

גדרון מודולרי - מודול 3

תאריך: 16.16.2013,-ID: 206386708
 נושא: מודול 3 - מבוא למודול 3 ועיבוד נתונים

לesson 1: מודול 3 של הפרויקט.

הנושא מודול 3 מוקדש לניתוח נתונים ועיבוד נתונים.

LinkedList.C + LinkedList.h: - מודול 3 מוקדש לניתוח נתונים

RUDP_Receiver -> RUDP_Sender -> LinkedList.h
 מודול מודול 3 מוקדש לניתוח נתונים ועיבוד נתונים ועיבוד נתונים.
 מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.
 מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

* מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

```

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: מודול LinkedList.h

הנושא מודול 3 מוקדש לניתוח נתונים ועיבוד נתונים.

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

②: fileList מוגדר בLinkedList.h

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

linked

③: מודול LinkedList.h

מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.
 מודול LinkedList.h מוקדש לניתוח נתונים ועיבוד נתונים.

(LinkedList.h מוגדר בLinkedList.h) מודול LinkedList.h מוגדר בLinkedList.h

```
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 }
```

סורה ה' לאלה נסרים.

6. יבירות זיכריה וענין גהינמיה.

הוכין תזום גענאה (כון-הנתקנות)

(1) Israe

הנורווגי הילן גראנץ נולדה ב-1982 וגדלה ב-

פְּנָקְדָּמָן כְּבוֹדָיו. בְּמַעַן יְמִינָה נְפָאָת

11.06.06. 06.06.11. 11.06.06. 06.06.11.

የኢትዮጵያ የፌዴራል ቤት አገልግሎት

```
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("- %d Data: Time = %.2fms; Speed=% .2fMB/s\n", index, p->_time, calculateSpeed(
89            p->_size, p->_time));
90            p = p->_next;
91        }
92    }

```

כְּנָסָרִים גַּתְתָּאָה (9)

נְגִינָה

ریاضیات

(ANSWER) NON-IC

የርግኩ ስጋዬ — (C)

לְפָנֶיךָ יְהוָה אֱלֹהֵינוּ

ההירוט MBIs

June 2020

הסנה הגדולה

۱۹۷۰ء

```
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 }
```

לפניהם נתקל עבורי רופא גנטיקה מודרנית אשר מגדירה את ה-

הנושאים העיקריים בקורס:

11. קידום מוכן לאחסון ומעבר נתונים בין מודולים.

12. נציג את דוגמאות הלקוחות והלקוחות המודולים.

13. מושג המודולים ומבנה המודולים.

Sendet

LinkedList.h -

14. מושג סדרה גוזרת כ-`const`.

```
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);
```

A 107 Receiver נספחים בונן

S קורס Sender

Senderה מודעת לserver רשיון לאפשר לclient לשלוח וריאנטים

- נאכלה Receiver הא

הוירטואלי, handshake ו- /טוקן אכזרי שדרוג מודול RUDP נזקן

. נאכלה Sender נזקן

ליד RUDP נזקן קידמיים נזקן - ו- צ'ק

RUDP-API.h

```

1 #pragma once
2
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <string.h>
6 #include <arpa/inet.h>
7 #include <stdio.h>
8 #include <stdlib.h>
9 #include <sys/time.h>
10 #include <time.h>
11 #include <netinet/in.h>
12 #include <errno.h>
13 #include <stdint.h>
14 #include <unistd.h>
15
16
17 typedef struct UDP_Header Packet;
18
19 int rudp_socket();
20
21 int got_ACK(int sockfd, struct sockaddr * from, socklen_t * fromlen);
22
23 int handshake_connect(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr * respond_server, socklen_t * respond_
24
25 int rudp_send(int sockfd, const void*msg, int len, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr * respond_server, soc
26
27 int rdup_close(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr* respond_server, socklen_t * respond_server_l
28
29 int send_ACK(int sockfd, struct sockaddr * dest, socklen_t destlen);
30
31 int rudp_receive(int sockfd, struct sockaddr * Sender_adrr, socklen_t * Sender_len);
32
33 unsigned short int calculate_checksum(void *data, size_t bytes) ;
34
35 int verify_checksum(Packet *buffer, size_t bytes);

```

הוירטואלי
handshake ו-
רשיון אכזרי
RUDP-API.c, RUDP-Sender.c
RUDP-Receiver.c

```

#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <netinet/in.h>
#include <errno.h>
#include <stdint.h>
#include <unistd.h>

```

- הוראות סטטוס
- ב-6710 מושג רשות /-
- STRING /טוקן
- השוואת כתובת IP /227
- מילוי /-0000
- מילוי /-0000
- מילוי /-0000
- פירוט מנגנון קידם ורשיון
- הוראות נרכש גבירותי
- מילוי /-0000
- מילוי int ו-00 מושג checksum "
- מילוי /-0000

כגון ה `header` ו `body` הינה אוסף של נתונים לא-ordinalים אשר ערכיהם ניתן לרשום כ-

```
1 #include "RUDP_API.h"
2
3 #define MAX_BUFFER_SIZE 2048
4 #define MAX_RETRANSMISSION_ATTEMPTS 5
5
```

לעומת TCP יש לנו struct שמספק UDP-Header ו-Data. הערך הראשון ב-Header הוא Type (16bits), בו נקבע אם מדובר ב-data או ב-control. לאחר מכן נמצאים (16bits * 2) bytes של data. לאחר מכן נמצאים (16bits * 2) bytes של checksum. לאחר מכן נמצאים (8bits * 2) bytes של flag. לאחר מכן נמצאים (16bits * 2) bytes של max buffer size. סוף סוף נמצאים bytes של data.

```
5 // RUDP Header
6
7 typedef struct UDP_Header {
8     unsigned short Length; // 2 Bytes (Byte 0, Byte 1) for length of data
9     unsigned short Checksum; // 2 Bytes (Byte 2, Byte 3) for checksum
10    char Flag; // 1 Byte for: SYN = 'S', ACK = 'A', Data = 'D', FIN = 'F'
11    char Content [MAX_BUFFER_SIZE];
12 } Packet;
13
```

Socket API provides socket interface for TCP and UDP.

