

תיאור מערכת TCP AND Congestion Control Algorithms

2.....	סקירת מערכת
2.....	פונקציונליות המערכת
2.....	איך מריצים
3.....	עץ החלטות לאפשריות של המערכת
4-10.....	הסברים על הקוד (Sender)
11-17.....	הסברים על הקוד (Receiver)
18-22.....	תמונות הרצה של המערכת

סקירת מערכת

המערכת מאפשרת שליחת קובץ בין 2 צדדים, צד שולח (Sender) וצד מקבל (Receiver).

בשלב ראשון מגדירים את שם הקובץ וגודלו ולאחר מכן נפתח בין ה-Sender ל-Receiver סוקט תחילה שולחים את החצי הראשון של הקובץ לצד המקבל, כשהוא סיים לקבל את כול חצי הקובץ הוא שולח בחזרה הודעת אותנטיקציה לצד השולח כדי לאמת שכול הביטים הגיעו כמו שצריך.

לאחר שהצד השולח קיבל את האותנטיקציה ואימת אותה שהיא אכן נכונה הוא ממשיך לשלב הבא שהוא לשלוח את החלק השני של הקובץ, כעת הצד המקבל שוב ישלח לצד השולח את הודעת האותנטיקציה כי לאמת שאכן החלק השני הגיע כמו שצריך והצד השולח יאמת שאכן האותנטיקציה נכונה ובכך יסתיים שליחת הקובץ בין 2 הצדדים.

פונקציונליות המערכת

למערכת יש פונקציונליות לשליחת הקובץ בחלקים מספר פעמים לפי בקשת המשתמש, בסיום כול שליחת קובץ ב-2 חלקים הצד השולח יוצג הודעה האם לשלוח את קובץ שוב אם הוא יבחר שכן הקובץ ישלח שוב לצד המקבל ב-2 חלקים.

בנוסף כדי לוודא שהצד השולח רוצה לסיים את שליחת הקובץ ולסיים את ההתקשרות מוצגת לשולח הודעה לאחר שהוא בחר לא לשלוח את הקובץ שוב הודעה האם לצאת אם בחר שכן נגמר שליחת הקובץ וסוגרים את הסוקט, אם בחר שלא הקובץ ישלח שוב כיון שהוא לא רוצה לצאת משליחת הקובץ.

איך מריצים

המערכת רצה על מערכת הפעלה Linux בלבד.

מצורף למערכת קובץ Makefile שבו כתובים פקודות שמקמפלות את הקבצים ומכינים אותם להרצה.

לכן בכדי להריץ יש לבצע את השלבים הבאים:

1. להיכנס לתיקייה שבה יש את הקבצים של המערכת, לפתוח את הקובץ Sender ובו בשורה 15 להכניס את שם הקובץ שאותו נרצה לשלוח.

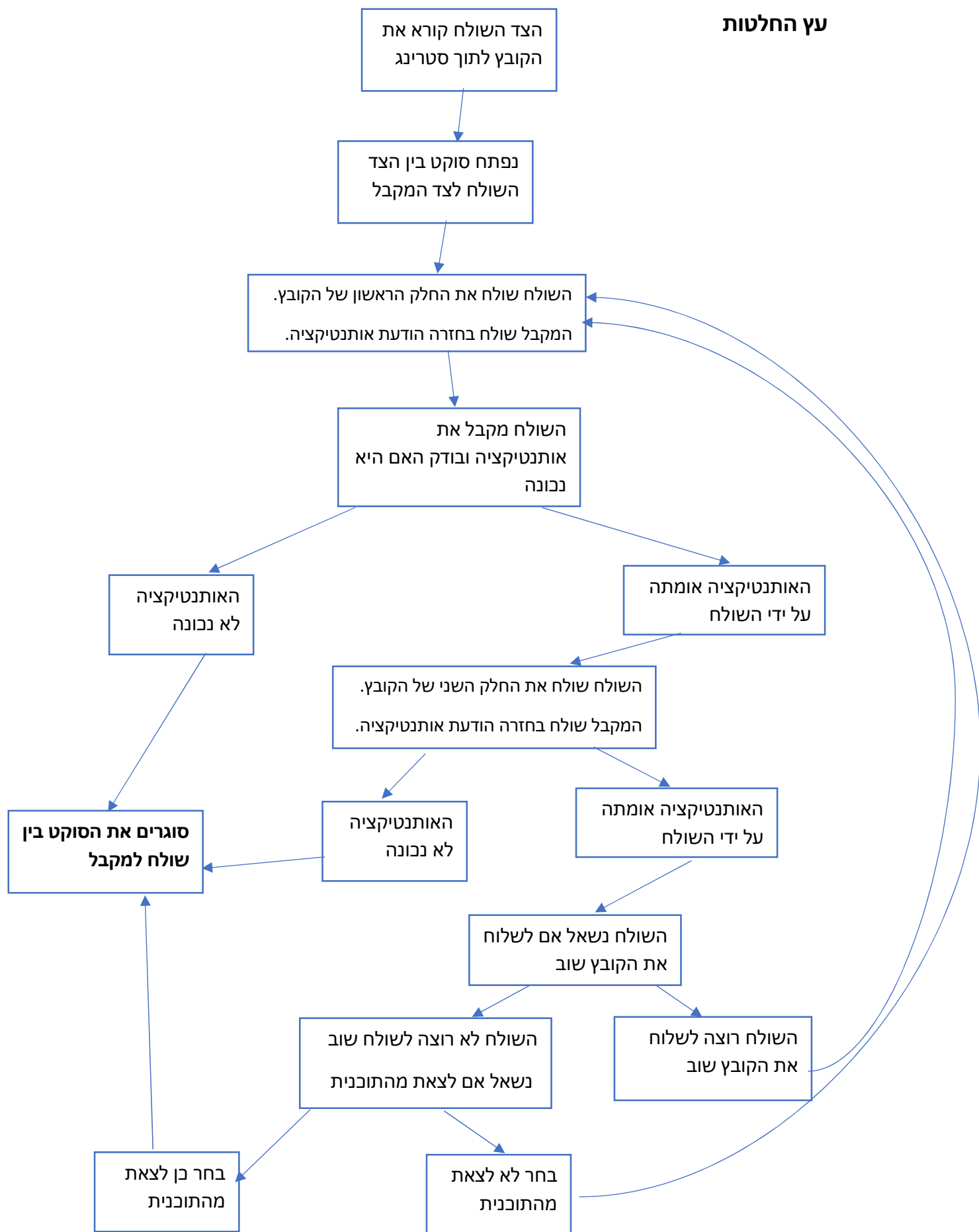
בשורה 20 נשנה את גודל הקובץ שאותה נרצה לשלוח, נשמור את הקובץ ונסגור אותו.

2. נוודא שאנחנו בתיקייה שבה יש את הקבצים של המערכת ומתוכה להיכנס ל-terminal ולרשום את הפקודה Make all בכדי שהקבצים יתקמפלו ויהיו מוכנים להרצה.

3. באותו חלון של terminal שפתחנו קודם נרשום את הפקודה Make runr בכדי להריץ את ה-Receiver כדי שנוכל להתחיל את פעולת השליחה.

4. לפתוח חלון terminal חדש עם הנתיב של התיקייה שבה הקבצים נמצאים ולכתוב את הפקודה Make runs כדי להריץ את ה-Sender ולהתחיל את שליחת הקובץ.

עץ החלטות



Sender

```
2
3 #define PORT 9999
4 #define SERVER_IP_ADDRESS "0.0.0.0"
5 #define SENDFILE "text.txt" // the file to send
6 #define id1 7084
7 #define id2 383 // first dig is 0 so 0383
8
9 int checkauthentication(int sock);
10 long SizeFile=1048580 ;
```

תחילה הגדרנו את הקבועים של התוכנית : הפורט עליו נעבוד , כתובת הIP של המחשב , שם הקובץ שאותו נשלח , 4 ספרות האחרונות של התעודות זהות שלנו (ב ID השני הספרה הראשונה היא 0 לכן השתמשנו ב 3 ספרות ולא ב 4 כי פעולת XOR לא התייחסה לספרה 0).

