

## **מטלה 3 רשתות תקשורת:**

### **תוכן עניינים:**

- תיאור מערכת
- מבנה התוכנה
- CC Algorithms-
- צילומי מסך מהוירשארק

### תיאור מערכת:

#### הרכבת התוכנה:

על מנת להריץ את התוכנה נצטרך מערכת הפעלה לינוקס ו4 קבצים:

1. קובץ טקסט בשם test.txt, תוכן ווגדל הקובץ לבחירתכם

Receiver.c .2

Sender.c .3

makefile .4

הרכבת התוכנה תבוצע לפי השלבים הבאים:

1. ניגש לתיקיה בה נמצאים הקבצים הנ"ל נלחץ על מקש ימני ונבחר באפשרות: open in terminal

2. נרץ את הפקודה all

3. נרץ את הפקודה ./Receiver

4. נרץ את הפקודה ./Sender

#### מה התוכנה עשו?

התוכנה מתארת שולח חבילות ומתקבל חבילות ברשות ע"י חיבור TCP

שולח חבילות שולח את הקובץ test.txt אל המקלט.

השליחה מתבצעת ב-2 חלקים , 2 החלקים נשלחים באותו חיבור TCP, החלק הראשון נשלח באמצעות

algoriythm congestion control הנקרא cubic והחלק השני נשלח באמצעות algoriythm congestion control הנקרא reno.

למשתמש בתוכנה יש אפשרות לבחור לאחר כל שליחה האם לשולח את הקובץ בשנית לפי בחירת (זא).

7 להרכבת התוכנה פעם נוספת.

8 ליציאה מהתוכנה וסגירת החיבור בין השולח לבין המקלט.

לאחר יציאה מהתוכנה (E), התוכנה ב- receiver מדפסה זמנים המתארים את זמן שליחת חבילה

בשניות להלן:

(1) זמני שליחה של כל חלק שנשלח בכל פעם שהתוכנית רצתה

(2) זמן ממוצע של שליחת החלק הראשון

(3) זמן ממוצע של שליחת החלק השני

(4) זמן ממוצע של שליחת כל הקובץ.

## מבנה התוכנה:

### :Makefile

קובץ שבעזרתו ניתן להריץ את הפקודה all ויוצר 2 קבצים מוכנים להרצה: .Receiver, Sender

## Sender

**Sender:** מתאר את שלוח הקובץ.

מייצאים ספריות נחוצות לשימוש בסוקטים ובחזרות (את הייבואים מצוינו בקובץ תרגול 5) בנוסף, הגדרנו גודל קבוע המתאים לגודל הקובץ שלנו ומצביע לקובץ hn".l.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
```

### סיכום מה המחלקה sender עשו:

אנחנו קוראים מהקובץ test.txt מחשבים את גודלו בביטים ופתחים סוקט עם כל האלמנטים של addr וip בשבייל לפתח חיבור עם receiver. לאחר שפתחנו חיבור נשלח לreceiver את גודל הקובץ שמצאו כדי שיוכל לדעת כמה ביטים צריך להציג אליו בכל אחד משני החלקים. לאחר מכן נכנס לולאה שלא נצא ממנה עד שהמשתמש החליט לצאת מהתוכנית, ושם נתחיל בכר שנשלח את החלק הראשון של הקובץ, ולאחריו נבצע אוטנטיקציה בשבייל לוודא שהגיע החלק הראשון במלואו, נשנה את הערך CC algorithm cubic1 לשbio receiver בשביל השליחה של החלק השני. לאחר מכן נשלח את החלק השני לנבדק אישור שקיבל את כל החלק השני ובכך בעצם את כל הקובץ. לאחר מכן נבקש מהמשתמש להגיד אם הוא רוצה לשלוח שוב את הקובץ(להמשיך את התוכנית) או לצאת. אם בחר להמשיך נשלח לreceiver את הערך 1 שהוא יודע שאומר שהמשתמש בחר להמשיך, ונשנה שוב את הערך CC algorithm cubic1 לשbio receiver 0 שהוא יודע שאם הוא מקבל שם את המספר 0 זה אומר שהמשתמש בחר לצאת מהערך 0, receiver יחזיר להיות בזיהוי. אך אם המשתמש בחר לצאת אז נשלח את receiver 0, receiver יוציא שאם הוא מקבל שם את המספר 0 זה אומר שהמשתמש בחר לצאת מהתוכנית ואז גם receiver יוציאים מהולאות while המרכזית. והsender פשוט סוגר את הסוקט שפתח בהתחלה.

נתאר במפורט מה עשינו במחלקה sender ומה כל חלק במחלקה עשו: (הסבירים וצילומי מסך מהקובד) מהעמוד הבא.

### הסבר על הפונקציות:

פונקציה לשילוח קובץ המקלט סוקט, מצביע למערך שנרצה לשולח וגודל המערכת.

הfonkzia שולחת את הקובץ באמצעות הפונקציה () send עם ארגומנטים של סוקט, מצביע לבאפר וגודל הבאפר, הפונקציה מחזירה -1 אם אירעה שגיאה , 0 אם החיבור נסגר ומחזירה את כמות הביטים שנשלחו בפועל.

```
int send_file(int sock,char* data,int size){  
  
    int sentbytes=send(sock,data,size,0);  
    if(sentbytes== -1)  
    {  
        perror("Text didnt sent\n");  
        return -1;  
    }  
    else if(sentbytes==0)  
    {  
        printf("The TCP connection has closed before the send()\n");  
    }  
    else if(sentbytes<size)  
    {  
        printf("Sent only %d ot of %d bytes of the text.\n",sentbytes,size);  
    }  
    else  
    {  
        printf("Full massage sent successesfully.\n");  
    }  
    return sentbytes;  
}
```

פונקציה לבדיקת אוטנטיקציה שליחת הקובץ, הפונקציה מקבלת סוקט וממתינה לקבלת אוטנטיקציה מהמקבל, מטרת הפונקציה היא לאמת שהקובץ ששלחנו אכן נשלח באופן תקין, המקלט שולח צירוף ביטים מסוים מראש המבטאゾקס בין 4 ספרות אחרונות של תעודות זהות של המגישים והשלוח מאמת שאכן, הביטים שנשלחו אליו הם הביטים שהוסכם מראש.(היחסוב של הזקס של 4 ספרות האחרונות של שני תעודות זהות: 1553 ו-0319 נעשה כבר על ידינו ולמשתנה הוכנס כבר התוצאה של הזקס מכיוון שכאשר מספר מתחילה ב-0 אז המספר נחשב כמספר בסיס 8 וכך אי אפשר להשתמש בספרה 9 שם וכן לא נתן לנו להריץ, ולכן חישבנו את הזקס בעצמנו מראש).

```
int AuthenticationCheck(int sock)
{
    char* authentication_check="10100001000";
    char check[12];

    recv(sock,&check,sizeof(check),0);

    if(strcmp(authentication_check, check) != 0){
        printf("The receiver didnt get the message that the sender sent\n");
        return 0;
    }
    else
    {
        printf("The receiver get the message completely that the sender sent\n");
        return 0;
    }
}
```

## :Main of the Sender

חלק 1: נקרא את הקובץ test.txt, נמדד את הגודל שלו על ידי הפקנץיה fseek(שלוקחת את המצביע לטו הראשון בקובץ ומגע עד לטו האחרון) ותכניס לנו לתוך משתנה int Size\_of\_file(שמייצגים את הגודל של הקובץ). ולאחר מכן נגידר משתנה נוסף half\_size\_of\_file שיחזק את גודלו של חצי הקובץ.

לאחר מכן נגידר באפר שייחסן את המידע שנמצא בקובץ ונכנס לתוכו את תוכן הקובץ.

(בחנו למדוד את הגודל של הקובץ כדי לחת אפשרות להשתמש ולבדוק את הגודל של כל קובץ שיש לנו שלא בהכרח נדע מראש מה הגודל שלו, כי הקובץ שבו אנחנו משתמשים לא בהכרח באותו גודל של הקובץ של הבדיקה והשימוש בגודל חשוב כדי לדעת כמה זה החצי של השילחה לכל חלק וגם כמה זה חצי שציר לקלט כל חלק בoverver.(receiver).

```
int main()
{
    //step 1:
    //that part is to open the file and read it into a char array
    FILE *pointer_to_file=fopen("test.txt","r");
    if(pointer_to_file==NULL)
    {
        perror("unable to read file\n");
        return -1;
    }
    //fseek() is used to move file pointer associated with a given file to a specific position.
    //using fseek() we moving file pointer to the end
    //then using ftell(), we find its position which it actually size in bytes
    fseek(pointer_to_file,0L,SEEK_END);

    //calculating the size of the file
    int Size_of_file=ftell(pointer_to_file);
    printf("The amount of bytes of the file is: %d\n",Size_of_file);

    //split the size in half for the 2 parts
    int half_size_of_file=Size_of_file/2;

    fseek(pointer_to_file,0L,SEEK_SET);

    //read data from the file and copy it to the char array
    char file_data[Size_of_file];
    fread(file_data,Size_of_file,sizeof(char),pointer_to_file);

    //close the file
    fclose(pointer_to_file);
    printf("read file successfully\n");
```

חלק 2: ייצור חיבור TCP בין השולח למקבל. מצהירים על כתובת IPו שלנו ועל פורט פנוי לבחירתנו יוצרים המתאר חיבור TCP בין השולח למקבל.

```
//step 2:  
//the ip and the port that intended for the socket creation  
char* ip="127.0.0.1";  
int port=5556;  
  
//creating the TCP sender socket descriptor  
int sock=socket(AF_INET, SOCK_STREAM, 0);  
if (sock == -1)  
{  
    perror("Error in socket\n");  
    return -1;  
}  
printf("TCP socket descriptor created successfully.\n");
```

מגדירים מבנה כתובת אינטרנט המתאים לכתובת IP מסוג IPv4 ואליו משיכים את הפורט והIP שהגדרנו נאפס את מבנה הכתובת באמצעות פונקציית (memset). על מנת שנוכל לשולח את הקובץ עם כתובת מסוימת אנחנו ממירם את כתובת IP ליצוג בינארי בעזרת פונקציית ()inet\_ntop() ואות מספר הPORT ביצוג בינארי בתור BIG ENDIAN שזו צורה לקריאת המספר הבינארי המוסכמת מראש ע"י השולח והמקבל וזה מתבצע באמצעות הפונקציה ()htonl.

לאחר מכן אנחנו מקימים חיבור בעזרת פונקציית ()connect המקבלת ארגומנטים את ה-socket, descriptor, מבנה הכתובת שבתוכו הכתובת הרצiosa וגודל המבנה הרלוונטי, הפונקציה מהכחה למקבל שיאשר את החיבור עם השולח. בנוסף, על מנת שהמבנה יהיה עבר כתובת מסוג IPv4 נגדיר את .AF\_INET.sin\_family

```
//defining a ipv4 internet address struct that intended to make  
// the connection between the sender and receiver  
//the ip and the port are going to be converted into binary numbers using inet_ntop() for the ip  
// and htons() for the port in big endian  
//memset function deletes n first chars and replace it by 0  
//sin_family is to identify the address as ipv4  
struct sockaddr_in receiver_addr;  
memset(&receiver_addr, 0, sizeof(receiver_addr));  
receiver_addr.sin_family = AF_INET;  
receiver_addr.sin_port = htons(port);  
  
int addr_bin_conv=inet_ntop(AF_INET, ip , &(receiver_addr.sin_addr));  
if(addr_bin_conv<0){  
    perror("Error in change ip to binary\n");  
    return -1;  
}  
  
//establishing the connection with the receiver  
int connection_request= connect(sock, (struct sockaddr *)&receiver_addr, sizeof(receiver_addr));  
if (connection_request == -1)  
{  
    perror("Error in the connection\n");  
    return -1;  
}  
printf("Connected successfully.\n");
```

אחרי שהתחברנו למקבל, נשלח למקבל את גודל הקובץ שאנו חוננו ממכונינו לשלווח אליו על מנת שהמקבל יוכל לבנות 2 באפרים בגודל חצי מהגודל של כל הקובץ כדי שהמקבל ידע בקבלת המידע מהשולח שהוא קיבל את כל הכמות של ביטים לכל חצי קובץ שהוא מקבל.

```
//sending to the receiver the size of the file in bytes to know how much he need to received for each half of the file;
int send_size_file=Size_of_file;
send(sock,&send_size_file,sizeof(int),0);
printf("sending to the receiver the size of the file.\n ");
```

עתה, נרצה לשלווח את הקובץ שלנו ב-2 חלקים, בחרנו להכניס את כל התהילך זהה בתור לולאת while על מנת שנוכל לחזור על התהילך במידה והמשתמש יבחר 2 (לשלווח שוב את הקובץ).

הגדכנו באפר ואיפסנו אותו, באפר זה מיועד לשימוש בהחלפת CC אלגוריתם ב- socket כך שנוכל להודיע למקבל על שינוי במצב השליחה ובאיזה CC algorithm CC אנחנו משתמשים כרגע.

מכיון שבאופן אוטומטי אנחנו מוגדרים על אלגוריתם CUBIC כאשר פותחים סוקט, אז לא שינוינו שום דבר בנוגע(CC) לאלגוריתם כי הוא כבר מוגדר בזאלט cubic כ- שפוחחים סוקט חדש, ושלחנו את החלק הראשון של הקובץ(חצי מהכמות של כל הקובץ) (חלק 3), לאחר מכן נבצע בדיקת אוטנטיקציה (AuthenticationCheck) (חלק 4).

```
//sending to the receiver the size of the file in bytes to know how much he need to received for each half of the file;
int send_size_file=Size_of_file;
send(sock,&send_size_file,sizeof(int),0);
printf("sending to the receiver the size of the file.\n ");

//sending the 2 messages and changing the cc algorithm every iterate until we tell to exit
while(1)
{
    char buffer[8];
    bzero(buffer,sizeof(buffer));

    //part 3:
    //sending the first half of the file
    printf("sending part 1 of the text\n");
    send_file(sock,file_data,(half_size_of_file));

    //part 4:
    //checking authentication of the first half of the text
    printf("Authentication check\n");
    AuthenticationCheck(sock);
```

חלק 5: בחלק זה אנחנו בודקים באיזה CC אלגוריתם השתמשנו בעזרת הפונקציה () getsockopt(). ואז אנחנו מנסים את האלגוריתם, על מנת לשלוח את החלק השני נשנה את ה-CC אלגוריתם להיות מסוג . RENO

על כן לקחנו את הבאför ובתוכו הכנסנו את המחרוזת reno והשתמשנו בפונקציה () setsockopt() בשבייל לשנות את ה-CC algorithem בסתוקט מ-reno-cubic. לאחר השינוי, אנחנו מעדכנים את המשתמש באלגוריתם הנוכחי שהסתוקט משתמש בכל פעם.