```
14 // Creating a RUDP socket
15 int rudp_socket() {
16     int soc = socket(AF_INET, SOCK_DGRAM, 0); // Create a UDP socket for IPv4
17     if (soc == -1) {
18         perror("Socket creation failed");
19         exit(EXIT_FAILURE);
20     }
21     return soc;
22 }
```

סוקט Socket סוקט

- IPv4 נזרק לינט Af_inet-
- IPv6 נזרק לינט sock_Dgram -
- UDP שלSend _לינט O -

:checksum ↗ 13711 ③

- כוחם של יוזמות הניסיון מחייב הרכבת מנגנון אובייקטיבי לשליטה על החלטות.

• Checksum ✓ 37102

הנורווגי הופיע בפעם הראשונה ב-1992.

2023-24 15 जून

checksum ת"ר גביה false ו-true, נסמן ב-טפל header ⇒ טפל inline

(checksum) \rightarrow $\text{sum}(\text{all data bytes}) \oplus \text{sum}(\text{all parity bytes})$

הנורווגיה נסעה ברכבת לסקונה

Calculate checksum (c)

(כל רצויו ירוכב ב-16 ס"מ מטרים ו-28 ס"מ מטרים)

សរុប ពិសេស 1-8 និង ៩០១ ដែល នឹង

```
24 // Function to calculate checksum
25 unsigned short int calculate_checksum(void *data, size_t bytes) {
26     unsigned short int *data_pointer = (unsigned short int *)data;
27     unsigned int total_sum = 0;
28
29     // Main summing loop
30     while (bytes > 1) {
31         total_sum += *data_pointer++;
32         bytes -= 2;
33     }
34     // Add left-over byte, if any
35     if (bytes > 0){
36         total_sum += *((unsigned char *)data_pointer);
37         total_sum &= 0xFFFF; // Ensure that the sum is within 16 bits
38     }
39
40     // Fold 32-bit sum to 16 bits
41     while (total_sum >> 16)
42         total_sum = (total_sum & 0xFFFF) + (total_sum >> 16);
43
44     return (~(unsigned short int)total_sum);
45 }
```

: Verify - checksum (2)

בנין ופיזיקה checksumatic פיקטוריון זר. DATA BYTES 65-127 ופיזיקה גאות

• **11022 11812 191 1103 1104 116723 108**

לפנינו נתונים טרנספורמציוניים (transformed data) שמשתמשים בפונקציית \log .

הנתקה בפונקציית `base` ו`checknum` בפונקציית `pic`.

הנִּמְצָא בְּבֵית הַמִּזְבֵּחַ וְבְּבֵית הַלְּבָנָה

```
46 // Function to verify checksum
47 int verify_checksum(Packet *buffer, size_t bytes) {
48     // Extract data and checksum from the packet
49     unsigned short *data_pointer= (unsigned short int *)buffer->Content;
50     unsigned short checksum = buffer->Checksum;
51     unsigned int total_sum = checksum;
52
53     // Calculate the checksum of the received data combined with the checksum
54     // Main summing loop
55     while (bytes > 1) {
56         total_sum += *data_pointer++;
57         bytes -= 2;
58     }
59     // Add left-over byte, if any
60     if (bytes > 0){
61         total_sum += *((unsigned char *)data_pointer);
62         total_sum &= 0xFFFF; // Ensure that the sum is within 16 bits
63     }
64     // Fold 32-bit sum to 16 bits
65     while (total_sum >> 16)
66         total_sum = (total_sum & 0xFFFF) + (total_sum >> 16);
67
68     // If the sum is all ones, the checksum is correct
69     int result = total_sum == 0xFFFF ? 1 : -1;
70     return result;
71 }
```

: Sender → send message (4)

: Got_Ack - 3710 (c)

8. הַנִּזְעָם כִּי־יֵרֶא כִּי־יֵלֹגֶד

סְרִירָה אַקְטּוֹבֶרְגָּה מִלְּכָה-מִלְּכָה בְּעֵדָה מִלְּכָה, מִלְּכָה בְּעֵדָה מִלְּכָה

ନାଟ୍‌କର ମୂଲ୍ୟ ଦେଖି ପାଇଁ ୫୯ ମୁହଁ

בכזהו הולך, מכאן שטוחה, מכאן שטוחה.

לפנינו נתקה ביחס 0.231BK SO_ACY77INFO נערך, Socket 7ucc ס-ב נתקל

לפיכך אין לנו גיבוב data מ- \mathbb{R}^n ל- \mathbb{R}^m .

group timeout = 120s max with max 15.12s /group 1s /s 50 - ACTTIMEOUT
expires

flag CAPTURE מציין אם נתונים הועברו מצלול אחד לצלול אחר.

```
74 // Function that checks if Sender got ACK Packet
75 int got_ACK(int sockfd, struct sockaddr * from, socklen_t * fromlen) {
76     Packet buffer;
77     memset(&buffer, 0, sizeof(Packet));
78
79     ssize_t ACK = recvfrom(sockfd, &buffer, sizeof(Packet), 0, from, fromlen);
80
81     // setsockopt(SO_RCVTIMEO): Causes the receive operation to return with an error (-1 with errno set to EAGAIN or EWOULDBLOCK)
82     // if the timeout expires before data is received. Need to check for this error condition explicitly.
83     if (ACK == -1) {
84         if (errno == EAGAIN || errno == EWOULDBLOCK) {
85             perror("ACK packet failed to be received: the timeout expires before data is received.");
86             return -10;
87         } else {
88             perror("ACK packet failed to be received.");
89             close(sockfd);
90             return -1;
91         }
92     }
93     if (ACK == 0) {
94         perror("The peer has closed the connection");
95         return 0;
96     }
97     if (buffer.Flag == 'A') {
98         return 1; // Successfully received ACK
99     }
100 }
101
102 }
```

: hand shake /'hændʃeɪk/ ②

הנתקה מהתפקידים הדרושים במקומות העבודה, נסגרה תקופה של כ-23 שנים.

בכלי רשת מושג R(S) יתאפשר לאריך את תקופת TCP (לפחות פי 2).

• נִרְאָה מִבְּרִיאָה שֶׁבְּרִיאָה S(SYN) בְּזַעֲמָה כְּלֹבֶד הַשְׁמָרָה S אֲנָשָׁה. גַּם כֵּן יְהוָה יְהוָה

data types for the model

timeout → for better data selection performance

הנתקן אם RN ACK מתקבל מלקוח או מSERVER. אם לא מתקבל ACK מלקוח, נזקן את הלקוח.

הנתק –> סינון SYN נשלח מארון הלקוח לארון השרת
הנתק –> סינון ACK מארון השרת לארון הלקוח

A N ACK

