
11
12 TCP_Receiver: TCP_Receiver.o LinkedList.o
13 | $(CC) $(FLAGS) -o TCP_Receiver TCP_Receiver.o LinkedList.o
14
15 TCP_Receiver.o: TCP_Receiver.c LinkedList.h
16 | $(CC) $(FLAGS) -c TCP_Receiver.c
17
18 LinkedList.o: LinkedList.c LinkedList.h
19 | $(CC) $(FLAGS) -c LinkedList.c
20
21 .PHONY: clean
22
23 clean:
24 | rm -f *.o *txt TCP_Sender TCP_Receiver
```

நிலை நடவடிக்கை : 2 புதின்

Keno:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- yairco@yairco:~/reshatot/EX3/A\$ make clean
rm -f *.o *txt TCP_Sender TCP_Receiver
- yairco@yairco:~/reshatot/EX3/A\$ make all
gcc -Wall -g -c TCP_Sender.c
gcc -Wall -g -o TCP_Sender TCP_Sender.o
gcc -Wall -g -c TCP_Receiver.c
gcc -Wall -g -c LinkedList.c
gcc -Wall -g -o TCP_Receiver TCP_Receiver.o LinkedList.o
- yairco@yairco:~/reshatot/EX3/A\$./TCP_Receiver -p 1234 -algo re no
Starting Receiver...
Waiting for TCP connection...
Sender connected, beginning to receive file...
File transfer completed (5272711 bytes).
Waiting for Sender response...
File transfer completed (5272713 bytes).
Waiting for Sender response...
File transfer completed (5272713 bytes).
Waiting for Sender response...
File transfer completed (5272713 bytes).
Waiting for Sender response...
Sender sent exit message.

- * Statistics * -
- Run #1 Data: Time =0.01ms; Speed=838074.84MB/s
- Run #2 Data: Time =0.01ms; Speed=558716.56MB/s
- Run #3 Data: Time =0.01ms; Speed=359174.93MB/s
- Run #4 Data: Time =0.01ms; Speed=558716.56MB/s
- Run #5 Data: Time =0.01ms; Speed=718349.87MB/s
- Average time: 0.01ms
- Average bandwidth: 606606.55MB/s

Receicer end.
- yairco@yairco:~/reshatot/EX3/A\$./TCP_Sender -ip 127.0.0.1 -p 1234 -algo reno
Connecting to Receiver...
Connected to Receiver. Beginning file transfer...
Send the file...
a file with 5272711 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 5272711 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 5272711 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 5272711 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 5272711 bytes has been sent successfully
Do you want to send the file again? (yes/no): no
Exit
- yairco@yairco:~/reshatot/EX3/A\$

Cubic:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash + v

• yairco@yairco:~/reshatot/EX3/A$ make clean
rm -f *.o *txt TCP_Sender TCP_Receiver
• yairco@yairco:~/reshatot/EX3/A$ make all
gcc -Wall -g -c TCP_Sender.c
gcc -Wall -g -o TCP_Sender TCP_Sender.o
gcc -Wall -g -c TCP_Receiver.c
gcc -Wall -g -c LinkedList.c
gcc -Wall -g -o TCP_Receiver TCP_Receiver.o LinkedList.o
• yairco@yairco:~/reshatot/EX3/A$ ./TCP_Sender -ip 127.0.0.1 -p 1234
-algo cubic
Connecting to Receiver...
Connected to Receiver. Beginning file transfer...
Send the file...
a file with 4873495 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 4873495 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 4873495 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 4873495 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
a file with 4873495 bytes has been sent successfully
Do you want to send the file again? (yes/no): no
Exit
• yairco@yairco:~/reshatot/EX3/A$ █
```



```
• yairco@yairco:~/reshatot/EX3/A$ ./TCP_Receiver -p 1234 -alg
Starting Receiver...
Waiting for TCP connection...
Sender connected, beginning to receive file...
File transfer completed (4873495 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
File transfer completed (4873497 bytes).
Waiting for Sender response...
Sender sent exit message.

-----
- * Statistics * -
- Run #1 Data: Time =0.01ms; Speed=516414.11MB/s
- Run #2 Data: Time =0.01ms; Speed=580965.88MB/s
- Run #3 Data: Time =0.01ms; Speed=580965.88MB/s
- Run #4 Data: Time =0.01ms; Speed=516414.11MB/s
- Run #5 Data: Time =0.01ms; Speed=580965.88MB/s
- Average time: 0.01ms
- Average bandwidth: 555145.17MB/s

-----
Receicer end.
• yairco@yairco:~/reshatot/EX3/A$ █
```

