

מילה 3 ברשותות תקשורת

מגישיים –

שלומי זכרייה - 315141242

יסמין כהן - 212733836

דבר ראשון נסביר בדיק מה התוכנית עשו בכל שלב. זאת אומרת נסביר על הקוד שכתבנו ואיך צריך להפעיל אותו.

נתחליל ב-Sender

דבר ראשון נגידר את ה-port של server (receivern) שאיתו נרצה לתקשר, אנחנו הגדכנו פורט 9999. אקרואי שהחרנו שיהיה- 9999. דבר שני נגידר IP -server-, ומכוון שאנו כtabנו שתי תוכניות שרצות על המחשב שלנו בחרונו בIP - 127.0.0.1 . דבר שלישי נגידר את 4 הספרות האחורונות של הת"ז שלנו כדי שנוכל בעתיד להשתמש בוואך שלו בתור סימן שנקבל מה הreceivern שהקובץ שלנו התקבל.

```
#define SERVER_PORT 9999  
#define SERVER_IP_ADDRESS "127.0.0.1"  
#define ID1 1242  
#define ID2 3836
```

נפתח file שאנו יצרנו במחשב, שנקרא 1mb, באמצעות הפונקציה () fopen ונדיר אותו במצב קרייה "r".

```
FILE *fptr; // define pointer to a file type  
fptr = fopen("1mb.txt","r"); // open the file "1mb.txt" , "r" - read
```

פה יצרנו פוינטרים שצביעים על המיקום בקובץ. (זה יעוז לנו אחר כך בשילוח הקובץ).

```
fseek(fptr,0,SEEK_END); // Move the pointer to the end of the file  
long fileSize = ftell(fptr); // Size of the file  
fseek(fptr,0,SEEK_SET); // Move the pointer to the start of the file  
printf("File half size %ld\n",fileSize/2);
```

נדיר את ה socket של ה sender כ-:

AF_INET - מגדיר את הרשות מסוג TCP , IPv4 CHASSIS- SOCK_STREAM ו-0 מגדיר גם את החיבור מסוג TCP.

```
int sockfd = -1;  
if((sockfd = socket(AF_INET , SOCK_STREAM , 0 )) == -1) { // create file descriptor socket  
    perror("Could not create file descriptor socket ");  
    return -1;  
}
```

אנו נגידר בו struct sockaddrinet שהוא יכול לנו את פרטי ה receiver , שאל הכתובת שלו אנו נשלח את הקובץ.

מאתחלים את ה struct עם פונקציית () memset שממלאת את כל הגדל של ה struct באפסים.

יש בתוך ה struct כמה קטגוריות שצרכי למלא: server_port htons נכניס את ה-port של ה server family_AF_INET נכניס את AF_INET SHA-4, בקטgoriyת htons נכניס את ה-port שבודקת וממיר את ה-port להיות כתוב בצורה Big Indian . במאיצעות הפונקציה () htons שבודקת וממיר את ה-port להיות כתוב בצורה הנכונה.

במאיצעות הפונקציה () inet_pton אנו נמיר את הכתובת server זו במקום SHA יהיה כתוב ב-Char ועם

נקודות בין המספרים, זה ממיר למספר בינארי, ולאחר מכן נכניס לקטגוריה ב-`struct` של `sin_addr`.

```
struct sockaddr_in serveraddress;
memset(&serveraddress,0,sizeof(serveraddress)); // reset serveraddress
serveraddress.sin_family = AF_INET; // define address type(IPV4)
serveraddress.sin_port = htons(SERVER_PORT); // convert server port to network byte order

int rval = inet_pton(AF_INET,(const char*)SERVER_IP_ADDRESS,&serveraddress.sin_addr); // Convert IP Address: binary/decimal representation to network bytes order
```

לאחר האתוליטם, אנחנו מבצעים התחברות באמצעות פונקציית `connection()` שמאפשרת חיבור שייהי כמו צינור ישיר אל ה `receiver`. בזכות שאנו בחרנו בחיבור מסוג TCP.

מכניסים לפונקציית החיבור את הפרמטרים הבאים: את ה-`socket`, `sender`, את ה-`struct` שמכיל את פרטי כתובת ה `receiver` שאיתו נרצה לעשות את החיבור, ובגלל שאנו בuft C נרצה גם לדעת את גודל ה `struct` כדי לדעת כמה מקום נרצה לפנות בזיכרון.

```
printf("Connected .... \n");

int connection = connect(sockfd, (struct sockaddr*)&serveraddress, sizeof(serveraddress)); // Connect to sockfd
```

לאחר מכן נפתח לולאת (1) שהוא בעצם ממשיכה תמיד לזרע בלולאה אינסופית. נרצה לעשות את זה כדי שה ימשיך לשולח את הקובץ תמיד כברירת מחדל באמצעות האות אישור `Y` והיציאה מהלולאת שליחת תהיה כאשר האות `N` תלחץ ובמצב זהה נעשה ישר נעשה `break` ונמצא בתוך הלולאה נוכל לשולח את הקובץ.

בהתחלת חישוב שנאתחל את הפוינטר שמצוין על המיקום הנוכחי, במקרה הזה בהתחלה הוא יצביע על תחילת הקובץ (באמצעות הפונקציה `fseek()`).

```
while(1){ // while loop to resend the file

    int bytesSent = 0; // The bytes the sender send
    int bytesRecv = 0; // The bytes the receiver send
    fseek(fptr,0,SEEK_SET); // Move the pointer to the start of the file
```

באמצעות הפונקציה `setsockopt()` נשנה את אלגוריתם ה-CC. נגדיר את ה-`socket` שלנו מסוג חיבור TCP ושם האלגוריתם יקרא `cubic`, נסיף כמובן גם את גודל השם כדי לדעת כמה מקום לפנות בזיכרון.

```
if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCcub, strlen(CCcub)) == -1)
{
    perror("setsockopt");
    return -1;
}
```

נפתח לולאה שמטטרת היא לקרוא מהקובץ ולשלוח עד לחץ הקובץ. בפונקציה `fread()` קוראים לפי בתים את הקובץ שנמצא בז'ט `fptr` לתוך המערך `char bufferSend` שאתחלנו שתהייה בגודל חצי מהbytes של הקובץ (`bufferSend` קוראים לערך).

מה שהצחנו לקרוא לתוך הבארט אנו שולחים באמצעות הפונקציה `send()` (שמתקבלת את `socket` של `sender` מקבלת גם את הבארט המבוקש שבתוכו נמצאים הבטים שנרצה לשולח (הטקסט), וגם נציג את מספר הבטים הכלל שנרצה לשולח באמצעות המשתנה "connection" , מכון שהפונקציה `fread()` מוחירה כמה בתים נקראו).

```

while ((connection = fread(bufferSend, 1, sizeof(bufferSend), fptr)) > 0) { // Send the first half of the file
    bytesSent = send(sockfd, bufferSend, connection, 0);
    printf("LOOK HERE!!!! : %d \n", bytesSent);
    if (bytesSent == 0) {
        printf("peer has closed the TCP connection.\n");
    }
    else if (bytesSent == -1) {
        printf("send failed.\n");
    }
    else if (connection > bytesSent) {
        printf("sent only %ld bytes from the required %d.\n", strlen(bufferSend), bytesSent);
    }
    else {
        printf("Half one of the file send successfully sent (Total bytes sent : %d) .\n", bytesSent);
        break;
    }
}

```

לאחר השילחה נחכה לSIMN שakan כל השילחה שלנו התקבלה אצל receiver וначכה לקבל בפונקציית `recv()` את ה xor של הת"ז שהינו הסימן המוסכם שה receiver שלוח אם הכל אכן התקבל. receiver יודע שהכל התקבל בגלל שהוא ממחכה לקבל מספר בתים מהם בדיק במספר הבתים של חצי מהקובץ.

```

printf("Waiting For Authontication .. \n");

bytesRecv = recv(sockfd, &xorRecv, sizeof(int), 0); // Receive a authontication from the receiver

```

כאשר נרצה לשולח את החצי השני נרצה לשנות את האלגוריתם CC לאלגוריתם אחר מסוג `reno` ונשתמש כמו שהסבירנו לעלה בפונקציית `setsockopt()`

```

// Change CC
strcpy(CCren, "reno"); // change cc algorithm to rno

if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCren, strlen(CCren)) != 0) {
    perror("setsockopt error");
    return -1;
}

```

ושוב נצטרך להשתמש בפונקציית `fread()` ו`send` כדי לשולח את החצי השני.

ב`fread()` ממשיכים לקרוא מהקובץ ישר מהמקום בו עיצרנו פעם קודם קודמת ועד הסוף, את זה נוכל לעשות באמצעות הפעונט `lseek()` בפונקציה `fseek()`. כל פעם אנו קוראים כמה שאפשר שלוחים את הכמות שקרהנו, עד שמניעים למצב שבו קראנו עד לכמה בתים של חצי הקובץ ואז הלולאה נעצרת.

```

while ((connection = fread(bufferSend, 1, sizeof(bufferSend), fptr)) > 0) { // Send the second half of the file
    bytesSent = send(sockfd, bufferSend, connection, 0);
    if (bytesSent == 0) {
        printf("peer has closed the TCP connection.\n");
    }
    else if (bytesSent == -1) {
        printf("send failed.\n");
    }
    else if (connection > bytesSent) {
        printf("sent only %ld bytes from the required %d.\n", strlen(bufferSend), bytesSent);
    }
    else if(ftell(fptr) == filesize){
        printf("Half two of the file send seccessfully sent (Total bytes sent : %ld) .\n", ftell(fptr));
    }
}

```

לאוט אישור נקלט באמצעות פונקציית `Recv` שakan כל הקובץ הגיע אלינו. מכיוון רק נרצה לדעת האם המשתמש רצה לשולח שוב פעם. אנו ניתן לו אפשרות, אם ילחץ Y נשלח שוב (ונעליה שוב בלולאה) ואם ילחץ N נסגור את החיבור. אם לוחצים N אז פשוט נצא מהלולאה הראשית באמצעות `break`, הלולאה היא `while(1)` שרצה ושלחת מיד ונעצרת ורק אם לוחצים N.

כasher עושים N יוצאים מהלולה ומיד סוגרים את החיבור בין receiver ל sender וcmbon גם את socket הראשי שלנו.

כasher לוחצים Y מתחילה שוב את הלולאה מהתחלה שכוללת את השילחה של הקובץ מחדש.

```
char exit = '0'; // For Scanner(exit = 'Y' --> start while loop again , exit = 'N' --> break while loop)
int keepAlive; // Message for the connection
int check = -1; // Check if the receiver close the connection

recv(sockfd, &keepAlive, sizeof(int), 0); // received message from the receiver to keep connection
printf("Do you want to send the file again ? :) \n Yes- press 'Y' , No- press 'N' \n");

while(exit != 'Y' && exit != 'N'){ // while loop --> if the user type a char that is not Y or N send the ma
scanf(" %c" , &exit);
if(exit !='Y' && exit != 'N')
{
printf("Press only : Y - Send again || N - Close connection \n");
}
}
if(exit == 'N') // If exit = N --> close connection
{
printf("Closes the connection .. \n");
break;
}

send(sockfd, &keepAlive, sizeof(int), 0); // send message to the receiver to keep connection(confirm)
recv(sockfd, &keepAlive, sizeof(int), 0); // received a message from the receiver to check if the receiver get the confirm message from the sender
}

fclose(fptr);
close(sockfd);
```

nbsp; נסביר עבשו על Receiver

ונדר את השדות כך שהserver יהיה זהה ל-port server שהגדכנו ב-.Sender.

את הת"ז נדריר גם פה כי הוסף בניהם גם כמובן יהיה הסימן המוסכם פה שאכן הכל הגיע מהשולח.

וגם הגדרנו את גודל הקובץ בbytes.

(ונוכל לעשות את זה כיון שהתאמנו את הקוד שלנו שיטאים רק לקובץ הספציפי שלנו.)

```
#define SERVER_PORT 9999
#define ID1 1242
#define ID2 3836
#define fileSize 1048576
```

ונדר `sockaddr_in` מסוג (in). struct .
 אחד מכיל את פרטי הכתובת של sender והשני את של receiver.

בנוסף נדריר `timeval` structים מיוחדים לזמןיהם שהם מסוג (timeval) שבעל אחד מהם יש קטגוריות שמורות זמן. נשתמשanza במחצית הזמן לשמרות זמן של ההגעה של הקבצים.

`socklen_t` זה סוג משתנה השומר גודל של structים.

ונדר מערכת צ'ארים בגודל הקובץ ונקרא לו `bufferSend`.

(בגל שbyte זה אז אפשר ממש בקומות כהה להגדיר כי גודל הקובץ גם בbytes).

במערכת הcz'ארים של CCub CCrencCCCub נשמרו בעמידה את השמות של האלגוריתמים של CC.

```
struct sockaddr_in serverAddress ,clientAddress;
struct timeval timeStart, timeEnd;
socklen_t clientAddressLen = sizeof(clientAddress);
char bufferSend[fileSize]; // The half size of the message sent from the sender(file size)
char CCub[8]; // Cubic size
char CCrenc[8]; // Reno size
```

נפתח receiver לsocket ומלא אותו באוֹתָה צורה כמו שהסבירנו בsender.s.

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror("Could not create listening socket \n");
    return -1;
}
```

עכשו נתחל את ה`struct` כמו שהסבירנו ב sender וק שיש פה שוני קטןشبקטוגריה של struct sin_addr.s_addr אנו מכנים ים INADDR_ANY זהה פשוט יותר קי אקיי מבין הסוגי ה`אפשרים` שיש במחשב המיעודים לכך.

```
serverAddress.sin_family = AF_INET; // define address type(IPV4)
serverAddress.sin_port = htons(SERVER_PORT); // convert server port to network byte order
serverAddress.sin_addr.s_addr = INADDR_ANY;
```

הfonקציה () bind מאפשרת לנו להאזין לחיבורים כמו החיבור עם ה Sender. הfonקציה מקבלת בפרמטרים את פרטii receiver , receiver שוכאים ב`struct` שמכיל את כל פרטי הכתובות שלו ובנוסף ל`struct` פרמטר נוסף הוא ה`socket` של receiver וגם גודל ה`struct` בשביל לדעת להקצות מקומות. הfonקציה מבהה לחיבור עם ה`sender`. אם לא תקבל חיבור הקוד לא ימשור לרוץ.