```

104 // Creating handshake between two peers (Sender send SYN message, Rec received the SYN message & send ACK, Sender receive ACK)
105 // An image of the TCP connect() function that will ensure a handshake
106 int handshake_connect(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr* respond_server, socklen_t * respond_
107     printf("Sending request for RDUP connection\n");
108
109     // Send SYN message - send just flag without a real data
110     Packet SYN;
111     memset(&SYN, 0, sizeof(SYN)); // ensure struct is clean
112     SYN.Length = 0;
113     SYN.Flag = 'S';
114
115     int attempts = 0;
116     // The function wait for an acknowledgment packet, if it didnt receive any, retransmits the packet till default max_attempts
117     // This for preventing infinite loops in case of persistent failures
118     while (attempts < MAX_RETRANSMISSION_ATTEMPTS) {
119         // send SYN packet
120         ssize_t size_SYN = sendto(sockfd,&SYN,sizeof(Packet), 0, serv_addr,addrlen);
121         if (size_SYN<0) {
122             perror("SYN packet failed to be send");
123             close(sockfd);
124             return -1;
125         }
126
127         int ACK_Status = got_ACK (sockfd, respond_server, respond_server_len);
128         // Check if no timeout, out of the loop
129         if (ACK_Status != -10) {
130             if (ACK_Status == 1)
131                 printf("Got ACK from Receiver.\n");
132             return ACK_Status;
133         }
134         attempts++;
135         printf("Retransmission attempt %d\n", attempts);
136     }
137
138     // If maximum retransmission attempts reached
139     printf("Maximum retransmission attempts reached. Handshake failed.\n");
140     return -1;
141 }
```

:Send Data –> סינון ②

על מנת לשלוח נתונים מארון הלקוח לארון השרת יש לבצע את הפעולות הבאות:
 1. שולח נתונים (Data) עם FLAG D (Data)
 2. אSEG מארון הלקוח ישלחChecksum ו-ACK (ACK)
 3. מארון השרת ישלחChecksum ו-ACK (ACK)
 4. מארון הלקוח ישלחChecksum ו-ACK (ACK)

```

143 // Sending data to the peer (after receive ACK packet)
144 int rudp_send(int sockfd, const void*msg, int len, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr* respond_server, soc
145     // Send data message
146     Packet data;
147     memset(&data, 0, sizeof(data)); // ensure struct is clean
148     data.Length = len;
149     data.Flag = 'D';
150     strcpy(data.Content, msg);
151     data.Checksum = calculate_checksum(data.Content, data.Length);
152
153     int attempts = 0;
154
155     // The function wait for an acknowledgment packet, if it didnt receive any, retransmits the data till default max_attempts
156     // This for preventing infinite loops in case of persistent failures
157     while (attempts < MAX_RETRANSMISSION_ATTEMPTS) {
158         ssize_t dataSent = sendto(sockfd, &data, sizeof(Packet), 0, (const struct sockaddr *)serv_addr,addrlen);
159         if (dataSent<0) {
160             perror("Data packet failed to be send");
161             close(sockfd);
162             return -1;
163         }
164         int ACK_Status = got_ACK (sockfd, respond_server, respond_server_len);
165         // Check if no timeout, out of the loop (if ACK is 1, return size of dataSent)
166         if (ACK_Status == -1) {
167             return -1;
168         }
169         if (ACK_Status == 0) {
170             return 0;
171         }
172         if (ACK_Status == 1) {
173             return dataSent;
174         }
175         attempts++;
176         printf("Retransmission attempt %d\n", attempts);
177     }
178     // If maximum retransmission attempts reached
179     printf("Maximum retransmission attempts reached. Sending the data failed.\n");
180     return -1;
181 }
```

:Receive Data –> סינון ③

AUDP-close (2)

```
183 int rdup_close(int sockfd, struct sockaddr *serv_addr, socklen_t addrlen, struct sockaddr * respond_server, socklen_t * respond_server_len)
184 {
185     printf("Sending request for exit.\n");
186
187     // Send FIN message - send just flag without a real data
188     Packet FIN;
189     memset(&FIN, 0, sizeof(FIN)); // ensure struct is clean
190     strcpy(FIN.Content,"Exit");
191     FIN.Length = 0;
192     FIN.Flag = 'F';
193
194     int attempts = 0;
195     // The function wait for an acknowledgment packet, if it didnt receive any, retransmits the packet till default max_attempts
196     // This for preventing infinite loops in case of persistent failures
197     while (attempts < MAX_RETRANSMISSION_ATTEMPTS) {
198         // send FIN packet
199         ssize_t size_FIN = sendto(sockfd, &FIN, sizeof(Packet), 0, (const struct sockaddr *)serv_addr,addrlen);
200         if (size_FIN<=0) {
201             perror("FIN packet failed to be send");
202             close(sockfd);
203             return -1;
204         }
205
206         int ACK_Status = got_ACK (sockfd, respond_server, respond_server_len);
207         // Check if no timeout, out of the loop
208         if (ACK_Status != -10) {
209             if (ACK_Status == 1) {
210                 printf("Got ACK from Receiver.\n");
211                 printf("Exit.\n");
212             }
213         }
214         attempts++;
215         printf("Retransmission attempt %d\n", attempts);
216     }
217
218     // If maximum retransmission attempts reached
219     printf("Maximum retransmission attempts reached. disconnect failed.\n");
220     return -1;
221 }
222
223 // *** Receiver's functions... ***
```

: Receiver → played / ɪ'pɪld / ⑤

. Send-Ack (IC)