၁၆၂၃၃ : ၃ ရပ်

:reno 12fic

• 1. גֶּפֶן מִזְבֵּחַ וְתָמֵן כְּלֵי קְרֻבָּה וְתָמֵן כְּלֵי קְרֻבָּה

No.	Time	Source	Destination	Protocol	Length	Info
6	37.892314128	127.0.0.1	127.0.0.1	TCP	74	41136 → 1234 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
7	37.892341949	127.0.0.1	127.0.0.1	TCP	74	1234 → 41136 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS
8	37.892353539	127.0.0.1	127.0.0.1	TCP	66	41136 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=34
9	37.892405921	127.0.0.1	127.0.0.1	TCP	70	41136 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSv
10	37.892411670	127.0.0.1	127.0.0.1	TCP	66	1234 → 41136 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=34
11	37.892466733	127.0.0.1	127.0.0.1	TCP	2114	41136 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=2048
12	37.892472665	127.0.0.1	127.0.0.1	TCP	66	1234 → 41136 [ACK] Seq=1 Ack=2053 Win=64128 Len=0 TSval
13	37.892485649	127.0.0.1	127.0.0.1	TCP	2114	41136 → 1234 [PSH, ACK] Seq=2053 Ack=1 Win=65536 Len=20
14	37.892494974	127.0.0.1	127.0.0.1	TCP	66	1234 → 41136 [ACK] Seq=1 Ack=4101 Win=63104 Len=0 TSval
15	37.892503111	127.0.0.1	127.0.0.1	TCP	2114	41136 → 1234 [PSH, ACK] Seq=4101 Ack=1 Win=65536 Len=20
16	37.892507164	127.0.0.1	127.0.0.1	TCP	66	1234 → 41136 [ACK] Seq=1 Ack=6149 Win=62080 Len=0 TSval
17	37.892514427	127.0.0.1	127.0.0.1	TCP	2114	41136 → 1234 [PSH, ACK] Seq=6149 Ack=1 Win=65536 Len=20

Sender = nijδ }
receiver = pje }

הלקוח שולח לשרת בקשה לפתיחת קשר [SYN]

השרת מקלט את ההודעה ושולח בתגובה להודעת אישור קבלה ואישור פתיחת קשר מצדיו שלוח [SYN, ACK].

הלקוח מידייע את השרת על סיום מיסוד הקשר בהודעת [ACK].

רֹאשׁ הַקָּבֵד בְּצִוְתִּים מִצְרָיִם Receiver of his Sender וְגַם
בְּצִוְתִּים מִצְרָיִם Receiver of his Sender וְגַם

bytes de מין Receiver (niseg ſunm ſender) ſur ſur ſe .
bytes ſe ſigur ſe ſigur ſo . niseg mode ſo ſigur ſe data ſo
ſigur receiver ſo , ſigur ſo

The screenshot shows the Wireshark interface with the following details:

- File**: reno.pcapng
- Frame List View**: Shows the first four frames of the capture. The fourth frame is highlighted.
- Selected Frame**: Frame 331 (TCP Segment 1):
 - Source: 127.0.0.1
 - Destination: 127.0.0.1
 - Protocol: TCP
 - Length: 69 bytes
 - Info: 69 41136 → 1234 [PSH, ACK] Seq=5272716 Ack=1 Win=65536 Len=3 TS=10000000000000000000000000000000
- Hex View**: Displays the raw hex and ASCII data for the selected frame.
- Selected Hex Value**: The value 44 (D) is selected in the hex view.
- Selected ASCII Value**: The value yes is selected in the ASCII view.

במקרה יישן מ-Receiver של Sender שולץ מ-
Sender, מושג יתאפשר על ידי קידום תקן קבוצה (במקרה
הנ"ל, קידום תקן קבוצה מ-1 ל-2). אולם במקרה
הנ"ל, קידום תקן קבוצה מ-1 ל-2 לא יאפשר,

The screenshot shows a Wireshark session titled "reno.pcapng" with the filter "tcp.port == 1234". The packet list pane displays 1512 TCP packets. The selected packet (1506) is highlighted in blue and has its details and bytes panes expanded. The details pane shows the following information:

- Frame 1506: 68 bytes on wire (544 bits), 68 bytes captured (544 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: 41136, Dst Port: 1234, Seq: 26363572, Ack: 1, Len: 2

The bytes pane shows the raw hex and ASCII data of the selected packet, which starts with the byte sequence 8a f6 6e 6f.