```
//part 5:  
//checking what cc algorithm we using and changing to another cc algo  
//getsockopt() is a function that tells us what cc algorithm the socket uses  
//setsockopt() is a function that change the cc algorithm for the socket  
socklen_t len=sizeof(buffer);  
if(getsockopt(sock,IPPROTO_TCP,TCP_CONGESTION,buffer,&len)!=0)  
{  
    perror("getsockopt failed\n");  
    return -1;  
}  
  
printf("The current CC algo: %s\n",buffer);  
  
strcpy(buffer,"reno");  
len=strlen(buffer);  
  
if(setsockopt(sock,IPPROTO_TCP,TCP_CONGESTION,buffer,len)!=0)  
{  
    perror("setsockopt failed\n");  
    return -1;  
}  
  
len=sizeof(buffer);  
  
if(getsockopt(sock,IPPROTO_TCP,TCP_CONGESTION,buffer,&len)!=0)  
{  
    perror("getsockopt failed\n");  
    return -1;  
}  
  
printf("after changing, the new CC algo: %s\n",buffer);
```

חלק 6: בחלק זה אנחנו שולחים את החלק השני של הקובץ לאחר ששינו את האלגוריתם ל-reno.

נשים לב שהפונקציה (`send_file`) מקבלת הפעם את המצביע לחצי השני של הקובץ.

לאחר השילחה, אנחנו בודקים האם המקבל קיבל את החבילה השנייה כמו שצריך ובצム בכך גם מודיע לנו האם הוא קיבל את הקובץ כלו כמו שצריך, כי אחרי החלק הראשון החלק של האוטנטיקציה בעצם משמש אותנו כמו החלק פה, שבעצם אומר לך אם הוא קיבל את כל החלק הראשון בשלמותו, וכך בודק האם קיבל את כל החלק השני בשלמותו ואם בשניהם נשלח מהמקבל לשולח שהם קיבלו הכל אז זה אומר שהמקבל קיבל את כל הקובץ בשלמותו. כמובן, מחכים לשילוח מהמקבל שוויידיע לנו האם קיבל את כל החלק השני של הקובץ(כל החצי קובץ השני) כדי שנדע אם כל הקובץ התקבל בהצלחה.

אחרי שקיבלנו מהرسיבר את הערך המוסכם אנחנו בודקים האם זה שווה לערך 1 `ack=1`.

אם כן אז המשמש מדפסה הודעת `sync-ok` שאומר שכל הקובץ הגיע בשלמותו.

אך אם קיבל מהرسיבר ערך אחר שלא 1 (אלא 0) אז זה אומר שהرسיבר לא קיבל את כל הקובץ.(זה סתתם בשבייל הבדיקה אבל לפני הבנתי זה לא משווה שי יכול לקרה כי אנחנו בתקשורת TCP וזה תקשורת אמינה כך שלא יכול לקרה מצב שלא הגיע כל מה שלחת).

```
//part 6:  
//start sending the part 2 with the new cc algorothm  
printf("part 2\n");  
printf("sending part 2 of the text\n");  
send_file(sock,(file_data+(half_size_of_file)-1),half_size_of_file);  
  
//checking with the receiver if he received the whole file completely  
int ack=1;  
printf("Waiting for OK signal.\n");  
int ack_got;  
recv(sock,&ack_got,sizeof(int),0);  
if(ack_got==ack)  
{  
    printf("Sync-OK\n");  
}  
else  
{  
    printf("The receiver didnt get the whole file.\n");  
}
```

חלק 7: בחלק זה אנחנו מדפיסים הודעה למשתמש בנוגע להחלטתו האם לשЛОח את הקובץ פעם נוספת או האם לצאת מהתוכנית (Y/E), במידה והקלט לא תקין, נבקש מהמשתמש להכניס את הערך פעם נוספת, אם המשתמש בחר בקלט תקין נמשיך בתוכנית.

```
//part 7:  
//user decision: send the file again (Y) or exit (E)  
char c;  
printf("To send the file again please enter Y, to exit please enter E\n");  
scanf(" %c",&c);  
  
while(c != 'Y' && c != 'E')  
{  
    printf("There is no such option, please enter again Y/E:\n");  
    scanf(" %c",&c);  
    if(c == 'Y' || c == 'E'){  
        break;  
    }  
}
```

כעת, לאחר שהמשתמש בחר Y או E אנחנו בודקים מה הוא הכניס.

אם הכניס Y (שאומר לשЛОח שוב פעם את הקובץ) אז ניצור משתנה send\_shicil את הערך 1 ונשלח אותו לשידוע שם הוא מקבל את הערך 1 והוא ממשיר בתוכנית, ונעדכן את הalgo להזוז להיות שוב פעם cubic בעזרת פונקציית() על מנת שבאייטרציה הבאה החלק הראשון ישלח בmodo CUBIC CC algorithem .RENO ולא באמצעות

```
//send_if_yes gets the value 1 if the sender decided to send the file again  
//then we send the send_if_yes to the receiver so the receiver will know that if he got the value 1 so he need to continue running the while loop again  
//and send_if_exit get the value 0 if the sender decided to exit  
//then we send the send_if_exit to the receiver so the receiver will know that if he got the value 0 he will know that the sender decide to exit and he will exit too  
  
//if the decision is yes (send the file again)  
if(c == 'Y')  
{  
    //send the bufferReply with the value 1  
    int send_if_yes=1;  
    send(sock,&send_if_yes,sizeof(int),0);  
  
    //change back to cc algo cubic  
    printf("changing back the CC algo to cubic.\n");  
    strcpy(buffer,"cubic");  
  
    if(setsockopt(sock,IPPROTO_TCP,TCP_CONGESTION,buffer,strlen(buffer))!=0)  
    {  
        perror("setsockopt failed\n");  
        return -1;  
    }  
}
```

אך אם המשתמש בחר E אז המשתמש יכנס למשתנה 0 וישלח לreceiver את הערך 0 ויגד שהוא יצא ואז יצא מהלולאת while הראשית. כך receiver יידע שאם הוא מקבל 0 אז הוא יודע שהמשתמש בחר לצאת מהתוכנית וכך גם receiver יעשה.

לאחר יציאה מהלולאה הראשית נסגור את הסקט שיצרנו ונודיע שהסקט נסגר ונחזיר 0 כי זה פונקציית .int main

```
        else
        {
            //send the bufferReply with the value 0
            int send_if_exit=0;
            send(sock,&send_if_exit,sizeof(int),0);

            printf("exiting...\n");
            break;
        }

    }
    //closing the sender socket
    close(sock);

    printf("Connection closed.\n");

    return 0;
}
```

## Receiver

**מתאר את מקבל הקובץ: Receiver**

בקובץ זה בחרנו לא ליצר פונקציות כמו שיצרנו בsender מכיוון שלא היה לנו זמן.

מייבאים ספריות נוחיות לשימוש בסוקטים, במחוזות ולמדידת זמנים.

(את הייבואים מצאנו בקובץ תרגול 5)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <netinet/tcp.h>
#include <netinet/in.h>
#include <sys/time.h>
```

**סיכום מה מחלוקת receiver עשו:**

יצירם חיבור TCP בין השולח למקבל , אנחנו לא פותחים סוקט חדש לגמרי אלא יוצרים סוקט המתאים לսוקט שהשולח יצר. כמובן שנעשה bind listening לכמה האזנות אפשר במקביל להקליב. לאחר מכן נתחבר לsender על ידי זה שנעשה את פונקציה accept ובכך ניצור סוקט חדש בעצם שבוא יעביר הדבירים שנשלחים אחד לשני. נקלט מהsender את הגודל המלא של כל הקובץ כדי שנוכל לדעת כמה צריך לקבל בכל אחד משני החלקים. לאחר מכן ניכנס לולאה שתסתהים רק כאשר קיבל 0 מהמשתמש בסוף שיאמר שהוא בחר לצאת מהתוכנית. בתוך הלולאה יתקיים קבלת הביטים בשני החלקים כל אחדחצי מהגודל של כל הקובץ כולל ויחסוב הזמןם של חלק לכל חלק להגיע והנכמתם למערך כדי לעשות את החישובים של ממוצעים אחרי היציאה המתוכנית. בין הקבלה של החלק הראשון לקבלה של החלק השני נוצרך שוב פעם לשנות את המוחם CC cubic algoritm לשונו redo ובכך שזה יהיה בהתאם לאלגוריתם שיש בשילוחה של שני החלקים. אחרי קבלת החלק הראשון נשלח את האוטנטיקציה לsender כדי לאשר לו שהגיע כל החלק הראשון, ובסוף קבלת החלק השני במלואו נשלח לsender אישור שקיבלנו את כל החלק השני של הקובץ ובכך נאשר לו שקיבלנו את כל הקובץ במלואו. לאחר מכן נמתין לקבלת הערך 1 או 0 מהsender שיעיד אם הוא בחר שלשלוח שוב את הקובץ או שבחר לצאת. אם בחר להשאר נשנה שוב את המוחם CC לשונו cubic כדי שייעבוד בהתאם עם השילוחה של החלק הראשון בזיהוי ונעשה איטרציה נוספת לקובץ(נעשה שוב את הלולאת while המרכזית), אך אם בחר לצאת (קובל 0) ולכן יצא מהLOOP את המרכזית נסגור את הסוקט של החיבור ביןינו ואת הסוקט שפתחנו בהתחלה receiver ונדפס את הזמן של כל חלק בכל איטרציה שהיא ונחשב את הממוצעים שהיא לכל חלק וכל הקובץ כולו.

**נתאר במפורט מה עשינו בחלוקת receiver ומה כל חלק בחלוקת עשו:**  
**(הסברים וצילומי מסך מהקובץ מהעמוד הבא.)**

## Main of the Receiver

חלק 1: בחלק זה אנחנו יוצרים חיבור TCP בין השולח למקבל , אנחנו לא פותחים סוקט חדש לגמרי אלא יוצרים סוקט המתאים לסוקט שהשלוח יצר.

מגדירים את הפורט להיות הפורט המתאים לפורט שהשלוח יצר על מנת שהשלוח והמקבל ידעו דרך איזה פורט מתקשרים בשכבה האפליקציה (שכן אם הפורט בינם יהיה שונה השולח והמקבל לא יוכל לתקשר בינם)

יוצרים socket descriptor , סוקט זה מיועד להיות הסוקט שדרכו נאזרן לחיבור TCP דרך הפורט הנ"לvr כרל מעשה , החיבור הראשון בין השולח למקבל נוצר.

מגדירים מבנה כתובת אינטרנט המתאים לכתובת IP מסוג IPv4 ואליו נשאיר את הפורט שהגדכנו נאפס את מבנה הכתובת באמצעות פונקציית () memset .

את הפורט אנחנו מושנים לייצוג בינהרி בצורה BIG ENDIAN ע"י פונקציית () htons

את כתובת הIP אנחנו מגדירים להיות כתובת כללית שתתקבל כל חיבור TCP ע"י שימוש בכתובת inet\_nton ANY\_INADDR כתובות זו כבר מומרת לייצוג בינהרி ועל כן לא השתמשנו בפונקציית () htons

כמו כן, על מנת שהמבנה יהיה עבור כתובת מסוג IPv4 נגדיר את sin\_family להיות AF\_INET

```
int main(){
    int port=5556;

    //step 1:
    //creating the TCP receiver socket descriptor
    int listen_socket=socket(AF_INET,SOCK_STREAM,0);
    if(listen_socket== -1)
    {
        perror("Error in socket\n");
        return -1;
    }
    printf("TCP socket descriptor created successfully.\n");

    //defining a ipv4 internet address struct that intended to make
    //the connection between the sender and receiver
    //the ip and the port are going to be converted into binary numbers
    //using htons() for the port in big endian
    //memset function deletes n first chars and replace it by 0
    //sin_family is to identify the address as ipv4
    struct sockaddr_in receiver_addr;
    memset(&(receiver_addr),0,sizeof(receiver_addr));
    receiver_addr.sin_family=AF_INET;
    receiver_addr.sin_port=htons(port);
    receiver_addr.sin_addr.s_addr=INADDR_ANY;
```

לפעמים מנסים להפעיל שרת אך קישור bind נופל עם הודעה הבאה : “Address already in use”  
הבעיה נובעת מכך שבית קטן של סוקט שהיה מחובר, עדין בשימוש זהה תופס פורט. על מנת לאפשר שימוש חוזר של פורט, השתמשנו בפונקציה ()setsockopt עם הכתובת בזיכרון של הערך 1 (יכול להיות כל ערך)

לאחר מכן נפעיל את הפונקציה ()bind על מנת לקבוע את כתובות הIP והPORT על מנת שנוכל להאזין דרך הsockt שלנו, בפועל זה אנחנו בעצם ”קושרים“ את הsockt ואת מבנה הכתובת ייחודי.

אכן ניתן לראות כי הפונקציה ()bind מקבלת ארגומנטים את הsockt, את מבנה הכתובת ואת גודל המבנה הנ”ל ומחזירה 1 - אם המבצעת שגיאה. אחרת, ()bind ה被执行 בהצלחה .

כעת, נפעיל את הפונקציה ()listen על מנת להתחיל להאזין לחבריהם דרך הsockt.

הפונקציה ()listen מקבלת ארגומנטים את הsockt שלנו ואת מספר החיבורים שניין לבצע בו זמנית, במקרה זה בחורנו במספר 5 באופן שירוטי. הפונקציה מחזיר 1 - אם התרחשה שגיאה.

לאחר מכן אנחנו מגדרים מבנה כתובות חדש שיכיל את הכתובת שנקל מהשולח לאחר יצירת החיבור הראשוני, מבנה זה מיועד להיות המבנה שדרכו נשלח לשולח סימנים מוסכמים מראש מבנה זה יקרא .sender\_addr

```
//to enable reuse with this port, we do this:
int yes=1;
if(setsockopt(listen_socket,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int))==-1)
{
    perror("setsockopt failed\n");
    return -1;
}

//bind the senders' ip and port for the TCP connection
int bind_check = bind(listen_socket , (struct sockaddr*)&receiver_addr,sizeof(receiver_addr));
if(bind_check== -1){
    perror("Error in binding\n");
    return -1;
}
printf("bind successfully\n");

//waiting for incoming connections by listen() function
if(listen(listen_socket,5)==-1){
    perror("Error in listening\n");
    return -1;
}
printf("Listening...\n");

//defining the sender ipv4 address struct to save the senders data
struct sockaddr_in sender_addr;
socklen_t sender_len= sizeof(sender_addr);
memset(&sender_addr,0,sizeof(sender_addr));
```

חלק 2: בחלק זה אנחנו מקבלים את בקשת החיבור של השולח באמצעות הפונקציה `(accept)` המקבלת כารוגומנטים את הsocket ואת מבנה הכתובת `sender_addr` כך שלמעשה הפונקציה תכניס את הערכים המתקיים מהפעלה `(connect)` אצל השולח אל הsocket החדש שנוצר בשם `sender_socket` שיכיל את המבנה החדש שלנו וכן למשה גם נשמר את כתובתו של השולח.

cutet בשבייל כל הש寥ות וקבלת של דברים מהסנדר נשתמש בסוקט החדש `sender_socket`.  
הפונקציה מחזירה 1 - אם ארצה בעיה בחיבור.

לאחר שקרנו לפונקציה `(accept)` החיבור הראשון בין השולח למקבל הטענו בהצלחה  
לאחר מכן, `recv` מקבל את גודל הקובץ שהשולח מתכוון לשולח על מנת לאותל את הבאים  
בהתקם(כל באפר מקבל חצי מהגודל של הקובץ).

ניתן לראות בסוף התמונה שהגדכנו משתנים חדשים, משתנים אלה מייעדים להיות מערכים שייקבלו את  
זמן השליחה של כל חלק שנשלח, כאשר כל אינדקס 0 יסמן את הפעם הוא קיבלנו את הקובץ, גודל  
המערך הוא 1000 מכיוון שרצינו לקחת טווח בטיחות למספר הש寥ות שייתבצע.(לפי מיל שלוחתם לנו,  
הגדרתם לנו שאפשר להגדיר את הגודל של המערך בגודל 1000 כי אנחנו יכולים להניח שלא ישלח לנו  
יותר מ1000 בקשות לש寥ת קובץ ושמירת הזמן של).

```
//step 2:  
//accepting the senders' connect() request.  
//When a sender connects, the method returns a new socket object representing the connection.  
//so that our new socket contains the senders socket data  
int sender_socket=accept(listen_socket,(struct sockaddr*)&sender_addr,&sender_len);  
if(sender_socket==1)  
{  
    perror("Error in accept\n");  
    return -1;  
}  
printf("Connection accepted!\n");  
  
//receiving from the sender the size of the file that we will know how much we need to recv in each half  
int file_size_from_sender;  
recv(sender_socket,&file_size_from_sender,sizeof(int),0);  
printf("The size of the file is: %d\n",file_size_from_sender);  
  
//define 2 arrays that will save the time of each half  
//the size of the arrays is 1000 because we assume that we can't ask more than 1000  
//requests (sent mail about it, that we can assume that)  
  
double time_of_first_half[1000];  
bzero(time_of_first_half,sizeof(time_of_first_half));  
double time_of_second_half[1000];  
bzero(time_of_second_half,sizeof(time_of_second_half));
```

נאתחל משתנה count\_times שיספור את כמות הפעמים שהמשתמש בחר לשלוח את הקובץ פעם נוספת.

בנוסף, משתנה still\_running המזענד להגדיר לולאת while מתי להפסיק לשלוח שוב, נשים לב שהמשתנה זהה לשטנה -0 כאשר הבוחר יבחר לצאת או כאשר תתרחש שגיאה.

כעת, נרצה לשלוח את הקובץ שלו ב-2 חלקים, בחרנו להכניס את כל התהילך זהה בתור לולאת while על מנת שנוכל לחזור על התהילך במידה והמשתמש יבחר 2.

בתוך הלולאה, נאתחל משתנה total\_byte\_received שיספור את כמות הביטים הכלולות שהתקבלו מהשלוח(כדי לוודא שבשני החלקים קיבלנו בדיקן חצי מהקובץ המקורי).

בתוך הלולאה מכינים באפר בשם bufferReply שיקבל את החלק הראשון של הקובץ ומ-appendים אותו.

נשים לב שאנו רוצים שככל פעםCSI יש שילחה של החלק הראשון ונרצה ששני הסוקטים אחד של החיבור ואחד של הסנדר יהיו באותו cubic(CC algorithm cubic) ובשליחת חלק השני מקבלת השילחה של החלק השני ונרצה שני הסוקטים יהיו באותו reno(CC algorithm reno).

אנחנו מדפיסים שכרגע ה-CC algorithm הוא cubic(מכיוון שכשפותחים סוקט חדש אין default CC algorithm ולכן אין מה לשנות לו את האלגוריתם כי השילחה של החלק הראשון הוא cubic).

```
//counts the amount of time that the file received
int count_times=0;

//we start a loop to receive the 2 part of the file every iterate until the sender tells to exit

int stillRunning=1;
while(stillRunning==1){

    int total_bytes_received=0;
    char bufferReply[file_size_from_sender/2];
    bzero(bufferReply,sizeof(bufferReply));

    //the first time we create the socket his defult cc algo is cubic(so in the first time we enter
    //but after this, every iterate we change back from reno to cubic
    //((the change from reno to cubic is before the end of the while loop, that it will happend if t
    printf("now the cc algorithm is cubic.\n");
    printf("Ready to receive the first half of the file.\n");
}
```

אתחול זמנים על מנת להתחיל לספור את הזמן שלוקח לחיבור להגיון.(זכור את הזמן המדယק בשעון שהגענו אליו כרגע, שנגיעה לאחר recv\_time\_half1 של gettimeofday().

אנו יכולים לחשב את הזמן על ידי החישוב בהמשך).

```
//measure the time of receive of the first half of the file
struct timeval before_recv_time_half1, after_recv_time_half1;
gettimeofday(&before_recv_time_half1,NULL);
```

חלק 3: בחלק זה המקלט יקבל את החלק הראשון של הקובץ, פועלה ממוקמת בולולאת while על מנת לוודא שקיבלנו את כל החלק הראשון מהשלוח, נוודה זאת באמצעות המשתנה total\_byte\_received

נשתמש בפונקציה () recv על מנת לקבל את הקובץ, פונקציה זו מקבלת כ\_ARGUMENTים את sender\_socket ותפקיד bufferReply ובוסף את גודל ה

- bufferReply

 פונקציה זו תואמת לפונקציית () send של השולח.

אם הפונקציה מחזירה -1 – אז התרחשה שגיאה במהלך הקבלה ו-0 אם השולח סגר את החיבור, במקרה בו הפונקציה החזירה 0 נשנה את stillRunning ל-0, שני המקרים נמצאים מהתכנית, אחרת זה אומר שאין בעיה והתתקבל כמות בייטים מסויימת שבסיום צריכה שהגיעו להגעה להיות חצי מגודל הקובץ.

```
//step 3:  
//receiving the data from the sender until we get the full amount of the first half  
while(total_bytes_received<(file_size_from_sender/2))  
{  
    int ReceivedBytes=0;  
    ReceivedBytes=recv(sender_socket,bufferReply,sizeof(bufferReply),0);  
    total_bytes_received+=ReceivedBytes;  
    //checking if there is no exception with the ReceivedBytes bytes  
    if(ReceivedBytes== -1)  
    {  
        perror("recv failed\n");  
        return -1;  
    }  
    else if(ReceivedBytes==0)  
    {  
        printf("Sender exited, exiting...\n");  
        stillRunning = 0;  
        break;  
    }  
}
```

נוודה שם המשתנה ערך stillRunning נמצא מהלולאה הראשית ונסגור את החיבור.  
בנוסף אחרי שייצאנו מLOOPLET ה

while
 שגיאות זה אומר שקיבלנו את כל הכמות של החלק הראשון של הקובץ וכן נשים שוב את the time gettimeofday כדי שיבדק את הזמן(זמן המקיים השעון באותו רגע) שהגענו אליו כדי לבצע את החישוב שכמה זמן לערך קיבל את החלק הראשון של הקובץ.

```
if (stillRunning == 0)  
{  
    ...  
    break;  
}  
  
//measure the time that the message was arrived  
gettimeofday(&after_recv_time_half1,NULL);
```

חלק 4: בחלק זה נמדד את הזמן שלקח לחלק הראשון של הקובץ להגעה, בacr שבמיסטנה השניות נבצע חישוב של הזמן אחרי הקבלה של כל החלק הראשון של הקובץ בזמן שלפני הקבלה של הקובץ ונקבל את השניות שלקחת. ובוצעו אותו דבר בשבייל המיקרו שניות כפי שאפשר לראות בתמונה.

```
//step 4:
double seconds=(double) (after_recv_time_half1.tv_sec - before_recv_time_half1.tv_sec);
double microseconds=(double) (after_recv_time_half1.tv_usec - before_recv_time_half1.tv_usec);
```

חלק 5: בחלק זה נבצע את המשוואה שמחשבת את הזמן לחלק הראשון של הקובץ לפי החישובים בחלק 4 ושמור את הזמן, שלקח מרגע השילוח הראשונה עד קבלת כל החלק הראשון של הקובץ, בתוך מערך הזמן של החלק הראשון של הקובץ ונדפס שהחלק הראשון של הקובץ התקבל בהצלחה.

```
//step 5:
time_of_first_half[count_times]=seconds+ microseconds*le-6;

printf("The first part received.\n");
```

חלק 6: בחלק זה נשלח אוטנטיקציה לשולח בעזרת פונקציית send כדי לוודא שהחלק הראשון של הקובץ התקבל בשילומו, נשים לב שאט ערך הוזע חישבנו מראש. לאחר מכן נשנה את CC אלגוריתם להיות מסוג reno. ניצור מערך בשם changeCC\_buffer שם נקבעו אותו. נכנס לתוכו את הסטרנאג reno ואז משתמש בפונקציה (setsockopt) בשבייל לשנות את האלגוריתם בסוקט לreno. ונדפס למשתמש ששינויו את האלגוריתם.

```
//step 6:
//sending the authentication to the sender again to confirm that the
//first half of the file received completely
//in authentication_check we calculated the xor already
char* authentication_check="10100001000";

int authentication_send=send(sender_socket,authentication_check,strlen(authentication_check),0);

if(authentication_send > 0)
{
    printf("sent ACK to the sender.\n");
}

printf("the first half of the file received completely.\n");

//changing the CC algo from cubic in to reno
char changeCC_buffer[8];
bzero(changeCC_buffer,sizeof(changeCC_buffer));
strcpy(changeCC_buffer,"reno");
int len=strlen(changeCC_buffer);

if(setsockopt(listen_socket,IPPROTO_TCP,TCP_CONGESTION,changeCC_buffer,len) != 0)
{
    perror("setsockopt failed\n");
    return -1;
}

printf("now the cc algorithm is reno\n");
```

לאחר מכן נכין באפר נוספת שיקבל את החלק השני של הקובץ ונתחל משתנה שיספור את סך כמות הביטים שהגיעו בחבילת השניה כדי לבדוק שהגיע כל הכמות של החלק השני של הקובץ, ובנוסף נתחל את המשתנים המיעדים למדידת הזמן של רקח לחלק השני של הקובץ להגעה.

```

total_bytes_received=0;
char bufferReply2[file_size_from_sender/2];
bzero(bufferReply2,sizeof(bufferReply2));

//measure the time of receive the second half of the file
struct timeval before_recv_time_half2,after_recv_time_half2;

```

שלב 7: בחלק זה המקלט יקבל את החלק השני של הקובץ , פועלה ממוקמת בלולאת while על מנת לוודא שקיבלנו את כל החלק השני מהשלוח, נוודא זאת באמצעות המשתנה .total\_byte\_received

נשים לב שלפי הכניסה ללולאת while יש את הערך gettimeofday של לפני ההתחלה של הקבלת הביטים מהשלוח בשבייל לחשב את הזמן של החלק השני של הקובץ כמו שהוא בחלק הראשון של הקובץ.

נשתמש בפונקציה () recv על מנת לקבל את הקובץ, פונקציה זו מקבלת ארגומנטים את sender\_socket וגודל bufferReply2 ובנוסף את גודל ה-2 פונקציה זו תואמת לפונקציית () send של השולח.

אם הפונקציה מחזירה -1 אז התרחשה שגיאה במהלך הקבלה ו-0 אם השולח סגר את החיבור, במקרה בו הפונקציה החזירה 0 נשנה את stillRunning ל-0, בשני המקרים נמצא מהתכנית, אחרת זה אומר שאין בעיה והתקבל כמות ביטים מסוימת שבסיום צריכה להגיע להיות חצי מגודל הקובץ .

```

//step 7:
//receiving the second half of the file
printf("Ready to receive the second half of the file.\n");

gettimeofday(&before_recv_time_half2, NULL);

while(total_bytes_received<(file_size_from_sender/2))
{
    int ReceivedBytes=0;
    ReceivedBytes=recv(sender_socket,bufferReply2,sizeof(bufferReply2),0);
    total_bytes_received+=ReceivedBytes;

    //checking if there is no exception with the ReceivedBytes bytes
    // if its dont -1 or 0 its mean the we got amount of bytes of the first half of the file from the sender
    if(ReceivedBytes==1)
    {
        perror("Error with recv \n");
        return -1;
    }
    else if(ReceivedBytes==0)
    {
        printf("Sender exited, exiting...\n");
        stillRunning = 0;
        break;
    }
}

```

נודא האם השתנה ערך stillRunning נזע מהלולאה הראשית ונסגור את החיבור.

בנוסף אחריו שיצאנו מLOOPת הwhile בלי שגיאות זה אומר שקיבלו את כל הכמות של החלק השני של הקובץ וכן נשים שוב את gettimeofday כדי לבדוק את הזמן(הזמן המקורי השוען באותו רגע)שהגענו אליו כדי לבצע את החישוב שכמה זמן לוקח לקבל את החלק השני של הקובץ.

```
if (stillRunning == 0)
{
    break;
}

gettimeofday(&after_recv_time_half2 , NULL);
```

חלק 8: בחלק זה נמדד את הזמן שלוקח לחלק השני של הקובץ להגיע, בכך שבמשתנה השניות נבצע חיסור של הזמן אחרי הקבלה של כל החלק הראשון של הקובץ לזמן שלפני הקבלה של הקובץ ונקבל את השניות שלוקח. ובוצע אותו דבר בשביל המיקרו שניות כפי שאפשר לראות בתמונה.

```
//step 8:
seconds=(double) (after_recv_time_half2.tv_sec - before_recv_time_half2.tv_sec);
microseconds=(double) (after_recv_time_half2.tv_usec - before_recv_time_half2.tv_usec);
```

חלק 9: בחלק זה נבצע את המשווה שמחשבת את הזמן לחלק השני של הקובץ לפי החישובים בחלק 8 ונשמר את הזמן, שלוקח מרגע השיליחה הראשונה עד קבלת כל החלק של הקובץ, בטור מערכ הזמן של החלק השני של הקובץ ונדפס שהחלק השני של הקובץ התקבל בהצלחה.

נשים לב שעכשו נגדיל את count\_times ב-1 מכיוון שיש לנו לקבל את כל הקובץ ועכשו אם משתמש יבחר שוב פעם לשלוח את הקובץ אז החישוב זמנים של הפעם הבאה יהיה בתא אחריו בזיכרון ולא ידרשו את הזמן שהוא באותו איטרציה.

נשלח לSENDר אישו הودעת אישור שקיבלו את כל הקובץ ובחורנו לשלוח לו אישזהו סימן מסכם(כן בחרנו שנשלח את הערך של stillRunning [שזה 1]).

ומדפיס שנשלח הודעת אישור לSENDר.

```
//step 9:
time_of_second_half[count_times]=seconds+ microseconds*1e-6;

count_times++;

printf("Second part received.\n");

//sending a ack for receiving the full file
send(sender_socket, &stillRunning, sizeof(int), 0);

printf("Sync packet sent.\n");
```

חלק 10: בחלק זה נבדוק האם המשתמש בחר לשלוח את הקובץ פעם נוספת, בכינוס למשתנה מסוג int not run או run\_not את הערך שנקבל מהשולח ע"י פונקציית recv().

במידה ותוכן not run הוא 0 אז מקבל יודע שהשולח הולך לסגור את חיבור TCP, מדפיס שהשולח החליט לצאת ולכן גם הרסיבר יצא, בנוסף הוא משנה את הערך של stillRunning ל-0 ועושה break כדי לצאת מהלולאה ולא לעשות אותה שוב וסגור את הסוקטים שפתחנו.

לאחר מכן מקבל ידפיס את הזמןים ממערכיו הזמינים.

נשים לב שבמידה והמשתמש בחר להמשיך לשלוח שוב את הקובץ אז הערך של not run הוא 1 וכאן הערך של still\_running לא השתנה וכך נחזור אליו לאיטרציה הבאה כולם, מקבל את הקובץ פעם נוספת.

בנוסף נשים לב, שאם המשתמש בחר לשלוח שוב פעם את הקובץ אז צריך לשנות שוב את ה CC algorithm Reno לcubic, כי כל פעם צריך להיות התאמה בין האלגוריתם שיש בשילוח לאלגוריתם שיש בסוקט של הקבלה, כך שבחלק הראשון יהיה שימוש בcubic ובחלק השני יהיה שימוש בסוכן, ומכךון שאנו חזרנו חזרם לשילוח החלהן צריך לשנות שוב לcubic.

לכן משתמש בbabper שיצרנו לפניו (changeCC\_buffer), נאפס את הבابر, ונגדי לו שם שוב את CC algorithm cubic. ואז נשתמש שוב פעם בפונקציה (setsockopt) בשילוב לשנות את המתריג Cubic Reno.

```
//step 10:  
//receiving the resending status  
int run_or_not;  
recv(sender_socket,&run_or_not,sizeof(int),0);  
if(run_or_not==0){  
    printf("Sender exited, exiting...\n");  
    stillRunning=0;  
    break;  
}  
//if the user decided to continue the program and send the file again  
//changing back cc algo from rno to cubic if the sender decide to continue  
//the program(the defult cc aglo that the socket have is cubic)  
bzero(changeCC_buffer,sizeof(changeCC_buffer));  
strcpy(changeCC_buffer,"cubic");  
len=strlen(changeCC_buffer);  
  
if(setsockopt(listen_socket,IPPROTO_TCP,TCP_CONGESTION,changeCC_buffer,len) != 0)  
{  
    perror("setsockopt failed\n");  
    return -1;  
}  
  
//closing the sender socket and the listen socket  
close(sender_socket);  
close(listen_socket);
```

אחרי שיצאנו מהלולאת while המרכזית, נחשב את סך הזמן הכללי ואת ממוצע הזמןים לכל אחד מחלק קובץ ובנוסף נחשב ממוצע גם לכל הקובץ ולאחר מכן נdfs למשתמש את הזמן הממוצע שלקח לכל חיבור להגעה ואת הזמן הממוצע שלקח לכל הקובץ להגעה(חלק ראשון של הקובץ, חלק שני של הקובץ והקובץ המלא).

```
//printing the time took to each half in each iterate
for(int i=0;i<count_times;i++){
    printf("Run %d of first half using cubic: %lf seconds.\n", i+1 , time_of_first_half[i]);
    printf("Run %d of second half using rno: %lf seconds.\n", i+1 , time_of_second_half[i]);
}
//calculating the sum of time for each half sent
double sum_half_1=0;
double sum_half_2=0;

for(int i=0;i<count_times;i++){
    sum_half_1+=time_of_first_half[i];
    sum_half_2+=time_of_second_half[i];
}

//calculating time average for each half and for the while file
double average_half_1=sum_half_1/count_times;
double average_half_2=sum_half_2/count_times;
double average_full_file= (sum_half_1+sum_half_2)/(2*count_times);

printf("The average time for the first half: %lf seconds.\n",average_half_1);
printf("The average time for the second half: %lf seconds.\n",average_half_2);
printf("The average time of the full file is : %lf\n",average_full_file);

printf("connection closed.\n");

return 0;
}
```

## :CC Algorithms

Congestion control או בעברית: בקרת גודש הוא מנגנון השולט בכניסת חבילות של נתונים לרשת, המאפשר שימוש טוב יותר ברשת והימנע מקריסה הנגרמת בשל גודש.

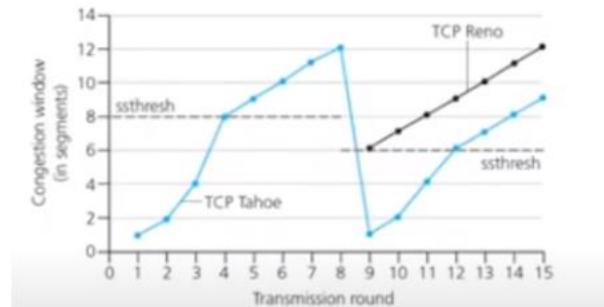
הנתונים CC Algorithms הם אלגוריתמים למניעת גודש ברשת הפעלים על ידי זיהוי גודש והתאמת קצב העברת הנתונים כדי להימנע מכך. זה מבטיח שהרשת תהיה שמישה לכלם.

על מנת להסביר טוב יותר את המושגים נזכיר כמה מושגים הרלוונטיים לנושא זה:  
(Congestion Window) Cwnd : מספר חבילות לא מאושרות (MSS) בכל רגע נתון שיכולות לעבור.  
חולון הגודש גדול, יורד או נשאר זהה בהתאם לכמה מהחבילות הראשונות אושרו וכמה זמן לוקח לאשר.  
(Slow Start threshold) Ssthresh : חסם עליון.

## :Reno

זהו אלגוריתם המבוסס על מנגנון ההתחלה איטית.

זה מתחילה בשלב ההתחלה האיטית שבו ה-cwnd גדל באופן אקספוננציאלי עד שהוא מגיע לערך ssthresh, ולאחר מכן ה-cwnd גדל באופן ליניארי. כאשר אובדן מזוהה בגלל ACKs כפולים, ה-cwnd וה-ssthresh מוגדרים שניהם למחצית הערך הנוכחי שלהם, ולאחר מכן, ה-cwnd גדל באופן ליניארי.



קיבלנו עזרה מהאתר: <https://granulate.io/blog/understanding-congestion-control>

### :Cubic

האלגוריתם זהה מיישם קונספט של התחלת היברידית.

במקרה ההתחלה האיטית שנמצאת באלגוריתם CUBIC , cwnd מובסס על הזמן מאז התרחש אירוע החדש האחרון ולא על מהירות שבתא ACKים מתקיים:

$$cwnd = C^*(t-K)^3 + W_{max}$$

$$K = \text{cubic\_root}(W_{max} * (1 - \text{beta\_cubic}) / C)$$

כasher:

Beta\_cubic הוא גורם הירידה ההפוך.

$W_{max}$  הוא גודל החלון ממש לפני ההפחתה האחרונות.

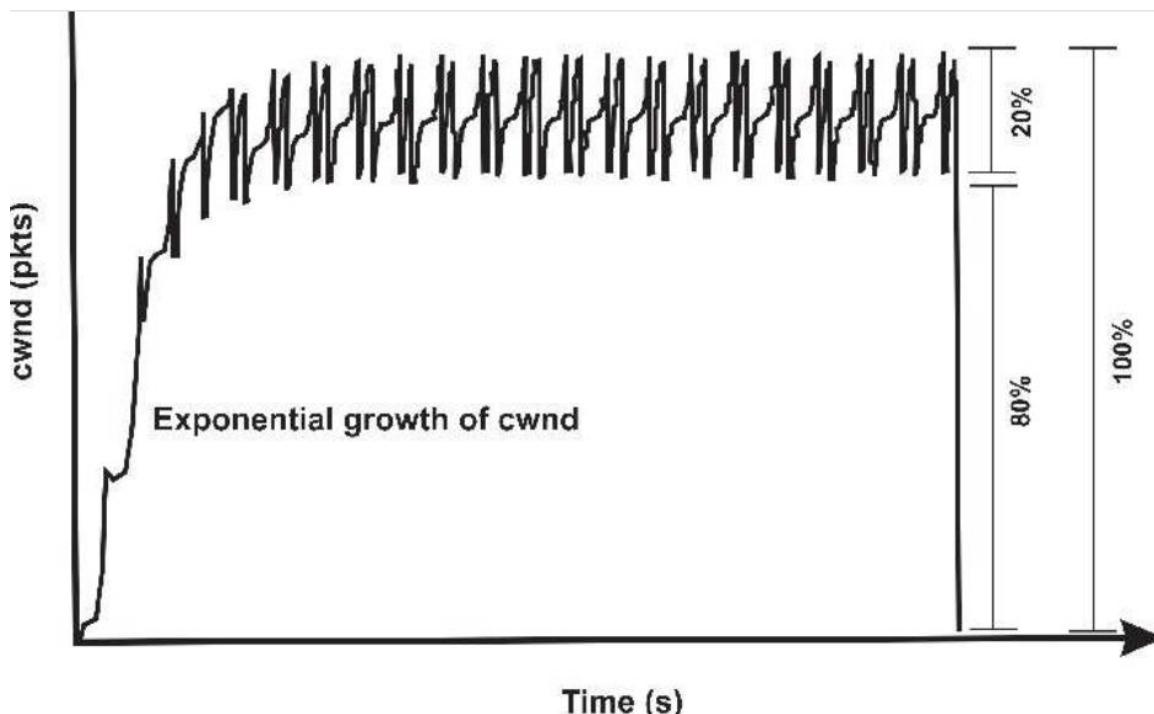
T הוא הזמן שהלך מאז ההפחתה האחרונה.

C הוא קנה מידת קבוע.

cwnd הוא חילון החדש בזמן הנוכחי.

זה מאפשר להתחלה היברידית לצאת במהירות משלב ההתחלה האיטית ומעבר לשלב ההימנעות מוגדר ללא איבוד רב של מנוקות. לפי מחקר השוואתי, התחלת היברידית משפרת את זמן האתחול של חיבור TCP פי שניים עד שלושה.

לכן זמן הריצה הממוצע של cubic יהיה מהיר יותר מאשר reno על הניר, כי הוא יותר מיועד לאיבוד פאקטות, וכן בזמן ממוצע של השילחה עד רגע הקבלה הוא מהיר יותר. אך נשים לב שזה לא צה חד צדדי (קורה לפעמים שבהן מהיר יותר ככל שהאחוז האיבודים עולה).



קיבלנו עזרה מהאתר: <https://granulate.io/blog/understanding-congestion-control>

## צילומי מסך מהווירשאrk:

ראשית, נכנסנו לוירשאrk ובחרנו באפשרות הבאה:

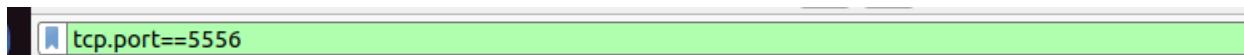
The screenshot shows the Wireshark interface with the 'Capture' tab selected. A filter bar at the top contains the placeholder 'Enter a capture filter ...'. Below it is a list of interfaces and other capture options:

- enp0s3
- any
- Loopback: lo** (highlighted with a blue background)
- bluetooth-monitor
- nflog
- nfqueue
- dbus-system
- dbus-session
- Cisco remote capture: ciscodump
- DisplayPort AUX channel monitor capture: dpauxmon
- Random packet generator: randpkt
- systemd Journal Export: sdjournal
- SSH remote capture: sshdump
- UDP Listener remote capture: udppdump

לאחר מכן, נשתמש בכל הגדינה איבוד פקודות לפי אחוזים.

בסך הכל נבעצ' זאת 4 פעמים עבור: 0%, 10%, 15%, 20%, עברו כל איבוד פקודות בגובה % נשלח את הקובץ 5 פעמים.

נשים לב שאנו מסננים בכל ההקלות של כל איבוד הפקודות לפי פרוטוקול TCP ולפי הפורט 5556(שזה הפורט שבו השתמשנו לsocket).



נסביר ונראה צילומי מסך מה-wireshark ומהטרמינל עברו כל אחד מאחוזי איבוד הפקודות:

## עבור 0% איבוד פאקטות:

הריצו את התוכנית 5 פעמים ולבסוף התוכנית הדפיסה לנו את הזמן הרלוונטיים:

```
Run 1 of first half using cubic: 0.000319 seconds.
Run 1 of second half using reno: 0.000316 seconds.
Run 2 of first half using cubic: 0.000120 seconds.
Run 2 of second half using reno: 0.000552 seconds.
Run 3 of first half using cubic: 0.000159 seconds.
Run 3 of second half using reno: 0.000331 seconds.
Run 4 of first half using cubic: 0.000256 seconds.
Run 4 of second half using reno: 0.002520 seconds.
Run 5 of first half using cubic: 0.000400 seconds.
Run 5 of second half using reno: 0.000513 seconds.
The average time for the first half: 0.000251 seconds.
The average time for the second half: 0.000846 seconds.
The average time of the full file is : 0.000549
connection closed.
```

## להלן צילומי מסך מהוירשארק כאשר יש 0% איבוד פאקטות:(הקלטה בשם "lost.pcapng")

ניתן לראות את ביצוע החיבור הראשמי (SYN,SYN ACK) בין השולח לבין המקלט ותחילת שליחת חבילות ביןיהם וקבלתן (ACK)(חץ שחזור),בנוסף ניתן לראות שליחת חבילות מידע (PSH, ACK) (חץ אדום)

1 0.000000000	127.0.0.1	127.0.0.1	TCP	74 44436 - 5556 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=762169325 TSeср=0 WS=128
2 0.000014849	127.0.0.1	127.0.0.1	TCP	74 5556 - 4436 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=762169325 TSeср=762169325
3 0.000025993	127.0.0.1	127.0.0.1	TCP	66 44436 - 5556 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=762169325 TSeср=762169325
4 0.000066841	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=762169325 TSeср=762169325
5 0.000073496	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=32742 Win=48640 Len=0 TSval=762169325 TSeср=762169325
6 0.000085934	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=32742 Ack=1 Win=48640 Len=0 TSval=762169325 TSeср=762169325
7 0.00009458743	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=32741 Win=65536 Len=0 TSval=762169325 TSeср=762169325
8 0.0000985870	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=65483 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
9 0.0000973380	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=65483 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
10 0.0000949200	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=130965 Win=65536 Len=0 TSval=762169325 TSeср=762169325
11 0.0000951877	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=130965 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
12 0.0000956261	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=163769 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
13 0.00009513829	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=163769 Ack=1 Win=163766 Len=0 TSval=762169325 TSeср=762169325
14 0.00009519749	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=196447 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
15 0.00009524614	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=229180 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
16 0.00009529643	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=261929 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
17 0.00009534218	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=294674 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
18 0.00009538377	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=327411 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
19 0.00009556445	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=196447 Win=327424 Len=0 TSval=762169325 TSeср=762169325
20 0.00009562714	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=360152 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
21 0.00009567481	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=392893 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
22 0.00009571886	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=425634 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
23 0.00009576247	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=458375 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
24 0.00009580887	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=491116 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
25 0.0000959164	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=229188 Win=458496 Len=0 TSval=762169325 TSeср=762169325
26 0.00009665197	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=523857 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
27 0.00009669949	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=556594 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
28 0.00009614384	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=589339 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
29 0.00009626862	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=261929 Win=589440 Len=0 TSval=762169325 TSeср=762169325
30 0.00009633807	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=622088 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
31 0.00009638881	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=654821 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
32 0.00009646911	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=294670 Win=720384 Len=0 TSval=762169325 TSeср=762169325
33 0.00009652976	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=687562 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
34 0.00009657435	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [ACK] Seq=720383 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325
35 0.00009666307	127.0.0.1	127.0.0.1	TCP	66 5556 - 4436 [ACK] Seq=1 Ack=327411 Win=851328 Len=0 TSval=762169325 TSeср=762169325
36 0.0000972496	127.0.0.1	127.0.0.1	TCP	32807 44436 - 5556 [PSH, ACK] Seq=753044 Ack=1 Win=65536 Len=0 TSval=762169325 TSeср=762169325

השורות השחורות מסמנות לנו שהוירשארק לא ראה ACK על החבילה שנשלחה (חץ כחול), או שהחביבה הקודמת לא נתפסה(חץ אדום).

ניתן לראות שמכיוון שזה ב% 0 איבוד פאקטות אז אין לנו שום שורות שביהם רשום ack או tcp dup ack ובה tcp connection על tcp retransmissiontcp out of order תקשורת אמינה ולכן סתם ככה לא יאבדו לנו פאקטות.

114 6.534266112	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 5556 - 44436 [ACK] Seq=46 Ack=7641843 Win=3112448 Len=0 TSval=762175859 TSecr=
115 6.534302040	127.0.0.1	127.0.0.1	TCP	77 [TCP ACKed unseen segment] 5556 - 44436 [PSH, ACK] Seq=46 Ack=7693133 Win=3112448 Len=11 TSval=762175859
116 6.534306589	127.0.0.1	127.0.0.1	TCP	66 [TCP Previous segment not captured] 44436 - 5556 [ACK] Seq=7693133 Ack=57 Win=65536 Len=0 TSval=762175859
117 6.534391986	127.0.0.1	127.0.0.1	TCP	65549 44436 - 5556 [ACK] Seq=7693133 Ack=57 Win=65536 Len=65483 TSval=762175859 TSecr=762175859
118 6.534458068	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 44436 - 5556 [ACK] Seq=7824099 Ack=57 Win=65536 Len=65483 TSval=762175859
119 6.534654216	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 44436 - 5556 [ACK] Seq=8413446 Ack=57 Win=65536 Len=65483 TSval=762175859
120 6.534735249	127.0.0.1	127.0.0.1	TCP	65549 [TCP Previous segment not captured] 44436 - 5556 [ACK] Seq=8675378 Ack=57 Win=65536 Len=65483 TSval=762175859
121 6.534745432	127.0.0.1	127.0.0.1	TCP	66 [TCP ACKed unseen segment] 5556 - 44436 [ACK] Seq=57 Ack=8740861 Win=3112448 Len=0 TSval=762175859 TSecr=

## עבור 10% איבוד פאקטות:

נפעיל בטרמינל את הפקודה הבאה:

```
amitim@amitim-VirtualBox:~/CLionProjects/EX_3_networks$ sudo tc qdisc add dev lo root netem loss 10%
amitim@amitim-VirtualBox:~/CLionProjects/EX_3_networks$
```

הפעלנו את שליחת החבילה 6 פעמים וקיבלנו את הזמן:

```
sender exiting, exiting...
Run 1 of first half using cubic: 0.001485 seconds.
Run 1 of second half using reno: 0.001585 seconds.
Run 2 of first half using cubic: 0.000291 seconds.
Run 2 of second half using reno: 0.000631 seconds.
Run 3 of first half using cubic: 0.000221 seconds.
Run 3 of second half using reno: 0.043739 seconds.
Run 4 of first half using cubic: 0.071671 seconds.
Run 4 of second half using reno: 0.069699 seconds.
Run 5 of first half using cubic: 0.000473 seconds.
Run 5 of second half using reno: 0.001364 seconds.
Run 6 of first half using cubic: 0.014891 seconds.
Run 6 of second half using reno: 0.458318 seconds.
The average time for the first half: 0.014839 seconds.
The average time for the second half: 0.095889 seconds.
The average time of the full file is : 0.055364
connection closed.
```

ניתן לראות שבממוצע cubic (CC algo for first half of the file)reno (CC algo for second half of the file) יתור מאריך מרגע השילחה עד רגע הקבלה הוא מהיר יותר. אך נשים לב שהזמן לא כזה חד צדי (קורה לפעמים שהסוכן בממוצע מהיר יותר ככל שהאחוז האיבודים עולה).

נשים לב בנוסף שהזמן שלקח בממוצע בשני החלקים גבוה יותר מאשר ב-0% איבוד, שמדובר שהזמן שלקח לחבילות להגעה לקח יותר זמן, שמעיד על זה שהיא איבוד פאקטות ובכמות גדולה יותר.

## להלן צילומי מסך מהוירשארק כאשר יש 10% איבוד פאקטות:(הקלטה בשם "lost.pcapng")

נשים לב שכשרשים localhost ב-source וב-destination איז הכוונה לכתובת ip . 127.0.0.1

Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)

אנו רואים בצילום שנאבדים פאקטות בכל מין דרכיהם:

דבר זה מעיד לנו שפאקטה נאבדה (כי קיבלנו בעצם הודעה ack פעמיים על אותה פאקטה) מה שאומר שהשולח יctrer שוב לשולח את החבילות האלה.(חץ שחזור).

Tcp : זה אומר שפאקטה נשלחה בסדר לא נכון(שאומר ששלחנו אותו במקומות מסוימים אך הוא התקבל במקומות שונים[נגיד הוא נשלח ראשון אך התקבל אצל הרטסיבר רק במקום השלישי]).(חץ אדום).

Tcp retransmission : זה שידור חוזר של פאקטה שלא הגיע עלייה ack.(חץ יrok)

ניתן לראות מזאת שיש איבוד פאקטות, אך בכמות קטנה יותר מאשר באיבוד של 15% או 20%.

כמובן שישנו עד פאקטות שנתקפסו שלא עשינו עליהם צילומי מסך שאפשר לראות אותם בהקלטה בשם "10% lost.pcapng".

No.	Time	Source	Destination	Protocol	Length	Info
316	7.156791141	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=11775987 Ack=76 Win=65536 Len=65483 TStamp=1526135870 TSecr=1526135870
317	7.156792810	localhost	localhost	TCP	66	freeciv(5556) → 39996 [ACK] Seq=76 Ack=11775987 Win=3145728 Len=0 TStamp=1526135870 TSecr=1526135870
318	7.156927982	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 39996 → freeciv(5556) [ACK] Seq=11906953 Ack=76 Win=65536 Len=65483 TStamp=1526135870 TSecr=1526135870
319	7.156941079	localhost	localhost	TCP	78	freeciv(5556) → 39996 [ACK] Seq=76 Ack=11841470 Win=3080320 Len=0 TStamp=1526135870 TSecr=1526135870 SLE=11906953 SRE=1...
320	7.156952706	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=11972436 Ack=76 Win=65536 Len=65483 TStamp=1526135870 TSecr=1526135870
321	7.156962871	localhost	localhost	TCP	78	[TCP Dup ACK 319#1] freeciv(5556) → 39996 [ACK] Seq=76 Ack=11841470 Win=3080320 Len=0 TStamp=1526135870 TSecr=1526135870
322	7.156977357	localhost	localhost	TCP	51356	39996 → freeciv(5556) [PSH, ACK] Seq=12837919 Ack=76 Win=65536 Len=51290 TStamp=1526135870 TSecr=1526135870
323	7.156985149	localhost	localhost	TCP	78	[TCP Dup ACK 319#2] freeciv(5556) → 39996 [ACK] Seq=76 Ack=11841470 Win=3080320 Len=0 TStamp=1526135870 TSecr=1526135870
324	7.157008352	localhost	localhost	TCP	65549	[TCP Fast Retransmission] 39996 → freeciv(5556) [ACK] Seq=11841470 Ack=76 Win=65536 Len=65483 TStamp=1526135870 TSecr=1...
325	7.157023263	localhost	localhost	TCP	66	freeciv(5556) → 39996 [ACK] Seq=76 Ack=12089209 Win=2988544 Len=0 TStamp=1526135870 TSecr=1526135870
326	7.378935997	localhost	localhost	TCP	77	freeciv(5556) → 39996 [PSH, ACK] Seq=76 Ack=12089209 Win=3145728 Len=11 TStamp=1526136092 TSecr=1526135870
327	7.379101717	localhost	localhost	TCP	66	39996 → freeciv(5556) [ACK] Seq=12089209 Ack=87 Win=65536 Len=0 TStamp=1526136092 TSecr=1526136092
328	7.379738560	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12089209 Ack=87 Win=65536 Len=65483 TStamp=1526136092 TSecr=1526136092
329	7.379934249	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 39996 → freeciv(5556) [ACK] Seq=12220175 Ack=87 Win=65536 Len=65483 TStamp=1526136092
330	7.379949719	localhost	localhost	TCP	78	freeciv(5556) → 39996 [ACK] Seq=87 Ack=12154692 Win=3080320 Len=0 TStamp=1526136093 TSecr=1526136093 SLE=12220175 SRE=1...
331	7.379984638	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12285658 Ack=87 Win=65536 Len=65483 TStamp=1526136093 TSecr=1526136093
332	7.380505703	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12351141 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136093
333	7.3898522600	localhost	localhost	TCP	78	[TCP Dup ACK 330#1] freeciv(5556) → 39996 [ACK] Seq=87 Ack=12154692 Win=3080320 Len=0 TStamp=1526136094 TSecr=1526136094
334	7.380537658	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12416624 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136093
335	7.380667994	localhost	localhost	TCP	65549	[TCP Retransmission] 39996 → freeciv(5556) [ACK] Seq=12154692 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136094
336	7.381047962	localhost	localhost	TCP	66	freeciv(5556) → 39996 [ACK] Seq=87 Ack=12482107 Win=2980864 Len=0 TStamp=1526136094 TSecr=1526136094
337	7.381075765	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12482107 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136094
338	7.381101382	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12547590 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136094
339	7.381124578	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12613073 Ack=87 Win=65536 Len=65483 TStamp=1526136094 TSecr=1526136094
340	7.381126189	localhost	localhost	TCP	66	freeciv(5556) → 39996 [ACK] Seq=87 Ack=12613073 Win=3012224 Len=0 TStamp=1526136095 TSecr=1526136094
341	7.381159507	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 39996 → freeciv(5556) [ACK] Seq=12744039 Ack=87 Win=65536 Len=65483 TStamp=1526136094
342	7.381173449	localhost	localhost	TCP	78	freeciv(5556) → 39996 [ACK] Seq=87 Ack=12678556 Win=3112448 Len=0 TStamp=1526136095 TSecr=1526136094 SLE=12744039 SRE=1...
343	7.381193186	localhost	localhost	TCP	65549	39996 → freeciv(5556) [ACK] Seq=12800922 Ack=87 Win=65536 Len=65483 TStamp=1526136095 TSecr=1526136095
344	7.381205656	localhost	localhost	TCP	78	[TCP Dup ACK 342#1] freeciv(5556) → 39996 [ACK] Seq=87 Ack=12678556 Win=3112448 Len=0 TStamp=1526136095 TSecr=1526136095
345	7.381564732	localhost	localhost	TCP	66	[TCP ACKed unseen segment] freeciv(5556) → 39996 [ACK] Seq=87 Ack=13071454 Win=3145728 Len=0 TStamp=1526136095 TSecr=15...
346	7.381589711	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 39996 → freeciv(5556) [ACK] Seq=13071454 Ack=87 Win=65536 Len=65483 TStamp=1526136095
347	7.381871288	localhost	localhost	TCP	51356	39996 → freeciv(5556) [PSH, ACK] Seq=13138937 Ack=87 Win=65536 Len=51290 TStamp=1526136095 TSecr=1526136095
348	7.382348077	localhost	localhost	TCP	66	[TCP ACKed unseen segment] freeciv(5556) → 39996 [ACK] Seq=87 Ack=13188227 Win=3145728 Len=0 TStamp=1526136096 TSecr=15...
349	7.382406834	localhost	localhost	TCP	70	freeciv(5556) → 39996 [PSH, ACK] Seq=87 Ack=13188227 Win=3145728 Len=4 TStamp=1526136096 TSecr=1526136095
350	7.382606656	localhost	localhost	TCP	66	39996 → freeciv(5556) [ACK] Seq=13200927 Ack=87 Win=65536 Len=65483 TStamp=1526136095 TSecr=1526136095

## עבור 15% איבוד פאקטות:

נפעיל בטרמינל את הפקודה הבאה:

```
matan@matan-VirtualBox:~/Desktop/Network and communicait/matala 3$ sudo tc qdisc change dev lo root netem loss 15%
matan@matan-VirtualBox:~/Desktop/Network and communicait/matala 3$
```

העלונו את שליחת החבילה 5 פעמים וקיבלנו את הזמןאים הבאים:

```
Run 1 of first half using cubic: 0.000958 seconds.
Run 1 of second half using reno: 0.005652 seconds.
Run 2 of first half using cubic: 0.015723 seconds.
Run 2 of second half using reno: 0.043598 seconds.
Run 3 of first half using cubic: 1.106662 seconds.
Run 3 of second half using reno: 2.440889 seconds.
Run 4 of first half using cubic: 0.000564 seconds.
Run 4 of second half using reno: 0.086371 seconds.
Run 5 of first half using cubic: 0.081786 seconds.
Run 5 of second half using reno: 0.003845 seconds.
Run 6 of first half using cubic: 0.000799 seconds.
Run 6 of second half using reno: 0.235035 seconds.
Run 7 of first half using cubic: 0.454529 seconds.
Run 7 of second half using reno: 0.771639 seconds.
The average time for the first half: 0.237289 seconds.
The average time for the second half: 0.512433 seconds.
The average time of the full file is : 0.374861
```

נשים לב ששוב פעם הזמן הממוצע של cubic(CC algo for first half of the file) הוא מהיר יותר מאשר reno(CC algo for second half of the file). cubic מיועד לאיבוד פאקטות ולכן בממוצע הזמן מרגע השילחה עד רגע הקבלה הוא מהיר יותר. אך נשים לב שהזמן לא זהה חד צדדי (קורה לפעמים שהreno בממוצע מהיר יותר ככל שהאחוז האיבודים עולה).

נשים לב בנוסף ששוב הזמן שלקח בממוצע בשני החלקים גבוה יותר מאשר ב-10% איבוד, שמדובר שהזמן שלקח לחבילות להגעה למשך זמן, שמעיד על זה שהיא איבוד פאקטות ובכמות גדולה יותר.

### להלן צילומי מסך מהוירשארק כאשר יש 15% איבוד פאקטות:(הקלטה בשם "15% lost.pcapng")

נשים לב שכשרשים localhost ב-source וב-destination אז הכוונה לכתובת ip . 127.0.0.1

ב-3 שורות הראשונות שוב אנחנו פותחים את הקשר בין sender ל-receiver.(חץ שחור).

כמובן שיש שליחת פאקטות וקבלתם שניתן לראות לפיה ack בשביל לשולח ו- ack בשביל קבלת של השילחה.(שורת לבנות)

נשים לב ששוב פעם אנחנו מקבלים שורות שחזרות שמיידות לנו על איבוד פאקטות כפי שראינו באיבוד של 10% אחז (ect, tcp dup ack, tcp retransmission), אך אפשר לשים לב שהם מגיעות יותר בריציפות (אחד אחרי השני) ובכמות גדולה יותר שמשיים איבוד פאקטות מאשר ב-10% איבוד. (חץ שחור). ניתן לראות בברור בצלום מסך השני והשלישי את הכמות הגדולה יותר של האיבוד פאקטות מאשר ב-10% איבוד ואת הרצף ארוך יותר של האיבוד.

בנוסף אפשר לראות עוד דברים:

Tcp window full : זה אומר שהשלוח שולח יותר קבצים ממה שהחלון יכול להכיל כמו מסוימת בקשות מהsender של receiver ואם receiver שלוח יותר מהר ממה שהחלון מוציא לאישור אזחלון מתמלא ואין לו יותר מקום).(חץ כחוי).

Tcp zero window : זה אומר לנו שהחלון השהEDA של החלון התאפס כך שהTCP מבין אם צריך להגדיל או להקטין את קצב שליחת ההודעות מהשלוח.

כמובן שישנו עוד פאקטות שנתפסו שלא עשינו עליהם צילומי מסך שאפשר לראות אותם בהקלטה בשם ."15% lost.pcapng"

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	localhost	localhost	TCP	74	35650 → freeciv(5556) [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1537787716 TSeср=0 WS=128
2	0.000413378	localhost	localhost	TCP	74	freeciv(5556) → 35650 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1537787716 TSeср=1537787716 W...
3	0.000468243	localhost	localhost	TCP	66	35650 → freeciv(5556) [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1537787716 TSeср=1537787716
4	0.001125954	localhost	localhost	TCP	70	35650 → freeciv(5556) [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=1537787717 TSeср=1537787716
5	0.001146045	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=1537787717 TSeср=1537787717
6	0.001762705	localhost	localhost	TCP	32834	[TCP Window Full] [TCP Previous segment not captured] 35650 → freeciv(5556) [PSH, ACK] Seq=32773 Ack=1 Win=65536 Len=3...
7	0.001793815	localhost	localhost	TCP	78	[TCP Dup ACK 5#1] freeciv(5556) → 35650 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=1537787717 TSeср=1537787717 SLE=32773
8	0.002231858	localhost	localhost	TCP	32834	[TCP Retransmission] 35650 → freeciv(5556) [ACK] Seq=5 Ack=1 Win=65536 Len=32768 TSval=1537787718 TSeср=1537787717
9	0.002260871	localhost	localhost	TCP	66	[TCP ZeroWindow] freeciv(5556) → 35650 [ACK] Seq=1 Ack=65541 Win=0 Len=0 TSval=1537787718 TSeср=1537787718
10	0.003170381	localhost	localhost	TCP	66	[TCP Window Update] freeciv(5556) → 35650 [ACK] Seq=1 Ack=65541 Win=65536 Len=0 TSval=1537787719 TSeср=1537787718
11	0.003193263	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=65541 Ack=1 Win=65536 Len=32768 TSval=1537787719 TSeср=1537787719
12	0.0032604725	localhost	localhost	TCP	32834	[TCP Window Full] 35650 → freeciv(5556) [PSH, ACK] Seq=98309 Ack=1 Win=65536 Len=32768 TSval=1537787719 TSeср=15377877...
13	0.003268693	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=131077 Win=65536 Len=0 TSval=1537787719 TSeср=1537787719
14	0.003286157	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=131077 Ack=1 Win=65536 Len=32768 TSval=1537787719 TSeср=1537787719
15	0.015996899	localhost	localhost	TCP	32834	[TCP Window Full] 35650 → freeciv(5556) [PSH, ACK] Seq=163845 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=15377877...
16	0.016056452	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=196612 Win=196608 Len=0 TSval=1537787732 TSeср=1537787732
17	0.016094988	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=196613 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
18	0.016106478	localhost	localhost	TCP	32834	35650 → freeciv(5556) [PSH, ACK] Seq=229381 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
19	0.016325152	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=262149 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
20	0.016346226	localhost	localhost	TCP	32834	35650 → freeciv(5556) [PSH, ACK] Seq=284917 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
21	0.016358835	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=327685 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
22	0.016382024	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=229381 Win=327552 Len=0 TSval=1537787732 TSeср=1537787732
23	0.016411544	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=294917 Win=589440 Len=0 TSval=1537787732 TSeср=1537787732
24	0.016422551	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=327685 Win=720384 Len=0 TSval=1537787732 TSeср=1537787732
25	0.016435969	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=360453 Win=851456 Len=0 TSval=1537787732 TSeср=1537787732
26	0.016462112	localhost	localhost	TCP	32834	[TCP Previous segment not captured] 35650 → freeciv(5556) [ACK] Seq=393221 Ack=1 Win=65536 Len=32768 TSval=1537787732 ...
27	0.016478806	localhost	localhost	TCP	32834	35650 → freeciv(5556) [PSH, ACK] Seq=425989 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=1537787732
28	0.016497918	localhost	localhost	TCP	78	[TCP Window Update] freeciv(5556) → 35650 [ACK] Seq=1 Ack=360453 Win=982400 Len=0 TSval=1537787732 TSeср=1537787732 SL...
29	0.0165098930	localhost	localhost	TCP	78	[TCP Window Update] freeciv(5556) → 35650 [ACK] Seq=1 Ack=360453 Win=1113344 Len=0 TSval=1537787732 TSeср=1537787732 S...
30	0.016528595	localhost	localhost	TCP	32834	[TCP Out-Of-Order] 35650 → freeciv(5556) [PSH, ACK] Seq=360453 Ack=1 Win=65536 Len=32768 TSval=1537787732 TSeср=153778...
31	0.016554633	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=458757 Win=1244288 Len=0 TSval=1537787732 TSeср=1537787732
32	0.042683977	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=458757 Ack=1 Win=65536 Len=32768 TSval=1537788358 TSeср=1537787732
33	0.042932763	localhost	localhost	TCP	66	freeciv(5556) → 35650 [ACK] Seq=1 Ack=491525 Win=1375232 Len=0 TSval=1537788359 TSeср=1537788358
34	0.042953714	localhost	localhost	TCP	32834	[TCP Previous segment not captured] 35650 → freeciv(5556) [PSH, ACK] Seq=557061 Ack=1 Win=65536 Len=32768 TSval=153778...
35	0.042963368	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=589829 Ack=1 Win=65536 Len=32768 TSval=1537788359 TSeср=1537788359
36	0.042975631	localhost	localhost	TCP	78	[TCP Window Update] freeciv(5556) → 35650 [ACK] Seq=1 Ack=491525 Win=1506176 Len=0 TSval=1537788359 TSeср=1537788359 S...
37	0.042999486	localhost	localhost	TCP	32834	[TCP Out-Of-Order] 35650 → freeciv(5556) [PSH, ACK] Seq=491525 Ack=1 Win=65536 Len=32768 TSval=1537788359 TSeср=153778...
38	0.043208429	localhost	localhost	TCP	78	freeciv(5556) → 35650 [ACK] Seq=1 Ack=524293 Win=1768192 Len=0 TSval=1537788359 TSeср=1537788359 SRE=622597
39	0.043236174	localhost	localhost	TCP	32834	35650 → freeciv(5556) [PSH, ACK] Seq=622597 Ack=1 Win=65536 Len=32768 TSval=1537788359 TSeср=1537788359
40	0.043245943	localhost	localhost	TCP	32834	35650 → freeciv(5556) [ACK] Seq=655365 Ack=1 Win=65536 Len=32768 TSval=1537788359 TSeср=1537788359

No.	Time	Source	Destination	Protocol	Length	Info
232	5.913429666	localhost	localhost	TCP	78	[TCP Window Update] freeciv(5556) -> 35650 [ACK] Seq=57 Ack=7824109 Win=3879040 Len=0 TSval=1537793629 TSecr=1537793629..
233	5.913443036	localhost	localhost	TCP	65549	[TCP Out-Of-Order] 35650 -> freeciv(5556) [ACK] Seq=7824109 Ack=57 Win=65536 Len=65483 TSval=1537793629 TSecr=1537793629
234	5.913511766	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8020558 Win=3946272 Len=0 TSval=1537793629 TSecr=1537793629
235	6.629988604	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8020558 Ack=57 Win=65536 Len=65483 TSval=1537794337 TSecr=1537793629
236	7.580938817	localhost	localhost	TCP	65549	[TCP Retransmission] 35650 -> freeciv(5556) [ACK] Seq=8020558 Ack=57 Win=65536 Len=65483 TSval=1537795297 TSecr=1537793..
237	7.581048133	localhost	localhost	TCP	78	freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8086041 Win=3145728 Len=0 TSval=1537795297 TSecr=1537795297 SLE=808..
238	7.581091579	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=8115124 Ack=57 Win=65536 Len=65483 TSval=153779529..
239	7.581445308	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8217007 Ack=57 Win=65536 Len=65483 TSval=1537795297 TSecr=1537795297
240	7.581505195	localhost	localhost	TCP	78	[TCP Dup ACK 237#1] freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8086041 Win=3145728 Len=0 TSval=1537795297 TSecr=1537795297..
241	7.581564022	localhost	localhost	TCP	65549	[TCP Out-Of-Order] 35650 -> freeciv(5556) [ACK] Seq=8086041 Ack=57 Win=65536 Len=65483 TSval=1537795297 TSecr=1537795297
242	7.581790385	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8282490 Win=3045632 Len=0 TSval=1537795297 TSecr=1537795297
243	7.581831132	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8282490 Ack=57 Win=65536 Len=65483 TSval=1537795297 TSecr=1537795297
244	7.581856689	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8347973 Ack=57 Win=65536 Len=65483 TSval=1537795297 TSecr=1537795297
245	7.581911940	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=8413456 Win=2980224 Len=0 TSval=1537795298 TSecr=1537795297
246	7.581947434	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=8478939 Ack=57 Win=65536 Len=65483 TSval=153779529..
247	7.581973743	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8544422 Ack=57 Win=65536 Len=65483 TSval=1537795298 TSecr=1537795298
248	7.581998680	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8609905 Ack=57 Win=65536 Len=65483 TSval=1537795298 TSecr=1537795298
249	7.582023077	localhost	localhost	TCP	78	[TCP Dup ACK 245#1] freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8413456 Win=2980224 Len=0 TSval=1537795298 TSecr=1537795297..
250	7.582043270	localhost	localhost	TCP	78	[TCP Dup ACK 245#2] freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8413456 Win=2980224 Len=0 TSval=1537795298 TSecr=1537795297..
251	7.582063849	localhost	localhost	TCP	78	[TCP Dup ACK 245#3] freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8413456 Win=2980224 Len=0 TSval=1537795298 TSecr=1537795297..
252	7.582094174	localhost	localhost	TCP	65549	[TCP Fast Retransmission] 35650 -> freeciv(5556) [ACK] Seq=8413456 Ack=57 Win=65536 Len=65483 TSval=1537795298 TSecr=15..
253	7.582136705	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8675388 Ack=57 Win=65536 Len=65483 TSval=1537795298 TSecr=1537795298
254	7.582163324	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8675388 Win=2848512 Len=0 TSval=1537795298 TSecr=1537795298
255	7.582187238	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=57 Ack=8740871 Win=2815872 Len=0 TSval=1537795298 TSecr=1537795298
256	7.582214539	localhost	localhost	TCP	51356	35650 -> freeciv(5556) [PSH, ACK] Seq=8740871 Ack=57 Win=65536 Len=51290 TSval=1537795298 TSecr=1537795298
257	7.582847892	localhost	localhost	TCP	70	freeciv(5556) -> 35650 [PSH, ACK] Seq=57 Ack=8792161 Win=3145728 Len=4 TSval=1537795299 TSecr=1537795298
258	7.628046893	localhost	localhost	TCP	66	35650 -> freeciv(5556) [ACK] Seq=8792161 Ack=61 Win=65536 Len=0 TSval=1537795344 TSecr=1537795299
259	8.282736137	localhost	localhost	TCP	70	35650 -> freeciv(5556) [PSH, ACK] Seq=8792161 Ack=61 Win=65536 Len=4 TSval=1537795998 TSecr=1537795299
260	8.283095124	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=8857648 Ack=61 Win=65536 Len=65483 TSval=153779599..
261	8.283114518	localhost	localhost	TCP	78	freeciv(5556) -> 35650 [ACK] Seq=61 Ack=8792165 Win=3145728 Len=0 TSval=1537795999 TSecr=1537795998 SLE=892..
262	8.283477704	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8923131 Ack=61 Win=65536 Len=65483 TSval=1537795999 TSecr=1537795999
263	8.283650616	localhost	localhost	TCP	78	[TCP Dup ACK 261#1] freeciv(5556) -> 35650 [ACK] Seq=61 Ack=8792165 Win=3145728 Len=0 TSval=1537795999 TSecr=153779599..
264	8.283667726	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=8988614 Ack=61 Win=65536 Len=65483 TSval=1537795999 TSecr=1537795999
265	8.283680421	localhost	localhost	TCP	78	[TCP Dup ACK 261#2] freeciv(5556) -> 35650 [ACK] Seq=61 Ack=8792165 Win=3145728 Len=0 TSval=1537795999 TSecr=153779599..
266	8.283710983	localhost	localhost	TCP	65549	[TCP Fast Retransmission] 35650 -> freeciv(5556) [ACK] Seq=8792165 Ack=61 Win=65536 Len=65483 TSval=1537795999 TSecr=15..
267	8.284086434	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=61 Ack=9054097 Win=3013504 Len=0 TSval=1537795999 TSecr=1537795999
268	8.284125484	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=9119588 Ack=61 Win=65536 Len=65483 TSval=153779600..
269	8.284347079	localhost	localhost	TCP	78	[TCP Window Update] freeciv(5556) -> 35650 [ACK] Seq=61 Ack=9054097 Win=3112448 Len=0 TSval=1537796000 TSecr=1537795999..
270	8.496663538	localhost	localhost	TCP	65549	[TCP Retransmission] 35650 -> freeciv(5556) [ACK] Seq=9054097 Ack=61 Win=65536 Len=65483 TSval=1537796212 TSecr=153779..
271	8.496997786	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=61 Ack=9185063 Win=3079040 Len=0 TSval=1537796212 TSecr=1537796212

149	4.088395888	localhost	localhost	TCP	65549	[TCP Retransmission] 35650 -> freeciv(5556) [ACK] Seq=4985432 Ack=31 Win=65536 Len=65483 TSval=1537791804 TSecr=1537791..
150	4.088485460	localhost	localhost	TCP	78	freeciv(5556) -> 35650 [ACK] Seq=31 Ack=5116398 Win=3145728 Len=0 TSval=1537791804 TSecr=1537791583 SLE=4985432 SRE=505..
151	4.088577827	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=5181881 Ack=31 Win=65536 Len=65483 TSval=153779180..
152	4.088621707	localhost	localhost	TCP	78	[TCP Dup ACK 150#1] freeciv(5556) -> 35650 [ACK] Seq=31 Ack=5116398 Win=3145728 Len=0 TSval=1537791804 TSecr=1537791583..
153	4.088658589	localhost	localhost	TCP	65549	[TCP Out-Of-Order] 35650 -> freeciv(5556) [ACK] Seq=5116398 Ack=31 Win=65536 Len=65483 TSval=1537791804 TSecr=1537791804..
154	4.088703886	localhost	localhost	TCP	66	freeciv(5556) -> 35650 [ACK] Seq=31 Ack=5247364 Win=3079040 Len=0 TSval=1537791804 TSecr=1537791804
155	4.088742851	localhost	localhost	TCP	65549	[TCP Previous segment not captured] 35650 -> freeciv(5556) [ACK] Seq=5312847 Ack=31 Win=65536 Len=65483 TSval=153779180..
156	4.088770247	localhost	localhost	TCP	78	[TCP Dup ACK 154#1] freeciv(5556) -> 35650 [ACK] Seq=31 Ack=5247364 Win=3079040 Len=0 TSval=1537791804 TSecr=1537791804..
157	4.088804882	localhost	localhost	TCP	65549	35650 -> freeciv(5556) [ACK] Seq=5378330 Ack=31 Win=65536 Len=65483 TSval=1537791804 TSecr=1537791804
158	4.088832505	localhost	localhost	TCP	78	[TCP Dup ACK 154#2] freeciv(5556) -> 35650 [ACK] Seq=31 Ack=5247364 Win=3079040 Len=0 TSval=1537791805 TSecr=1537791804..
159	4.088872577	localhost	localhost	TCP	65549	[TCP Fast Retransmission] 35650 -> freeciv(5556) [ACK] Seq=5247364 Ack=31 Win=65536 Len=65483 TSval=1537791805 TSecr=15..
160	4.3232554586	localhost	localhost	TCP	65549	[TCP Retransmission] 35650 -> freeciv(5556) [ACK] Seq=5247364 Ack=31 Win=65536 Len=65483 TSval=1537792048 TSecr=1537791..

## עבור 20% איבוד פאקטות:

נפעיל בטרמינל את הפקודה הבאה:

```
matan@matan-VirtualBox:~/Desktop/Network and communicait/matala 3$ sudo tc qdisc change dev lo root netem loss 20%
matan@matan-VirtualBox:~/Desktop/Network and communicait/matala 3$
```

העלונו את שליחת החבילה 5 פעמים וקיבلونו את הזמן们的 האלו: (הראתי פה שני הרצות שונות(כל אחד מהרצות 5 פעמים) שראים שבמוצע cubic מהיר יותר מreno)

```
Run 1 of first half using cubic: 0.472846 seconds.
Run 1 of second half using reno: 0.656756 seconds.
Run 2 of first half using cubic: 0.393666 seconds.
Run 2 of second half using reno: 5.406023 seconds.
Run 3 of first half using cubic: 0.003219 seconds.
Run 3 of second half using reno: 0.111086 seconds.
Run 4 of first half using cubic: 0.655263 seconds.
Run 4 of second half using reno: 1.362991 seconds.
Run 5 of first half using cubic: 1.390553 seconds.
Run 5 of second half using reno: 2.799124 seconds.
The average time for the first half: 0.583109 seconds.
The average time for the second half: 2.067196 seconds.
The average time of the full file is : 1.325153
```

```
Run 1 of first half using cubic: 1.257051 seconds.
Run 1 of second half using reno: 34.615698 seconds.
Run 2 of first half using cubic: 5.782601 seconds.
Run 2 of second half using reno: 0.446715 seconds.
Run 3 of first half using cubic: 4.612690 seconds.
Run 3 of second half using reno: 0.119274 seconds.
Run 4 of first half using cubic: 0.073692 seconds.
Run 4 of second half using reno: 0.229813 seconds.
Run 5 of first half using cubic: 0.867508 seconds.
Run 5 of second half using reno: 1.023086 seconds.
The average time for the first half: 2.518708 seconds.
The average time for the second half: 7.286917 seconds.
The average time of the full file is : 4.902813
```

נשים לב ששוב פעם הזמן המוצע של cubic(CC algo for first half of the file) הוא מהיר יותר מאשרreno(CC algo for second half of the file) כי cubic מיועד לאיבוד פאקטות ולכן במוצע הזמן מרגע השילחה עד רגע הקבלה הוא מהיר יותר.

נשים לב בនוסף לשוב הזמן שלקח בשני החלקים גבוה יותר מאשר ב-15% איבוד, שמדובר שהזמן שלקח לחבילות להגיא,濂קח יותר זמן שמעיד על זה שהוא איבוד פאקטות ובכמות גדולה יותר.

## להלן צילומי מסך מהוירשארק כאשר יש 20% איבוד פאקטות:(הקלטה בשם "20% lost.pcapng")

נשים לב שכשרשים localhost ב-source וב-destination אז הכוונה לכתובת ip 127.0.0.1

Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)

שוב פעם ונראה שב-3 שורות הראשונות אנחנו פותחים את הקשר בין receiver-ל-sender.(חץ שchor).

כמוון שיש שליחת פאקטות וקבלתם שניתן לראות לפיה ack psh בשבייל לשולח ו- ack בשבייל קבלת של השילחה.(שורות לבנות)

נשים לב ששוב פעם אנחנו מקבלים שורות שחזרות שמעידות לנו על איבוד פאקטות כפי שראינו באיבוד של 10%-15% אוחז (tcp dup ack, tcp retransmission, ect). (חץ אדום) ( שמתו חצים רק ב-2 צילומי מסך אך זה קורה גם בשאר הצלומיות כך שזה מראה על הכמות הגדולה של איבוד הפאקטות).

אך אפשר לשים לב שם מגיעות יותר בריציפות (אחד אחרי השני) ובכמות גדולה יותר שמעידה שיש יותר איבוד פאקטות מאשר ב-10% איבוד ו-15% איבוד.

אך ניתן לראות בוורור בצלומי המסך הבאים (5 צילומי המסך הבאים) את התדריות הגבואה יותר של איבוד פאקטות ואת הכמות הגדולה יותר של השורות שמראות על איבוד פאקטות יותר מאשר ב-10% איבוד וב-15% איבוד ואת הרצף ארוך יותר של האיבוד.

כמובן שישנו עוד פאקטות שנתפסו שלא עשינו עליהם צילומי מסך שאפשר לראות אותם בהקלטה בשם "20% lost.pcapng".

1 0.000000000	localhost	localhost	TCP	74 34648 - freeciv(5556) [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1544012748 TSecr=0 WS=128
2 1.012046998	localhost	localhost	TCP	74 [TCP Retransmission] [TCP Port numbers reused] 34648 - freeciv(5556) [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1...
3 1.012059682	localhost	localhost	TCP	74 freeciv(5556) - 34648 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1544013760 TSecr=1544012748 W...
4 1.012121929	localhost	localhost	TCP	74 [TCP Out-Of-Order] freeciv(5556) - 34648 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1544013760...
5 1.012139832	localhost	localhost	TCP	66 34648 - freeciv(5556) [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1544013760 TSecr=1544013760
6 1.012142062	localhost	localhost	TCP	66 [TCP Dup ACK 5#1] 34648 - freeciv(5556) [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1544013760 TSecr=1544013760
7 1.012226345	localhost	localhost	TCP	32807 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=5 Ack=1 Win=65536 Len=32741 TSval=1544013760 TSecr...
8 1.012232563	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1544013760 TSecr=1544013760 SLE=5 SR...
9 1.012367261	localhost	localhost	TCP	32807 34648 - freeciv(5556) [PSH, ACK] Seq=32746 Ack=1 Win=65536 Len=32741 TSval=1544013761 TSecr=1544013760
10 1.012372388	localhost	localhost	TCP	78 [TCP Dup ACK 3#1] freeciv(5556) - 34648 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1544013761 TSecr=1544013760 SLE=5 SR=...
11 1.012379695	localhost	localhost	TCP	70 [TCP Out-Of-Order] 34648 - freeciv(5556) [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=1544013761 TSecr=1544013761
12 1.013293900	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=65487 Win=65536 Len=0 TSval=1544013762 TSecr=1544013761
13 1.013310389	localhost	localhost	TCP	32807 34648 - freeciv(5556) [ACK] Seq=65487 Ack=1 Win=65536 Len=32741 TSval=1544013762 TSecr=1544013762
14 1.073291017	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=98228 Win=65536 Len=0 TSval=1544013822 TSecr=1544013762
15 1.073334577	localhost	localhost	TCP	32807 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=130969 Ack=1 Win=65536 Len=32741 TSval=1544013822 ...
16 1.073422756	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=98228 Win=196608 Len=0 TSval=1544013822 TSecr=1544013762 SLE...
17 1.073434838	localhost	localhost	TCP	32807 [TCP Retransmission] 34648 - freeciv(5556) [PSH, ACK] Seq=98228 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=15440...
18 1.073446606	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=163710 Win=327552 Len=0 TSval=1544013822 TSecr=1544013822
19 1.073454206	localhost	localhost	TCP	32807 34648 - freeciv(5556) [PSH, ACK] Seq=163710 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
20 1.073460250	localhost	localhost	TCP	32807 34648 - freeciv(5556) [ACK] Seq=196451 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
21 1.073465379	localhost	localhost	TCP	32807 34648 - freeciv(5556) [PSH, ACK] Seq=229192 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
22 1.073471550	localhost	localhost	TCP	32807 34648 - freeciv(5556) [ACK] Seq=261933 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
23 1.073479364	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=196451 Win=458496 Len=0 TSval=1544013822 TSecr=1544013822
24 1.073484052	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=229192 Win=589440 Len=0 TSval=1544013822 TSecr=1544013822
25 1.073489657	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=261933 Win=720384 Len=0 TSval=1544013822 TSecr=1544013822
26 1.073494534	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=294674 Win=851456 Len=0 TSval=1544013822 TSecr=1544013822
27 1.073501732	localhost	localhost	TCP	32807 34648 - freeciv(5556) [PSH, ACK] Seq=294674 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
28 1.073511976	localhost	localhost	TCP	32807 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=392897 Ack=1 Win=65536 Len=32741 TSval=1544013822 ...
29 1.073519307	localhost	localhost	TCP	32807 34648 - freeciv(5556) [PSH, ACK] Seq=425638 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
30 1.073529606	localhost	localhost	TCP	78 freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1113344 Len=0 TSval=1544013822 TSecr=1544013822 SLE=425638
31 1.073534393	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1244288 Len=0 TSval=1544013822 TSecr=1544013822 S...
32 1.073541832	localhost	localhost	TCP	32807 34648 - freeciv(5556) [ACK] Seq=458379 Ack=1 Win=65536 Len=32741 TSval=1544013822 TSecr=1544013822
33 1.073547871	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=491120 Ack=1 Win=65536 Len=32768 TSval=1544013822 TSecr=1544013822
34 1.073558273	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1375232 Len=0 TSval=1544013822 TSecr=1544013822 S...
35 1.073580688	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1506176 Len=0 TSval=1544013822 TSecr=1544013822 S...
36 1.073593250	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=523888 Ack=1 Win=65536 Len=32768 TSval=1544013822 TSecr=1544013822
37 1.073601715	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=556656 Ack=1 Win=65536 Len=32768 TSval=1544013822 TSecr=1544013822
38 1.073606680	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1637248 Len=0 TSval=1544013822 TSecr=1544013822 S...
39 1.073610882	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=327415 Win=1768192 Len=0 TSval=1544013822 TSecr=1544013822 S...
40 1.073624292	localhost	localhost	TCP	65548 [TCP Out-Of-Order] 34648 - freeciv(5556) [PSH, ACK] Seq=327415 Ack=1 Win=65536 Len=65482 TSval=1544013822 TSecr=154401...

150 11.765407750	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=4279308 Ack=27 Win=65536 Len=65483 TSval=154402451...	
151 11.765434348	localhost	localhost	TCP	78 [TCP Dup ACK 147#1] freeciv(5556) - 34648 [ACK] Seq=27 Ack=4017376 Win=3145728 Len=0 TSval=1544024514 Tscr=1544021774...	←
152 11.765521707	localhost	localhost	TCP	78 [TCP Dup ACK 147#2] freeciv(5556) - 34648 [ACK] Seq=27 Ack=4017376 Win=3145728 Len=0 TSval=1544024514 Tscr=1544021774...	←
153 11.765542622	localhost	localhost	TCP	86 [TCP Dup ACK 147#3] freeciv(5556) - 34648 [ACK] Seq=27 Ack=4017376 Win=3145728 Len=0 TSval=1544024514 Tscr=1544021774...	←
154 11.765592870	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=4213825 Ack=27 Win=65536 Len=65483 TSval=1544024514 Tscr=1544024...	←
155 15.25210413	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=4017376 Ack=27 Win=65536 Len=65483 TSval=1544028000 Tscr=1544024...	←
277 24.0884343557	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=9250546 Ack=61 Win=65536 Len=65483 TSval=1544036833 Tscr=1544036...	
278 24.0884364450	localhost	localhost	TCP	78 freeciv(5556) - 34648 [ACK] Seq=61 Ack=9446995 Win=2914048 Len=0 TSval=1544036833 Tscr=1544036833 SLE=9512478 SRE=964...	
279 24.0884382071	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=9446995 Ack=61 Win=65536 Len=65483 TSval=1544036833 Tscr=1544036...	
280 24.0884397055	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=61 Ack=9643444 Win=2881920 Len=0 TSval=1544036833 Tscr=1544036833	
281 24.0884421031	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=9708927 Ack=61 Win=65536 Len=65483 TSval=1544036833...	
282 24.088446646	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=9774410 Ack=61 Win=65536 Len=65483 TSval=1544036833 Tscr=1544036833	
283 24.0884605628	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=61 Ack=9643444 Win=3045632 Len=0 TSval=1544036833 Tscr=1544036833...	
284 24.0884616851	localhost	localhost	TCP	78 [TCP Dup ACK 280#1] freeciv(5556) - 34648 [ACK] Seq=61 Ack=9643444 Win=3045632 Len=0 TSval=1544036833 Tscr=1544036833...	
285 24.0884609494	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=9643444 Ack=61 Win=65536 Len=65483 TSval=1544036833 Tscr=1544036...	
286 24.0884770878	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=61 Ack=9839893 Win=3045632 Len=0 TSval=1544036833 Tscr=1544036833	
287 24.0884800859	localhost	localhost	TCP	51356 34648 - freeciv(5556) [PSH, ACK] Seq=9839893 Ack=61 Win=65536 Len=51290 TSval=1544036833 Tscr=1544036833	
288 24.307980117	localhost	localhost	TCP	77 freeciv(5556) - 34648 [PSH, ACK] Seq=61 Ack=9891183 Win=3145728 Len=11 TSval=1544037056 Tscr=1544036833	
289 24.310031450	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=9891183 Ack=72 Win=65536 Len=65483 TSval=1544037056 Tscr=1544037056	
290 24.310152110	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=9956666 Ack=72 Win=65536 Len=65483 TSval=1544037058 Tscr=1544037056	
291 24.310287898	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10022149 Win=3145728 Len=0 TSval=1544037059 Tscr=1544037058	
292 24.310875857	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=10022149 Ack=72 Win=65536 Len=65483 TSval=1544037059 Tscr=1544037059	
293 24.351444062	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10087632 Win=3145728 Len=0 TSval=1544037100 Tscr=1544037059	
294 24.351591645	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=10218598 Ack=72 Win=65536 Len=65483 TSval=15440371...	
295 24.351613858	localhost	localhost	TCP	78 [TCP Dup ACK 293#1] freeciv(5556) - 34648 [ACK] Seq=72 Ack=10087632 Win=3145728 Len=0 TSval=1544037100 Tscr=154403705...	
296 24.351635107	localhost	localhost	TCP	65549 [TCP Out-Of-Order] 34648 - freeciv(5556) [ACK] Seq=10087632 Ack=72 Win=65536 Len=65483 TSval=1544037100 Tscr=15440371...	
297 24.351755865	localhost	localhost	TCP	78 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10153115 Win=3080320 Len=0 TSval=1544037100 Tscr=1544037100 SLE=10218598 SRE=1...	
298 24.351779331	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=10153115 Ack=72 Win=65536 Len=65483 TSval=1544037100 Tscr=154403...	
299 24.351795533	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=10284081 Ack=72 Win=65536 Len=65483 TSval=1544037100 Tscr=1544037100	
300 24.351814565	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10284081 Win=3046912 Len=0 TSval=1544037100 Tscr=1544037100	
301 24.351851381	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=10349564 Ack=72 Win=65536 Len=65483 TSval=1544037100 Tscr=1544037100	
302 24.351864856	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10415047 Win=2981504 Len=0 TSval=1544037100 Tscr=1544037100	
303 24.351884107	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=10480530 Ack=72 Win=65536 Len=65483 TSval=15440371...	
304 24.351899115	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=10546013 Ack=72 Win=65536 Len=65483 TSval=1544037100 Tscr=1544037100	
305 24.351912847	localhost	localhost	TCP	78 [TCP Dup ACK 302#1] freeciv(5556) - 34648 [ACK] Seq=72 Ack=10415047 Win=2981504 Len=0 TSval=1544037100 Tscr=15440371...	←
306 24.351924657	localhost	localhost	TCP	78 [TCP Dup ACK 302#2] freeciv(5556) - 34648 [ACK] Seq=72 Ack=10415047 Win=2981504 Len=0 TSval=1544037100 Tscr=15440371...	←
307 24.621796088	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=10415047 Ack=72 Win=65536 Len=65483 TSval=1544037370 Tscr=154403...	←
308 25.152675613	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=10415047 Ack=72 Win=65536 Len=65483 TSval=1544037901 Tscr=154403...	←
309 25.152979229	localhost	localhost	TCP	78 freeciv(5556) - 34648 [ACK] Seq=72 Ack=10611496 Win=3145728 Len=0 TSval=1544037901 Tscr=1544037370 SLE=10415047 SRE=1...	
310 25.153047534	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=10742462 Ack=72 Win=65536 Len=65483 TSval=1544037901...	
311 25.153353172	localhost	localhost	TCP	65549 34648 - freeciv(5556) [ACK] Seq=10897945 Ack=72 Win=65536 Len=65483 TSval=1544037901 Tscr=1544037901	
312 25.153419837	localhost	localhost	TCP	78 [TCP Dup ACK 309#1] freeciv(5556) - 34648 [ACK] Seq=72 Ack=10611496 Win=3145728 Len=0 TSval=1544037902 Tscr=154403737...	←
313 25.153449107	localhost	localhost	TCP	78 [TCP Dup ACK 309#2] freeciv(5556) - 34648 [ACK] Seq=72 Ack=10611496 Win=3145728 Len=0 TSval=1544037902 Tscr=154403737...	←
314 25.153493560	localhost	localhost	TCP	65549 [TCP Fast Retransmission] 34648 - freeciv(5556) [ACK] Seq=10611496 Ack=72 Win=65536 Len=65483 TSval=1544037902 Tscr=1...	
315 25.153543643	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=10676979 Ack=72 Win=65536 Len=65483 TSval=1544037902 Tscr=154403...	←

203 19.7606545100	localhost	localhost	TCP	51356 34648 → freeciv(5556) [PSH, ACK] Seq=6542831 Ack=42 Win=65536 Len=51290 TSval=1544032509 TSeср=1544032509
204 19.7606579777	localhost	localhost	TCP	78 [TCP Dup ACK 198#1] freeciv(5556) → 34648 [ACK] Seq=42 Ack=6149933 Win=2878728 Len=0 TSval=1544032509 TSeср=1544032509...
205 19.7606593830	localhost	localhost	TCP	65549 [TCP Out-of-Order] 34648 → freeciv(5556) [ACK] Seq=6149933 Ack=42 Win=65536 Len=65483 TSval=1544032509 TSeср=1544032509
206 19.760680977	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=42 Ack=6594121 Win=2496256 Len=0 TSval=1544032509 TSeср=1544032509
207 19.760951199	localhost	localhost	TCP	70 freeciv(5556) → 34648 [PSH, ACK] Seq=42 Ack=6594121 Win=3145728 Len=4 TSval=1544032509 TSeср=1544032509
208 19.811062472	localhost	localhost	TCP	66 34648 → freeciv(5556) [ACK] Seq=6594121 Ack=46 Win=65536 Len=0 TSval=1544032559 TSeср=1544032509
209 21.166046684	localhost	localhost	TCP	70 34648 → freeciv(5556) [PSH, ACK] Seq=6594121 Ack=46 Win=65536 Len=4 TSval=15440333914 TSeср=1544032509
210 21.166119464	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=6659668 Ack=46 Win=65536 Len=65483 TSval=1544033391...
211 21.166199534	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=6725091 Ack=46 Win=65536 Len=65483 TSval=15440333914 TSeср=1544032509
212 21.166222108	localhost	localhost	TCP	78 freeciv(5556) → 34648 [ACK] Seq=46 Ack=6594125 Win=3145728 Len=0 TSval=15440333914 TSeср=15440333914 SLE=665...
213 21.166986262	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=6856057 Ack=46 Win=65536 Len=65483 TSval=1544033391...
214 21.166955126	localhost	localhost	TCP	86 [TCP Dup ACK 212#1] freeciv(5556) → 34648 [ACK] Seq=46 Ack=6594125 Win=3145728 Len=0 TSval=15440333915 TSeср=15440333914...
215 21.167116349	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=6921540 Ack=46 Win=65536 Len=65483 TSval=15440333915 TSeср=15440333914
216 21.167170339	localhost	localhost	TCP	65549 [TCP Out-of-Order] 34648 → freeciv(5556) [ACK] Seq=6594125 Ack=46 Win=65536 Len=65483 TSval=15440333915 TSeср=15440333915
217 21.652207709	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=6594125 Ack=46 Win=65536 Len=65483 TSval=1544034400 TSeср=1544033...
218 21.652454381	localhost	localhost	TCP	86 freeciv(5556) → 34648 [ACK] Seq=46 Ack=6790574 Win=3079040 Len=0 TSval=1544034401 TSeср=1544033915 SLE=6594125 SRE=665...
219 21.652552725	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=6987023 Ack=46 Win=65536 Len=65483 TSval=1544034401 TSeср=1544034401
220 21.652815471	localhost	localhost	TCP	78 [TCP Dup ACK 218#1] freeciv(5556) → 34648 [ACK] Seq=46 Ack=6790574 Win=3079040 Len=0 TSval=1544034401 TSeср=15440333915...
221 21.652849788	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=6798574 Ack=46 Win=65536 Len=65483 TSval=1544034401 TSeср=1544034...
222 21.652873232	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7052506 Win=3013504 Len=0 TSval=1544034401 TSeср=1544034401
223 21.652901679	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=7117981 Ack=46 Win=65536 Len=65483 TSval=154403440...
224 21.652918460	localhost	localhost	TCP	78 [TCP Dup ACK 222#1] freeciv(5556) → 34648 [ACK] Seq=46 Ack=7052506 Win=3013504 Len=0 TSval=1544034401 TSeср=1544034401...
225 21.652939134	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=7183472 Ack=46 Win=65536 Len=65483 TSval=1544034401 TSeср=1544034401
226 21.652953842	localhost	localhost	TCP	78 [TCP Dup ACK 222#2] freeciv(5556) → 34648 [ACK] Seq=46 Ack=7052506 Win=3013504 Len=0 TSval=1544034401 TSeср=1544034401...
227 21.652975866	localhost	localhost	TCP	65549 [TCP Fast Retransmission] 34648 → freeciv(5556) [ACK] Seq=7052506 Ack=46 Win=65536 Len=65483 TSval=1544034401 TSeср=15...
228 22.324149877	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=7052506 Ack=46 Win=65536 Len=65483 TSval=1544035072 TSeср=1544034...
229 22.324577418	localhost	localhost	TCP	78 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7248955 Win=3145728 Len=0 TSval=1544035072 TSeср=1544034401 SLE=7052506 SRE=711...
230 22.324636282	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=7248955 Ack=46 Win=65536 Len=65483 TSval=1544035073 TSeср=1544035072
231 22.324871659	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=7314438 Ack=46 Win=65536 Len=65483 TSval=1544035073 TSeср=1544035072
232 22.324929800	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7314438 Win=3112448 Len=0 TSval=1544035073 TSeср=1544035072
233 22.324961965	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7379921 Win=3079040 Len=0 TSval=1544035073 TSeср=1544035073
234 22.324996469	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=7379921 Ack=46 Win=65536 Len=65483 TSval=1544035073 TSeср=1544035073
235 22.325025464	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=7445404 Ack=46 Win=65536 Len=65483 TSval=1544035073 TSeср=1544035073
236 22.325049667	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7445404 Win=3046272 Len=0 TSval=1544035073 TSeср=1544035073
237 22.325071224	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7510887 Win=3013504 Len=0 TSval=1544035073 TSeср=1544035073
238 22.484651884	localhost	localhost	TCP	51356 [TCP Previous segment not captured] 34648 → freeciv(5556) [PSH, ACK] Seq=7641853 Ack=46 Win=65536 Len=51290 TSval=1544...
239 22.484767466	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) → 34648 [ACK] Seq=46 Ack=7510887 Win=3119488 Len=0 TSval=1544035233 TSeср=1544035073...
240 22.484736475	localhost	localhost	TCP	65549 [TCP Out-of-Order] 34648 → freeciv(5556) [ACK] Seq=7510887 Ack=46 Win=65536 Len=65483 TSval=1544035233 TSeср=1544035233
241 22.484916294	localhost	localhost	TCP	78 freeciv(5556) → 34648 [ACK] Seq=46 Ack=7576370 Win=3086080 Len=0 TSval=1544035233 TSeср=1544035233 SLE=7641853 SRE=769...
242 22.484946885	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=7576370 Ack=46 Win=65536 Len=65483 TSval=1544035233 TSeср=1544035...

90 6.708775418	localhost	localhost	TCP	78 [TCP Dup ACK 89#1] freeciv(5556) → 34648 [ACK] Seq=12 Ack=1622887 Win=2946816 Len=0 TSval=1544019457 TSecr=1544019457 ...
91 6.708813394	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=1884819 Ack=12 Win=65536 Len=65483 TSval=1544019457 ...
92 6.708830859	localhost	localhost	TCP	86 [TCP Dup ACK 89#2] freeciv(5556) → 34648 [ACK] Seq=12 Ack=1622887 Win=2946816 Len=0 TSval=1544019457 TSecr=1544019457 ...
93 6.708856656	localhost	localhost	TCP	65549 [TCP Fast Retransmission] 34648 → freeciv(5556) [ACK] Seq=1622887 Ack=12 Win=65536 Len=65483 TSval=1544019457 TSecr=15...
94 6.708877730	localhost	localhost	TCP	78 freeciv(5556) → 34648 [ACK] Seq=12 Ack=1753853 Win=2847872 Len=0 TSval=1544019457 TSecr=1544019457 SLE=1884819 SRE=195...
95 6.708902868	localhost	localhost	TCP	65549 [TCP Out-Of-Order] 34648 → freeciv(5556) [ACK] Seq=1753853 Ack=12 Win=65536 Len=65483 TSval=1544019457 TSecr=1544019457 ...
96 6.708924449	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=1819336 Ack=12 Win=65536 Len=65483 TSval=1544019457 TSecr=1544019...
97 6.709174711	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=12 Ack=1950302 Win=2978944 Len=0 TSval=1544019457 TSecr=1544019457
98 6.709179874	localhost	localhost	TCP	70 [TCP ACKed unseen segment] freeciv(5556) → 34648 [PSH, ACK] Seq=12 Ack=2198041 Win=3112448 Len=4 TSval=1544019458 TSec...
99 6.751370159	localhost	localhost	TCP	66 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=2198041 Ack=16 Win=65536 Len=0 TSval=1544019500 TS...
100 7.925286695	localhost	localhost	TCP	70 34648 → freeciv(5556) [PSH, ACK] Seq=2198041 Ack=16 Win=65536 Len=4 TSval=1544020674 TSecr=1544019458
101 7.925431233	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2198045 Ack=16 Win=65536 Len=65483 TSval=1544020674 TSecr=1544019458
102 7.925562176	localhost	localhost	TCP	66 [TCP ACKed unseen segment] freeciv(5556) → 34648 [ACK] Seq=16 Ack=2263528 Win=3112448 Len=0 TSval=1544020674 TSecr=15...
103 7.925597205	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2263528 Ack=16 Win=65536 Len=65483 TSval=1544020674 TSecr=1544019458
104 7.925722741	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2329011 Ack=16 Win=65536 Len=65483 TSval=1544020674 TSecr=1544019458
105 7.926083938	localhost	localhost	TCP	66 freeciv(5556) → 34648 [ACK] Seq=16 Ack=2394494 Win=3112448 Len=0 TSval=1544020674 TSecr=1544020674
106 7.926201388	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 → freeciv(5556) [ACK] Seq=2459977 Ack=16 Win=65536 Len=65483 TSval=154402067...
107 7.926212584	localhost	localhost	TCP	78 [TCP Dup ACK 105#1] freeciv(5556) → 34648 [ACK] Seq=16 Ack=2394494 Win=3112448 Len=0 TSval=1544020674 TSecr=1544020674...
108 7.926225007	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2525460 Ack=16 Win=65536 Len=65483 TSval=1544020674 TSecr=1544020674
109 7.926234786	localhost	localhost	TCP	78 [TCP Dup ACK 105#2] freeciv(5556) → 34648 [ACK] Seq=16 Ack=2394494 Win=3112448 Len=0 TSval=1544020674 TSecr=1544020674...
110 7.926245778	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2590943 Ack=16 Win=65536 Len=65483 TSval=1544020674 TSecr=1544020674
111 7.926271154	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2656426 Ack=16 Win=65536 Len=65483 TSval=1544020675 TSecr=1544020674
112 7.926369232	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2721909 Ack=16 Win=65536 Len=65483 TSval=1544020675 TSecr=1544020674
113 7.926947863	localhost	localhost	TCP	78 [TCP Dup ACK 105#3] freeciv(5556) → 34648 [ACK] Seq=16 Ack=2394494 Win=3112448 Len=0 TSval=1544020675 TSecr=1544020674...
114 7.926971911	localhost	localhost	TCP	65549 34648 → freeciv(5556) [ACK] Seq=2787392 Ack=16 Win=65536 Len=65483 TSval=1544020675 TSecr=1544020675
115 7.929188392	localhost	localhost	TCP	78 [TCP Dup ACK 105#4] freeciv(5556) → 34648 [ACK] Seq=16 Ack=2394494 Win=3112448 Len=0 TSval=1544020675 TSecr=1544020674...
116 7.929131217	localhost	localhost	TCP	65549 [TCP Fast Retransmission] 34648 → freeciv(5556) [ACK] Seq=2394494 Ack=16 Win=65536 Len=65483 TSval=1544020677 TSecr=15...
117 9.018489046	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 → freeciv(5556) [ACK] Seq=2394494 Ack=16 Win=65536 Len=65483 TSval=1544021767 TSecr=1544020...

40 1.073624292	localhost	localhost	TCP	65548 [TCP Out-Of-Order] 34648 - freeciv(5556) [PSH, ACK] Seq=327415 Ack=1 Win=65536 TStamp=1544013822 TSectr=154401...
41 1.073637148	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=589424 Win=1899136 Len=0 TStamp=1544013822 TSectr=1544013822
42 1.073645805	localhost	localhost	TCP	32834 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=622192 Ack=1 Win=65536 Len=32768 TStamp=1544013822 ...
43 1.073651893	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=654960 Ack=1 Win=65536 Len=32768 TStamp=1544013822 TSectr=1544013822
44 1.073656893	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=589424 Win=2039080 Len=0 TStamp=1544013822 TSectr=1544013822 S...
45 1.073668670	localhost	localhost	TCP	32834 [TCP Out-Of-Order] 34648 - freeciv(5556) [PSH, ACK] Seq=589424 Ack=1 Win=65536 Len=32768 TStamp=1544013822 TSectr=154401...
46 1.073675550	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=687728 Win=2191488 Len=0 TStamp=1544013822 TSectr=1544013822
47 1.073684225	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=687728 Ack=1 Win=65536 Len=32768 TStamp=1544013822 TSectr=1544013822
48 1.073690197	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=720496 Ack=1 Win=65536 Len=32768 TStamp=1544013822 TSectr=1544013822
49 1.247687612	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=753264 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr=1544013822
50 1.247685135	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=786032 Win=2322432 Len=0 TStamp=1544013996 TSectr=1544013996
51 1.247766321	localhost	localhost	TCP	32834 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=818800 Ack=1 Win=65536 Len=32768 TStamp=1544013996 ...
52 1.247779737	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=851568 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr=1544013996
53 1.247786900	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=884336 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr=1544013996
54 1.247802966	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=786032 Win=2453376 Len=0 TStamp=1544013996 TSectr=1544013996 S...
55 1.247809779	localhost	localhost	TCP	78 [TCP Dup ACK 56#1] freeciv(5556) - 34648 [ACK] Seq=1 Ack=786032 Win=2453376 Len=0 TStamp=1544013996 TSectr=1544013996 SL...
56 1.247817718	localhost	localhost	TCP	78 [TCP Dup ACK 56#2] freeciv(5556) - 34648 [ACK] Seq=1 Ack=786032 Win=2453376 Len=0 TStamp=1544013996 TSectr=1544013996 SL...
57 1.247822815	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=917104 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr=1544013996
58 1.247839621	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=949872 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr=1544013996
59 1.247852776	localhost	localhost	TCP	32834 [TCP Fast Retransmission] 34648 - freeciv(5556) [PSH, ACK] Seq=786032 Ack=1 Win=65536 Len=32768 TStamp=1544013996 TSectr...
60 1.247860687	localhost	localhost	TCP	78 [TCP Dup ACK 56#3] freeciv(5556) - 34648 [ACK] Seq=1 Ack=786032 Win=2453376 Len=0 TStamp=1544013996 TSectr=1544013996 SL...
61 2.676614929	localhost	localhost	TCP	32834 [TCP Retransmission] 34648 - freeciv(5556) [PSH, ACK] Seq=786032 Ack=1 Win=65536 Len=32768 TStamp=1544015425 TSectr=1544...
62 2.677173707	localhost	localhost	TCP	78 freeciv(5556) - 34648 [ACK] Seq=1 Ack=982640 Win=2455424 Len=0 TStamp=1544015425 TSectr=1544013996 SLE=786032 SRE=818800
63 2.677268249	localhost	localhost	TCP	32834 34648 - freeciv(5556) [PSH, ACK] Seq=982640 Ack=1 Win=65536 Len=32768 TStamp=1544015426 TSectr=1544015425
64 2.677292385	localhost	localhost	TCP	32834 34648 - freeciv(5556) [ACK] Seq=1015408 Ack=1 Win=65536 Len=32768 TStamp=1544015426 TSectr=1544015425
65 2.677333326	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=1015408 Win=2586368 Len=0 TStamp=1544015426 TSectr=1544015426
66 2.677389469	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=1048176 Win=2717312 Len=0 TStamp=1544015426 TSectr=1544015426
67 3.252728448	localhost	localhost	TCP	18145 [TCP Previous segment not captured] 34648 - freeciv(5556) [PSH, ACK] Seq=1080944 Ack=1 Win=65536 Len=18079 TStamp=15440...
68 3.252783003	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=1 Ack=1048176 Win=2848256 Len=0 TStamp=1544016001 TSectr=1544015426 ...
69 5.043901753	localhost	localhost	TCP	32834 [TCP Retransmission] 34648 - freeciv(5556) [PSH, ACK] Seq=1048176 Ack=1 Win=65536 Len=32768 TStamp=1544017792 TSectr=154...
70 5.044044402	localhost	localhost	TCP	66 freeciv(5556) - 34648 [ACK] Seq=1 Ack=1099023 Win=2979200 Len=0 TStamp=1544017792 TSectr=1544017792
71 5.044873359	localhost	localhost	TCP	77 freeciv(5556) - 34648 [PSH, ACK] Seq=1 Ack=1099023 Win=2979200 Len=11 TStamp=1544017793 TSectr=1544017792
72 5.044893736	localhost	localhost	TCP	66 34648 - freeciv(5556) [ACK] Seq=1099023 Ack=12 Win=65536 Len=0 TStamp=1544017793 TSectr=1544017793
73 5.045812981	localhost	localhost	TCP	65549 [TCP Previous segment not captured] 34648 - freeciv(5556) [ACK] Seq=1164506 Ack=12 Win=65536 Len=65483 TStamp=154401779...
74 5.045846366	localhost	localhost	TCP	78 [TCP Window Update] freeciv(5556) - 34648 [ACK] Seq=12 Ack=1099023 Win=3110144 Len=0 TStamp=1544017794 TSectr=1544017793...
75 5.046363184	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=1099023 Ack=12 Win=65536 Len=65483 TStamp=1544017795 TSectr=1544017...
76 6.707840365	localhost	localhost	TCP	65549 [TCP Retransmission] 34648 - freeciv(5556) [ACK] Seq=1099023 Ack=12 Win=65536 Len=65483 TStamp=1544019456 TSectr=1544017...