```
225 // Function that send ACK packet to Sender
226 int send_ACK(int sockfd, struct sockaddr * dest, socklen_t destlen) {
227     // Send ACK message - send just flag without a real data
228     Packet ACK;
229     memset(&ACK, 0, sizeof(ACK)); // ensure struct is clean
230     ACK.Length = 0;
231     ACK.Flag = 'A';
232
233     ssize_t size_ACK = sendto(sockfd, &ACK, sizeof(Packet), 0, (const struct sockaddr *)dest, destlen);
234     if (size_ACK<=0) {
235         perror("ACK packet failed to be send");
236         close(sockfd);
237         return -1;
238     }
239     return 1;
240 }
```

: Find P_Receive

השלוחן מקבל מסר בודק את מסר הלקוח R ובודק אם מסר IS מוגדר כ-

רף, פיר checksum יוכן בפונקציית receive checksum י

הנשלח מאריך ומייד שדרט של ACK משלו.

הנשלח מאריך ומייד שדרט של ACK משלו.
(הספירה רוחנית מ-5 עד 12, 'f' ל-12 ו-ACK)

```
241 // Function to receive any type of packet,  
242 // after receive successfully, send ACK packet to the Sender  
243 int rudp_receive(int sockfd, struct sockaddr * Sender_addr, socklen_t * Sender_len) {  
244     Packet buffer;  
245     memset(&buffer, 0, sizeof(Packet));  
246  
247     ssize_t rec_size = recvfrom(sockfd, &buffer, sizeof(Packet), 0, Sender_addr, Sender_len);  
248     if (rec_size<0) {  
249         perror("packet failed to be received");  
250         close(sockfd);  
251         return -1;  
252     }  
253 }
```

```
54 // Connection closed  
55 if (rec_size == 0) {  
56     printf("Connection closed by peer.\n");  
57     close(sockfd);  
58     return 0;  
59 }  
60  
61 // Ensure that the buffer is null-terminated, no matter what message was received.  
62 // This is important to avoid SEGFAULTs when printing the buffer.  
63 if (buffer.Content[MAX_BUFFER_SIZE - 1] != '\0')  
64     buffer.Content[MAX_BUFFER_SIZE - 1] = '\0';  
65  
66 int ACK = 0;  
67 // Got a SYN packet (Sender wants to connect)  
68 if(buffer.Flag == 'S') {  
69     printf("Connection request received, sending ACK.\n");  
70     ACK = send_ACK(sockfd, Sender_addr,*Sender_len);  
71     if (ACK == -1){  
72         perror("packet ACK failed to be Send for start connection");  
73         close(sockfd);  
74         return -1;  
75     }  
76     printf("Sender connected, beginning to receive file...\n");  
77     return 1;  
78 }  
79 // Got a Data packet  
80 if(buffer.Flag == 'D') {  
81     int val_checksum = verify_checksum(&buffer,buffer.Length);  
82     if (val_checksum == -1) {  
83         perror("Checksum is not valid");  
84         close(sockfd);  
85         return -1;  
86     } else {  
87         ACK = send_ACK(sockfd, Sender_addr,*Sender_len);  
88         if (ACK == -1){  
89             perror("packet ACK failed to be Send data");  
90             close(sockfd);  
91             return -1;  
92         }  
93     }  
94 }
```

```
297 // Got a FIN packet (Sender wants to close connection)  
298 if(buffer.Flag == 'F') {  
299     printf("Sender sent exit message.\n");  
300     ACK = send_ACK(sockfd, Sender_addr,*Sender_len);  
301     if (ACK == -1){  
302         perror("packet ACK failed to be Send for start connection");  
303         close(sockfd);  
304         return -1;  
305     }  
306     printf("ACK sent.\n");  
307     memset(buffer.Content, 0, MAX_BUFFER_SIZE);  
308     return 2;  
309 }  
310  
311  
312 return 0;  
313 }
```

AhDP - Seh der c:

קווין ג'קסון זה ה-סינגל - בנט אודיס רנדולף גאנט גראן. נזקן עוזי ק' האניאר

RUDP-API ≈ UDP-התקנה ב-IP-socket > TCP-socket

```
100
101 // *** Part B: Create UDP socket between Sender - Receiver ***
102
103 printf("Starting Sender...\\n");
104
105 //creat socket
106 int _sockfd = rudp_socket();
107
```

בפועל נסן רצון מזמן המתוקף בTIMEOUT. במקרה שבו לא קיבל תגובה מACK מS, יתבצע ניסיון מחדש.

```
108 // Define a timeout for Sender to receive ACK  
109 // According to the article you published, the valid range for Timeout is 0 to 65536 (in ms), so I chose 10000ms randomly  
110 struct timeval timeout = {0,65530};  
111
```

struct sockaddr_in הינה ישות Receiver ב recvfrom()

3177 first 10⁴ 10⁵ 10⁶ headers ⇒ 2016 = 2016

לעומת זה, מושג זה מוגדר כמי שלב (self - himself) או מושג אחד (one).

כג' ינואר 2019 | IPV4 | הדריך לנטול IPV6

נגישותם של מנגנונים יוצרים מנגנון אחד אחד.

הנובע מINET_PTONL ש带回ו לנו מ-IPWIZ String ו-IPADDRESS י带回ו לנו מ-ICMPV6_NEXTHOP.

```
111 //create receiver address struct  
112 struct sockaddr_in server_address; // Struct sockaddr_in is defined in the <netinet/in.h> header file.  
113 memset(&server_address, 0, sizeof(server_address));
```

```
115 server_address.sin_family = AF_INET;
116 server_address.sin_port = htons(port); // htons() function ensures that the port number is properly formatted for network communication
117 int rval = inet_pton(AF_INET, ip_address, &server_address.sin_addr); // inet_pton() is a function that converts an IPv4 address in dotted decimal notation to a binary format
118 if (rval <= 0){
119     close(_sockfd);
120     free(data);
121     printf("inet_pton() failed");
122     return -1;
123 }
124
125 // The variable to store the server's address length.
126 socklen_t server_len = sizeof(server_address);
127
```

לעומת TCP, ב-UDP אין מנגנון של חיבור בין הלקוח והשרת, connect/accept/join/leave. UDP משליטה על התקשורת באמצעות ACK ו-NAK. ACK נשלח לאחר קבלת Paket, NAU נשלח לאחרtimeout. ACK ו-NAK מושכים ללקוח. R-req מושך לשרת. ACK מושך ללקוח. ACK ו-NAK מושכים ללקוח.

(፩፻፻፻) አማካይ

בנוסף ל-`SO_RCVTIMEO`, ניתן לרשום ב-`SO_RCVTIMEO` את ערך `timeo.tmrval` ו-`timeo.tmrhi` (ב-`struct timeval`) כזמן המתוקף למשך קיבול נתונים.