הזרה היא ריכוז Receiver ג' המכיל מוקדי Receiver ו- Sender . הזרה מוגדרת כ-

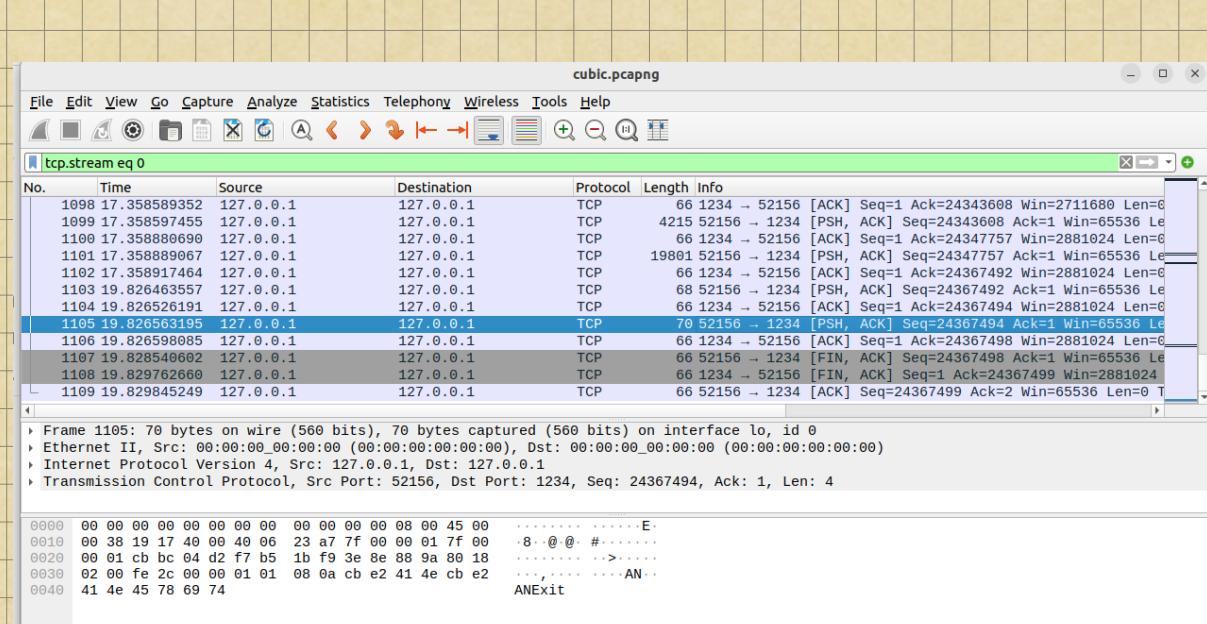
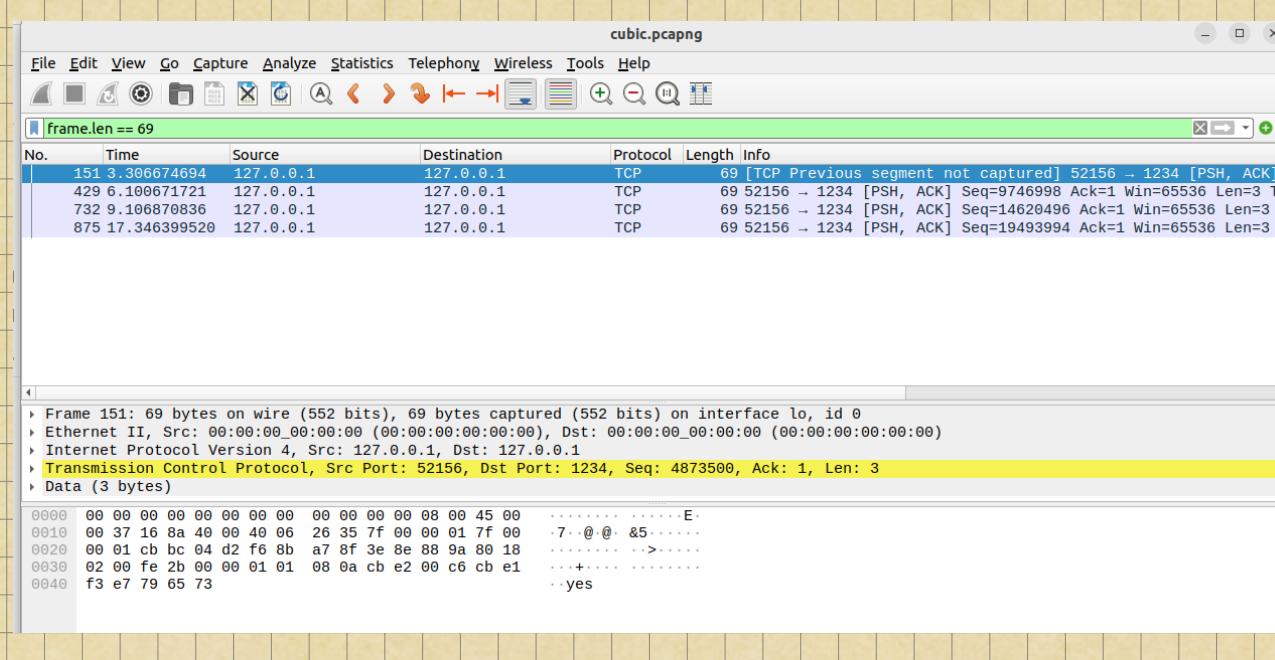
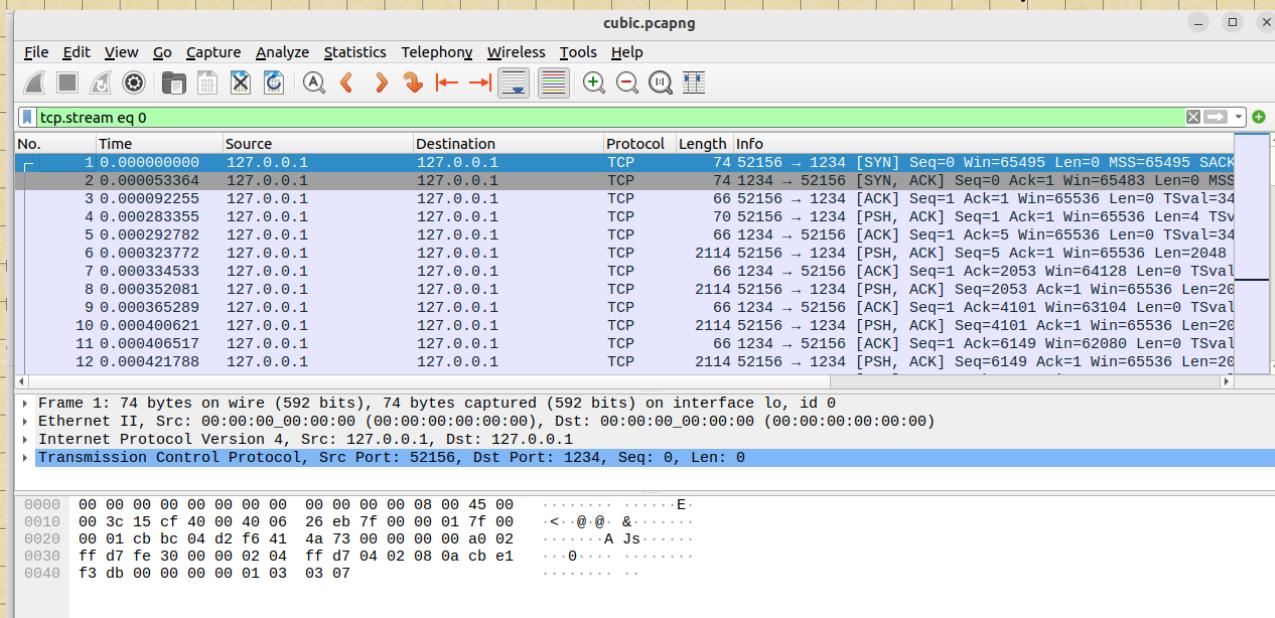
הודעה לסגירת קשר עם השרת מצד הלוקו [FIN ,ACK]

הודעה לסיירת קשר עם הלוקה מצד השירות [FIN ,ACK]

הוּא תְּלִיכָה כְּלֵינָה וְנִמְלֵאת הַמִּזְבֵּחַ בְּעַמְקֹם

: Cubic 'lllK

.201 215.3 173K 77 KIC G70K 115 pS, 101K 174K 852m 80K



בינה מאובטת טריינינג מילויים

Receiver:

Socket()



bind()



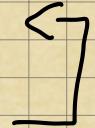
listen()



accept()



rec()



print()



exit()

Sender:

create file()



writel()



read()



socket()



connect()



send()



get answer
from user



close & exit

connection

1) filesize

2) data

3) yes/no

4) exit