הוספנו חתימה לפונקציה שמקבלת כפרמטר את הסוקט ובודקת אם האותנטיקציה תואמת להודעה מהשולח של האותנטיקציה ששלח אם כן מחזירה 1 , אם לא תואם מחזירה 0.

לבסוף הגדרנו את גודל הקובץ ששולחים.

```
int main(int argc, char const *argv[])
{
    FILE *fp;
    char *filename = SENDFILE;

    fp = fopen(filename, "r"); // open file to send

    if (fp == NULL) // check if the file open successfully
    {
        perror("Can't get filename");
        exit(1);
    }

    char message[SizeFile];

    fread(message,sizeof(char),SizeFile,fp); // insert the file in to the string

    fclose(fp); // close the file
```

שלב ראשון הגדרנו מציע מסוג FILE (fp) שיצביע על הקובץ ופתחנו את הקובץ עם הרשאת קריאה בלבד.

בדקנו שבאמת הצלחנו לפתוח את הקובץ בכך שהמציע fp הוא לא Null שהוא לא מציע על כתובת 0 .

הגדרנו מערך של char בגודל הקובץ בכדי לקרוא אליו את הקובץ.

קראנו את הקובץ למערך באמצעות fread ולבסוף סגרנו את הקובץ באמצעות fclose .

```

int sock = socket(AF_INET, SOCK_STREAM, 0); // create socket
if (sock == -1)
{
    printf("Unable to create a socket : %d", errno);
    exit(1);
}

// "sockaddr_in" used for IPv4 communication
struct sockaddr_in serverAddress;
memset(&serverAddress, 0, sizeof(serverAddress)); // file struct with 0
serverAddress.sin_family = AF_INET; // work with ipv4
serverAddress.sin_port = htons(PORT); // insert to struct the port
int rval = inet_pton(AF_INET, (const char *)SERVER_IP_ADDRESS, &serverAddress.sin_addr); // convert the address to binary
if (rval <= 0)
{
    printf("inet_pton failed");
    exit(1);
}

```

שלב שני יצרנו סוקט באמצעות הפקודה socket שמקבלת כפרמטים את סוג ה IP איתו אנחנו עובדים וסוג הפרוטוקול (אצלנו TCP), הפונקציה מחזירה ערך שלם לכן לאחר מכן בדקנו אם הפונקציה החזירה 1- אם כן יצירת הסוקט לא הצליחה הדפסנו הודעת שגיאה ויצאנו מהתוכנית.

שלב שלישי הגדרנו מבנה בשם sockaddr_in שהוא מבנה שמכיל מידע של כתובת IP פורט עליו עובדים וסוג ה IP שאיתו אנחנו עובדים גירסא 4 או 6.

תחילה איפסנו את המבנה באמצעות memset באפסים.

ואז הגרנו לו שאנחנו עובדים עם IP V4 והכנסנו לו את הפורט עליו אנחנו עובדים שאותו ההמרנו לביטים באמצעות htons.

ולבסוף המרנו את הכתובת לייצוג בינארי באמצעות פונקציית inet_pton שמקבלת כפרמטרים סוג ה IP עליו עובדים מצביע לכתובת ה IP ואת המקום במבנה שבוא מאחסנים את הכתובת לאחר ההמרה.

פונקציה inet_pton ערך שלם 1- אם יש שגיאה 0 אם ההמרה לא עברה בהצלחה או מספר גדול מ 0 אם ההמרה בוצע בהצלחה לכן הוספנו תנאי שבודק שהערך המוחזר גדול מ 0 ואם לא מציג הודעת שגיאה ויוצא מהתוכנית.

```

if (connect(sock, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) == -1) // Make a connection to the receiver with socket
{
    printf("connect failed with error code : %d", errno);
}

printf("connected to server\n\n");

```

ובשלב הרביעי הקמנו חיבור עם ה Receiver באמצעות הפונקציה connect שמקבלת כפרמטים socket descriptor שקיבלנו מפונקציה socket, מצביע למבנה שהגדרנו קודם וגודל המבנה שהגדרנו בביטים.

הפונקציה מחזירה 1- אם החיבור לא הוקדם לכן הוספנו בדיקה שהערך החזרה הוא לא 1- ואם הוא כן הפונקציה מדפיסה שגיאה.

```

while (1)
{
    char ccalgo[7]
    if (setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, ccalgo, strlen(ccalgo)) != 0) //change cc algorithm to reno
    {
        printf("Error in change cc algorithm to reno\n");
        exit(1);
    }
    else
    {
        printf("change cc algorithm to reno \n\n");
    }
}

```

לאחר שהקמנו חיבור עם ה Receiver הגדרנו לולאת while אינסופית כיון שלא ידעו לנו מספר הפעמים שנצטרך לשלוח את הקובץ.

בשלב ראשון בלולאה הגדרנו את CC אלגוריתם להיות reno באמצעות פונקציה setsockopt שמקבלת כפרמטרים את הסוקט, באיזה פרוטוקול אנחנו עובדים, אתה מה אנחנו רוצים לשנות (TCP CONGESTION), לאיזה שיטה לשנות, ואת כמות הביטים בסטרינג שמכיל את השיטה שאליה נרצה לשנות.

הפונקציה מחזירה מספר שלם, אם החזירה 0 זה אומר שהיה שגיאה לכן נבדוק שלא קיבלנו 0 ואם כן קיבלנו 0 נציג הודעת שגיאה ונצא מהתוכנית ואם הצליחה לשנות נציג הודעה שהאלגוריתם שונה לreno.

```

long BytesSent = 0; // counting how much byte are send to receiver
long SizeFileToSend = SizeFile / 2; // amount of half file
long BytesLeft = SizeFileToSend; // initialize how much byte left to send
while (BytesSent < SizeFileToSend)
{
    int SendMess = send(sock, message+BytesSent, BytesLeft, 0); // send message to receiver
    if (-1 == SendMess)
    {
        printf("Error in sending file: %d", errno);
    }
    else if (0 == SendMess)
    {
        printf("peer has closed the TCP connection prior to send().\n");
    }
    BytesSent += SendMess; // add the number of byte that arrive to receiver
    BytesLeft -= SendMess; // subtraction the number of byte that arrive receiver
}
printf("Send %ld bytes of file.\n", BytesSent); // print how much byte send to receiver

```

בשלב שני נגדיר מספר פרמטרים שאיתם נעבוד בשליחת 2 החלקים של הקובץ 1. כמה ביטים נשלחו עד כה מאותחל ב 0 2. כמה ביטים נרצה לשלוח בחלק הזה מאותחל בחצי מגודל הקובץ 3. כמה ביטים נשארו לשלוח מאותחל בחצי מגודל הקובץ.

לאחר מכן נכנס ללולאת while שרצה כול עוד לא שלחנו את כול הביטים שרצינו לשלוח.

בתוך הלולאה נשלח את הביטים באמצעות פונקציה send שמקבלת כפרמטרים את הסוקט, מצביע להודעה שנרצה לשלוח החל ממנה וכמה ביטים לשלוח.

הפרמטר של המצביע להודעה שיתנה בהתאם להתקדמות של הלולאה באמצעות הוספת כמות הביטים שנשלחו למצביע של תחילת ההודעה.

הפרמטר של כמות הביטים שישלחו יקטן בהתאם להתקדמות הלולאה ולביטים שנשארו לשלוח. הפונקציה מחזירה ערך שלם אם הוחזר 1- זה אומר שהייתה שגיאה בשליחה אם הוחזר 0 זה אומר שהצד השני סגר את החיבור וערך גדול מ 0 שהוא כמות הביטים שנשלחו לכן נבדוק שלא הוחזר לנו אף אחד מהערכים האלו ואם כן נציג הודעה בהתאם.

נוסיף למשתנה שהגדרנו בו את כמות ביטים את כמות הביטים שנשלחו בפונקציה Send ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לשלוח.

לסוף שנצא מהלולאה לאחר ששלחנו את כול הביטים נדפיס הודעה עם כמות הביטים ששלנו.

```
printf("first part of file send successfully\n");

int cheaut = checkauthentication(sock); // check authentication

if (cheaut == 1) // return 1 if the authentication is correct
{
    printf("authentication for first part are successfully\n\n\n");
} else {
    break;
}
```

לאחר שליחת החלק הראשון נדפיס שהחלק הראשון נשלח בהצלחה.

לאחר מכן נקרא לפונקציה checkauthentication שהיא פונקציה שמקבלת דרך הסוקט הודעת אותנטיקציה ובודקת האם היא נכונה ואם נכונה מחזירה 1 אחרת 0.

לאחר שקראנו ל checkauthentication נבדוק מה הערך המוחזר אם הערך הוא 1 נדפיס הודעה שהאותנטיקציה עברה בהצלחה אחרת נצא מהתוכנית ונסגור את הסוקט.

```
char ccalgo2[7] = "cubic";
if (setsockopt(sock, IPPROTO_TCP, TCP_CONGESTION, ccalgo2, strlen(ccalgo2)) != 0) //change cc algorithm
{
    printf("Error in change cc algo\n");
    exit(1);
}
else
{
    printf("change cc algo to cubic \n\n");
}
```

כעת נשנה את ה CC אלגוריתם להיות cublic באמצעות פונקציה setsockopt שמקבלת כפרמטרים את הסוקט, באיזה פרוטוקול אנחנו עובדים, אתה מה אנחנו רוצים לשנות (TCP CONGESTION), לאיזה שיטה לשנות, ואת כמות הביטים בסטרינג שמכיל את השיטה שאליה נרצה לשנות.

הפונקציה מחזירה מספר שלם, אם החזירה 0 זה אומר שהיה שגיאה לכן נבדוק שלא קיבלנו 0 ואם כן קיבלנו 0 נציג הודעת שגיאה ונצא מהתוכנית ואם הצליחה לשנות נציג הודעה שהאלגוריתם שונה ל cublic.

```

BytesSent = 0; // counting how much byte are send to receiver
BytesLeft = SizeFileToSend; // initialize how much byte left to send
while (BytesSent < SizeFileToSend)
{
    int SendMess = send(sock,message+BytesSent+SizeFile/2,BytesLeft, 0); //send message to r
    if (-1 == SendMess)
    {
        printf("Error in sending file: %d", errno);
    }
    else if (0 == SendMess)
    {
        printf("peer has closed the TCP connection prior to send().\n");
    }
    BytesSent += SendMess; // add the number of byte that arrive receiver
    BytesLeft -= SendMess; // subtraction the number of byte that arrive receiver
}
printf("Send %ld bytes of file.\n", BytesSent); // print how much byte send to receiver

```

עכשיו נרצה לשלוח את החלק השני של הקובץ לכן נאפס את המשתנה שמכיל את כמות הביטים שנשלחו להיות 0.

ואת המשתנה ביטים שנשארו להיות חצי מגדול הקובץ.

לאחר מכן נכנס ללולאת while שרצה כול עוד לא שלחנו את כול הביטים שרצינו לשלוח.

בתוך הלולאה נשלח את הביטים באמצעות פונקציה send שמקבלת כפרמטרים את הסוקט, מצביע להודעה שנרצה לשלוח החל ממנה וכמה ביטים לשלוח.

הפרמטר של המצביע להודעה שיתנה בהתאם להתקדמות של הלולאה באמצעות הוספתה כמות הביטים שנשלחו למצביע של תחילת ההודעה.

הפרמטר של כמות הביטים שישלחו יקטן בהתאם להתקדמות הלולאה ולביטים שנשארו לשלוח.

הפונקציה מחזירה ערך שלם אם הוחזר 1- זה אומר שהייתה שגיאה בשליחה אם הוחזר 0 זה אומר שהצד השני סגר את החיבור וערך גדול מ 0 שהוא כמות הביטים שנשלחו לכן נבדוק שלא הוחזר לנו אף אחד מהערכים האלו ואם כן נציג הודעה בהתאם.

נוסיף למשתנה שהגדרנו בו את כמות ביטים את כמות הביטים שנשלחו בפונקציה Send ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לשלוח.

לסוף שנצא מהלולאה לאחר ששלחנו את כול הביטים נדפיס הודעה עם כמות הביטים ששלנו.

```

printf("second part of file send successfully\n");

cheaut = checkauthentication(sock); // check authentication

if (cheaut == 1)
{
    printf("authentication for second part are successfully\n\n\n");
}else{
    break;
}

```


לאחר שליחת החלק השני נדפיס שהחלק השני נשלח בהצלחה.

לאחר מכן נקרא לפונקציה checkauthentication שהיא פונקציה שמקבלת דרך הסוקט הודעת אותנטיקציה ובודקת האם היא נכונה ואם נכונה מחזירה 1 אחרת 0.

לאחר שקראנו ל checkauthentication נבדוק מה הערך המוחזר אם הערך הוא 1 נדפיס הודעה שהאותנטיקציה עברה בהצלחה אחרת נצא מהתוכנית ונסגור את הסוקט.

```
char ch1 , ch2 ;
printf("Send the file again? y to yes or n to no\n"); //check if send file again
ch2 = getchar();
getchar();
if (ch2 == 'n' || ch2 == 'N') // if no check if exit
{
    printf("exit? y to yes or n to no\n");
    ch1 = getchar();
    getchar();
    if (ch1 == 'y' || ch1 == 'Y') //if exit
    {
        char ExitMess[] = "exit";
        int BytesSent = send(sock, ExitMess, sizeof(ExitMess), 0); // send exit message
        if (BytesSent == -1)
        {
            printf("Error in sending size of file");
            exit(1);
        }
        else if (BytesSent == 0)
        {
            printf("peer has closed the TCP connection prior to send.\n");
        }
        break; // break loop
    }
}
else if (ch2 == 'y')
{
    continue;
}
```

בחלק האחרון של הלולאה נשאל את השולח 2 שאלות האם לשלוח את הקובץ עוד פעם? אם הוא בוחר שכן אז מבצעים את הלולאה שוב אם הוא בוחר שלא מוצגת לו עוד הודעה האם צאת?

אם הוא בוחר לצאת הוא שולח הודעת EXIT לReceiver והשולח סוגר את החיבור עם לReceiver באמצעות הפקודה break וגם ה Receiver סוגר את החיבור מהצד שלו בשמגיעה לו הודעת EXIT.

אם הוא בוחר שלא לצאת אז הלולאה מתבצעת שוב.

את הבחירות אם לשלוח שוב ולצאת קולטים מ Buffer באמצעות פונקציית getchar כיון נאמר לשולח להכניס תו אחד שמייצג את הבחירה שלו.

```
printf("\nclose socket\n");
close(sock); // close socket
return 0;
```

ולבסוף שהלולאה נשברת באמצעות break מגיעים לחלק של סגירת החיבור ומדפיסים הודעה שהחיבור נסגר.

```
int checkauthentication(int sock)
{
    // receive the authentication code from the server
    char auth[5];
    char XorIds[5];
    sprintf(XorIds, "%d", id1 ^ id2); // insert to temp id1 xor id2
    int bytes = recv(sock, auth, sizeof(auth), 0); // recving the authentication code
    if (bytes == -1)
    {
        printf("Error in receiving authentication code\n");
        exit(1);
    }
    else if (bytes == 0)
    {
        printf("peer has closed the TCP connection prior to receive.\n");
        exit(1);
    }
    if (strcmp(auth, XorIds) == 0) //check if the authentication from receiver is correct
    {
        return 1;
    }
    return 0;
}
```

פונקציית checkauthentication

הפונקציה מקבל את הסוקט בפרמטר.

מגדירה 2 סטרינגים בגודל 5, בסטרינג בשם XorIds מכניסים את פעולת XOR בין 2 תעודות זהות באמצעות sprintf שממירה מספר מסוג int ל סטרינג.

נקבל מreceiver את הודעת האותנטיקציה שלו באמצעות פונקציה recv שמחזירה ערך שלם אם מחזירה -1 זה אומר שהיה שגיאה בקבלת ההודעה, אם 0 זה אומר שהחיבור נסגר מצד של Receiver אם מספר גדול מ 0 אז זה מייצג את כמות הביטים שנקראו.

לבסוף משווים בין פעולת XOR שעשינו להודעת האותנטיקציה שקיבלנו, אם הם שווים נחזיר 1 אם הם שונים נחזיר 0.

Receiver

```
#define SERVER_PORT 9999 // The port that the receiver listens
#define id1 7084
#define id2 383

int CountMessArrive = 0; // count how much time sender send part of t
double TotalTime=0; // sum all the time
long SizeFile=1048580;

void sendauthentication(int);
```

תחילה הגדרנו את הקבועים של התוכנית : הפורט עליו נעבוד, 4 ספרות האחרונות של התעודות זהות שלנו (ב ID השני הספרה הראשונה היא 0 לכן השתמשנו ב 3 ספרות ולא ב 4 כי פעולת XOR לא התייחסה לספרה 0).

הגדרנו משתנה שסופר את כמות הפעמים שקיבלנו חלק מסויים מהקובץ.

משתנה שסובם את זמני השליחה של כל חלק.

ולבסוף משתנה שמכיל את הגודל הקובץ שנשלח.

הוספנו חתימה לפונקציה ששולחת הודעת האותנטיקציה לשולח כשמקבל כפרמטר את הסוקט.

```
int main()
{
    int listenSocket = -1; // create listening socket
    if ((listenSocket = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        printf("Could not create listening socket : %d", errno);
        exit(1);
    }
    printf("Sokcet created\n");
```

שלב ראשון ניצור את הסוקט באמצעות הפקודה socket שמקבלת כפרמטים את סוג ה IP איתו אנחנו עובדים וסוג הפרוטוקול (אצלנו TCP), הפונקציה מחזירה ערך שלם לכן לאחר מכן בדקנו אם הפונקציה החזירה -1 אם כן יצירת הסוקט לא הצליחה הדפסנו הודעת שגיאה ויצאנו מהתוכנית.

אם הצליחה ליצור סוקט נדפיס הודעה בסוקט נוצר.

```
int yes = 1; // check if the ip in not in use
if (setsockopt(listenSocket, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof yes) == -1)
{
    perror("setsockopt");
    exit(1);
}
```

לאחר מכן נבדוק שה IP לא בשימוש באמצעות פונקציית `setsockopt` הפונקציה מחזירה ערך שלם אם הוחזר 1- זה אומר שעדיין יש ביטים של סוקט שעדיין מחוברים לכן נדפיס הודעת שגיאה ונצא מהתוכנית.

```
// "sockaddr_in" used for IPv4 communication
struct sockaddr_in serverAddress;
memset(&serverAddress, 0, sizeof(serverAddress));

serverAddress.sin_family = AF_INET;
serverAddress.sin_addr.s_addr = INADDR_ANY;
serverAddress.sin_port = htons(SERVER_PORT);
```

הגדרנו מבנה בשם `sockaddr_in` שהוא מבנה שמכיל מידע של כתובת IP פורט עליו עובדים וסוג ה IP שאיתנו אנחנו עובדים גירסא 4 או 6.

תחילה איפסנו את המבנה באמצעות `memset` באפסים.

ואז הגרנו לו שאנחנו עובדים עם IP V4 והכנסנו לו את הפורט עליו אנחנו עובדים שאותו ההמרנו לביטים באמצעות `htons`.

```
// Bind the socket to the port with any IP at this port
if (bind(listenSocket, (struct sockaddr *)&serverAddress, sizeof(serverAddress)) == -1)
{
    printf("Bind failed with error code : %d", errno);
    close(listenSocket);
    exit(1);
}

printf("Bind success\n");
```

נקשר את הכתובת והפורט עם הסוקט באמצעות הפונקציה `bind` שמקבלת כפרמטים את הסוקט, את המבנה שהגדרנו קודם ואת הגודל שלו.

הפונקציה מחזירה ערך שלם, אם הוחזר 1- זה אומר שהיה שגיאה בקישור בין הכתובת והפורט לסוקט אם הוחזר ערך אחר הקישור נוצר בהצלחה.

לכן נבדוק אם הוחזר ערך 1- אם כן נציג שגיאה נסגור את הסוקט ונצא מהתוכנית.

```
// Make the socket listening; actually mother of all client s
if (listen(listenSocket, 1) == -1) // 1 is a Maximum size of
{
    printf("listen failed with error code : %d", errno);
    close(listenSocket);
    exit(1);
}
```

נגדיר את הסוקט שלנו שיכול להאזין ללקוח אחד בו זמנית באמצעות פונקציית listen.

הפונקציה מחזיר ערך שלם אם מחזירה 1- זה אומר שהייתה שגיאה אם ערך אחר זה אומר ש ההגדרה בוצע בהצלחה , לכן נבדוק שלא הוחזר לנו 1- אם כן נדפיס הודעת שגיאה נסגור את הסוקט ונצא מהתוכנית.

```
printf("Waiting for incoming TCP-connections\n");

struct sockaddr_in ClientAddress;
socklen_t clientAddressLen = sizeof(ClientAddress);
memset(&ClientAddress, 0, sizeof(ClientAddress)); // file struct with 0

int ClientSocket = accept(listenSocket, (struct sockaddr *)&ClientAddress, &clientAddressLen); //
if (ClientSocket == -1)
{
    printf("listen failed with error code : %d", errno);
    return -1;
}

printf("A new client connection accepted\n\n");
```

בחלק הבאה נדפיס הודעה שאנחנו מחכים לחיבור.

נגדיר עוד מבנה שמכיל את ה IP והפורט של ה Sender , ונגדיר משתנה שמכיל את הגדול של המבנה הזה.

נאפס את המבנה באמצעות memset באפסים.

נוציא בקשה לחיבור מתוך תור הבקשות באמצעות הפונקציה accept שמקבל כפרמטר את הסוקט , מצביע למבנה שמכיל מידע על הכתובת והפורט של ה Sender ואת גודל המבנה בביטים.

הפונקציה מחזיר ערך שלם אם הוחזר 1- סימן שהייתה שגיאה אחרת החיבור בוצע בהצלחה , לכן נבדוק שלא הוחזר 1- אם כן נדפיס הודעת שגיאה ונצא מהתוכנית.

אם החיבור נוצר בהצלחה נדפיס הודעה של הבוצע התחברות של לקוח חדש.

```
double times[100]; // assuming that send the file not over the 50 times
while (1)
{
    char ccalgo[7]="reno";
    if (setsockopt(ClientSocket, IPPROTO_TCP, TCP_CONGESTION, ccalgo, strlen(ccalgo)) != 0) //
    {
        printf("Error in change cc reno\n");
        exit(1);
    }else{
        printf("change cc algo to reno \n");
    }
}
```

נגדיר מערך של דאבל בגדול 100 (בהנחה שלא יבוצעו 50 פעמים שליחת הקובץ מחדש) שיכיל את הזמנים שלכל לכול חלק להגיע.

נכנס ללולאת while אינסופית כיון שלא יודעים כמה פעמים נתבקש לשלוח את הקובץ ונגדיר את ה CC אלגוריתם להיות reno באמצעות פונקציה setsockopt שמקבלת כפרמטרים את הסוקט

באיזה פרוטוקול אנחנו עובדים , אתה מה אנחנו רוצים לשנות (TCP CONGESTION), לאיזה שיטה לשנות , ואת כמות הביטים בסטרינג שמכיל את השיטה שאליה נרצה לשנות.

הפונקציה מחזירה מספר שלם , אם החזירה 0 זה אומר שהיה שגיאה לכן נבדוק שלא קיבלנו 0 ואם כן קיבלנו 0 נציג הודעת שגיאה ונצא מהתוכנית ואם הצליחה לשנות נציג הודעה שהאלגוריתם שונה לreno.

```
long BytesLeft = SizeFile/2; // initialize how much byte left to received
char buffer[SizeFile /2];
long BytesReceived = 0; // counting how much byte received from sender
clock_t start = clock(); // start measure time
while (BytesReceived < SizeFile/2)
{
    int MessRecv = recv(ClientSocket, buffer, BytesLeft, 0); // receive the message
    BytesReceived += MessRecv; // add the number of byte that arrive from sender
    BytesLeft -= MessRecv; // subtraction the number of byte that left to receive
    if (MessRecv <= 0)
    {
        break;
    }
}
clock_t end = clock(); // stop measure time
double time= (double)(end - start)/CLOCKS_PER_SEC; // calculating the time to take the
times[CountMessArrive++]=time;
TotalTime += time; // add time to total time
```

נגדיר מספר פרמטים שאיתם נעבוד בקבלת 2 החלקים של הקובץ 1. כמה ביטים נשאר לקבל בחצי מגודל הקובץ 2. buffer שאליו נכניס את ההודעה שנקבל מהSender מאותחל בגודל חצי מגודל הקובץ 3. כמה ביטים קיבלנו מאותחל ב0 4. ומשתנה שמכיל את הזמן מתחילת התוכנית בזמן שמתחילים את קבלת הקובץ.

לאחר מכן נכנס ללולאת while שרצה כול עוד לא קיבלנו את כול הביטים שרצינו לקבל.

בתוך הלולאה נקבל את הביטים באמצעות פונקציה recv שמקבלת כפרמטרים את הסוקט, buffer להכנסת ההודעה לשם וכמה ביטים נשאר לקבל.

הפרמטר של כמות הביטים שנשאר לקבל יקטן בהתאם להתקדמות הלולאה ולביטים שהתקבלו.

הפונקציה מחזירה ערך שלם אם הוחזר 1- זה אומר שהייתה שגיאה בשליחה אם הוחזר 0 זה אומר שהצד השני סגר את החיבור וערך גדול מ 0 שהוא כמות הביטים שנשלחו לכן נבדוק שלא הוחזר לנו אף אחד מהערכים האלו ואם כן נציג הודעה בהתאם.

נוסיף למשתנה שהגדרנו בו את כמות ביטים שהתקבלו את כמות הביטים שהתקבלו בפונקציה recv ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לקבל.

כשנצא מהלולאה נדגום שוב את הזמן מתחילת התוכנית, נחסיר את מזמן הסיום את הזמן ההתחלה ונשמור במשתנה , נוסיף את הזמן למערך ונקדם באחד את המשתנה שמכיל את כמות הפעמים שקיבלנו את אחד החלקים של הקובץ.

ולבסוף נוסיף למשתנה שמכיל את סך כול הזמן את הזמן האחרון שמדדנו.

```

if(strcmp(buffer,"exit")==0){ // check if get exit message
    break;
}
printf("Received %ld bytes\n\n", BytesReceived); // print how much
sendauthentication(ClientSocket); // send authentication

```

כעת נבדוק שלא קיבלנו הודעת EXIT מהשולח אם כן קיבלנו הודעה נצא מהלולאה ונסגור את הסוקט ואם לא קיבלנו הודעת EXIT נמשיך בקבלת החלקים של הקובץ.

נדפיס את הכמות ביטים שהתקבלו בלולאה הראשונה ונשלח לשולח הודעת האותנטיקציה באמצעות הפונקציה sendauthentication שכתבנו.

```

char ccalgo2[7]="cubic";
if (setsockopt(ClientSocket, IPPROTO_TCP,TCP_CONGESTION, ccalgo2, strlen(ccalgo2)) != 0) //
{
    printf("Error in change cc algo\n");
    exit(1);
}else{
    printf("change cc algo to cubic \n");
}

```

נשנה את ה CC אלגוריתם להיות cubic באמצעות פונקציה setsockopt שמקבלת כפרמטרים את הסוקט, באיזה פרוטוקול אנחנו עובדים, אתה מה אנחנו רוצים לשנות (TCP CONGESTION), לאיזה שיטה לשנות, ואת כמות הביטים בסטרינג שמכיל את השיטה שאליה נרצה לשנות.

הפונקציה מחזירה מספר שלם, אם החזירה 0 זה אומר שהיה שגיאה לכן נבדוק שלא קיבלנו 0 ואם כן קיבלנו 0 נציג הודעת שגיאה ונצא מהתוכנית ואם הצליחה לשנות נציג הודעה שהאלגוריתם שונה cubic.

```

BytesLeft = SizeFile / 2; // initialize how much byte left to receive
buffer[SizeFile / 2];
BytesReceived = 0; // counting how much byte received from sender
start = clock(); // start measure time
while (BytesReceived < SizeFile/2)
{
    int MessRecv = recv(ClientSocket, buffer, BytesLeft, 0); // receive message
    BytesReceived += MessRecv; // add the number of byte that arrived
    BytesLeft -= MessRecv; // subtraction the number of byte that arrived
    if (MessRecv <= 0)
    {
        break;
    }
}
end = clock(); // stop measure time
time = (double)(end - start)/CLOCKS_PER_SEC; // calculating the time
times[CountMessArrive++] = time;
TotalTime += time; // add time to total time

```

תחילה נאפס את המשתנים כמות הביטים שנשארו בחצי מגודל הקובץ, את הbuffer בגודל חצי מגודל הקובץ ואת המשתנה ביטים שהגיעו בערך 0.

ונדגום את הזמן מתחילת התוכנית.

לאחר מכן נכנס ללולאת while שרצה כול עוד לא קיבלנו את כול הביטים שרצינו לקבל.

בתוך הלולאה נקבל את הביטים באמצעות פונקציה recv שמקבלת כפרמטרים את הסוקט, buffer להכנסת הודעה לשם וכמה ביטים נשאר לקבל.

הפרמטר של כמות הביטים שנשאר לקבל יקטן בהתאם להתקדמות הלולאה ולביטים שהתקבלו.

הפונקציה מחזירה ערך שלם אם הוחזר 1- זה אומר שהייתה שגיאה בשליחה אם הוחזר 0 זה אומר שהצד השני סגר את החיבור וערך גדול מ 0 שהוא כמות הביטים שנשלחו לכן נבדוק שלא הוחזר לנו אף אחד מהערכים האלו ואם כן נציג הודעה בהתאם.

נוסיף למשתנה שהגדרנו בו את כמות ביטים שהתקבלו את כמות הביטים שהתקבלו בפונקציה recv ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לקבל.

כשנצא מהלולאה נדגום שוב את הזמן מתחילת התוכנית, נחסיר את מזמן הסיום את הזמן ההתחלה ונשמור במשתנה, נוסיף את הזמן למערך ונקדם באחד את המשתנה שמכיל את כמות הפעמים שקיבלנו את אחד החלקים של הקובץ.

ולבסוף נוסיף למשתנה שמכיל את סך כול הזמן את הזמן האחרון שמדדנו.

```
printf("Received %ld byte\n\n", BytesReceived); // print how mu
sendauthentication(ClientSocket); // send authentication
```

נדפיס את הכמות ביטים שהתקבלו בלולאה הראשונה ונשלח לשולח הודעת האותנטיקציה באמצעות הפונקציה sendauthentication שכתבנו.

```
printf("\next message arrived\n\n");
close(listenSocket); // close socket with sender
printf("socket close\n");
```

לבסוף לאחר שנצא מהלולאה נגיע לחלק של סגירת הסוקט לכן תחילה נדפיס הודעה שהגיע הודעת יציאה מהשולח, נסגור את הסוקט ונדפיס הודעה שהסוקט נסגר.

```
double TimeForFirstPart=0,TimeForSecondPart=0;
for (int i = 0; i <CountMessArrive; i=i+2)
{
    TimeForFirstPart+=times[i];
    TimeForSecondPart+=times[i+1];
    printf("time to receive first part in %d time is : %f", (i/2)+1, times[i]); //print how much tim
    printf("time to receive first second in %d time is : %f", (i/2)+1, times[i+1]); //print how much
}
printf("total average time for firts part = %f\n", TimeForFirstPart/((CountMessArrive)/2)); // pr
printf("total average time for second part = %f\n", TimeForSecondPart/((CountMessArrive)/2)); //
printf("total average time = %f\n", TotalTime/CountMessArrive); // print avarage time
```


ולבסוף נגיע לחלק של הזמנים.

נגדיר 2 משתנים מסוג דאבל ונאתחל אותם ב 0.

נעבור בלולאת FOR על המערך ונוסיף לכול משתנה את הזמן המתאים לו לפי החלק המתאים לו מ2 החלקים של הקובץ.

ונדפיס את הזמן לקח לקבל את ההודעה לכול חלק בכול פעם שנשלח הקובץ שוב.

ולבסוף לאחר הלולאה נדפיס את הזמן הממוצע של קבל החלק הראשון , את הזמן הממוצע של החלק השני , ואת הזמן הממוצע הכולל.

```
void sendauthentication(int sock) // send the authentication to the sender
{
    char authentication[5];
    sprintf(authentication, "%d", id1 ^ id2); // calculating the xor of id's
    long BytesSent = 0;
    long BytesLeft = 5;
    while (BytesSent < 5)
    {
        long bytes = send(sock, authentication, BytesLeft, 0); // send the au
        if (bytes == -1)
        {
            printf("Error in sending authentication");
            exit(1);
        }
        else if (bytes == 0)
        {
            printf("peer has closed the TCP connection prior to send.\n");
            exit(1);
        }
        BytesSent += bytes;
        BytesLeft -= bytes;
    }
}
```

פונקציית sendauthentication

נגדיר סטרינג בגודל 5 תווים ונכניס עליו את הפעולת XOR בין 2 התעודות הזהות.

נגדיר 2 משתנים שמכילים את כמות הביטים שנשלחו וכמות הביטים שנשארו לשלוח.

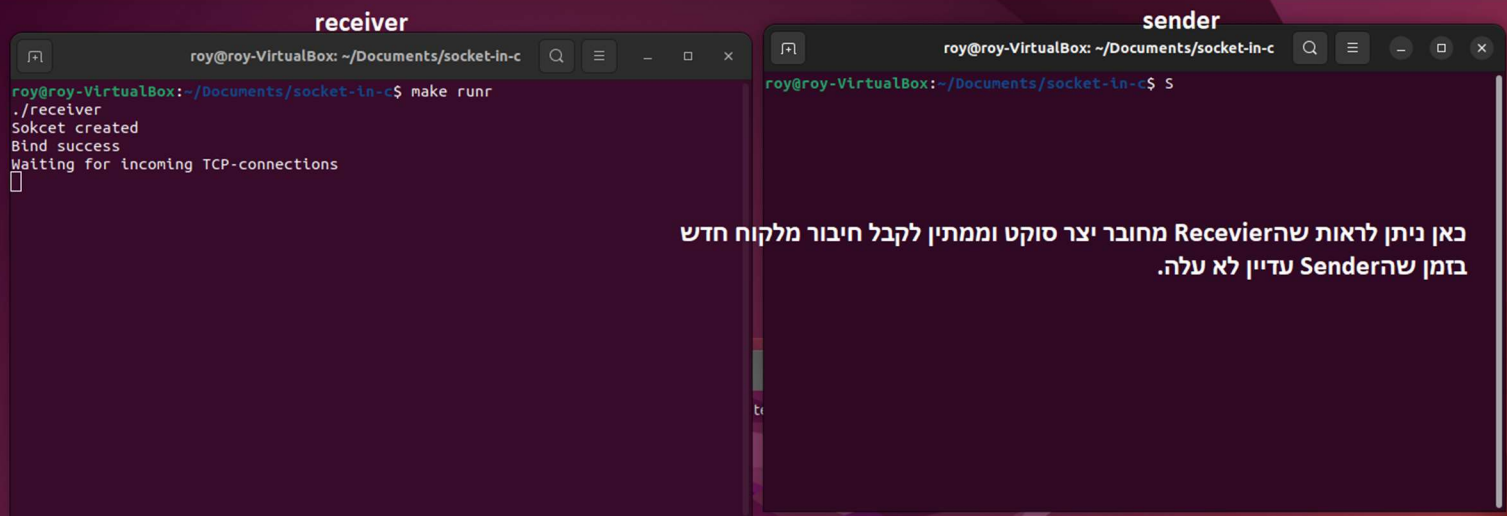
נרוץ בלולאת while עד שנשלח את כול 5 התווים , נשלח את האותנטיקציה באמצעות הפונקצייה send.

פונקציה זו מחזיר ערך 1- אם בוצע שגיאה בשליחה אם 0 עם החיבור נותק מצד השולח.

לכן נבדוק את ההערך החזרה ונציג הודעה בהתאם.

למשתנה כמות הביטים שנשלחו נוסיף את הכמות הביטים שנשלחו לפי פונקציה send ונחסיר את אותה כמות מכמות הביטים שנשארו לשלוח.

תמונות הרצה של ה sender ו receiver עם דוגמאות והצגת זמנים ממוצעים

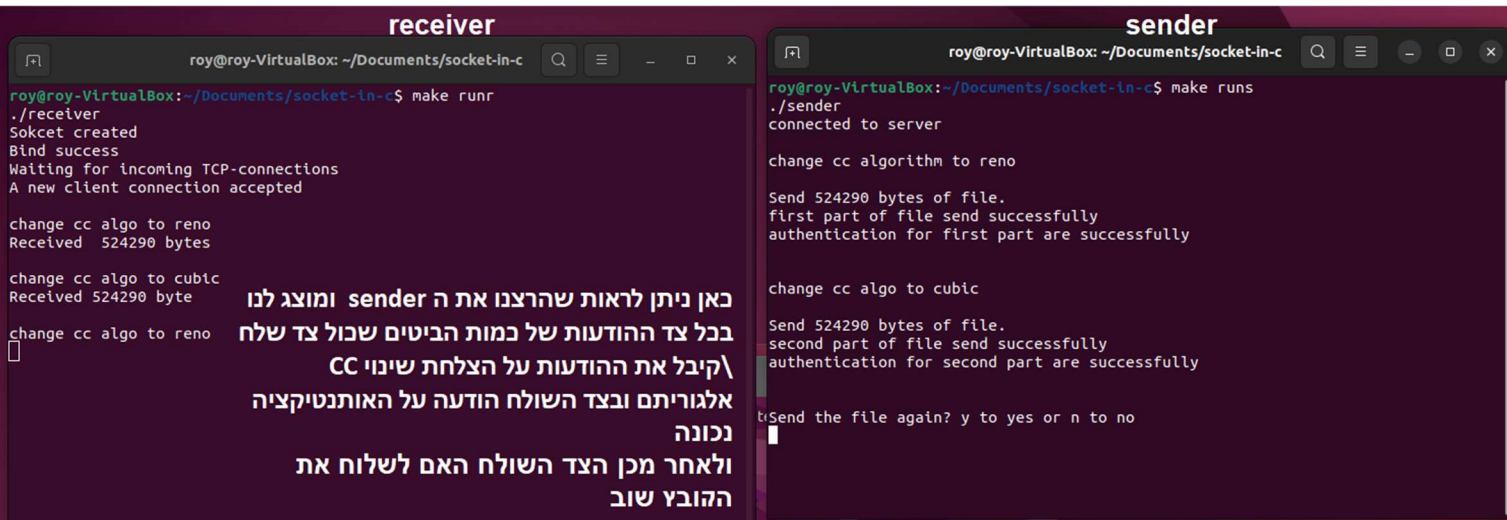


```
receiver
roy@roy-VirtualBox: ~/Documents/socket-in-c
roy@roy-VirtualBox:~/Documents/socket-in-c$ make runr
./receiver
Socket created
Bind success
Waiting for incoming TCP-connections
[ ]

sender
roy@roy-VirtualBox: ~/Documents/socket-in-c
roy@roy-VirtualBox:~/Documents/socket-in-c$ S
```

באן ניתן לראות שהReceiver מחובר יצר סוקט וממתין לקבל חיבור מלקוח חדש בזמן שהSender עדיין לא עלה.

בתמונה רואים את הטרימינל של השולח והמקבל בזמן שרק המקבל פתוח וממתין.



```
receiver
roy@roy-VirtualBox: ~/Documents/socket-in-c
roy@roy-VirtualBox:~/Documents/socket-in-c$ make runr
./receiver
Socket created
Bind success
Waiting for incoming TCP-connections
A new client connection accepted

change cc algo to reno
Received 524290 bytes

change cc algo to cubic
Received 524290 byte

change cc algo to reno

sender
roy@roy-VirtualBox: ~/Documents/socket-in-c
roy@roy-VirtualBox:~/Documents/socket-in-c$ make runs
./sender
connected to server

change cc algorithm to reno

Send 524290 bytes of file.
first part of file send successfully
authentication for first part are successfully

change cc algo to cubic

Send 524290 bytes of file.
second part of file send successfully
authentication for second part are successfully

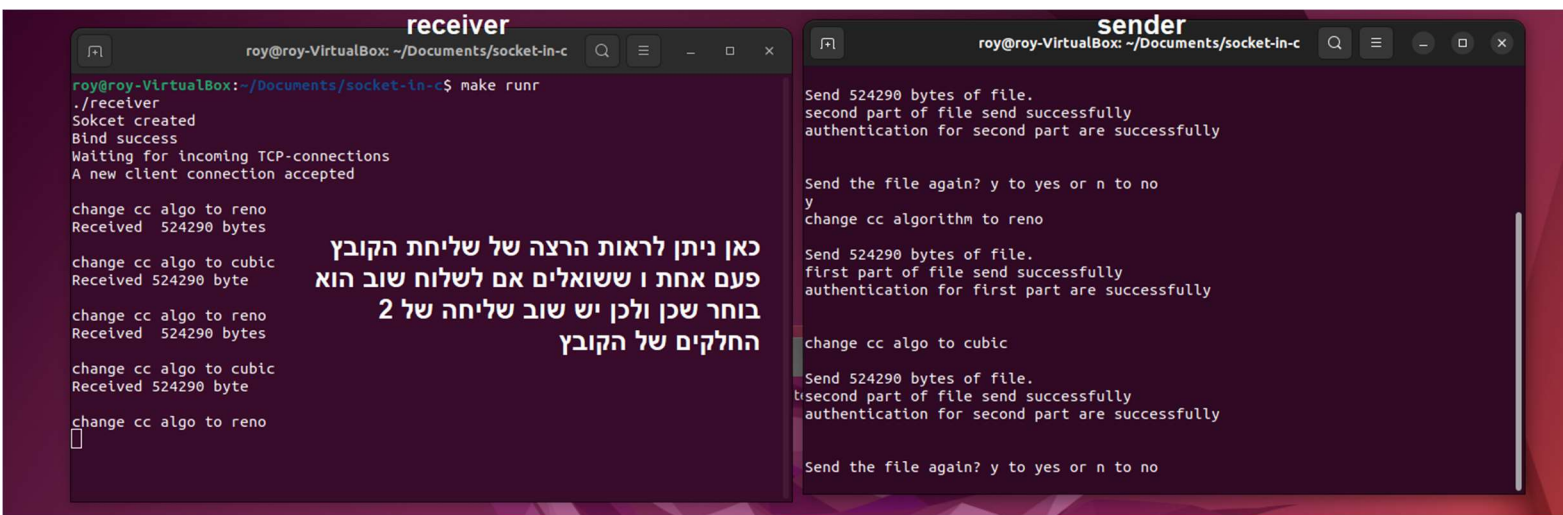
Send the file again? y to yes or n to no
[ ]
```

באן ניתן לראות שהרצנו את ה sender ומוצג לנו בכל צד ההודעות של כמות הביטים שכול צד שלח וקיבל את ההודעות על הצלחת שינוי CC אלגוריתם ובצד השולח הודעה על האותנטיקציה נכונה ולאחר מכן הצד השולח האם לשלוח את ההובץ שוב

באן ניתן לראות הרצה של שליחת הקובץ פעם אחת ואיך זה נראה בכול צד עם ההודעות המוצגות לכול צד.

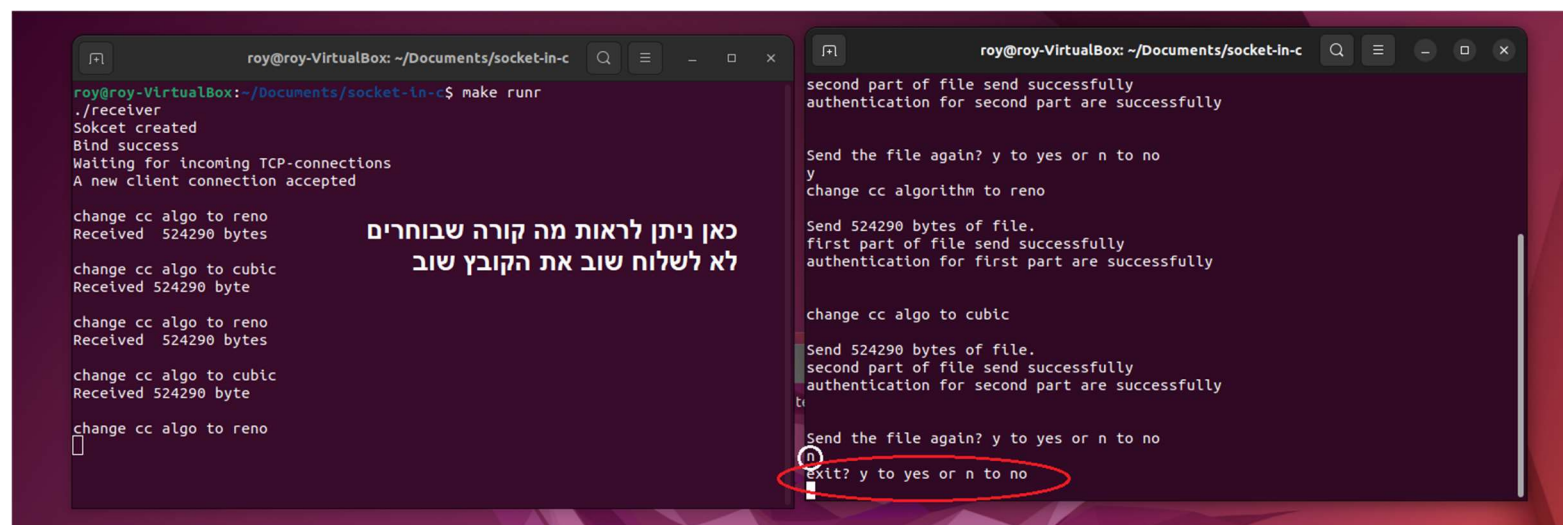
שינוי אלגוריתם של CC ב2 הצדדים אימות האותנטיקציה בצד של השולח שקיבל אותה מה receiver .

ולבסוף מוצג הודעה בצד של השולח ששואלת אותו האם לשלוח שוב את הקובץ וההודעה ממתינה לקלט מהשולח לכן למתקדמת/ יוצאת מהלולאה.



בתמונה זו ניתן לראות דוגמא להרצה של שליחת הקובץ פעם אחד ושהשולח נשאל אם לשלוח שוב הוא בוחר שכן והקובץ נשלח עוד פעם ואז יש סה"כ 4 פעמים שליחה של חלקי קבצים בין השולח למקבל.

בנוסף ניתן לראות ששולחים את הקובץ שוב אלגוריתם של ה CC משתנה חזרה ל reno לפני השליחה\ קבלה מחדש.



בתמונה רואים מה קורה שבחרים לא לשלוח את הקובץ שוב .

בתמונה ניתן לראות מה קורה שבחורים גם לצאת.
מה קורה בצד השולח ובצד המקבל ולאחר סגירת הקשר מודפס הזמנים והממוצעים ואת הזמן שלקח לכול כול חלק מהקבוצ להגיע

```

roy@roy-VirtualBox: ~/Documents/socket-in-c
change cc algo to cubic
Received 524290 byte

change cc algo to reno
Received 524290 bytes

change cc algo to cubic
Received 524290 byte

change cc algo to reno
exit message arrived
socket close

time to receive first part in 1 time is : 0.000150
time to receive second part in 1 time is : 0.000090
time to receive first part in 2 time is : 0.000105
time to receive second part in 2 time is : 0.000116
total average time for first part = 0.000128
total average time for second part = 0.000103
total average time = 0.000120
roy@roy-VirtualBox: ~/Documents/socket-in-c$

```

```

roy@roy-VirtualBox: ~/Documents/socket-in-c$
Send the file again? y to yes or n to no
y
change cc algorithm to reno

Send 524290 bytes of file.
first part of file send successfully
authentication for first part are successfully

change cc algo to cubic

Send 524290 bytes of file.
second part of file send successfully
authentication for second part are successfully

Send the file again? y to yes or n to no
n
exit? y to yes or n to no
y
close socket
roy@roy-VirtualBox: ~/Documents/socket-in-c$

```

בתמונה ניתן לראות מה קורה שיוצאים מהלולאה של שליחת הקבצים ב2 הצדדים מה מודפס בכל צד , סגירת הסוקט ובצד המקבל את הזמנים שלקח לחלקים להגיע.

הדפסת הזמנים

תחילה מודפסים הזמנים שלקח לכול חלק וחלק להגיע לפי הסדר שהם נשלחו.

לאחר מכן מודפס הזמן הממוצע לקבל החלק הראשון של הקובץ שהאלגוריתם CC היה מוגדר על שיטה של reno .

ואז מודפס הזמן הממוצע לקבל החלק השני של הקובץ שהאלגוריתם CC היה מוגדר על שיטה של cubic .

ולבסוף מודפס הזמן הממוצע שלקח לכול החלקים להגיע.

receiver

```

roy@roy-VirtualBox: ~/Documents/socket-in-c
./receiver
Socket created
Bind success
Waiting for incoming TCP-connections
A new client connection accepted

change cc algo to reno
Received 524290 bytes

change cc algo to cubic
Received 524290 byte

change cc algo to reno
exit message arrived
socket close

time to receive first part in 1 time is : 0.000395
time to receive second part in 1 time is : 0.000270
total average time for first part = 0.000395
total average time for second part = 0.000270
total average time = 0.000358
roy@roy-VirtualBox: ~/Documents/socket-in-c$

```

sender

```

roy@roy-VirtualBox: ~/Documents/socket-in-c
./sender
connected to server

change cc algorithm to reno

Send 524290 bytes of file.
first part of file send successfully
authentication for first part are successfully

change cc algo to cubic

Send 524290 bytes of file.
second part of file send successfully
authentication for second part are successfully

Send the file again? y to yes or n to no
n
exit? y to yes or n to no
y
close socket
roy@roy-VirtualBox: ~/Documents/socket-in-c$

```

הגדרת איבוד הפקטות

```

roy@roy-VirtualBox: ~
roy@roy-VirtualBox: $ sudo tc qdisc add dev lo root netem loss 20%
[sudo] password for roy:
roy@roy-VirtualBox: $

```

בתמונה ניתן לראות דוגמה להרצה של המערכת עם אחוז איבוד פקטות בגודל 20 אחוז

בתמונה ניתן לראות דוגמא להרצה של המערכת עם כלי של איבוד פאקטות בגודל של 20 אחוז.
ניתן לראות את ההבדלים לעומת התמונה הקודמת שבא לא היה איבוד פאקטות כלל שהזמנים הממוצעים עלו בכל חלק מהשניים של הקובץ.
וכמובן גם עלייה בזמן הממוצע הכולל של כל 2 הקבצים.

```

receiver
roy@roy-VirtualBox: ~/Documents/socket-in-c
./receiver
Socket created
Bind success
Waiting for incoming TCP-connections
A new client connection accepted

change cc algo to reno
Received 524290 bytes

change cc algo to cubic
Received 524290 byte

change cc algo to reno

exit message arrived

socket close

time to receive first part in 1 time is : 0.000408
time to receive second part in 1 time is : 0.000137
total average time for first part = 0.000408
total average time for second part = 0.000137
total average time = 0.000285
roy@roy-VirtualBox:~/Documents/socket-in-c

sender
roy@roy-VirtualBox: ~/Documents/socket-in-c
./sender
connected to server

change cc algorithm to reno

Send 524290 bytes of file.
first part of file send successfully
authentication for first part are successfully

change cc algo to cubic

Send 524290 bytes of file.
second part of file send successfully
authentication for second part are successfully

Send the file again? y to yes or n to no
n
exit? y to yes or n to no
y
te
close socket

roy@roy-VirtualBox: ~
roy@roy-VirtualBox:~$ sudo tc qdisc change dev lo root netem loss 10%
roy@roy-VirtualBox:~$

```

הגדרת איבוד הפאקטות

**בתמונה ניתן לראות דוגמא להרצה של המערכת עם
אחוז איבוד פאקטות בגובה 10 אחוז**

בתמונה ניתן לראות דוגמא להרצה של המערכת עם כלי של איבוד פאקטות בגודל של 10 אחוז.
ניתן לראות את ההבדלים לעומת התמונה הקודמת שבא היה איבוד פאקטות בגובה 20 אחוז
שהזמן הממוצע ש החלק הראשון עלה אך הזמן הממוצע של החלק השני ירדו לעומת הרצה של
20 אחוז.
וכמובן גם ירידה בזמן הממוצע הכולל של כל 2 הקבצים.

```
roy@roy-VirtualBox: ~/Documents/socket-ln-c
change cc algo to cubic
Received 524290 byte

change cc algo to reno

exit message arrived

socket close

time to receive first part in 1 time is : 0.000193
time to receive second part in 1 time is : 0.000098
time to receive first part in 2 time is : 0.000184
time to receive second part in 2 time is : 0.000268
time to receive first part in 3 time is : 0.000322
time to receive second part in 3 time is : 0.000273
time to receive first part in 4 time is : 0.000128
time to receive second part in 4 time is : 0.000149
time to receive first part in 5 time is : 0.000101
time to receive second part in 5 time is : 0.000105
total avarage time for firts part = 0.000186
total avarage time for second part = 0.000179
total avarage time = 0.000185
roy@roy-VirtualBox:~/Documents/socket-ln-c$

roy@roy-VirtualBox: ~/Documents/socket-ln-c
Send the file again? y to yes or n to no
y
change cc algorithm to reno

Send 524290 bytes of file.
first part of file send successfully
authentication for first part are successfully

change cc algo to cubic

Send 524290 bytes of file.
second part of file send successfully
authentication for second part are successfully

Send the file again? y to yes or n to no
n
exit? y to yes or n to no
y
close soket
roy@roy-VirtualBox:~/Documents/socket-ln-c$

roy@roy-VirtualBox:~$ sudo tc qdisc change dev lo root netem loss 20%
roy@roy-VirtualBox:~$
```

בתמונה ניתן לראות דוגמא להרצה של המערכת עם איבוד פאקטות
בגודל 20 אחוז ושליחת הקובץ 5 פעמים שוב ושוב

בתמונה אפשר לראות דוגמא להרצה של המערכת עם אחוז איבוד פאקטות בגובה 20 אחוז
ושליחה של הקובץ 5 פעמים שוב ושוב.