```
126 // Since in UDP there is no connection, there isn't a guarantee that the server is up and running.  
127 // Therefore, we set a timeout for the recvfrom() call using the setsockopt function.  
128 // If the server (RUDP_Receiver) does not respond within the timeout, the client (RUDP_Sender) will drop.  
129 if (setsockopt(_sockfd, SOL_SOCKET, SO_RCVTIMEO, (char*)&timeout, sizeof(timeout)) == -1) {  
130     perror("setsockopt() failed");  
131     close(_sockfd);  
132     free(data);  
133     return -1;  
134 }  
135  
136 }
```

לפונקציית Response מוסמך, Server י反响 עליה בפונקציית `Send` כ-`ACK` ו-`NACK`.
אם לא קיבל תווים מלקוח, Server י反响 עליה בפונקציית `Send` כ-`ACK` ו-`NACK`.
אם קיבל תווים מלקוח, Server י反响 עליהם בפונקציית `Send` כ-`ACK`.

```
139 // The variable to store the server's address, that responded to the message.  
140 // it is required by the recvfrom function (that included in rudp_handshake function)  
141 // Note that the target server might be different from the server that responded to the message.  
142 struct sockaddr_in respond_server;  
143  
144 // The variable to store the respond server's address length.  
145 socklen_t respond_server_len = sizeof(respond_server);  
146  
147 memset(&respond_server, 0, sizeof(respond_server));
```

```
// Ensure theres a handshake between Sender and Receiver
int handshake = handshake_connect(_sockfd, (struct sockaddr*)&server_address, server_len, (struct sockaddr*)&respond_server, &response);
if (handshake != 1) {
    perror("Handshake failed");
    close(_sockfd);
    free(data);
    return -1;
}

printf("Receiver connected, beginning to send file...\\n");
```

רוכסן בדרכו נשלח ACK מהלך זיהוי סדרה. נספח מודול צבאי.

• 10 miles per second = 165m/s
• Time since : 2 + 2 sec

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

הנתקן בראטון נספחים ל-
הנתקן בראטון נספחים ל-

הפריטים - הם הרוחניים הקיימים כמי יתגלו בדורות עתידיים. מנגנון ה传递 של התרבות מושפע מהתפקידים הנדרשים בתרבות.

```

// Ask user for decision
printf("Do you want to send the file again? (yes/no): ");

// Get only valid answers - yes or no
int valid_char = 0;
while (valid_char==0) {
    char decision[4];
    scanf("%s",decision);
    if(strcmp(decision, "no")==0 || strcmp(decision, "yes")==0) {
        ssize_t send_decision = sendto(_sockfd, decision, strlen(decision),0, (struct sockaddr*)&server_address, server_len);
        if(send_decision < 0){
            perror("Error sending decision");
            close(_sockfd);
            free(data);
            exit(EXIT_FAILURE);
        }
        if (strcmp(decision, "no")==0) {
            send_again = 0; // If decision is not "yes", exit loop
        }
        valid_char=1;
    } else {
        printf("not valid answer, try again\n");
    }
}

```

היכן שעשוי לבקש מהמשתמש אם הוא רוצה לשלוח את הקובץ מחדש או לא. במקרה שהמשתמש י選擇 'yes' או 'no' יתבצע פעולה בהתאם. אם המשתמש בחר ב'no' י跳出 את הLOOP ויפתח קובץ חדש, אם בחר ב'yes' יתבצע פעולה רקורסיבית ותשוב לLOOP.

: סטטוס

```

220 // *** Part E+ F: Send exit message to the receiver + Close the RUDP connection ***
221 int close_connection = rdup_close(_sockfd, (struct sockaddr*)&server_address, server_len, (struct sockaddr*)&respond_server, &response);
222 if (close_connection != 1) {
223     perror("disconnect failed");
224     close(_sockfd);
225     free(data);
226     return -1;
227 }
228 close(_sockfd);
229 free(data);
230 remove(file_path); // Remove temporary file
231
232 // *** Part G: Exit ***
233 return 0;
234
235 }
236

```

היכן שעשוי לשלוח מsg של סיום תקשורת למשתמש, ולחסום את ה-Socket. לאחר מכן, יCLOSE את ה-Socket ויפריט.

RUDP RECEIVER.C :

המודול מקבל מטודים מ-UDP SERVER וsetserv() מגדיר את הכתובת והפורט של SERVER. מטרת המודול היא לקבל מטודים מ-UDP SENDER וsetsender() מגדיר את הכתובת והפורט של SERVER. מטרת המודול היא לקבל מטודים מ-UDP SENDER וsetsender() מגדיר את הכתובת והפורט של SERVER.