```
if (bind(sockfd, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) == -1) // bind to connect - (AF_INET,SOCK_STREAM)
{
    perror("bind failed");
}
```

fonקציית listen() בעצם מאפשרת לנו להחיליט כמה חיבורים אפשר לעשות בו זמןית עם receiver.

```
if (listen(sockfd, 5) < 0) // Max 5 client connection requests
{
```

* חשוב לציין שבכל fonקציה שנמצאת במספרה socket ישנו משתמשים בה, אנחנו צריכים לבדוק האם הfonקציה מחזירה -1 או 0 (לפעמים) כי זה מראה שהfonקציה לא פעולה כמו שציריך, אז בכל פעם אנחנו עושים את הבדיקה הזאת. בכל הצלומי מסך כמעט לא צילמו את החלק הזה אלא רק את החלק העיקרי אבל לדוגמא נראה בצלום מסך הבא שהשתמשנו בהזה.

בfonקציית accept() אנו מקבלים את פרטי receiver שעשה איתנו connection (גילנו אותו כשהעשינו bind והזנו לו).

הfonקציה משתמשת בsocket שלה, ומכניסה את פרטי receiver ל`struct` המתאים לו שזה - clientAddress, שמחכה לקבל את כל פרטי הכתובות של receiver, ומכוון שהוא בשפת C אז אנחנו צריכים לצרף בfonקציה את גודל ה`struct` כדי לדעת כמה מקום בזיכרון להקצות.

```
int connection = accept(sockfd, (struct sockaddr *)&clientAddress, &clientAddressLen); // Accept sender connection to sockfd
if (connection == -1)
{
    printf("accept failed");
    close(sockfd);
    return -1;
}
printf("connection successful .. \n");
```

נרצה להציג מערכת דינמי של זמינים. כדי שנוכל לחשב את הזמן שלוקח לכל חלק להגיא- וכדי שבסוף נוכל לחשב ממוצע ולהדפיס את הזמינים כמו שבקישו. וכך מגדרים בס מערכת דינמי.

```
double* HalfOne = (double*) malloc(arraySize * sizeof(double)); // HalfOne pointer about dynamic array in length 8*8 (which contains the times of the first part)
double* HalfTwo = (double*) malloc(arraySize * sizeof(double)); // HalfTwo pointer about dynamic array in length 8*8 (which contains the times of the second part)
```

נפתח לולאה שבתוכה קיבל את הקובץ וכאשר מישו ירצה להפסיק לשЛОח את הקובץ יתרחש break אחריה היא תמשיך לקבל לנצח.

. cubic נשנה את החיבור מלאגוריתם CC אחד לאחר ולכז נשנה עכשו ל

```
while(1){ // while loop to resend the file
    int countHalfOne = 0; // Time of the first half of the file
    int countHalfTwo = 0; // Time of the second half of the file
    int bytesSent = 0; // The bytes the sender send to the receiver
    int bytesRecv = 0; // The bytes the receiver get from the sender
    bzero(bufferSend, sizeof(bufferSend)); // Reset buffer to zero
    // Change CC
    strcpy(CCcub, "cubic"); // Change cc algorithm to cubic

    if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCcub, strlen(CCcub)) == -1)
    {
        perror("setsockopt");
        return -1;
    }
```

בפונקציית () gettimeofday אנו נשמר את הזמן שלפני קבלת המידע מהsender לתוכה ה struct timeStart וואז גם לאחר הקבלה נשמר את הזמן שהסתיים הקבלה ב struct timeval. אנו נטענו בלולאה שמקבלת בפועל סימן עוד ועוד פאקטות, את הפונקציה שמקבלת את הקובץ () recv, אנו מושיכת עד שנקבל את כל הפקאות שיבידם בגודל בbytes של החצי קובץ. הלולאה מושיכה עד שנקבל את כל הפקאות שיבידם בגודל בbytes של החצי קובץ. כל פעם שהיא מקבלת מידע היא מכניסה אותו לבארור זהה המערך של הצ'רים.

```
gettimeofday(&timeStart, NULL); // Start counting time (How much time the receiver received the first half of the message)

while(countHalfOne != fileSize/2){ // While loop to received the first half of the file from the sender

    bytesRecv = recv(connection, bufferSend, sizeof(bufferSend), 0);
    printf("Packet size : %d\n", bytesRecv);
    countHalfOne+=bytesRecv;
}
if (bytesRecv <= 0)
{
    printf("Sender has closed the TCP connection.\n");
    break;
}
gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the first half of the message)
```

לאחר מכן נשמר את ההפרש בין הזמן התחלתו לזמן סוף בתחום המערך הדינמי. נעשה זאת כדי לדעת כמה זמן זה לוקח ההגעת חצי קובץ.

```
*(HalfOne + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + (((double)timeEnd.tv_usec - timeStart.tv_usec))/1000; // Total half one time to received
```

שליח את אותן מוסכם – xor שאקן התקבל ב receive כל המידע. אנו יודעים שהתקבל כל המידע כי הלולאה של recv נעצרת רק כאשר מגיע כל הגודל של החצי קובץ שלנו. אך אם המשכנו מהלואה זה אומר שהגיע כל החצי קובץ שהוא אמור.

```
bytesSent = send(connection, &xor, sizeof(int), 0); // Send the authentication to the sender
```

כדי לקבל את החצי קובץ השני נשנה שוב את אלגוריתם ה CC הפעם לreno

```
// Change CC

strcpy(CCren,"reno"); // Change CC algorithm to rno

if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCren, strlen(CCren)) == -1)
{
    perror("setsockopt");
    return -1;
}
```

שוב נקבל את החצי קובץ כמו שהסבירנו פעם קוימת ונחשב את הזמן באותה דרך.

```
gettimeofday(&timeStart, NULL); // Start counting time (How much time the receiver received the second half of the message)

while(countHalfTwo != fileSize/2){ // While loop to received the second half of the file from the sender

    bytesRecv = recv(connection, bufferSend, sizeof(bufferSend) , 0);
    printf("Packet size : %d\n",bytesRecv);
    countHalfTwo+=bytesRecv;
}

if (bytesSent == 0)
{
    printf("Sender has closed the TCP connection.\n");
    return -1;
    break;
}

else if (bytesSent < 0)
{
    perror("Receive");
    return -1;
}
gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the second half of the message)
```

נשמרו את ההפרש זמנים במערך הדינامي.

```
*HalfTwo + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + (((double)timeEnd.tv_usec - timeStart.tv_usec))/1000; // Total half two time to received
```

נרצה שתיהה לנו אפשרות להגדיל את המערך הדינמי כל פעם שיגמר המקום ולכן נעשה כך.
וכמוון נקדם גם את האינדקס `TIME` שמודדר להיות האינדקס במערך הדינامي.
חווב לציין שיש שתי מערכים דינמיים של זמנים, אחד שהוא אחראי על הזמן הראשון בחצי הראשון של הקובץ והשני אחראי על הזמן השני, והאינדקס שרצ (`time`) הוא משותף לשניהם כי הם מתקדים בהתקופה.

```
if (TIME >= arraySize) // If the array full ---> Change array size
{
    arraySize *= 2;
    HalfOne = (double*) realloc(HalfOne, arraySize * sizeof(double));
    HalfTwo = (double*) realloc(HalfTwo, arraySize * sizeof(double));
}

TIME++;
```

לאחר קבלת שני חצאי הקובץ נרצה לשלוח לsender באמצעות פונקציית `(void send (socket, const void*, size_t, int))`.
בנוסף חשוב לנו לדעת האם נוכל לקבל שוב קובץ או שצריך לסיים חיבור ואז נצא בbreak.

```
int keepAlive; // message for the connection
int check = -1; // check if the receiver close the connection

send(connection, &keepAlive, sizeof(int), 0); // Send message to the sender to check connection

if ((check = (recv(connection, &keepAlive, sizeof(int), 0))) == 0) // received message from the sender if to keep connection(confirm)
{
    printf("TCP Connection with the sender was closed.\n");
    break;
}
else
{
    send(connection, &keepAlive, sizeof(int), 0); // Send a message to the sender if to keep connection(confirm)
}

}
```

אם לא קיבל את N אז לא נמצא מהלולה ונחזיר ונקבל את הקובץ שוב.
אם כן קיבל N אנו יוצאים מהלולה שמקבלת, וישר מחשבים ממצאים של הזמן.
בכלולה הריאונה נדפיס מtower המערך הדינמי את כל הזמןים שלקחו לחצוי הראשן להגעה בכל הפעמים שהוא נשלח. וכך בולואה השניה גם לחלק השני את הזמןים.
בסוף אחרי שסיכמנו את כל הזמןים של שתי החצאים נעשה להם חישוב ממוצע ונדפיס אותו.

```
printf("\nCubic Total Run Times:\n");

for (int i = 0; i < TIME; i++) // Loop for calculate avarage time for the first half (CC cubic)
{
    avarage_HalfONETime += *(HalfONE + i);
    printf("Cubic run %d time = %0.3lf ms\n", (i+1), *(HalfONE + i));
}
avarage_HalfONETime = avarage_HalfONETime / TIME;
printf("Total avarage cubic: %0.3lf ms\n", avarage_HalfONETime);

printf("\nReno Total Run times:\n");

for (int i = 0; i < TIME; i++) // Loop for calculate avarage time for the second half (CC reno)
{
    avarage_HalfTwoTime += *(HalfTWO + i);
    printf("Reno run %d time = %0.3lf ms\n", (i+1), *(HalfTWO + i));
}
avarage_HalfTwoTime = avarage_HalfONETime / TIME;
printf("Total avarage reno: %0.3lf ms\n", avarage_HalfTwoTime);

close(connection);
close(sockfd);
return 0;
```

ולסיום אנו סוגרים את החיבור שפתחנו עם sender וגם את הsocket הראשי.
כדי להריץ את התוכנה קודם כל צורך לקמפל את הקבצים שיינו מוכנים להרצה באמצעות make all בטרמינל
כל אחד בנפרד. כדי להריץ נבצע את הרצה עם הפיקודות ./Sender ./Receiver ./.makefile 1mb והקובץ receiverserver.
צריך לפתוח 2 טרמינלים בתקיה בה נמצאים ה sender receiver יסתוים הרצה של שליחת פעם אחת
ולכתוב את הפיקודות שרשמננו.
באחד מהטרמינלים להריץ את ה sender ובשני את receiver וכאשר יסתוים הרצה של שליחת פעם אחת
של קובץ יהיה למשתמש אפשרות להחליט אם הוא רוצה לשלוח שוב לפ' לחיצת Y או N לבחירתו.

: SENDER

C sender.c

```

1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <string.h>
4 #include <arpa/inet.h>
5 #include <stdio.h>
6 #include <unistd.h>
7 #include <netinet/in.h>
8 #include <netinet/tcp.h>
9
10#define SERVER_PORT 9999
11#define SERVER_IP_ADDRESS "127.0.0.1"
12#define ID1 1242
13#define ID2 3836
14
15 int main(){
16     FILE *fptr; // define pointer to a file type
17     fptr = fopen("1mb.txt","r"); // open the file "1mb.txt" , "r" - read
18     if (fptr == NULL)
19     {
20         perror("Failed open file");
21         return 2;
22     }
23     fseek(fptr,0,SEEK_END); // Move the pointer to the end of the file
24     long fileSize = ftell(fptr); // Size of the file
25     fseek(fptr,0,SEEK_SET); // Move the pointer to the start of the file
26     printf("LOOK HERE %ld\n",fileSize/2);
27
28     char bufferSend[fileSize/2]; // The half size of the message sent from the sender(file size)
29     char CCcub[8];
30     char CCren[8];
31
32     int sockfd = -1;
33     if((sockfd = socket(AF_INET , SOCK_STREAM , 0 )) == -1) { // create file descriptor socket
34         perror("Could not create file descriptor socket ");
35         return -1;
36     }
37
38     struct sockaddr_in serveraddress;
39     memset(&serveraddress,0,sizeof(serveraddress)); // reset serveraddress
40     serveraddress.sin_family = AF_INET; // define address type(IPV4)
41     serveraddress.sin_port = htons(SERVER_PORT); // convert server port to network byte order
42

```

הספריות שמכילות את ההוראות לפונקציות שבם משתמש

נגידר :
port - 9999
IPv4-127.0.0.1

4 ספירות אחרונות של התשודות זהות (לשימוש באוטנטיקציה) - ID1/ID2

נפתח את הקובץ בуורת הפונקציה (**fopen**)
 בשימוש ב - **"z"** כדי לקרוא את הקובץ
 והגדרכנו **fptr** להיות מצביע לקובץ

בשימוש בפונקציה **fseek** כדי להזין את הגודל של הקובץ ("בנהחה שאנו חסן לא
 לסוף הקובץ כדי להשיג את הגודל הקובץ") , ונחיר אותו לתחילה הקובץ
 נודיעים את גודל הקובץ (" , ונהיר אותו לתחילה הקובץ

נגידר את גודל החודעה יהיה ביחס מוגדל הקובץ
 משוך של תווים עברו כל אחד מאלגוריתמים CC

נקראת ה- **socket** שיכיל
IPv4-AF_INET
TCP PORT-SOCK_STREAM

נכלה את השדות המבוקשים :
 (in - for internet) **struct sockaddr_in**
 הממיר את IP,PORt הפהונקציה **htonS**, **htonL**, **network bytes order**

זיכרון שאנו אין יודעים איך המעבד בצד השני עובד ומתקבל מידע לכך כדי שהמידע יגיע באופן אמין)

C sender.c

```

19     perror("Failed open file");
20     return 2;
21 }
22 fseek(fptr,0,SEEK_END); // Move the pointer to the end of the file
23 long fileSize = ftell(fptr); // Size of the file
24 fseek(fptr,0,SEEK_SET); // Move the pointer to the start of the file
25 printf("LOOK HERE %ld\n",fileSize/2);
26
27     char bufferSend[fileSize/2]; // The half size of the message sent from the sender(file size)
28     char CCcub[8]; // The size of
29     char CCren[8];
30
31     int sockfd = -1;
32     if((sockfd = socket(AF_INET , SOCK_STREAM , 0 )) == -1) { // create file descriptor socket
33         perror("Could not create file descriptor socket ");
34         return -1;
35     }
36
37     struct sockaddr_in serveraddress;
38     memset(&serveraddress,0,sizeof(serveraddress)); // reset serveraddress
39     serveraddress.sin_family = AF_INET; // define address type(IPV4)
40     serveraddress.sin_port = htons(SERVER_PORT); // convert server port to network byte order
41
42     int rval = inet_pton(AF_INET,(const char*)SERVER_IP_ADDRESS,&serveraddress.sin_addr); // Convert IP Address: binary/decimal representation
43
44     if(rval <= 0){
45         perror("Inet pton failed");
46         return -1;
47     }
48
49     printf("Connected .... \n");
50
51     int connection = connect(sockfd, (struct sockaddr*)&serveraddress, sizeof(serveraddress)); // Connect to sockfd
52
53     if(connection == -1){
54         perror("connection failed");
55         return -1;
56     }
57
58     int xor = (ID1^ID2); // The message the sender expects to receive (for authentication)
59     int xorRecv = 0; // The message from the receiver (for authentication)
60
61

```

inet_pton ביחסnetwork bytes order ביחס הפהונקציה **Network Bytes Order** -

וציהות **socket** לחיבור (זכרנו נושאר את המידע)

2 משתנים שבעזרתם נבדוק האם המידע
 באמת הגיא לצד השני

Run Terminal Help

C receiver.c

C sender.c X

C sender.c

```

63     while(1) // while loop to resend the file
64
65     int bytesSent = 0; // The bytes the sender sent
66     int bytesRecv = 0; // The bytes the sender received from the receiver
67
68     fseek(fp, 0, SEEK_SET); // Move the pointer to the start of the file
69     // Change CC
70     strcpy(CC, "cubic"); // Change cc algorithm to cubic
71     if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CC, strlen(CC)) == -1)
72     {
73         perror("setsockopt");
74         return -1;
75     }
76     printf("CC algorithm changed to : cubic\n");
77
78     while ((connection = fread(bufferSend, 1, sizeof(bufferSend), fp)) > 0) { // Send the first half of the file
79         bytesSent = send(sockfd, bufferSend, connection, 0);
80         printf("LOOK HERE: %d\n", bytesSent);
81         if (bytesSent == 0) {
82             printf("peer has closed the TCP connection.\n");
83         }
84         else if (bytesSent == -1) {
85             printf("send failed.\n");
86         }
87         else if (connection > bytesSent) {
88             printf("sent only %ld bytes from the required %d.\n", strlen(bufferSend), bytesSent);
89         }
90         else {
91             printf("Half one of the file send successfully sent (Total bytes sent : %d)\n", bytesSent);
92             break;
93         }
94     }
95     printf("Waiting For Authentication ..\n");
96
97     bytesRecv = recv(sockfd, &xorRecv, sizeof(int), 0); // Receive a authentication from the receiver
98
99     if (bytesRecv == 0)
100    {
101        printf("peer has closed the TCP connection.\n");
102    }
103
104    else if (bytesRecv == -1)
105    {
106        printf("Receive failed");
107    }
108
109
110    if (xor != xorRecv)
111    {
112        printf("Authentication failed\n");
113        break;
114    }
115
116    else
117        printf("Autentication completed.\n"); // The first half of the file successfully received to the receiver
118
119
120    // Change CC
121    strcpy(C, "reno"); // Change cc algorithm to reno
122
123    if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, C, strlen(C)) != 0) {
124        perror("setsockopt error");
125        return -1;
126    }
127    printf("CC algorithm changed to : reno\n");
128
129
130    while ((connection = fread(bufferSend, 1, sizeof(bufferSend), fp)) > 0) { // Send the second half of the file
131        bytesSent = send(sockfd, bufferSend, connection, 0);
132        if (bytesSent == 0) {
133            printf("peer has closed the TCP connection.\n");
134        }
135        else if (bytesSent == -1) {
136            printf("send failed.\n");
137        }
138        else if (connection > bytesSent) {
139            printf("sent only %ld bytes from the required %d.\n", strlen(bufferSend), bytesSent);
140        }
141        else if (ftell(fp) == fileSize) {
142            printf("Half two of the file send successfully sent (Total bytes sent : %ld)\n", ftell(fp));
143        }
144    }
145
146
147    char exit = 'Y'; // For Scanner(exit = 'Y' --> start while loop again , exit = 'N' --> break while loop)

```

היבטים שמנחלים
והיבטים שמתתקלים

מציאת המצביע לתחילת
הקובץ בכל פעם שנשלח
את הקובץ מחדש

נשנה את האלגוריתם ל- cubic
(נעביר את המצביע של המחרוזת אל CC)

נקראת הקובץ ונשלח את החצי הראשון
fread()-to read the file into buffersend
send()-send the file into socket

קבלת האותנטיקציה
recv() - to receive message

שינוי האלגוריתם ל- reno
CcRen - השינוי במאחורית

נעביר את המצביע של המחרוזת לתוך - CCRen

שלוח את החצי השני של
הקובץ, הסברתו בתמונות
הקודמות על הפקנציות
fread,send

הולאה לתגמור לאחר שנשלחה
בל הקובץ מכיוון שהמצביע
יעזר לאחר שגדלו שווה לגודל
הקובץ

ftell - return the size of the pointer

```

146
147     char exit = '0'; // For Scanner(exit = 'Y' --> start while loop again , exit = 'N' --> break while loop)
148     int keepAlive; // Massage for the connection
149     int check = -1; // Check if the receiver close the connection
150
151     recv(sockfd, &keepAlive, sizeof(int), 0); // received message from the receiver to keep connection
152     printf("Do you want to send the file again ? :) \n Yes- press 'Y' , No- press 'N' \n");
153
154     while(exit != 'Y' && exit != 'N'){ // While loop --> if the user type a char that is not Y or N send the ma
155         scanf(" %c" , &exit);
156         if(exit !='Y' && exit != 'N')
157         {
158             printf("Press only : Y - Send again || N - Close connection \n");
159         }
160     }
161     if(exit == 'N') // If exit = N --> close connection
162     {
163         printf("Closes the connection .. \n");
164         break;
165     }
166     send(sockfd, &keepAlive, sizeof(int), 0); // send message to the receiver to keep connection(confirm)
167     recv(sockfd, &keepAlive, sizeof(int), 0); // received a message from the receiver to check if the receiver get the confirm message from 1
168
169
170     fclose(fptr);
171     close(sockfd);
172

```

כדי שהתקשות תמשין להתקיים במידה והמשתמש ירצה לשלוח את אותו קובץ עוד פעם , אנחנו נקבל הודעה מהשורה כדי לראות אם אנחנו מעוניינים להמשיך את הקשר ולאחר מכן השורה יזכה לקבל בחזרה תגובה מייתנו, במידה ואנחנו לא סגנו את הקשר נשלח לו שעניינו נרצה להמשיך את הקשר ונחכה לקבל ממנו הודעה אם הקשר המשיך ואם הוא קיבל את הבקשה להמשיך את הקשר, במידה ואנו סגנו את הקשר אנחנו נצא מהלואה ונסגור את הפסוק של השולח.

:Receiver

receiver.c - 315141242_212733836 - Visual Studio Code

```

Selection View Go Run Terminal Help
EXPLORER ... C receiver.c X C sender.c
315141242_212733836 > Captured 1mb.txt C receiver.c C sender.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <arpa/inet.h>
6 #include <netinet/in.h>
7 #include <netinet/tcp.h>
8 #include <errno.h>
9 #include <string.h>
10 #include <unistd.h>
11 #include <sys/time.h>
12
13 #define SERVER_PORT 9999
14 #define ID1 1242
15 #define ID2 3836
16 #define fileSize 1048576
17
18 int main(){
19
20     struct sockaddr_in serverAddress ,clientAddress;
21     struct timeval timeStart, timeEnd;
22     socklen_t clientAddressLen = sizeof(clientAddress);
23     char bufferSend[fileSize]; // The half size of the message sent from the sender(file size)
24     char CCCub[8]; // Cubic size
25     char CCren[8]; // Reno size
26
27     memset(&serverAddress, 0, sizeof(serverAddress));
28     memset(&clientAddress, 0, sizeof(clientAddress));
29     clientAddressLen = sizeof(clientAddress);
30
31
32     int sockfd = -1;
33     int reuse = 1;
34     if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
35     {
36         perror("Could not create listening socket \n");
37         return -1;
38     }
39
40     if ((setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof(int))) == -1)
41     {
42         printf("setsockopt failed \n");
43     }

```

הספריות המובילות את הヘルפרים ששימושם נשתמש

בניגוד ל **sender** את גודל הקובץ fileSize = 1048576
ולכן הקוד יעבור בשלהמו עבר הקובץ הספציפי שלו
הגדומו מראש את ה - PORT יהיה 9999
וכמובן את ה - 4 ספנות האחרונות של התשומות דוחת כדי לבצע אונטיקציה: ID1, ID2

struct timeval timeStart, timeEnd
נשתמש במבנה זה כדי למדוד כמה זמן לקובץ להגיע בשלהמו.
נדיר את האורך של הכתובת של הלוקו (את השאר הסברתו בתמונות הקודמות)
שদע כמה מקום נדרש לפונקציית העבורה. בוגר איפוס בכתובות של הלוקו ושל השרת
בנוסף לשימוש בפונקציה memset כדי לבצע איפוס בפונקציה: memset

receiver.c - 315141242_212733836 - Visual Studio Code

```

Selection View Go Run Terminal Help
EXPLORER ... C receiver.c X C sender.c
315141242_212733836 > Captured 1mb.txt C receiver.c C sender.c
46 serverAddress.sin_family = AF_INET; // define address type(IPV4)
47 serverAddress.sin_port = htons(SERVER_PORT); // convert server port to network byte order
48 serverAddress.sin_addr.s_addr = INADDR_ANY;
49
50 //bind
51
52 printf("Binding\n");
53
54 if (bind(sockfd, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) == -1) // bind to connect - (AF_INET,SOCK_STREAM)
55 {
56     perror("Bind failed");
57     close(sockfd);
58     return -1;
59 }
60
61 printf("Binded\n");
62
63 if (listen(sockfd, 5) < 0) // Max 5 client connection requests
64 {
65     printf("listen failed");
66     close(sockfd);
67     return -1;
68 }
69
70 printf("Listening ..\n");
71
72 int connection = accept(sockfd, (struct sockaddr *)&clientAddress, &clientAddressLen); // Accept sender connection to sockfd
73
74 if (connection == -1)
75 {
76     printf("accept failed");
77     close(sockfd);
78     return -1;
79 }
80
81 printf("connection successful .. \n");
82
83 int arrayIndex = 0;
84 int arraySize = 8;
85
86 double* HalfONE = (double*) malloc(arraySize * sizeof(double)); // HalfONE pointer about dinamic array in length 8*8 (which contains the
87 double* HalfTWO = (double*) malloc(arraySize * sizeof(double)); // HalfTWO pointer about dinamic array in length 8*8 (which contains the
88
89 if (HalfONE == NULL || HalfTWO == NULL ){
90     perror("malloc");
91     close(sockfd);
92     return -1;
93 }

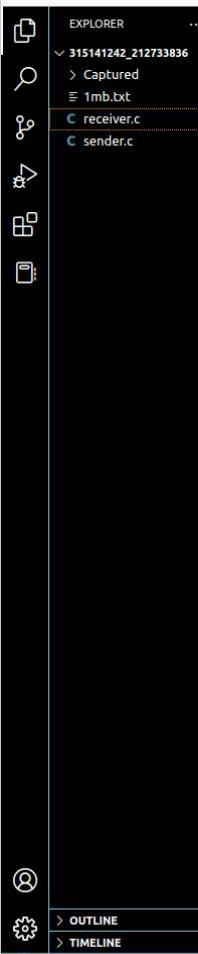
```

בתוכ הפונקציה bind מילאמן את השדות:
AT_HFSOCK
AF_INET
SOCK_STREAM
listen

ובפונקציה listen הגדתו כמה קריאות יכול השרת לענות עבורה

הגדו 2 מבאים עבור המערךים שיכיל את הנקודות של שליחת שני חצאי הקובץ

File Edit Selection View Go Run Terminal Help



```

C receiver.c x C sender.c
C receiver.c
C receiver.c
int xor = (ID1^ID2); // The message the sender expects to receive
int xorRecv; // The message from the receiver
int TIME = 0; // Time between the send and the receive
while(1){ // while loop to resend the file
    int countHalfOne = 0; // Time of the first half of the file
    int countHalfTwo = 0; // TIme of the second half of the file
    int bytesSent = 0; // The bytes the sender send to the receiver
    int bytesRecv = 0; // The bytes the sender get from the receiver
    bzero(bufferSend, sizeof(bufferSend)); // Reset buffer to zero
    // Change CC
    strcpy(CCcub,"cubic"); // Change CC algorithm to cubic
    if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCcub, strlen(CCcub)) == -1)
    {
        perror("setsockopt");
        return -1;
    }
    printf("CC algorithm changed to : cubic\n");

    printf("Wait for receiving the first half of the file...\n");
    gettimeofday(&timeStart, NULL); // Start counting time (How much time the receiver received the first half of the message)

    while(countHalfOne != fileSize/2){ // While loop to received the first half of the file from the sender
        bytesRecv = recv(connection, bufferSend, sizeof(bufferSend), 0);
        printf("Packet size : %d\n",bytesRecv);
        countHalfOne+=bytesRecv;
    }
    if (bytesRecv <= 0)
    {
        printf("Sender has closed the TCP connection.\n");
        break;
    }

    gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the first half of the message)
    *(HalfONE + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + ((double)timeEnd.tv_usec - timeStart.tv_usec)/1000; // Total half or
    printf("The time of the first half to receive : %f \n", *(HalfONE+TIME));
    printf("The first half of the file received successfully \n");
    printf("Total bytes received: (%d bytes)\n", countHalfOne);
}

```

countHalfOne/countHalfTwo
נשותם בהם כדי לבדוק את גודל
הBITSIM המתקבלם
בשנה את האלגוריתם ל cubic

bzero() reset bufferSend
בגדרם שופט, וכל שבר
מופיע כמה ביטים נשלחו
נעצור את חישוב הזמן

```

C receiver.c x C sender.c
C receiver.c
130 gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the first half of the message)
131
132 *(HalfONE + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + ((double)timeEnd.tv_usec - timeStart.tv_usec)/1000; // Total half or
133 printf("The time of the first half to receive : %f \n", *(HalfONE+TIME));
134 printf("The first half of the file received successfully \n");
135 printf("Total bytes received: (%d bytes)\n", countHalfOne);
136
137 printf("Sending authontication .. \n");
138
139 bytesSent = send(connection, &xor, sizeof(int), 0); // Send the authontication to the sender
140
141 if (bytesSent == 0)
142 {
143     printf("Sender has closed the TCP connection.\n");
144     return -1;
145     break;
146 }
147
148 else if (bytesSent < 0)
149 {
150     perror("send");
151     return -1;
152 }
153 else{
154
155     printf("Authontication sent .. \n");
156 }
157
158 // Change CC
159
160 strcpy(CCren,"reno"); // Change CC algorithm to reno
161
162 if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCren, strlen(CCren)) == -1)
163 {
164     perror("setsockopt");
165     return -1;
166 }
167
168 printf("CC algorithm changed to : reno\n");
169 printf("Receiving the second half of the file \n");
170
171 gettimeofday(&timeStart, NULL); // Start counting time (How much time the receiver received the second half of the message)
172
173 while(countHalfTwo != fileSize/2){ // While loop to received the second half of the file from the sender
174
175 }

```

שלוח את האותנטיקציה
לקוח לאחר קבלת החצי
הראשון של הקוביץ

משנה את האלגוריתם ל reno
(הסבירו יותר בתמונות הקוממות)

View Go Run Terminal Help

3836

C receiver.c X C sender.c

```

154     else{
155
156         printf("Authontication sent .. \n");
157     }
158
159     // Change CC
160
161     strcpy(CCren,"reno"); // Change CC algorithm to reno
162
163     if (setsockopt(sockfd, IPPROTO_TCP, TCP_CONGESTION, CCren, strlen(CCren)) == -1)
164     {
165         perror("setsockopt");
166         return -1;
167     }
168
169     printf("CC algorithm changed to : reno\n");
170     printf("Receiving the second half of the file \n");
171
172     gettimeofday(&timeStart, NULL); // Start counting time (How much time the receiver received the second half of the message)
173
174     while(countHalfTwo != fileSize/2){ // While loop to received the second half of the file from the sender
175
176         bytesRecv = recv(connection, bufferSend, sizeof(bufferSend) , 0);
177         printf("Packet size : %d\n",bytesRecv);
178         countHalfTwo+=bytesRecv;
179     }
180
181     if (bytesSent == 0)
182     {
183         printf("Sender has closed the TCP connection.\n");
184         return -1;
185         break;
186     }
187
188     else if (bytesSent < 0)
189     {
190         perror("Receive");
191         return -1;
192     }
193
194     gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the second half of the message)
195
196     *(HalfTWO + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + (((double)timeEnd.tv_usec - timeStart.tv_usec))/1000; // Total half tv

```

מחילה לחשב את הזמן
שליחת החזי השמי של
הΚονβאַק, בפונק שוב
השתמשן בללאה כי
המידע יוביל להגיע בכמה
פאקטאות באגדים שונים,
געצור את החישוב של הזמן

Section View Go Run Terminal Help

24_212733836
tured
o.txt
iver.c
der.c

C receiver.c X C sender.c

```

180     if (bytesSent == 0)
181     {
182         printf("Sender has closed the TCP connection.\n");
183         return -1;
184         break;
185     }
186
187
188     else if (bytesSent < 0)
189     {
190         perror("Receive");
191         return -1;
192     }
193
194     gettimeofday(&timeEnd, NULL); // End counting time (How much time the receiver received the second half of the message)
195
196     *(HalfTWO + TIME) = (timeEnd.tv_sec - timeStart.tv_sec)*1000 + (((double)timeEnd.tv_usec - timeStart.tv_usec))/1000; // Total hal
197     printf("The time of the second half to receive : %f \n", *(HalfTWO+TIME));
198     printf("The second half of the file received successfully\n");
199     printf("Total bytes received: (%d bytes)\n", countHalfTwo);
200
201     if (TIME >= arraySize) // If the array full ---> Change array size
202     {
203         arraySize *= 2;
204         HalfONE = (double*) realloc(HalfONE, arraySize * sizeof(double));
205         HalfTWO = (double*) realloc(HalfTWO, arraySize * sizeof(double));
206     }
207
208     TIME++;
209
210     int keepAlive; // message for the connection
211     int check = -1; // Check if the receiver close the connection
212
213     send(connection, &keepAlive, sizeof(int), 0); // Send message to the sender to check connection
214
215     if ((check = (recv(connection, &keepAlive, sizeof(int), 0))) == 0) // received message from the sender if to keep connection(confirm)
216     {
217         printf("TCP Connection with the sender was closed.\n");
218         break;
219     }
220     else
221     {
222         send(connection, &keepAlive, sizeof(int), 0); // Send a massage to the sender if to keep connection(confirm)
223     }

```

בשלב זהה אנחנו בודקים
שאם ספרטו את זמן
הריצה יותר פעמים מוגדר
המערך איז נבדיל אותו

הסביר על כך בתמונות הקודמות

File Edit Selection View Go Run Terminal Help

EXPLORER ... C receiver.c C sender.c

```

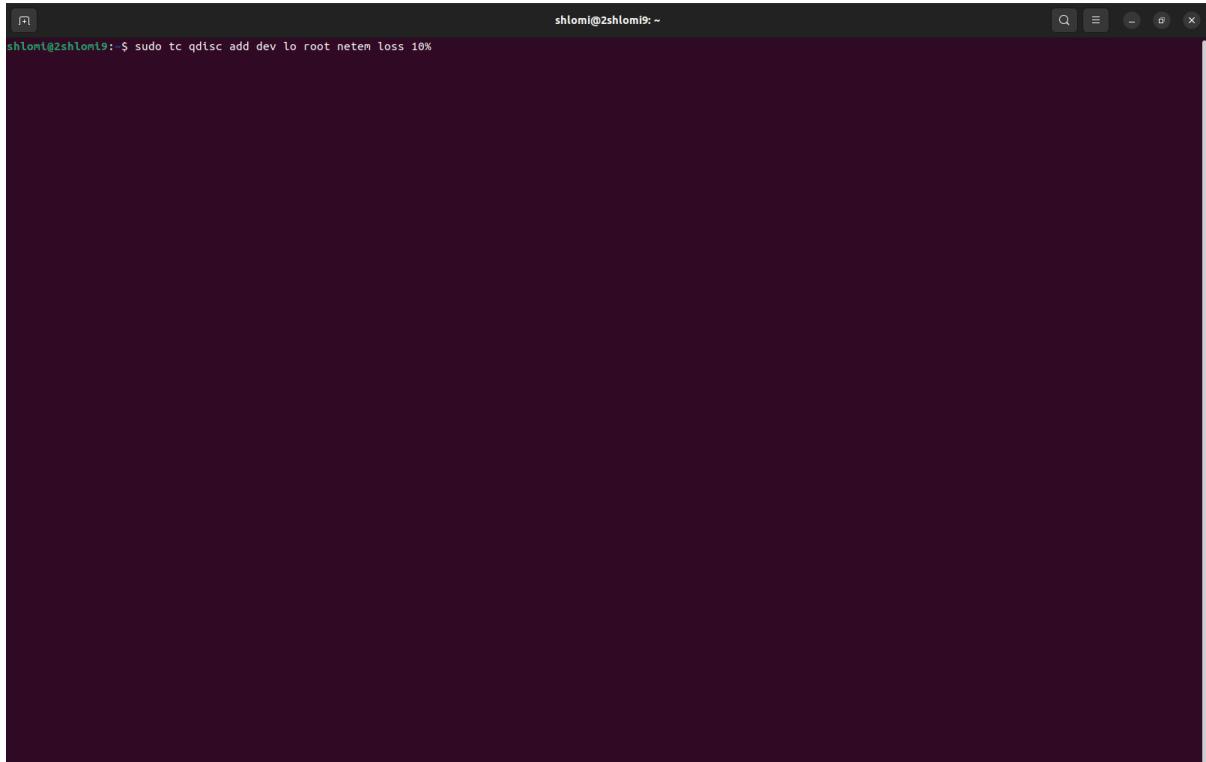
315141242_212733836
Captured
1mb.txt
C receiver.c
C sender.c

211  #include <sys/types.h>
212  #include <sys/socket.h>
213  #include <netdb.h>
214  #include <errno.h>
215  #include <stdio.h>
216  #include <string.h>
217  #include <time.h>
218  #include <math.h>
219
220  #define TIME 100
221
222  int check = 0; // check if the receiver close the connection
223
224  send(connection, &keepAlive, sizeof(int), 0); // Send message to the sender to check connection
225
226  if ((check = (recv(connection, &keepAlive, sizeof(int), 0))) == 0) // received message from the sender if to keep connection(confirm)
227  {
228      printf("TCP Connection with the sender was closed.\n");
229      break;
230  }
231  else
232  {
233      send(connection, &keepAlive, sizeof(int), 0); // Send a message to the sender if to keep connection(confirm)
234  }
235
236
237 double avarage_HalfONETime = 0, avarage_HalfTwoTime = 0; // Avarage time for each half of the file to received
238
239 printf("\nCubic Total Run Times:\n");
240
241 for (int i = 0; i < TIME; i++) // Loop for calculate avarage time for the first half (CC cubic)
242 {
243     avarage_HalfONETime += *(HalfONE + i);
244     printf("Cubic run %d time = %0.3lf ms\n", (i+1), *(HalfONE + i));
245 }
246 avarage_HalfONETime = avarage_HalfONETime / TIME;
247 printf("Total avarage cubic: %0.3lf ms\n", avarage_HalfONETime);
248
249 printf("\nReno Total Run times:\n");
250
251 for (int i = 0; i < TIME; i++) // Loop for calculate avarage time for the second half (CC reno)
252 {
253     avarage_HalfTwoTime += *(HalfTWO + i);
254     printf("Reno run %d time = %0.3lf ms\n", (i+1), *(HalfTWO + i));
255 }
256 avarage_HalfTwoTime = avarage_HalfONETime / TIME;
257 printf("Total avarage reno: %0.3lf ms\n", avarage_HalfTwoTime);
258
259
260 close(connection);
261 close(sockfd);
262 return 0;
263
264 }
```

לآخر שהמשתמש קבע
שהוא לא רוצה לשלוח יותר
את הקובץ א חט נא המלחאה ווחשב כמו שכח הכל הטעמים שייצאו לנו בשני האלגוריתמים : **cubic,reno** |

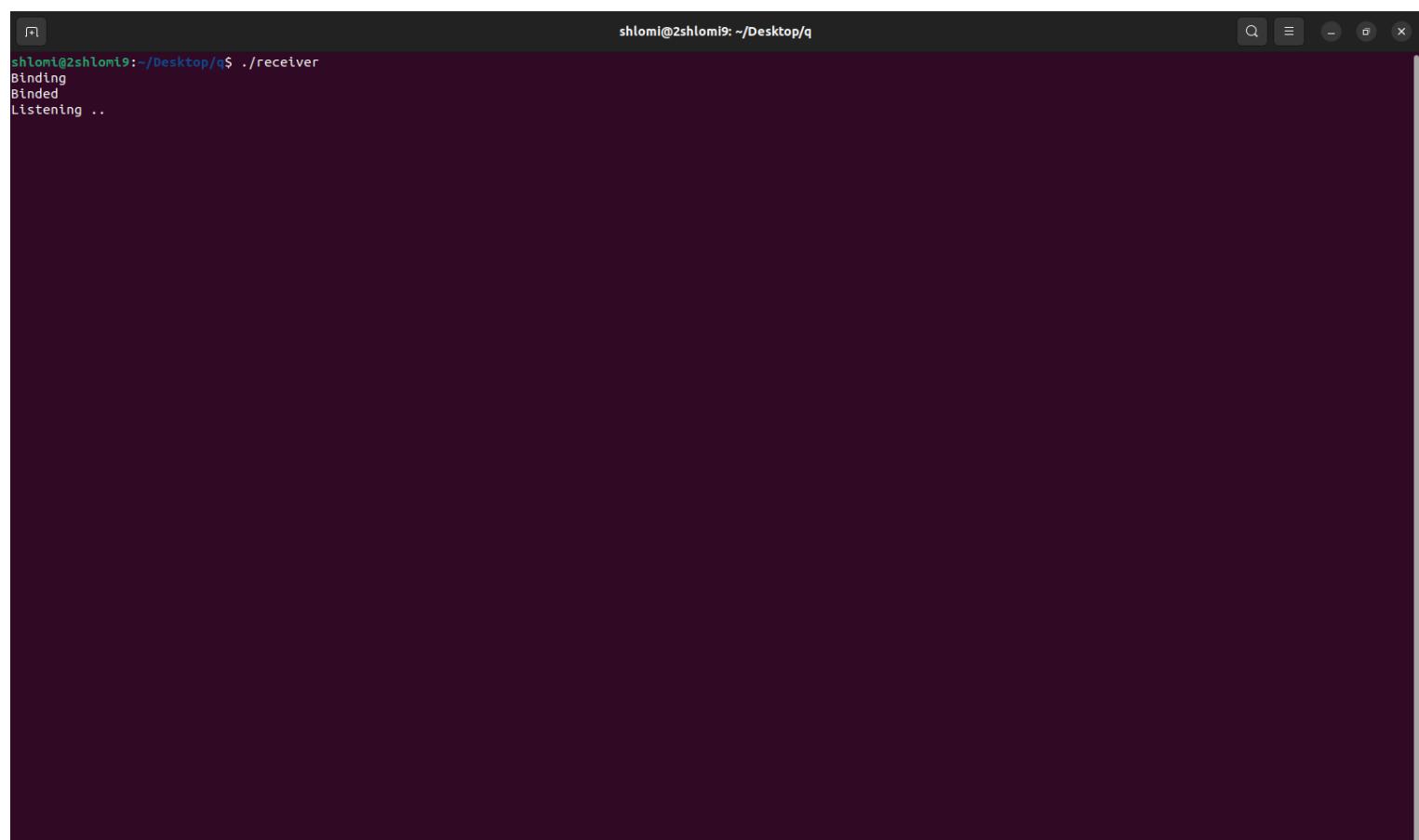
סירות הסקיטים(הקשר נסגר)

נעביר ל – wireshark וنبצע הソンפה :
נשלח את הקובץ 5 פעמים , ובכל פעם נשנה את אחזו
איבוד הפקודות. (שנרצה לשנות במקומפֿפֿפֿ נרשם change בתמונה למטה)



```
shlomi@2shlomi9: $ sudo tc qdisc add dev lo root netem loss 10%
```

נפעיל את ה – server –



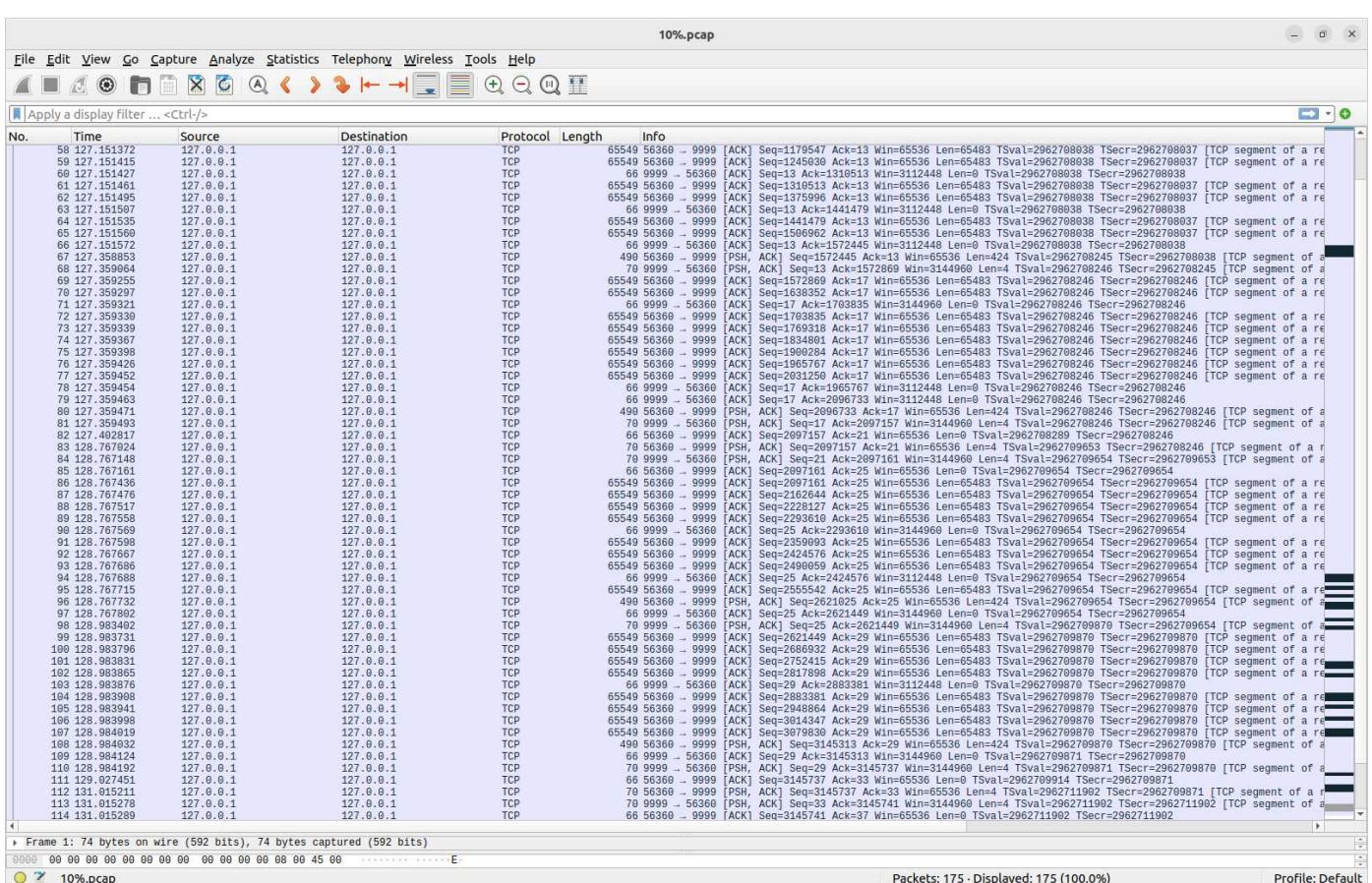
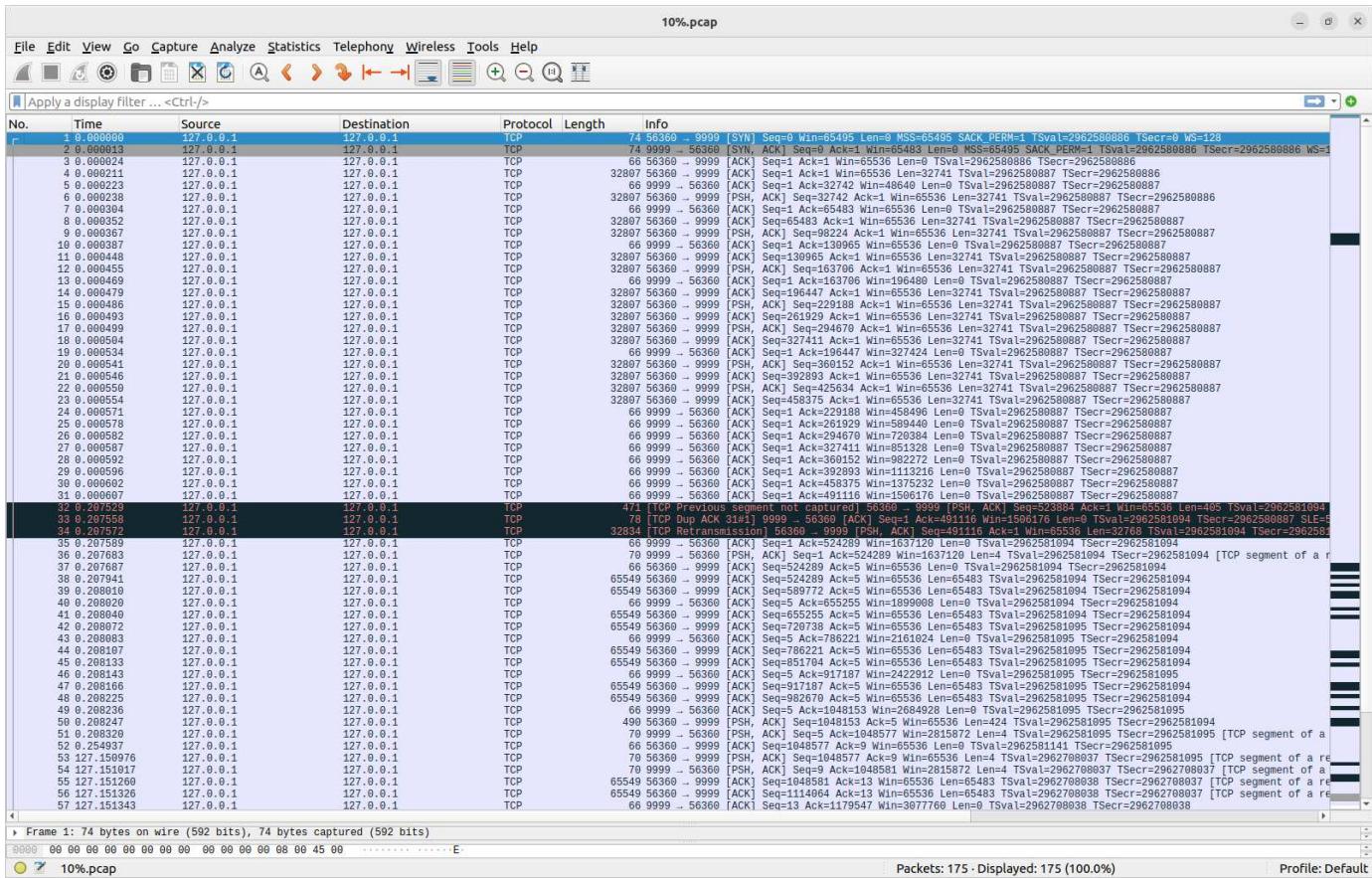
```
shlomi@2shlomi9: ~/Desktop/q$ ./receiver
Binding
Binded
Listening ..
```

נשלח את הקובץ 5 פעמים ונסגור את הקשר

```
shlomi@2shlomi9: ~/Desktop/q$ ./sender
LOOK HERE 524288
Connected ...
CC algorithm changed to : cubic
LOOK HEREEE 524288
Half one of the file send successfully sent (Total bytes sent : 524288) .
Waiting For Authentication ..
Received only 4 bytes from the required.
Authentication completed.
(CC algorithm changed to : rno
Half two of the file send successfully sent (Total bytes sent : 1048576) .
Do you want to send the file again ? :)
Yes - press 'Y' , No- press 'N'
Y
CC algorithm changed to : cubic
LOOK HEREEE : 524288
Half one of the file send successfully sent (Total bytes sent : 524288) .
Waiting For Authentication ..
Received only 4 bytes from the required.
Authentication completed.
(CC algorithm changed to : rno
Half two of the file send successfully sent (Total bytes sent : 1048576) .
Do you want to send the file again ? :)
Yes - press 'Y' , No- press 'N'
Y
CC algorithm changed to : cubic
LOOK HEREEE : 524288
Half one of the file send successfully sent (Total bytes sent : 524288) .
Waiting For Authentication ..
Received only 4 bytes from the required.
Authentication completed.
(CC algorithm changed to : rno
Half two of the file send successfully sent (Total bytes sent : 1048576) .
Do you want to send the file again ? :)
Yes - press 'Y' , No- press 'N'
Y
CC algorithm changed to : cubic
LOOK HEREEE : 524288
Half one of the file send successfully sent (Total bytes sent : 524288) .
Waiting For Authentication ..
Received only 4 bytes from the required.
Authentication completed.
(CC algorithm changed to : rno
Half two of the file send successfully sent (Total bytes sent : 1048576) .
Do you want to send the file again ? :)
Yes - press 'Y' , No- press 'N'
Y
CC algorithm changed to : cubic
LOOK HEREEE : 524288
Half one of the file send successfully sent (Total bytes sent : 524288) .
Waiting For Authentication ..
Received only 4 bytes from the required.
Authentication completed.
(CC algorithm changed to : rno
Half two of the file send successfully sent (Total bytes sent : 1048576) .
Do you want to send the file again ? :)
Yes - press 'Y' , No- press 'N'
Y
Closes the connection ..
shlomi@2shlomi9: ~/Desktop/q$
```

```
shlomi@2shlomi9: ~/Desktop/q$ ./receiver
Binding
Binded
Listening ..
connection successful ..
CC algorithm changed to : cubic
Wait for receiving the first half of the file
Packet size : 65482
Packet size : 32741
Packet size : 32741
Packet size : 32741
The time of the first half to receive : 0.365000
The first half of the file received successfully
Total bytes received: (524288 bytes)
Sending authentication ..
Authentication sent ..
CC algorithm changed to : rno
Receiving the second half of the file
Packet size : 196449
Packet size : 65483
Packet size : 65483
Packet size : 65483
Packet size : 65483
Packet size : 65907
The time of the second half to receive : 0.513000
The second half of the file received successfully
Total bytes received: (524288 bytes)
CC algorithm changed to : cubic
Wait for receiving the first half of the file
Packet size : 196449
Packet size : 65483
Packet size : 65483
Packet size : 65483
Packet size : 65483
Packet size : 65907
The time of the second half to receive : 0.357000
The first half of the file received successfully
Total bytes received: (524288 bytes)
Sending authentication ..
Authentication sent ..
CC algorithm changed to : rno
Receiving the second half of the file
Packet size : 196449
Packet size : 130966
Packet size : 65483
Packet size : 65483
The time of the second half to receive : 0.399000
The second half of the file received successfully
Total bytes received: (524288 bytes)
CC algorithm changed to : cubic
Wait for receiving the first half of the file
Packet size : 65483
Packet size : 65907
The time of the first half to receive : 0.342000
The first half of the file received successfully
Total bytes received: (524288 bytes)
Sending authentication ..
Authentication sent ..
CC algorithm changed to : rno
```

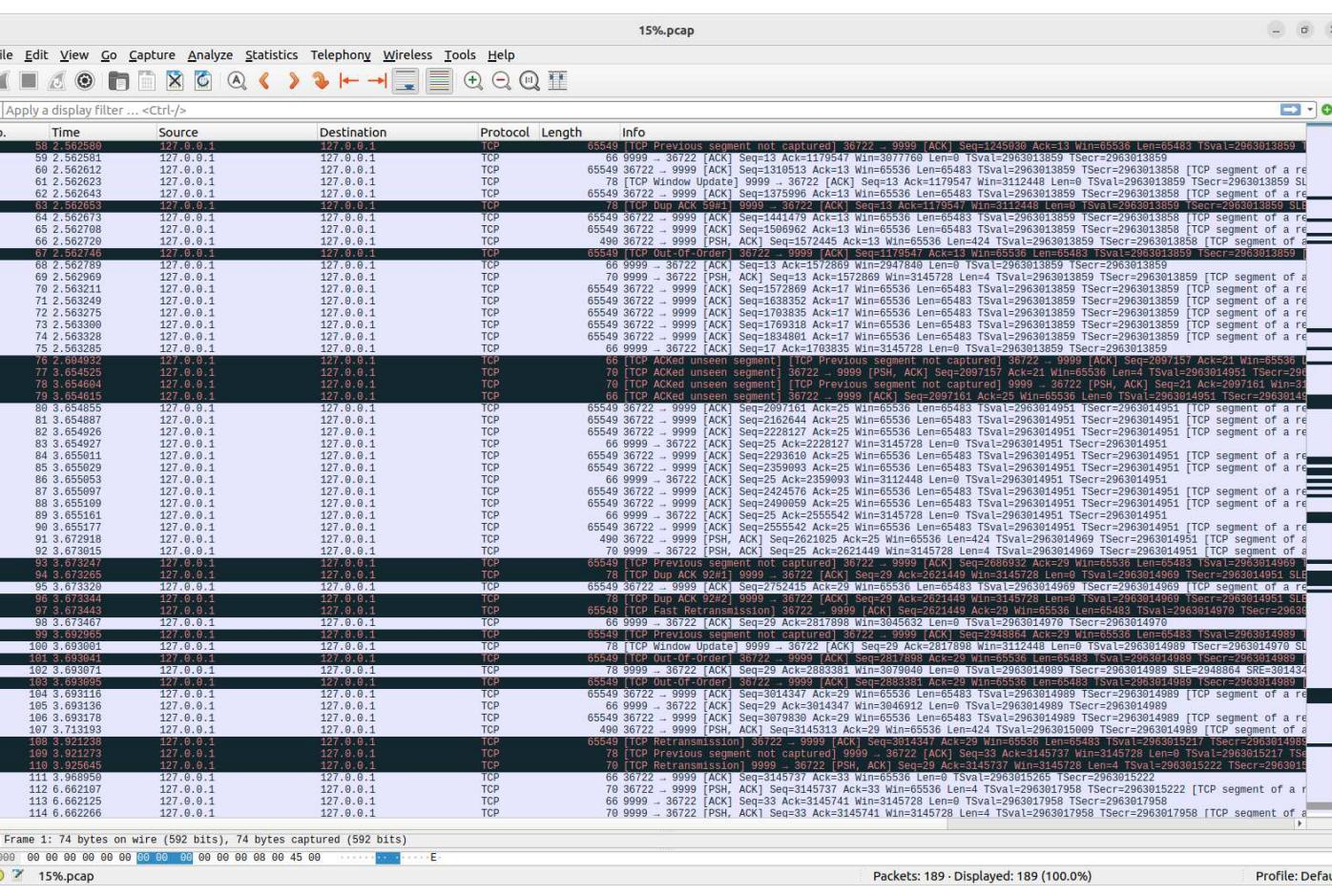
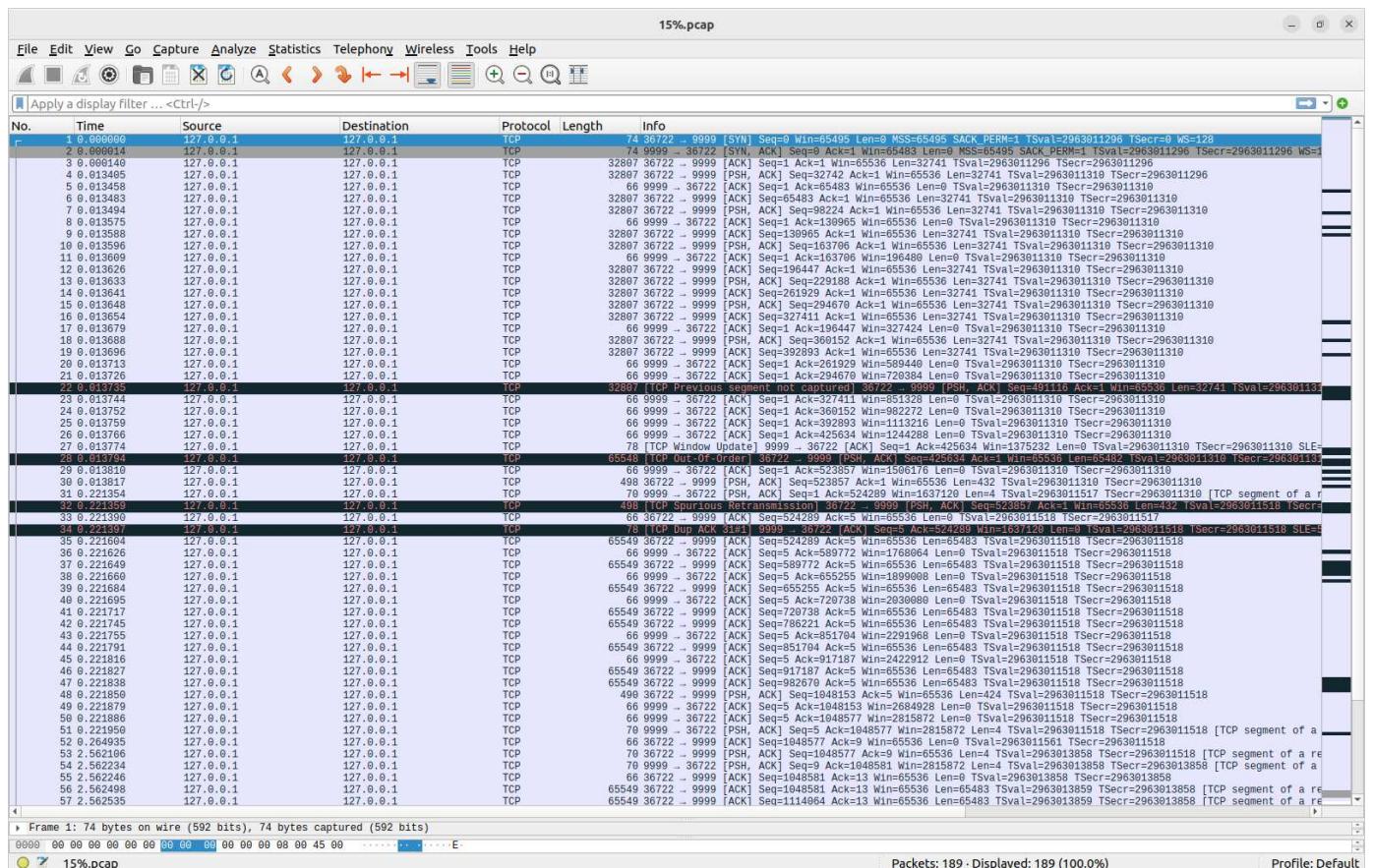
עובר.1 : 10%



No.	Time	Source	Destination	Protocol	Length	Info
115	131.015653	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=3211224 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 S
116	131.015671	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3276707 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
117	131.015697	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3276707 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
118	131.015718	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3342190 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
119	131.015748	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3342190 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
120	131.015769	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3342190 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
121	131.015788	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3342190 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962 S
122	131.015844	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=3458694 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 S
123	131.015892	127.0.0.1	127.0.0.1	TCP	65549	[TCP East Retransmission] 56360 - 9999 [ACK] Seq=3458694 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 SLE=5338639 SRE=360412
124	131.015915	127.0.0.1	127.0.0.1	TCP	78	9999 - 56360 [ACK] Seq=37 Ack=3473156 Win=2945536 Len=4 TSval=1-2962711962 TSscr=2962711962 SLE=5338639 SRE=360412
125	131.015944	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3604122 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
126	131.015958	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 124#] 9999 - 56360 [ACK] Seq=37 Ack=3473156 Win=2945536 Len=4 TSval=1-2962711962 TSscr=2962711962 S
127	131.015967	127.0.0.1	127.0.0.1	TCP	490	56360 - 9999 [PSH, ACK] Seq=3669695 Ack=37 Win=65536 Len=424 TSval=1-2962711962 TSscr=2962711962
128	131.015992	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 56360 - 9999 [ACK] Seq=3473156 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
129	131.016081	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 56360 [ACK] Seq=3473156 Win=3044608 Len=0 TSval=1-2962711962 TSscr=2962711962 S
130	131.016095	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=3670929 Win=3011840 Len=0 TSval=1-2962711963 TSscr=2962711962
131	131.016164	127.0.0.1	127.0.0.1	TCP	70	9999 - 56360 [PSH, ACK] Seq=37 Ack=3670929 Win=3145728 Len=4 TSval=1-2962711963 TSscr=2962711962
132	131.016366	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3670929 Ack=41 Win=65536 Len=65483 TSval=1-2962711963 TSscr=2962711963
133	131.016410	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3735512 Ack=41 Win=65536 Len=65483 TSval=1-2962711963 TSscr=2962711963
134	131.016442	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=3809995 Win=3145726 Len=4 TSval=1-2962711963 TSscr=2962711963
135	131.016454	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3809995 Ack=41 Win=65536 Len=65483 TSval=1-2962711963 TSscr=2962711963
136	131.016496	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3866478 Ack=41 Win=65536 Len=65483 TSval=1-2962711963 TSscr=2962711963
137	131.016778	127.0.0.1	127.0.0.1	TCP	70	[TCP ACKed unseen segment] 9999 - 56360 [PSH, ACK] Seq=397078 Ack=41 Win=65536 Len=0 TSval=1-2962711963 TSscr=2962711962
138	131.016779	127.0.0.1	127.0.0.1	TCP	66	[TCP Previous segment not captured] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=8 TSval=1-2962711949 TSscr=2962711949
139	131.020736	127.0.0.1	127.0.0.1	TCP	65549	[TCP Out-of-Order] 56360 - 9999 [ACK] Seq=4252878 Ack=41 Win=65536 Len=65483 TSval=1-2962711947 TSscr=2962711947
140	131.020745	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Ack=41 Win=65536 Len=65483 TSval=1-2962711947 TSscr=2962711947
141	133.287435	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4194321 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
142	133.287669	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4194321 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474 S
143	133.287701	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=42589492 Ack=53 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
144	133.287729	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Win=3145728 Len=0 TSval=1-2962711474 TSscr=2962711474
145	133.287793	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=43997078 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 S
146	133.287805	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 144#] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474 S
147	133.287820	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4456253 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
148	133.302620	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 56360 - 9999 [ACK] Seq=43252878 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474 S
149	133.302666	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4521736 Win=3045632 Len=0 TSval=1-2962711474 TSscr=2962711474
150	133.302685	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
151	133.302696	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 S
152	133.320638	127.0.0.1	127.0.0.1	TCP	70	9999 - 56360 [ACK] Seq=4521736 Win=3145728 Len=0 TSval=1-2962711474 TSscr=2962711474
153	133.320651	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
154	133.320695	127.0.0.1	127.0.0.1	TCP	65549	[TCP Dup ACK 152#] 9999 - 56360 [PSH, ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474 S
155	133.320702	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
156	133.320729	127.0.0.1	127.0.0.1	TCP	65549	[TCP Out-of-Order] 56360 - 9999 [ACK] Seq=4521736 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
157	131.016778	127.0.0.1	127.0.0.1	TCP	70	[TCP ACKed unseen segment] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
158	131.016779	127.0.0.1	127.0.0.1	TCP	66	[TCP Previous segment not captured] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=8 TSval=1-2962711474 TSscr=2962711474
159	133.287336	127.0.0.1	127.0.0.1	TCP	70	56360 - 9999 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
160	133.287455	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=4194321 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
161	133.287669	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4194321 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
162	133.287701	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4258904 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
163	133.287729	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Win=3145728 Len=0 TSval=1-2962711474 TSscr=2962711474
164	133.287793	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=43997078 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 S
165	131.016778	127.0.0.1	127.0.0.1	TCP	70	[TCP ACKed unseen segment] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
166	131.016779	127.0.0.1	127.0.0.1	TCP	66	[TCP Previous segment not captured] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=8 TSval=1-2962711474 TSscr=2962711474
167	131.016781	127.0.0.1	127.0.0.1	TCP	490	56360 - 9999 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
168	131.016784	127.0.0.1	127.0.0.1	TCP	70	[TCP Dup ACK 153#] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
169	131.016785	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4252878 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
170	131.016786	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 56360 - 9999 [ACK] Seq=4252878 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
171	131.016787	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
172	131.016788	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=4252878 Ack=49 Win=65536 Len=65483 TSval=1-2962711474 TSscr=2962711474
173	131.016789	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
174	131.016790	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474
175	131.016791	127.0.0.1	127.0.0.1	TCP	66	9999 - 56360 [ACK] Seq=4252878 Ack=49 Win=65536 Len=0 TSval=1-2962711474 TSscr=2962711474

No.	Time	Source	Destination	Protocol	Length	Info
129	131.015769	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3407673 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
131	131.015788	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 56360 - 9999 [ACK] Seq=3458694 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 S
132	131.015844	127.0.0.1	127.0.0.1	TCP	65549	[TCP East Retransmission] 56360 - 9999 [ACK] Seq=3458694 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 S
133	131.015892	127.0.0.1	127.0.0.1	TCP	70	9999 - 56360 [ACK] Seq=3604122 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
134	131.015915	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3604122 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
135	131.015944	127.0.0.1	127.0.0.1	TCP	65549	56360 - 9999 [ACK] Seq=3604122 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
136	131.015967	127.0.0.1	127.0.0.1	TCP	490	56360 - 9999 [PSH, ACK] Seq=3669695 Ack=37 Win=65536 Len=424 TSval=1-2962711962 TSscr=2962711962
137	131.016778	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 56360 - 9999 [ACK] Seq=3473156 Ack=37 Win=65536 Len=65483 TSval=1-2962711962 TSscr=2962711962
138	131.016779	127.0.0.1	127.0.0.1	TCP	66	[TCP Previous segment not captured] 9999 - 56360 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=8 TSval=1-2962711949 TSscr=2962711949
139	131.020736	127.0.0.1	127.0.0.1	TCP	70	56360 - 9999 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=424 TSval=1-2962711949 TSscr=2962711949
140	131.020745	127.0.0.1	127.0.0.1	TCP	66	9999 - 5636

15% עבר .2



No.	Time	Source	Destination	Protocol	Length	Info
115 8.662272	127.0.0.1	127.0.0.1	TCP	66	36722	9999 [ACK] Seq=2145741 Ack=37 Win=65536 Len=9 TsvaL=2963017958 TSerC=2963017958
116 8.662507	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=2145741 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
117 6.662545	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=2121244 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
118 6.662578	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=2127687 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
119 6.662608	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3241909 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
120 6.662648	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=3342196 Win=3112448 Len=0 TsvaL=2963017959 TSerC=2963017959
121 6.662691	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured]	36722 9999 [ACK] Seq=3473159 Ack=37 Win=65536 Len=65483 TsvaL=2963017959
122 6.662702	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3407673 Win=3112448 Len=0 TsvaL=2963017959 TSerC=2963017959 SLE=453156 SRE=353805
123 6.662712	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=538639 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
124 6.662722	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3473159 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
125 6.662732	127.0.0.1	127.0.0.1	TCP	489	36722	[TCP Previous segment not captured] 9999 [ACK] Seq=3407673 Win=3112448 Len=0 TsvaL=2963017959 TSerC=2963017959
126 6.662734	127.0.0.1	127.0.0.1	TCP	66	36722	[TCP Dup ACK 122(2)] 9999 36722 [ACK] Seq=3407673 Win=3112448 Len=0 TsvaL=2963017959 TSerC=2963017959
127 6.662778	127.0.0.1	127.0.0.1	TCP	65549	[TCP Fast Retransmission]	36722 9999 [ACK] Seq=3497673 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
128 6.662795	127.0.0.1	127.0.0.1	TCP	78	9999	36722 [ACK] Seq=3407673 Win=3112448 Len=0 TsvaL=2963017959 TSerC=2963017959 SLE=367009
129 6.662808	127.0.0.1	127.0.0.1	TCP	65549	[TCP Out-of-order]	36722 9999 [ACK] Seq=3694122 Ack=37 Win=65536 Len=65483 TsvaL=2963017959 TSerC=2963017959
130 6.662818	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=3670829 Win=3103376 Len=0 TsvaL=2963017959 TSerC=2963017959
131 6.677633	127.0.0.1	127.0.0.1	TCP	70	9999	36722 [PSH, ACK] Seq=3476982 Win=3145728 Len=9 TsvaL=2963018174 TSerC=2963018174
132 6.677792	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3670829 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
133 6.677892	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
134 6.677902	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
135 6.677915	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
136 6.678165	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
137 6.678188	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
138 6.678217	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
139 6.678226	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
140 6.678237	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
141 6.678296	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
142 6.678335	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
143 6.678346	127.0.0.1	127.0.0.1	TCP	499	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
144 6.690452	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
145 6.690923	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3476982 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
146 6.695397	127.0.0.1	127.0.0.1	TCP	66	36722	[ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
147 6.743426	127.0.0.1	127.0.0.1	TCP	70	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
148 7.434337	127.0.0.1	127.0.0.1	TCP	70	9999	36722 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
149 7.434348	127.0.0.1	127.0.0.1	TCP	70	9999	36722 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
150 7.434557	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
151 7.434557	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
152 7.434558	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
153 7.434712	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
154 7.434739	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
155 7.434895	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=3494317 Ack=41 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
156 7.434826	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured]	36722 9999 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018174 TSerC=2963018174
157 7.434840	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 155(1)]	36722 [ACK] Seq=49 Ack=4456253 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
158 7.453500	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4456253 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
159 7.665138	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission]	36722 9999 [ACK] Seq=4456253 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
160 7.665171	127.0.0.1	127.0.0.1	TCP	78	9999	36722 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
161 7.665208	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
162 7.665233	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
163 7.665337	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
164 7.665351	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
165 7.665359	127.0.0.1	127.0.0.1	TCP	499	36722	9999 [PSH, ACK] Seq=4718185 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
166 7.665387	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=4521739 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
167 7.665497	127.0.0.1	127.0.0.1	TCP	70	9999	36722 [PSH, ACK] Seq=4718069 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
168 6.665571	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4718069 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
169 6.665581	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4718069 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
170 7.721139	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4718069 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
171 8.500249	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission]	36722 9999 [ACK] Seq=4718069 Ack=49 Win=65536 Len=65483 TsvaL=2963018173 TSerC=2963018173
172 8.509299	127.0.0.1	127.0.0.1	TCP	78	9999	36722 [ACK] Seq=53 Ack=4915058 Win=3145728 Len=0 TsvaL=2963018917 TSerC=2963018917 SLE=4784895
173 8.509331	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4915058 Win=65536 Len=65483 TsvaL=2963018905 TSerC=2963018905
174 8.509360	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=4988541 Ack=53 Win=65536 Len=65483 TsvaL=2963018905 TSerC=2963018905
175 8.509374	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=5046024 Win=3079049 Len=0 TsvaL=2963018905 TSerC=2963018905
176 8.509392	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=511507 Ack=53 Win=65536 Len=65483 TsvaL=2963018905 TSerC=2963018905
177 8.509421	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=53 Ack=4988541 Win=3112448 Len=0 TsvaL=2963018905 TSerC=2963018905
178 8.509437	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=54 Ack=4988541 Win=3079049 Len=0 TsvaL=2963018905 TSerC=2963018905
179 8.509451	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=53 Ack=511597 Win=3045632 Len=0 TsvaL=2963018905 TSerC=2963018905
180 8.509483	127.0.0.1	127.0.0.1	TCP	65549	36722	9999 [ACK] Seq=53 Ack=511597 Win=3045632 Len=0 TsvaL=2963018905 TSerC=2963018905
181 8.509530	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=53 Ack=5176890 Win=3079049 Len=0 TsvaL=2963018905 TSerC=2963018905
182 8.509521	127.0.0.1	127.0.0.1	TCP	499	36722	9999 [PSH, ACK] Seq=53 Ack=5242743 Win=53 Len=124 TsvaL=2963018906 TSerC=2963018906
183 8.509547	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=53 Ack=5176890 Win=3079049 Len=0 TsvaL=2963018906 TSerC=2963018906
184 8.509554	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=53 Ack=5242897 Win=8013632 Len=0 TsvaL=2963018906 TSerC=2963018906
185 8.509561	127.0.0.1	127.0.0.1	TCP	70	9999	36722 [PSH, ACK] Seq=53 Ack=5242897 Win=3145728 Len=0 TsvaL=2963018906 TSerC=2963018906
186 8.559233	127.0.0.1	127.0.0.1	TCP	66	36722	9999 [ACK] Seq=5242897 Ack=57 Win=65536 Len=0 TsvaL=2963018949 TSerC=2963018949
187 8.662378	127.0.0.1	127.0.0.1	TCP	66	36722	9999 [FIN, ACK] Seq=5242897 Ack=57 Win=65536 Len=0 TsvaL=2963018959 TSerC=2963018959
188 8.662502	127.0.0.1	127.0.0.1	TCP	66	9999	36722 [ACK] Seq=5242898 Win=3145728 Len=0 TsvaL=2963018959 TSerC=2963018959
189 8.662516	127.0.0.1	127.0.0.1	TCP	66	36722	9999 [ACK] Seq=5242898 Ack=58 Win=65536 Len=0 TsvaL=2963018959 TSerC=2963018959

No.	Time	Source	Destination	Protocol	Length</th
-----	------	--------	-------------	----------	------------

3. עברו : 20%

The Wireshark interface displays a list of network captures. The current capture is titled "20%pcap". The packet list shows 192.168.0.1 sending data to 192.168.0.2. The details pane shows the structure of a TCP segment, and the bytes pane shows the raw binary data.

Key details from the Wireshark interface:

- Capture:** 20%pcap
- File:** File → Open...
- Edit:** Edit → Preferences...
- View:** View → Show/Hide Columns...
- Capture:** Capture → Start...
- Analyze:** Analyze → Compute...
- Statistics:** Statistics → Summary...
- Telephony:** Telephony → Call...
- Wireless:** Wireless → Scan...
- Tools:** Tools → Network Minimize...
- Help:** Help → About Wireshark...

Selected packet details:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	74	42392 - 9999 [SYN] Seq=0 Win=65495 SACK_PERM=1 TStamp=2963170656 TSecr=2963170656 WS=128
2	8.000014	127.0.0.1	127.0.0.1	TCP	74	9999 - 42392 [SYN, ACK] Seq=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TStamp=2963170656 TSecr=2963170656
3	8.000026	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TStamp=2963170656 TSecr=2963170656
4	8.000033	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
5	8.000042	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
6	8.000047	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=32742 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
7	8.000055	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=1 Win=65536 Len=9 TStamp=2963170656 TSecr=2963170656
8	8.000062	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=1 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
9	8.000072	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=98224 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
10	8.000089	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TStamp=2963170656 TSecr=2963170656
11	8.000096	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=139965 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
12	8.000103	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=163708 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
13	8.000108	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=163708 Win=65486 Len=0 TStamp=2963170656 TSecr=2963170656
14	8.000134	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=196447 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
15	8.000168	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=229184 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
16	8.000173	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=261939 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
17	8.000177	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=294674 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
18	8.000196	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=196447 Win=327424 Len=0 TStamp=2963170656 TSecr=2963170656
19	8.000203	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [TCP Previous segment not captured] [ACK] Seq=32741 Ack=1 Win=65536 Len=32741 TStamp=2963170656
20	8.000406	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=382918 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
21	8.000411	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=426593 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
22	8.000415	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [ACK] Seq=480775 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
23	8.000419	127.0.0.1	127.0.0.1	TCP	32807	42392 - 9999 [PSH, ACK] Seq=101161 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
24	8.000457	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=229188 Win=65486 Len=0 TStamp=2963170656 TSecr=2963170656
25	8.000463	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=261939 Win=65949 Len=0 TStamp=2963170656 TSecr=2963170656
26	8.000468	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=294670 Win=72038 Len=0 TStamp=2963170656 TSecr=2963170656
27	8.000472	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=1 Ack=327411 Win=85132 Len=0 TStamp=2963170656 TSecr=2963170656
28	8.000477	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=1 Ack=327411 Win=982272 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
29	8.000485	127.0.0.1	127.0.0.1	TCP	32807	[TCP Retransmission] 42392 - 9999 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
30	8.000523	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=1 Ack=327411 Len=1244288 TStamp=2963170656 TSecr=2963170656 SLE=3
31	8.000531	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=1 Ack=327411 Len=137532 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
32	8.000536	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=1 Ack=327411 Len=1506176 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
33	8.00515	127.0.0.1	127.0.0.1	TCP	32807	[TCP Retransmission] 42392 - 9999 [ACK] Seq=327411 Ack=1 Win=65536 Len=32741 TStamp=2963170656 TSecr=2963170656
34	8.005249	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [ACK] Seq=1 Ack=523857 Win=653129 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=327411 SRE=360152
35	8.005368	127.0.0.1	127.0.0.1	TCP	498	42392 - 9999 [PSH, ACK] Seq=523857 Ack=1 Win=65536 Len=432 TStamp=2963170656 TSecr=2963170656
36	8.005479	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [PSH, ACK] Seq=1 Ack=524289 Win=1768064 Len=4 TStamp=2963170656 TSecr=2963170656
37	8.005493	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=524289 Ack=5 Win=65536 Len=0 TStamp=2963170656 TSecr=2963170656
38	8.00516	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=5897/2 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656 SLE=3
39	8.005735	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=5 Ack=524289 Win=1899008 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
40	8.005759	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=655255 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656
41	8.005772	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=5 Ack=524289 Win=2030080 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
42	8.005863	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=720738 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656
43	8.005921	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=5 Ack=524289 Win=2161924 Len=0 TStamp=2963170656 TSecr=2963170656 SLE=3
44	8.005939	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=5 Ack=524289 Win=65483 TStamp=2963170656 TSecr=2963170656
45	8.005957	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=5 Ack=786221 Win=2291968 Len=0 TStamp=2963170656 TSecr=2963170656
46	8.005974	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=786221 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656
47	8.006003	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=851704 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656
48	8.006018	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=5 Ack=851704 Win=65536 TStamp=2963170656 TSecr=2963170656
49	8.006037	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=918178 Win=2553855 Len=0 TStamp=2963170656 TSecr=2963170656
50	8.006077	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=918178 Ack=5 Win=65536 TStamp=2963170656 TSecr=2963170656
51	8.006127	127.0.0.1	127.0.0.1	TCP	499	42392 - 9999 [PSH, ACK] Seq=1048153 Ack=5 Win=65538 TStamp=2963170656 TSecr=2963170656
52	8.006146	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=5 Ack=1048153 Win=24 Win=9999 TStamp=2963170656 TSecr=2963170656
53	8.014151	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [PSH, ACK] Seq=5 Ack=1048577 Win=2946816 Len=4 TStamp=2963170656 TSecr=2963170656
54	8.045155	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=1048577 Ack=5 Win=65536 Len=0 TStamp=2963170656 TSecr=2963170656
55	8.125657	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [PSH, ACK] Seq=9 Ack=1048583 Win=2946816 Len=4 TStamp=2963170656 TSecr=2963170656
56	1.745455	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [PSH, ACK] Seq=9 Ack=1048583 Ack=13 Win=65536 Len=0 TStamp=2963172481 TStamp=2963172182 TSecr=2963170656
57	1.745483	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=1048583 Ack=13 Win=65536 Len=0 TStamp=2963172481 TStamp=2963172182 TSecr=2963170656

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

 20% pcap

Packets: 207 : Displayed: 207 (100.0%)

Profile: Default

20%.pcap							
No.	Time	Source	Destination	Protocol	Length	Info	
115	2.874330	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=29 Ack=2752415 Win=3079949 Len=9 Tsv=2963173539 Tscr=2963173539	
116	2.874347	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=29178949 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
117	2.874358	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 115#1] 9999 - 42392 [ACK] Seq=29178949 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
118	2.874371	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=2883381 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
119	2.874383	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 115#2] 9999 - 42392 [ACK] Seq=29 Ack=2752415 Win=3079949 Len=9 Tsv=2963173539 Tscr=2963173539	
120	2.874398	127.0.0.1	127.0.0.1	TCP	65549	[TCP Fast Retransmission] 42392 - 9999 [ACK] Seq=2752415 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
121	2.874415	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=29 Ack=2948864 Win=2986864 Len=9 Tsv=2963173539 Tscr=2963173539	
122	2.874432	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=2914344 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
123	2.874506	127.0.0.1	127.0.0.1	TCP	78	[TCP Window Update] 9999 - 42392 [ACK] Seq=29 Ack=2948864 Win=3079949 Len=9 Tsv=2963173539 Tscr=2963173539	
124	2.874598	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=2948864 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
125	4.413196	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=2948864 Ack=29 Win=65536 Len=65483 Tsv=2963173539 Tscr=2963173539	
126	4.413352	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [ACK] Seq=29 Ack=3079949 Win=3145728 Len=9 Tsv=2963175067 Tscr=2963175067	
127	4.413369	127.0.0.1	127.0.0.1	TCP	499	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=2914344 Ack=29 Win=65536 Len=65483 Tsv=2963175067 Tscr=2963175067	
128	4.413405	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=2914344 Ack=29 Win=65536 Len=65483 Tsv=2963175067 Tscr=2963175067	
129	4.413426	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=2914344 Ack=29 Win=65536 Len=65483 Tsv=2963175067 Tscr=2963175067	
130	4.413528	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=29 Ack=3145737 Win=3111552 Len=9 Tsv=2963175069 Tscr=2963175069	
131	4.413738	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [PSH, ACK] Seq=29 Ack=3145737 Win=3145728 Len=4 Tsv=2963175069 Tscr=2963175069	
132	4.529580	127.0.0.1	127.0.0.1	TCP	78	[TCP Retransmission] 9999 - 42392 [PSH, ACK] Seq=29 Ack=3145728 Len=4 Tsv=2963175069 Tscr=2963175069	
133	4.626649	127.0.0.1	127.0.0.1	TCP	78	42392 - 9999 [ACK] Seq=3145737 Win=3111552 Len=9 Tsv=2963175069 Tscr=2963175069	
134	6.013347	127.0.0.1	127.0.0.1	TCP	70	42392 - 9999 [PSH, ACK] Seq=3145737 Ack=33 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
135	6.013529	127.0.0.1	127.0.0.1	TCP	70	9999 - 42392 [PSH, ACK] Seq=3145741 Win=3145728 Len=9 Tsv=2963176669 Tscr=2963176669	
136	6.013541	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=3145741 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
137	6.013859	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3145741 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
138	6.013896	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3145741 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
139	6.013975	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=3145741 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
140	6.014097	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3276769 Win=3145728 Len=9 Tsv=2963176669 Tscr=2963176669	
141	6.014130	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3407693 Win=3145728 Len=9 Tsv=2963176669 Tscr=2963176669	
142	6.014167	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=3407693 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
143	6.014191	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3407673 Win=3145728 Len=9 Tsv=2963176669 Tscr=2963176669	
144	6.014255	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3473156 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
145	6.014258	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=3473156 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
146	6.014295	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3473156 Ack=37 Win=65536 Len=65483 Tsv=2963176669 Tscr=2963176669	
147	6.014313	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 145#1] 9999 - 42392 [ACK] Seq=3473156 Win=3145728 Len=4 Tsv=2963176669 Tscr=2963176669	
148	6.014322	127.0.0.1	127.0.0.1	TCP	499	42392 - 9999 [PSH, ACK] Seq=3473156 Win=3145728 Len=4 Tsv=2963176669 Tscr=2963176669	
149	6.014325	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 145#2] 9999 - 42392 [ACK] Seq=3473156 Win=3145728 Len=4 Tsv=2963176669 Tscr=2963176669	
150	6.014362	127.0.0.1	127.0.0.1	TCP	65549	[TCP Fast Retransmission] 42392 - 9999 [ACK] Seq=3473156 Win=3145728 Len=4 Tsv=2963176669 Tscr=2963176669	
151	6.226355	127.0.0.1	127.0.0.1	TCP	70	9999 - 42392 [PSH, ACK] Seq=3670029 Win=3145728 Len=9 Tsv=2963176682 Tscr=2963176682	
152	6.226669	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3670029 Ack=34 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
153	6.226719	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3735512 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
154	6.226752	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=41 Ack=3809959 Win=3145728 Len=9 Tsv=2963176682 Tscr=2963176682	
155	6.226810	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3809959 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
156	6.226810	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3809959 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
157	6.226852	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=3931961 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
158	6.226894	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3931961 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
159	6.226917	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=3931961 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
160	6.226929	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4062927 Ack=44 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
161	6.329727	127.0.0.1	127.0.0.1	TCP	499	[TCP Previous segment not captured] 42392 - 9999 [PSH, ACK] Seq=4193893 Ack=41 Win=65536 Len=424 Tsv=2963176682 Tscr=2963176682	
162	6.845814	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=3931961 Ack=41 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
163	6.845843	127.0.0.1	127.0.0.1	TCP	88	9999 - 42392 [ACK] Seq=41 Ack=3407693 Win=3145728 Len=9 Tsv=2963176682 Tscr=2963176682	
164	6.861539	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=4128419 Ack=41 Win=65536 Len=65483 Tsv=2963176682 Tscr=2963176682	
165	6.861592	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=4194317 Ack=45 Win=65536 Len=65483 Tsv=2963177517 Tscr=2963177517	
166	6.861676	127.0.0.1	127.0.0.1	TCP	70	9999 - 42392 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=65483 Tsv=2963177517 Tscr=2963177517	
167	6.869158	127.0.0.1	127.0.0.1	TCP	66	42392 - 9999 [ACK] Seq=4194317 Ack=45 Win=65536 Len=65483 Tsv=2963177517 Tscr=2963177517	
168	6.892245	127.0.0.1	127.0.0.1	TCP	70	42392 - 9999 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=65483 Tsv=2963177517 Tscr=2963177517	
169	6.892318	127.0.0.1	127.0.0.1	TCP	70	42392 - 9999 [PSH, ACK] Seq=4194317 Ack=45 Win=65536 Len=65483 Tsv=2963177517 Tscr=2963177517	
170	6.892329	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4194321 Ack=49 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
171	6.892327	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4259881 Ack=49 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
172	6.892334	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=4325287 Win=3145728 Len=9 Tsv=2963179079 Tscr=2963179079	
173	6.892337	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4325287 Ack=49 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
174	6.892401	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4397077 Ack=49 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
175	6.892434	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=4456253 Ack=50 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
176	6.892461	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment not captured] 42392 - 9999 [ACK] Seq=457219 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
177	6.892475	127.0.0.1	127.0.0.1	TCP	78	[TCP Dup ACK 151#1] 9999 - 42392 [ACK] Seq=4456253 Win=3145728 Len=9 Tsv=2963179079 Tscr=2963179079	
178	6.892488	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=4456253 Ack=51 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
179	6.918982	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=4456253 Ack=51 Win=65536 Len=65483 Tsv=2963179079 Tscr=2963179079	
180	9.118131	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=4567202 Win=3045632 Len=9 Tsv=2963179774 Tscr=2963179774	
181	9.118151	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4652702 Ack=49 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
182	9.118154	127.0.0.1	127.0.0.1	TCP	499	42392 - 9999 [PSH, ACK] Seq=4652702 Ack=49 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
183	9.118181	127.0.0.1	127.0.0.1	TCP	66	9999 - 42392 [ACK] Seq=4718699 Win=3045632 Len=9 Tsv=2963179774 Tscr=2963179774	
184	9.118344	127.0.0.1	127.0.0.1	TCP	70	42392 - 9999 [ACK] Seq=4718699 Ack=53 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
185	9.118649	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4718699 Ack=53 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
186	9.118701	127.0.0.1	127.0.0.1	TCP	65549	42392 - 9999 [ACK] Seq=4718699 Ack=53 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
187	9.118709	127.0.0.1	127.0.0.1	TCP	65549	[TCP Retransmission] 42392 - 9999 [ACK] Seq=4718699 Ack=53 Win=65536 Len=65483 Tsv=2963179774 Tscr=2963179774	
188	9.118763	127.0.0.1	127.0.0.1	TCP	78	9999 - 42392 [ACK] Seq=4839575 Win=3145728 Len=9 Tsv=2963179774 Tscr=2963179774	
189	9.118787	127.0.0.1	127.0.0.1	TCP	65549	[TCP Previous segment	

: 0% עבר 4

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

Time	Source	Destination	Protocol	Length	Info
58 7.887123	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=11140864 Ack=13 Win=65536 Len=65483 TStamp=2964232673 TSecr=2964232673 [TCP segment of a r...
59 7.887139	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=13 Ack=1179547 Win=3079040 Len=0 TStamp=2964232673 TSecr=2964232673
60 7.887157	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1179547 Ack=13 Win=65536 Len=65483 TStamp=2964232673 TSecr=2964232673 [TCP segment of a r...
61 7.887184	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1245030 Ack=13 Win=65536 Len=65483 TStamp=2964232673 TSecr=2964232673 [TCP segment of a r...
62 7.887193	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=13 Ack=1310513 Win=3112448 Len=0 TStamp=2964232673 TSecr=2964232673
63 7.887212	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1310513 Ack=13 Win=65536 Len=65483 TStamp=2964232673 TSecr=2964232673 [TCP segment of a r...
64 7.887239	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1310513 Ack=13 Win=65536 Len=65483 TStamp=2964232673 TSecr=2964232673 [TCP segment of a r...
65 7.887248	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=13 Ack=14141479 Win=3112448 Len=0 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
66 7.887266	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1441479 Ack=13 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
67 7.887293	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1506962 Ack=13 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
68 7.887302	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=13 Ack=1572445 Win=3112448 Len=0 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
69 7.887310	127.0.0.1	127.0.0.1	TCP	490	53316 - 9999 [PSH, ACK] Seq=1572445 Ack=13 Win=65536 Len=424 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
70 7.887369	127.0.0.1	127.0.0.1	TCP	79	9999 - 53316 [PSH, ACK] Seq=1572869 Ack=17 Win=65536 Len=9 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
71 7.887574	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1572869 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
72 7.887606	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1638352 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
73 7.887629	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1703835 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
74 7.887655	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1769318 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232673 [TCP segment of a r...
75 7.887668	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=17 Ack=1769318 Win=3112448 Len=0 TStamp=2964232674 TSecr=2964232674
76 7.887678	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1834861 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232674 [TCP segment of a r...
77 7.887768	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=1900834 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232674 [TCP segment of a r...
78 7.887767	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=1900834 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232674 [TCP segment of a r...
79 7.887729	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=965767 Ack=17 Win=65536 Len=65483 TStamp=2964232674 TSecr=2964232674 [TCP segment of a r...
80 7.887727	127.0.0.1	127.0.0.1	TCP	66	9999 - 53216 [ACK] Seq=17 Ack=2021259 Win=3112449 Len=0 TStamp=2964232674 TSecr=2964232674
81 7.930919	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] [TCP Previous segment not captured] 53316 - 9999 [ACK] Seq=1607157 Ack=25 Win=65536
82 9.463697	127.0.0.1	127.0.0.1	TCP	79	[TCP ACKed unseen segment] [TCP Previous segment not captured] 9999 - 53316 [PSH, ACK] Seq=2097157 Ack=26 Win=65536
83 9.463611	127.0.0.1	127.0.0.1	TCP	79	[TCP ACKed unseen segment] [TCP Previous segment not captured] 9999 - 53316 [PSH, ACK] Seq=21 Ack=2097161 Win=3...
84 9.463175	127.0.0.1	127.0.0.1	TCP	66	[TCP ACKed unseen segment] [TCP Previous segment not captured] 53316 - 9999 [ACK] Seq=2097161 Ack=25 Win=65536 Len=65483 TStamp=2964234249 TSecr=2964234249
85 9.463390	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2097161 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
86 9.463427	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2162644 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
87 9.463447	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=25 Ack=2228127 Win=3112448 Len=0 TStamp=2964234250 TSecr=2964234250
88 9.463468	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2228127 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
89 9.463561	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2293610 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
90 9.463514	127.0.0.1	127.0.0.1	TCP	66	9999 - 53216 [ACK] Seq=2293610 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
91 9.463528	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2385016 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
92 9.463567	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=242576 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
93 9.463576	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=25 Ack=2409059 Win=3112449 Len=0 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
94 9.463597	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2499959 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
95 9.463628	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2555542 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
96 9.463638	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=25 Ack=2621025 Win=3112448 Len=0 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
97 9.463644	127.0.0.1	127.0.0.1	TCP	490	53316 - 9999 [PSH, ACK] Seq=2621025 Ack=25 Win=65536 Len=424 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
98 9.463683	127.0.0.1	127.0.0.1	TCP	70	9999 - 53316 [PSH, ACK] Seq=25 Ack=2621449 Win=3144064 Len=4 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
99 9.463965	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2621449 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234249 [TCP segment of a r...
100 9.464067	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2668692 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
101 9.464087	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2752415 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
102 9.464084	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=2752415 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
103 9.464065	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2817898 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
104 9.464082	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2817898 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
105 9.464084	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=2820381 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
106 9.464122	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=2820381 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
107 9.464140	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=3014347 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
108 9.464150	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=29 Ack=3014347 Win=3144064 Len=4 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
109 9.464181	127.0.0.1	127.0.0.1	TCP	65549	53316 - 9999 [ACK] Seq=3079830 Ack=25 Win=65536 Len=65483 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
110 9.464192	127.0.0.1	127.0.0.1	TCP	499	53316 - 9999 [PSH, ACK] Seq=3145313 Ack=29 Win=65536 Len=424 TStamp=2964234250 TSecr=2964234250 [TCP segment of a r...
111 9.464265	127.0.0.1	127.0.0.1	TCP	66	9999 - 53316 [ACK] Seq=3145313 Ack=29 Win=65536 Len=65483 TStamp=2964234251 TSecr=2964234251 [TCP segment of a r...
112 9.464236	127.0.0.1	127.0.0.1	TCP	79	9999 - 53316 [PSH, ACK] Seq=29 Ack=3145737 Win=3144064 Len=4 TStamp=2964234251 TSecr=2964234251 [TCP segment of a r...
113 9.511115	127.0.0.1	127.0.0.1	TCP	66	53316 - 9999 [ACK] Seq=3145737 Ack=33 Win=65536 Len=65483 TStamp=2964234251 TSecr=2964234251 [TCP segment of a r...
114 11.111356	127.0.0.1	127.0.0.1	TCP	70	53316 - 9999 [PSH, ACK] Seq=3145737 Ack=33 Win=65536 Len=4 TStamp=2964234251 TSecr=2964234251 [TCP segment of a r...

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 E
0% pcap Packets: 176 · Displayed: 176 (100.0%) Profile: Default

*הפאקטות המסומנות בצבע שחור, הם פאקטות שהמידע לא נשלח בשלמותו

*נשים לב שעבור 0% יש לנו הci פחות פאקטות מסומנות בשחור

-סביר למה cubic עדיף על reno :

TCP Cubic, היא ברירת המחדל הנוכחית בשרתים לינוקס המפעילים חלק גדול מהאינטרנט. ההבדל העיקרי בין TCP Reno ל-TCP Cubic הוא פונקציית הגדלת החלון. בעוד reno משתמש בהגדלה הליניארית המסורתית ($W+1$), Cubic מימושה הגדלת חיפוש ביןארי שיכולה להגיע לרווח הפס הזמין הרבה יותר מהר מאשר reno

-סביר על אלגוריתם CC (congestion control algorithm) CC הינו הטכניקה העיקרית שמונעת מהאינטרנט להאט עקב מצב של ניצול יתר מצב זה נקרא "congestion collapse".

הweeneyון הבסיסי של congestion control הוא שהשלוח משדר פאקטות TCP בראשת, וזו מגיב לאירועים הניטנים לצפיה כדי להגדיל או להקטין את קצב השליחה שלו. אם סבור שהרשות מנוצלת יתר על המידה, עליה להפחית את קצב השליחה שלו, ואם הוא מאמין שהרשות לא מנוצלת, אז היא צריכה להגדיל את קצב השליחה שלו.

מקורות המידע שדריכם עשינו את המחבר :

1. שימוש בספריה- h/Time-Sys

<https://www.youtube.com/watch?v=cunJcNgtxMk>

2. תכנות סוקטים ב-C, שליחת קובץ ב-C:

*תרגולים אנה פרבר

*במדריך הבא (עזר המון ביצירת הפרויקט) :

https://www.youtube.com/watch?v=HWVVEa3seX_E&t=874s

:TCP congestion control .3

*האתר הבא (יש בו הסברים רבים על congestion

:control

<https://witestlab.poly.edu/blog/tcp-congestion-control-basics>

*מדריכים מהיוטיוב מהיוטיוב:

<https://www.youtube.com/watch?v=Aw31XgfXkOs>

https://www.youtube.com/watch?v=XBq_UOe4VS_c

https://www.youtube.com/watch?v=RQ8O13utXQ_w