```

1 #include "RUDP_API.h"
2 #include "LinkedList.h"
3
4 #define MAX_BUFFER_SIZE 2048
5
6 // a function to calculate milliseconds
7 double get_time_in_milliseconds(struct timeval start, struct timeval end) {
8     return (double)(end.tv_sec - start.tv_sec) * 1000.0 + (double)(end.tv_usec - start.tv_usec) / 1000.0;
9 }
10
11 int main(int argc, char *argv[]) {
12
13     // *** Pre-Parts : Get from the user the command from terminal ***
14
15     if (argc != 3) {           // ./RUDP_Receiver -p 12345
16         fprintf(stderr, "Please provide the correct usage for the program: %s -ip IP -p PORT -algo ALGO <FILE_PATH>\n", argv[0]);
17         exit(EXIT_FAILURE);
18     }
19
20     // Extract command-line arguments
21     int port = 0;
22
23     // Process command-line arguments
24     for (int i = 1; i < argc; i++) {
25         if (strcmp(argv[i], "-p") == 0 && i + 1 < argc) {
26             port = atoi(argv[i + 1]); // Convert a string representing an integer (ASCII string) to an integer value.
27             i++;
28         }
29     }
30
31     // Check if required arguments are provided
32     if (port == 0) {
33         fprintf(stderr, "Invalid arguments. Please provide the correct usage.\n");
34         exit(EXIT_FAILURE);
35     }
36 }
```

מונע בזבז זמן
mil ~

```

37 // *** Part A: Create a UDP connection between the Receiver and the Sender ***
38 printf("Starting Receiver...\n");
39
40 //Create socket
41 int listeningSocket = rudp_socket(); // Create a UDP socket for IPv4
42
43 // The variable to store the socket option for reusing the server's address.
44 int opt = 1;
45
46 // Set the socket option to reuse the server's address.
47 // This is useful to avoid the "Address already in use" error message when restarting the server.
48 if (setsockopt(listeningSocket, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt)) < 0)
49 {
50     perror("Error settingsockopt");
51     close(listeningSocket);
52     return 1;
53 }
54
55 // Define Receiver's values
56 struct sockaddr_in server_address; // Struct sockaddr_in is defined in the <netinet/in.h> header file.
57 memset(&server_address, 0, sizeof(server_address));
58
59 server_address.sin_family = AF_INET;
60 server_address.sin_addr.s_addr = INADDR_ANY; // The server will listen on all available network interfaces.
61 server_address.sin_port = htons(port); // htons() function ensures that the port number is properly formatted for network communication
62
63 // Bind the socket to the specified address and port
64 if (bind(listeningSocket, (struct sockaddr *)&server_address, sizeof(server_address)) == -1) {
65     perror("Bind failed");
66     close(listeningSocket);
67     exit(EXIT_FAILURE);
68 }
```

ריצויו יוכנה אוטומטית
ポート自動化

:טיפסן

(R) יוזם מנגנון בידור בין ה-3 ביטים ביחד () —

תוקן נתון הינו ערך ה-0. כראוי IP הילוגיק ה-

ה-1 הוא יוזם מנגנון בידור בין ה-3 ביטים ביחד () —

ה-2 הוא יוזם מנגנון בידור בין ה-3 ביטים ביחד () —

לע"ז מילא בפונט העברית את המילה bind (bind) במשמעותה של גיבוב.

* קב' נג', הילן היה נושא למחקרים (bihl) של סנדר גוטסברג (Gottschanger) במכון למדעי הרוח של אוניברסיטת תל אביב.

. Sender ה בר זלען: זי גהן

```
69 // *** Part B: Get a connection from the sender ***
70
71 // Define Sender
72 struct sockaddr_in client_address;
73 socklen_t client_address_len = sizeof(client_address);
```

- . יונת ביב-32 סדרת מונט קולברג

נוצר נבון ביחס לשליטה על הרים.

```
77 // Accept a connection
78 printf("Waiting for RUDP connection...\n");
79 ssize_t recvSYN = rudp_receive(listeningSocket, (struct sockaddr *)&client_address, &client_address_len);
80 if (recvSYN<=0) {
81     exit(EXIT_FAILURE);
82 }
```

.../vs רְקָבָל הַגָּדָה רְסָבָה:

```
83
84     // *** Part C: Receive the file, measure the time it took to save ***
85
86     struct timeval start_time, end_time;
87     fileList* files = fileList_alloc(); // Start a new null list of files
88
```

תְּבוּ פָנָס = יְרֵאָה הַיּוֹתָה כְּלִילָה כְּלִילָה. תְּבוּ נָסָתָה כְּלִילָה כְּלִילָה.

```
88 // Receive the size of the file from the sender
89 unsigned int file_size;
90 ssize_t size_received = recvfrom(listeningSocket, &file_size, sizeof(file_size), 0, (struct sockaddr *)&client_address, &client_address_length);
91 if (size_received <= 0) {
92     perror("Error receiving file size");
93     close(listeningSocket);
94     exit(EXIT_FAILURE);
95 }
```

נוסף ב-
הנוסףrecvfrom() שנותן גודל ה-
recvfrom().
recvfrom() שנותן גודל ה-recvfrom().
recvfrom() שנותן גודל ה-recvfrom().

```
97
98     // Receive the data of the file from the sender
99     while (1)
100    {
101        int total_received = 0;
102        while(total_received<file_size) {
103            gettimeofday(&start_time,NULL);
104            ssize_t bytes_received = rudp_receive(listeningSocket, (struct sockaddr *)&client_address, &client_address_len);
105            if (bytes_received <= 0) {
106                exit(EXIT_FAILURE);
107            }
108            total_received+=bytes_received;
109        }
110    }
```

הנ'ן הינה גורם לנטול מילוי המטרים
לפניהם. מילוי המטרים נקבע על ידי
ההנ'ן הינה גורם לנטול מילוי המטרים
לפניהם. מילוי המטרים נקבע על ידי

לפניהם נקבעו שלושה מטרים: קיבולת דוחות (total_rbc), גודל דוחות (file_size) ומספר דוחות (num_rbc).
המטרה היא לחלק כל דוחות למספר קבוצות שווה גודל (bins), ולבצע סכום כל דוחות בקבוצה.

```
110
111     gettimeofday(&end_time, NULL); // Set the end time before measuring elapsed time
112     double elapsed_time = get_time_in_milliseconds(start_time, end_time);
113     fileList_insertLast (files,elapsed_time,file_size); // Create new file
114     printf("File transfer completed (%d bytes).\n",total_received);
115     printf("ACK sent.\n");
116     printf("Waiting for Sender response...\n");
117
```

לפניהם נסמן נווטרול (neutral) ו-
ב-טבלה 1 מוצגים נתונים עבורם.

ה-תבנית מוגדרת כ-**תבנית טרנספורמצייתית**.
ה-תבנית מוגדרת כ-**תבנית טרנספורמצייתית**.

בכדי לסייע לאפשרות אוטומטית ביצוע הוראות, ניתן לשים מנגנון `checkSum` ו-`timeout` ב-

• Sender - גן הגנה פיזית גוף גוף

```
118 // ** Part D: Wait for Sender Response **
119 char decision[4];
120 ssize_t decision_rec = recvfrom(listeningSocket, decision, sizeof(decision), 0, (struct sockaddr *)&client_address, &client_add
121 if (decision_rec <= 0) {
122     perror("Error receiving decision");
123     close(listeningSocket);
124     exit(EXIT_FAILURE);
125 }
126 // 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
127 if(decision[0]=='n') {
128     memset(decision, 0, strlen(decision));
129     break;
130 }
131 }
```

רְגֵבֶן אַלְמָנָה וְסִינָה, רְגֵבֶן סִינָה קְרֹבָה.

שְׁלֵמָה

```
131
132
133     // Get Exit message from Sender
134     ssize_t Exit = rudp_receive(listeningSocket, (struct sockaddr *)&client_address, &client_address_len);
135     if (Exit <= 0) {
136         perror("Error receiving Exit message");
137         exit(EXIT_FAILURE);
138     }
139
140     // ** Part E: Print out statistics **
```

```
133 // Get Exit message from Sender
134 ssize_t Exit = rudp_receive(listeningSocket, (struct sockaddr *)&client_address, &client_address_len);
135     if (Exit <= 0) {
136         perror("Error receiving Exit message");
137         exit(EXIT_FAILURE);
138     }
139
140 // ** Part E: Print out statistics **
141
142 printf("-----\n");
143 printf("- * Statistics * -\n");
144 fileList_print(files); //fucntion that prints time and speed for each file
145
146 // ** Part F: Calculate the average time and the total average bandwidth **
147
148 fileList_AverageT_print(files);
149 fileList_AverageBT_print(files);
150 printf("-----\n");
151 close(listeningSocket);
152 fileList_free(files);
153
154 // ** Part G: Exit **
155
156 printf("Receicer end.\n");
157 return 0;
158 }
```

הטבות מילויים בפונקציית `close()` בsender
הפעלה של `close()` על sockfd מושג[הנימוקים](http://www.cs.tau.ac.il/~moriel/courses/OS/2013/lectures/06/06.html)
הפעלה של `close()` על sockfd מושג[הנימוקים](http://www.cs.tau.ac.il/~moriel/courses/OS/2013/lectures/06/06.html)

Makefile:

```
1 CC=gcc
2 FLAGS=-Wall -g
3
4 all: RUDP_Sender RUDP_Receiver
5
6 RUDP_Sender: RUDP_Sender.o RUDP_API.o
7     $(CC) $(FLAGS) -o RUDP_Sender RUDP_Sender.o RUDP_API.o
8
9 RUDP_Sender.o: RUDP_Sender.c RUDP_API.h
10    $(CC) $(FLAGS) -c RUDP_Sender.c
11
12 RUDP_Receiver: RUDP_Receiver.o RUDP_API.o LinkedList.o
13     $(CC) $(FLAGS) -o RUDP_Receiver RUDP_Receiver.o RUDP_API.o LinkedList.o
14
15 RUDP_Receiver.o: RUDP_Receiver.c LinkedList.h RUDP_API.h
16     $(CC) $(FLAGS) -c RUDP_Receiver.c
17
18 RUDP_API.o: RUDP_API.c RUDP_API.h
19     $(CC) $(FLAGS) -c RUDP_API.c
20
21 LinkedList.o: LinkedList.c LinkedList.h
22     $(CC) $(FLAGS) -c LinkedList.c
23
24 .PHONY: clean
25
26 clean:
27     rm -f *.o *txt RUDP_Sender RUDP_Receiver
```

niksiz nesn : 2 pın

. Receiver d'ına Sender → -ic 1234 nesn 112500 nesn 112311 1131
. 11-1825 s 112500 -ic 112500

```
yairco@yairco:~/reshatot/EX3/B$ ./RUDP_Sender -ip 127.0.0.1 -p 1234
Starting Sender...
Sending request for RDUP connection
Got ACK from Receiver.
Receiver connected, beginning to send file...
Send the file...
Got ACK from Receiver.
a file with 4262050 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
Got ACK from Receiver.
a file with 4262050 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
Got ACK from Receiver.
a file with 4262050 bytes has been sent successfully
Do you want to send the file again? (yes/no): yes
Send the file...
Got ACK from Receiver.
a file with 4262050 bytes has been sent successfully
Do you want to send the file again? (yes/no): no
Sending request for exit.
Got ACK from Receiver.
Exit.
```

```
yairco@yairco:~/reshatot/EX3/B$ make all
gcc -Wall -g -c RUDP_Sender.c
gcc -Wall -g -c RUDP_API.c
gcc -Wall -g -o RUDP_Sender RUDP_Sender.o RUDP_API.o
gcc -Wall -g -c RUDP_Receiver.c
gcc -Wall -g -c LinkedList.c
gcc -Wall -g -o RUDP_Receiver RUDP_Receiver.o RUDP_API.o LinkedList.o
yairco@yairco:~/reshatot/EX3/B$ ./RUDP_Receiver -p 1234
Starting Receiver...
Waiting for RUDP connection...
Connection request received, sending ACK.
Sender connected, beginning to receive file...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
File transfer completed (4262050 bytes).
ACK sent.
Waiting for Sender response...
Sender sent exit message.
ACK sent.
-----
- * Statistics *
- Run #1 Data: Time =0.05ms; Speed=76678.46MB/s
- Run #2 Data: Time =0.05ms; Speed=82937.92MB/s
- Run #3 Data: Time =0.06ms; Speed=71297.51MB/s
- Run #4 Data: Time =0.06ms; Speed=71297.51MB/s
- Run #5 Data: Time =0.03ms; Speed=131095.42MB/s
- Average time: 0.05ms
- Average bandwidth: 86661.37MB/s
-----
Receiver end.
```

• 1. גְּדוֹלָה יְהוָה כָּל־עַמִּים פָּתַח תְּבִנָּה בְּבֵית־יְהוָה

16% : 3 ရွေ့

RUDP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len=2054
2	0.0002659821	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len=2054
3	0.0003434317	127.0.0.1	127.0.0.1	UDP	46	43131 - 1234 Len=2
4	0.000419765	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len=2054
5	0.0004523262	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len=2054
6	0.000489246	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len=2054
7	0.000544394	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len=2054
8	0.0005757536	127.0.0.1	127.0.0.1	UDP	2096	42321 - 1234 Len=2054

1

Sender = nijδ
Receiver = pje

. הלקוח שולח לשרת בקשה לפתיחת קשר [SYN]

. הרשות מתקבלת ההודעה ושולחים בתגובה הודעת אישור קבלה ואישור פתיחת קשר מצדיהם שלוח [SYN, ACK]

• የጊዜ ተከራካሪ አይነት ስርቃት የሚያስፈልግ ይችላል፡፡

bytes de מילון Receiver f nises f.nrr sender n f.nrr bytes f.e mode 28 הינה f.nrr bytes f.e receiver n , f.nrr n

The figure shows a Wireshark capture of UDP traffic on interface lo. A yellow bracket highlights four consecutive UDP packets (4154, 8305, 11525, 15676) from 127.0.0.1 to 127.0.0.1. A yellow callout points to the third byte of the fourth packet (Data bytes 3-4), which contains the ASCII value 'yes'. The packet details pane shows the following:

No.	Time	Source	Destination	Protocol	Length	Info
4154	7.129763043	127.0.0.1	127.0.0.1	UDP	45	43131 - 1234 Len=3
8305	10.522343831	127.0.0.1	127.0.0.1	UDP	45	43131 - 1234 Len=3
11525	12.164431789	127.0.0.1	127.0.0.1	UDP	45	43131 - 1234 Len=3
15676	14.626036988	127.0.0.1	127.0.0.1	UDP	45	43131 - 1234 Len=3

Frame 4154: 45 bytes on wire (360 bits), 45 bytes captured (360 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: 43131, Dst Port: 1234
Data (3 bytes)

0000 00 00 00 00 00 00 00 00 00 00 00 00 45 00E
0010 00 1f ce b9 40 00 40 11 6e 12 7f 00 00 01 7f 00@ @ n.....
0020 00 01 a8 7b 04 d2 00 0b fe 1e 79 65 73{.....yes

בנין גירויים משלים Sender של Receiver על מנת
לענות על גירויים משלים Receiver של Sender.

RUDP.pcapng								
File	Edit	View	Go	Capture	Analyze	Statistics	Telephony	Wireless
No.	Time	Source	Destination	Protocol	Length	Info		
19812	14.740942912	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19813	14.740954721	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19814	14.740967142	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19815	14.740980675	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19816	14.740992711	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19817	14.741012484	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19818	14.741027138	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19819	14.741042087	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19820	14.741055761	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19821	14.741070872	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19822	14.741084904	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19823	14.741099863	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19824	14.741114657	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19825	14.741130351	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19826	14.741145147	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		
19827	16.983772634	127.0.0.1	127.0.0.1	UDP	44	43131 - 1234 Len:2		
19828	16.984657099	127.0.0.1	127.0.0.1	UDP	2096	43131 - 1234 Len:2054		
19829	16.984158352	127.0.0.1	127.0.0.1	UDP	2096	1234 - 43131 Len:2054		

```
> Frame 19827: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 43131, Dst Port: 1234
Data (2 bytes)
```

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 E
0010 00 1e 13 84 40 00 40 11 29 49 7f 00 00 00 01 7f 00 . @@ )I
0020 00 01 8b 7b 0d d8 00 fa fe 1d 6e 6f { . no
```

הReceiver ימוך נספחים ל-
ה-Receiver.

RUDP.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
19812	14.749942612	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19813	14.749954721	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19814	14.749967142	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19815	14.74998675	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19816	14.749992711	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19817	14.7410212484	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19818	14.741027135	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19819	14.741042087	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19820	14.741055762	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19821	14.741070872	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19822	14.741084994	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19823	14.741099868	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19824	14.741114657	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19825	14.741130353	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19826	14.741145147	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054
19827	16.983772634	127.0.0.1	127.0.0.1	UDP	44	43131 .. 1234 Len=2
19828	16.984057090	127.0.0.1	127.0.0.1	UDP	2096	43131 .. 1234 Len=2054
19829	16.984158352	127.0.0.1	127.0.0.1	UDP	2096	1234 .. 43131 Len=2054

Frame 19828: 2096 bytes on wire (16768 bits), 2096 bytes captured (16768 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: 43131, Dst Port: 1234

Data (2054 bytes)

$f_2 \rightarrow f = f_{FIN}$

אם יש לנו Sender,/no הגדרה מוכנהReceiver יתבצע.

RUDP.pcapng							
File	Edit	View	Capture	Analyze	Statistics	Telephone	
No.	Time	Source	Destination	Protocol	Length	Info	
19812	14.749454910	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19813	14.749464927	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19814	14.749861742	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19815	14.749898075	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19816	14.749905715	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19817	14.749913381	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19818	14.741027138	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19819	14.741024287	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19820	14.741024297	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19821	14.741078972	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19822	14.741084964	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19823	14.741114657	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19824	14.741114657	127.0.0.1	127.0.0.1	UDP	2996	1234 - 43131 Len:2994	
19825	14.741139352	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19826	14.741139352	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
19827	16.983772834	127.0.0.1	127.0.0.1	UDP	44	43131 - 1234 Len:2	
19828	16.983772834	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994	
L	19828	16.984053955	127.0.0.1	127.0.0.1	UDP	2996	43131 - 1234 Len:2994

ה- 8.15... E
ג'כיה ס.נ. ר'זבג כ-0 ג'כיה ס.נ. ר'זבג

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

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

הוֹדָעָה לְסֶגִירַת קְשָׁר עִם הַלְּקוֹחַ מִצְדֵּךְ הַשְּׁרָת [F I N , A C K]

בְּכָל תְּמִזְמָרָה תְּפִלָּה כְּלָמָדָה:

