תיאור מערכת TCP AND Congestion Control Algorithms

סקירת מערכת..................................................................................................................2

פונקציונליות המערכת.......................................................................................................2

איך מריצים......................................................................................................................2

עץ החלטות לאפשריות של המערכת...................................................................................3

הסברים על הקוד (Sender)...........................................................................................4-10

הסברים על הקוד (Receiver)......................................................................................11-18

תמונות הרצה של המערכת ........................................................................................19-23

**סקירת מערכת**

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

בשלב ראשון מגדירים את שם הקובץ וגודלו ולאחר מכן נפתח בין ה Senderל Receiverסוקט, 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 ב2 חלקים הצד השולח יוצג הודעה האם לשלוח את קובץ שוב אם הוא יבחר שכן הקובץ ישלח שוב לצד המקבל ב2 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 ה 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 הSender ולהתחיל את שליחת הקובץ.

**עץ החלטות**

הצד השולח קורא את הקובץ לתוך סטרינג

האותנטיקציה לא נכונה

בחר לא לצאת מהתוכנית

בחר כן לצאת מהתוכנית

השולח לא רוצה לשולח שוב

נשאל אם לצאת מהתוכנית

השולח רוצה לשלוח את הקובץ שוב

השולח נשאל אם לשלוח את הקובץ שוב

האותנטיקציה אומתה על ידי השולח

השולח שולח את החלק השני של הקובץ.

המקבל שולח בחזרה הודעת אותנטיקציה.

האותנטיקציה אומתה על ידי השולח

**סוגרים את הסוקט בין שולח למקבל**

האותנטיקציה לא נכונה

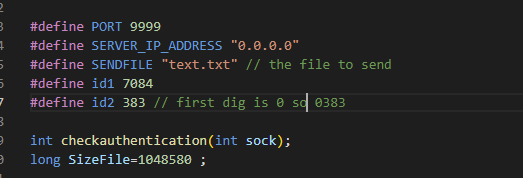
השולח מקבל את אותנטיקציה ובודק האם היא נכונה

השולח שולח את החלק הראשון של הקובץ.

המקבל שולח בחזרה הודעת אותנטיקציה.

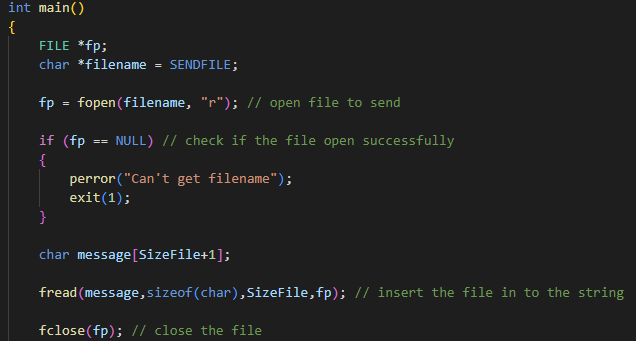
נפתח סוקט בין הצד השולח לצד המקבל

**Sender**



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

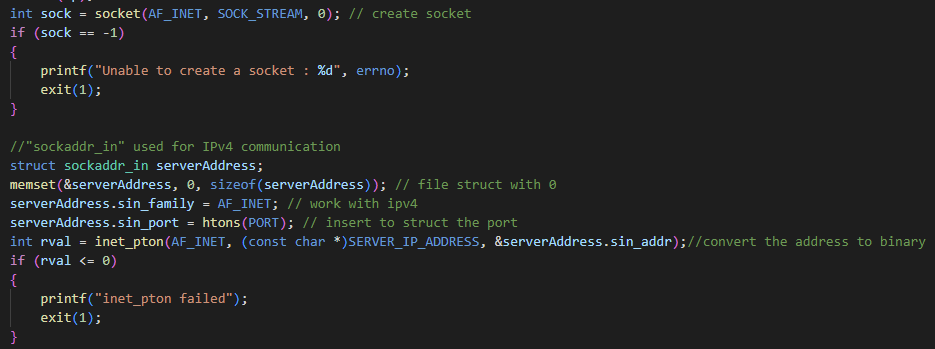
לבסוף הגדרנו את גודל הקובץ ששולחים.

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 לקראו אליו את הקובץ.

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

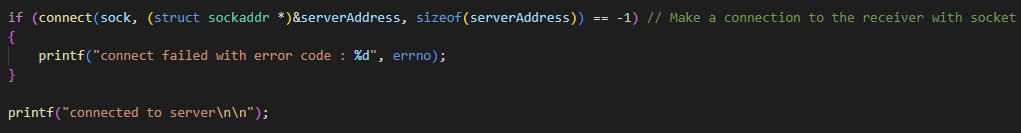
שלב שני יצרנו סוקט באמצעות הפקודה socket שמקבלת כפרמטים את סוג הIP איתו אנחנו עובדים וסוג הפרוטוקול (אצלנו TCP) , הפונקציה מחזירה ערך שלם לכן לאחר מכן בדקנו אם הפונקציה החזירה 1- אם כן יצירת הסוקט לא הצליחה הדפסנו הודעת שגיאה ויצאנו מהתוכנית.

שלב שלישי הגדרנו מבנה בשם sockaddr\_in שהוא מבנה שמכיל מידע של כתובת IP פורט עליו עובדים וסוג הIP שאיתנו אנחנו עובדים גירסא 4 או 6.

תחילה איפסנו את המבנה באמצעות memset באפסים.

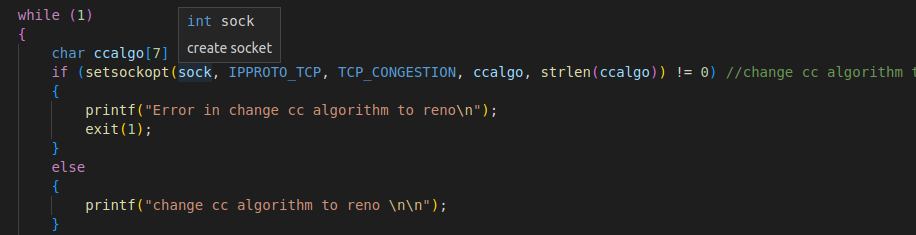
ואז הגרנו לו שאנחנו עובדים עם IP V4 והכנסנו לו את הפורט עליו אנחנו עובדים שאותו ההמרנו לביטים באמצעות htons.

ולבסוף המרנו את הכתובת לייצוג בינארי באמצעות פונקציית inet\_pton שמקבלת כפרמטרים סוג הIP 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.

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

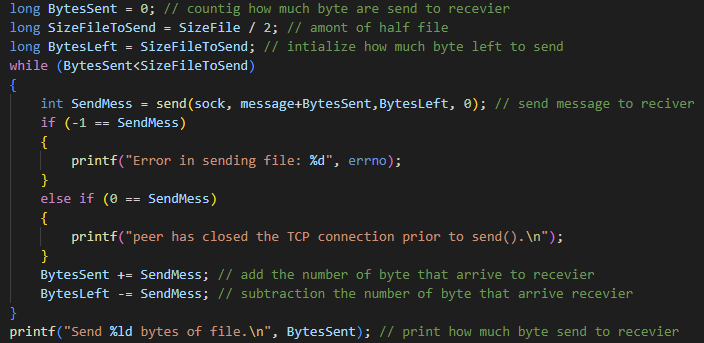
הפונקציה מחזירה 1- אם החיבור לא הוקדם לכן הוספנו בדיקה שהערך החזרה הוא לא 1- ואם הוא כן הפונקציה מדפיסה שגיאה.



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

כשלב ראשון בלולאה הגדרנו את CC אלגוריתם להיות reno באמצעות פונקציה setsocketop 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.



כשלב שני נגדיר מספר פרמטים שאיתם נעבוד בשליחת 2 החלקים של הקבוץ 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.

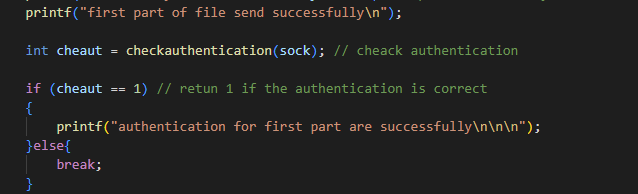
הפרמטר של המצביע להודעה שיתנה בהתאם להתקדמות של הלולאה באמצעות הוספתה כמות הביטים שנשלחו למצביע של תחילת ההודעה.

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.

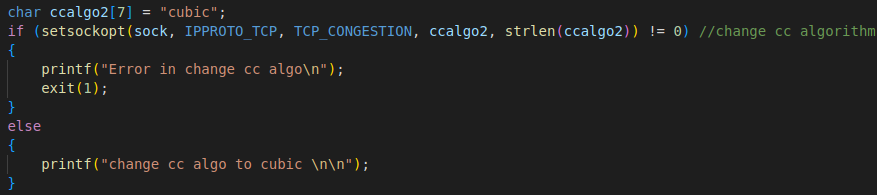
לסוף שנצא מהלולאה לאחר ששלחנו את כול הביטים נדפיס הודעה עם כמות הביטים ששלנו.



לאחר שליחת החלק הראשון נדפיס שהחלק הראשון נשלח בהצלחה.

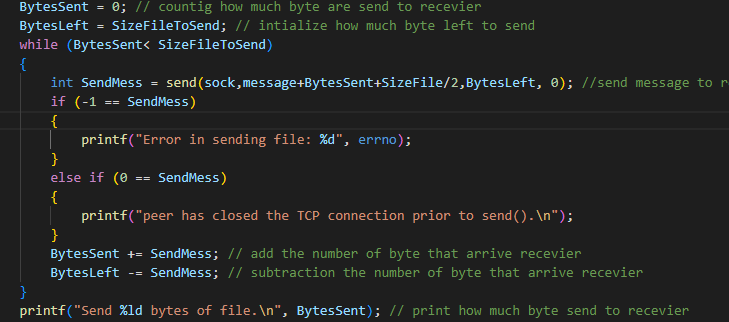
לאחר מכן נקרא לפונקציה checkauthentication שהיא פונקציה שמקבלת דרך הסוקט הודעת אותנטיקציה ובודקת האם היא נכונה ואם נכונה מחזירה 1 אחרת 0.

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



כעת נשנה את ה CC אלגוריתם להיות cublic באמצעות פונקציה setsocketop 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 .



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.

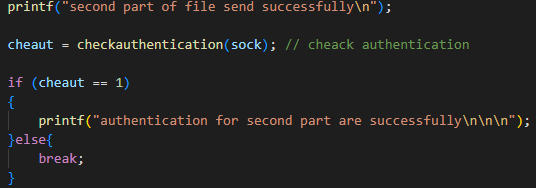
הפרמטר של המצביע להודעה שיתנה בהתאם להתקדמות של הלולאה באמצעות הוספתה כמות הביטים שנשלחו למצביע של תחילת ההודעה.

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.

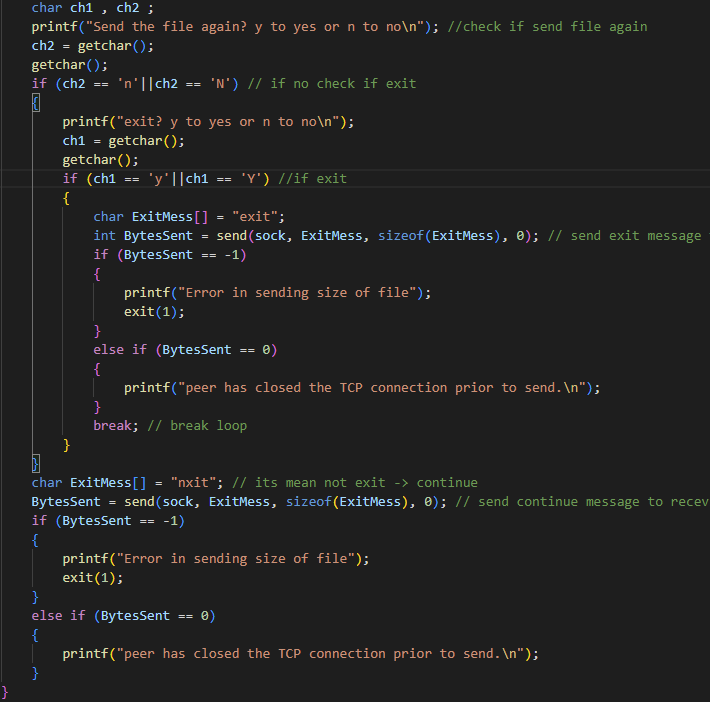
נוסיף למשתנה שהגדרנו בו את כמות ביטים את כמות הביטים שנשלחו בפונקציה Send ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לשלוח.

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



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.

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?

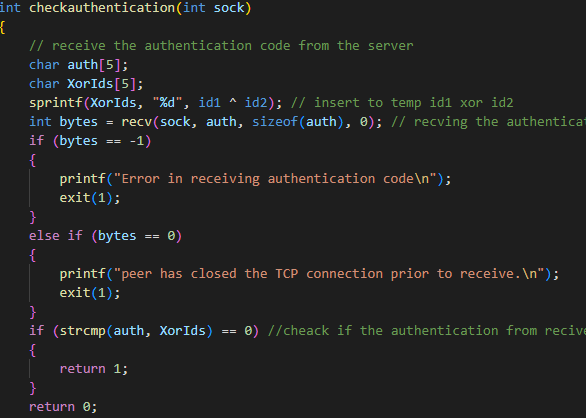
אם הוא בוחר לצאת הוא שולח הודעת EXIT לReceiver והשולח סוגר את החיבור עם לReceiver באמצעות הפקודה break וגם ה לReceiver סוגר את החיבור מהצד שלו כשמגיעה לו הודעת EXIT .

אם הוא בוחר שלא לצאת אז הSender שולח הודעת NXIT לReceiver שמייצג not exit כלומר להמשיך ובכך הReceiver יודע שהוא אמור לקבל שוב את 2 החלקים של הקובץ.

במידה והשולח בחר שלא לשלוח שוב אבל גם לא לצאת הקבצים נשלחים שוב לReceiver.

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



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

פונקציית checkauthentication

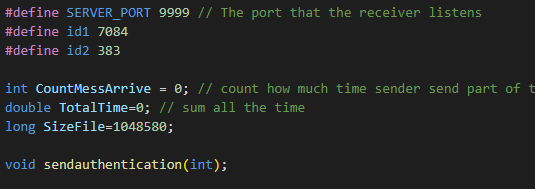
הפונקציה מקבל את הסוקט כפרמטר.

מגדירה 2 סטרינגים בגודל 5 , בסטרינג בשם XorIds מכניסים את פעול הXOR בין 2 תעודות הזהות באמצעות sprintf שממירה מספר מסוג int ל סטרינג.

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

לבסוף משווים בין פעול הXOR שעשינו להודעת האותנטיקציה שקיבלנו , אם הם שווים נחזיר 1

אם הם שונים נחזיר 0.

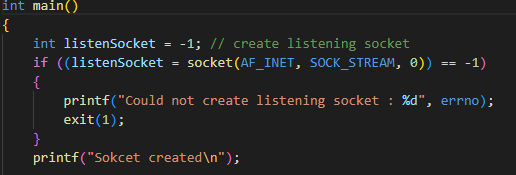
**Receiver**

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

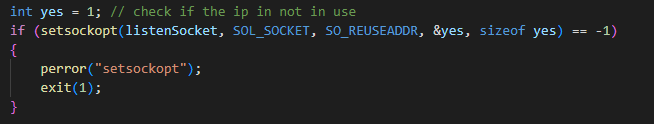
הגדרנו משתנה שסופר את כמות הפעמים שקיבלנו חלק מסויים מהקובץ.

משתנה שסוכם את זמני השליחה של כל חלק.

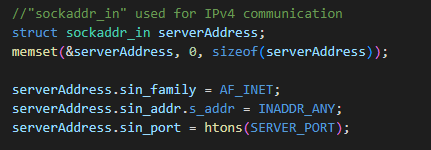
ולבסוף משתנה שמכיל את הגודל הקובץ שנשלח.

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

שלב ראשון ניצור את הסוקט באמצעות הפקודה socket שמקבלת כפרמטים את סוג הIP 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.

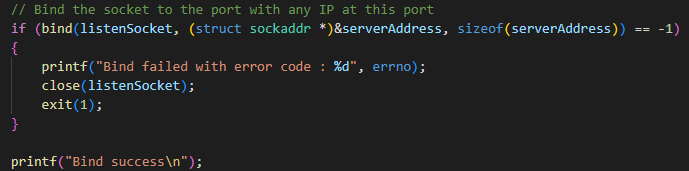
אם הצליחה ליצור סוקט נדפיס הודעה בסוקט נוצר.

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



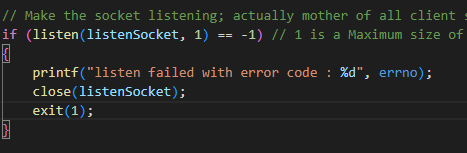
הגדרנו מבנה בשם sockaddr\_in שהוא מבנה שמכיל מידע של כתובת IP פורט עליו עובדים וסוג הIP שאיתנו אנחנו עובדים גירסא 4 או 6.

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.

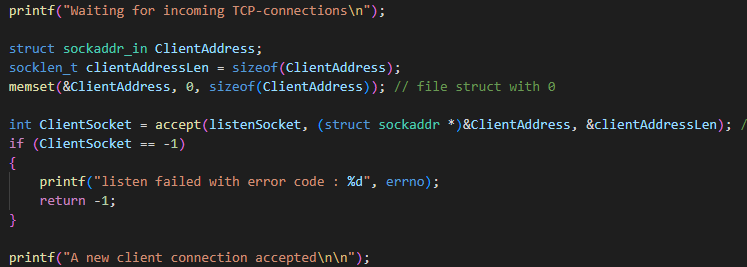
נקשר את הכתובת והפורט עם הסוקט באמצעות הפונקציה bind שמקבלת כפרמטים את הסוקט, את המבנה שהגדרנו קודם ואת הגודל שלו.

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.

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.



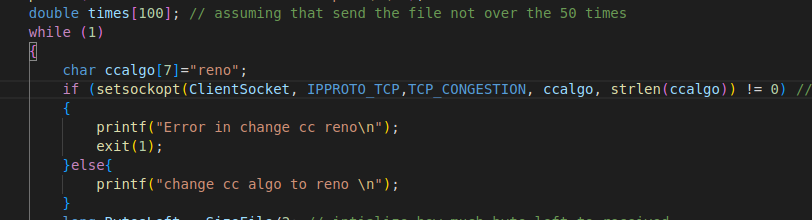
בחלק הבאה נדפיס הודעה שאנחנו מחכים לחיבור.

נגדיר עוד מבנה שמכיל את הIP והפורט של הSender , ונגדיר משתנה שמכיל את הגדול של המבנה הזה.

נאפס את המבנה באמצעות memset באפסים.

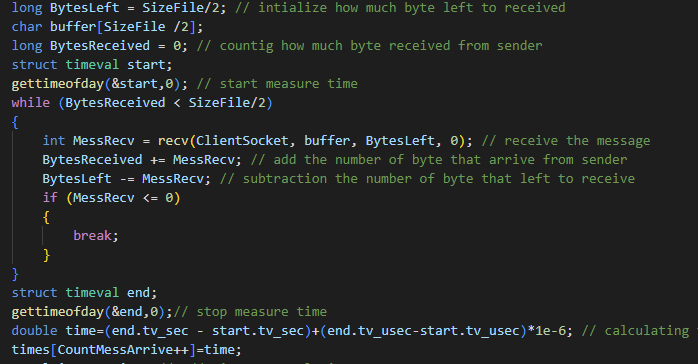
נוציא בקשה לחיבור מתוך תור הבקשות באמצעות הפונקציה accept שמקבל כפרמטר את הסוקט , מצביע למבנה שמכיל מידע על הכתובת והפורט של הSender ואת גודל המבנה בביטים.

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.

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 setsocketop 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.

נגדיר מספר פרמטים שאיתם נעבוד בקבלת 2 החלקים של הקבוץ 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 הנוכחי ברגע שהפעלנו את הפונקציה gettimeofday.

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 ,נחסיר מזמן הסיום את הזמן ההתחלה בדיוק של שניות ומיקרושניות 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.



לאחר שקיבלנו את החלק הראשון של הקובץ נדפיס הודעה עם כמות הביטים שקיבלנו.

ונשלח לשולח הודעת האותנטיקציה באמצעות הפונקציה sendauthentication שכתבנו.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

נשנה את ה CC אלגוריתם להיות cubic באמצעות פונקציה setsocketop 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.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

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

ונדגום את הזמן הנוכחי באמצעות gettimeofday.

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.

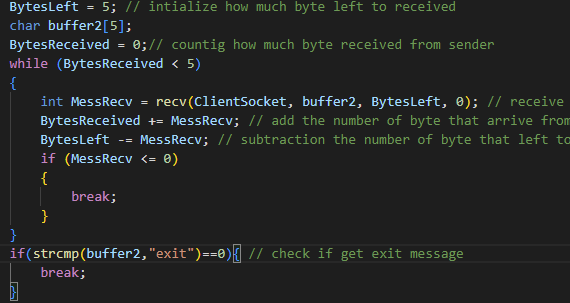
הפרמטר של כמות הביטים שנשאר לקבל יקטן בהתאם להתקדמות הלולאה ולביטים שהתקבלו.

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, נחסיר מזמן הסיום את הזמן ההתחלה בדיוק של שניות ומיקרושניות 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.

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

בחלק האחרון של הלולאה נבדוק איזה הודעה קיבלנו מהשולח exit או nxit.

נאפס את המשתנה כמה ביטים נשארו עם הערך5 כיון שיש להודעה 4 תווים ועוד 1 שמייצג את סוף ההודעה.

נגדיר buffer חדש בגודל 5 תווים, וכמות הביטים שהתקבלו ל0.

נרוץ בלולאה שרצה עד שנקבל 5 ביטים ובתוך 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.

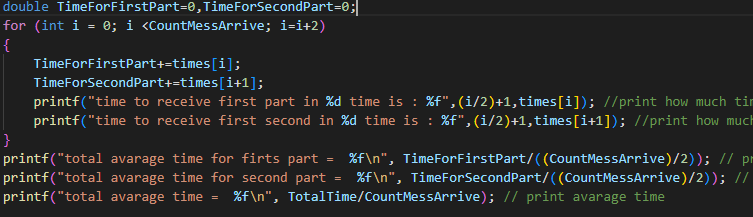
נוסיף למשתנה שהגדרנו בו את כמות ביטים שהתקבלו את כמות הביטים שהתקבלו בפונקציה recv ונחסיר את אותה כמות מהמשתנה שמכיל כמה ביטים נשארו לקבל.

לאחר קיבלנו את כול ביטים נצא מהלולאה ונבדוק אם ההודעה שקיבלנו היא exit אם כן נצא מהלולאה באמצעות הפקודה break.

אם לא קיבלנו הודעת exit משמע שקיבלנו הודעת nxit שמשמעות היא אנחנו ההולכים לקבל את הקובץ שוב ב 2חלקים לכן הלולאה תבצע עוד איטרציה.



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



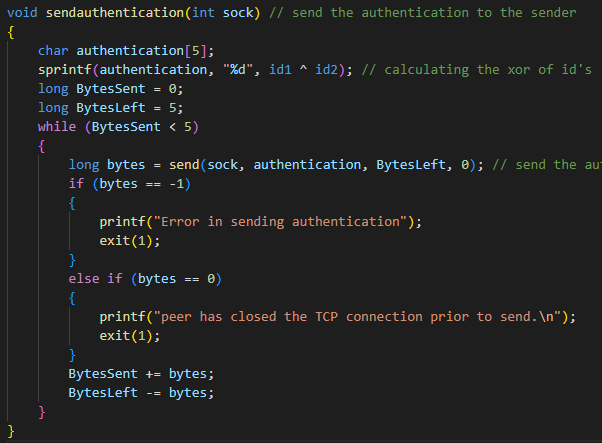
ולבסוף נגיע לחלק של הזמנים.

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.



פונקציית sendauthentication

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

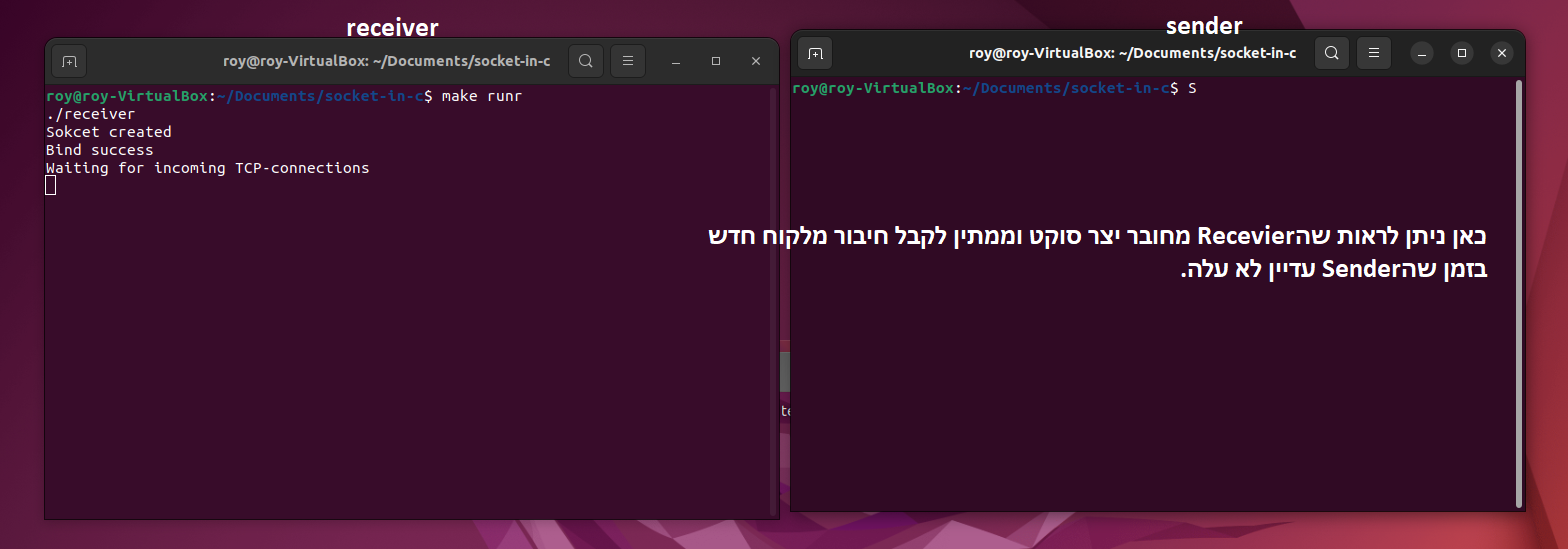
נגדיר 2 משתנים שמכילים את כמות הביטים שנשלחו וכמות הביטים שנשארו לשלוח.

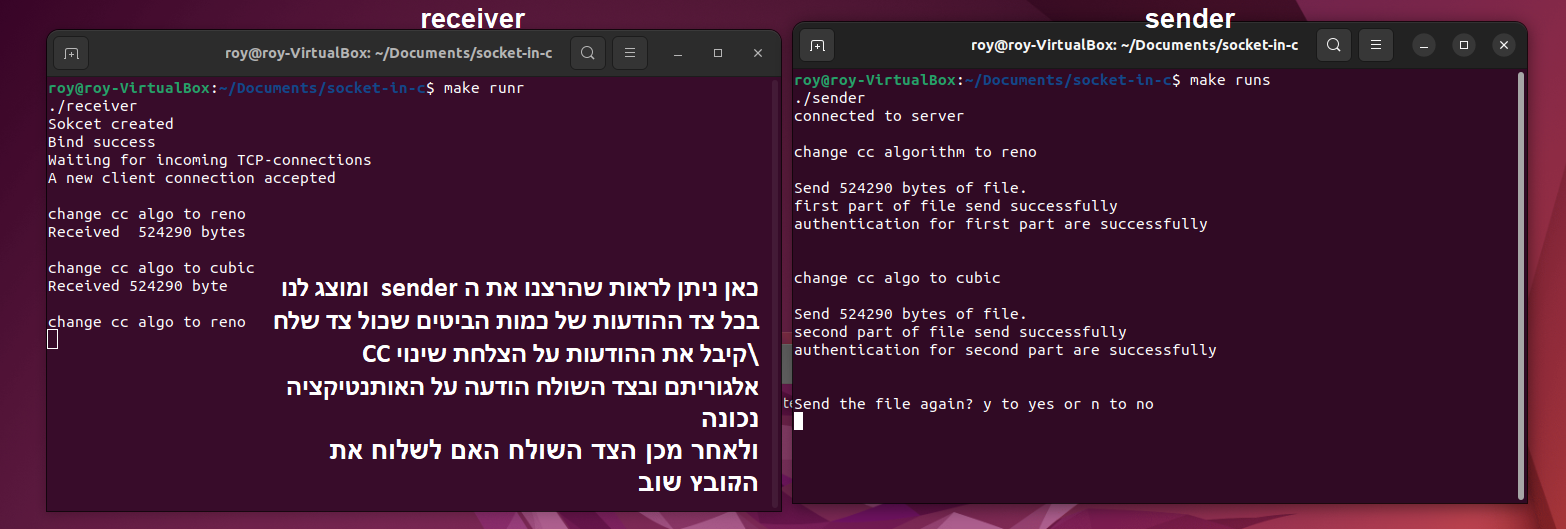
נרוץ בלולאת while עד שנשלח את כול 5 התווים , נשלח את האותנטיקציה באמצעות הפונקצייה send.

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.

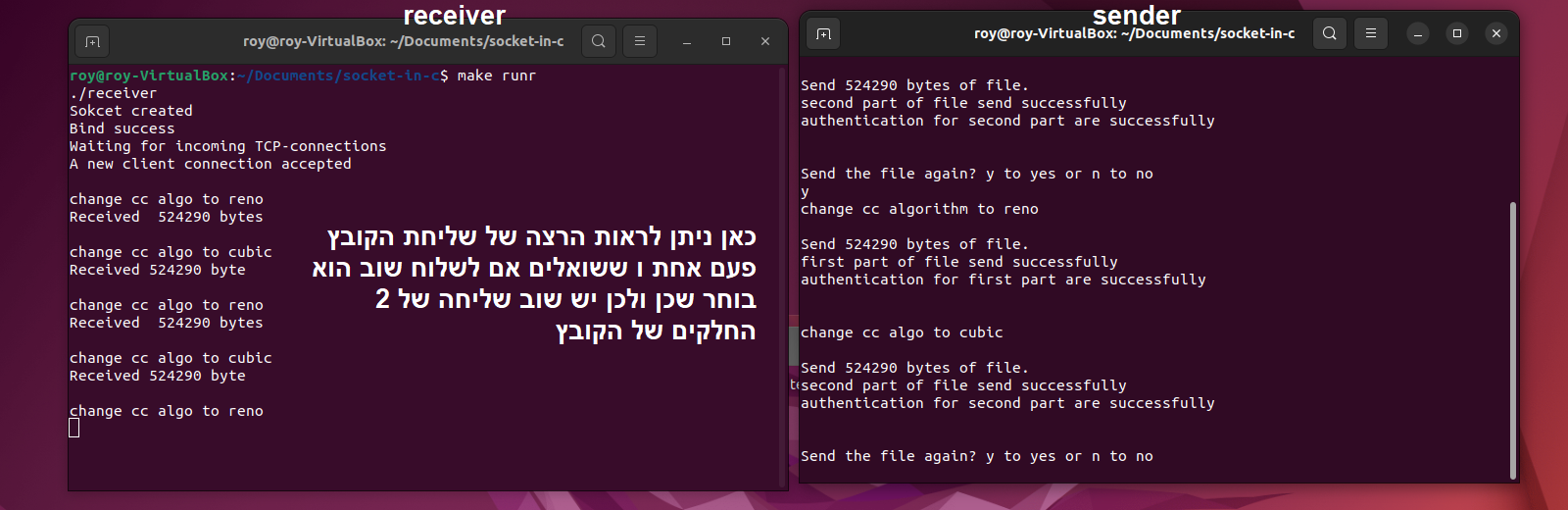
**תמונות הרצה של ה sender וreciever עם דוגמאות והצגת זמנים ממוצעים**

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

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

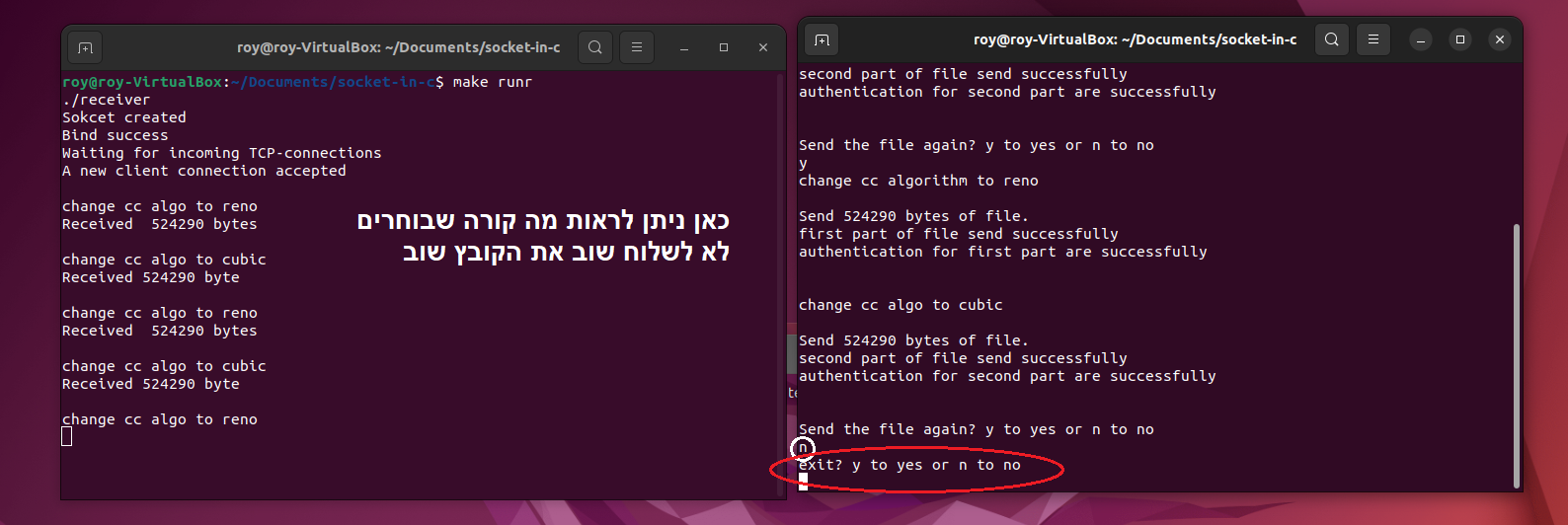
שינוי אלגוריתם של CC ב2 הצדדים אימות האותנטיקציה בצד של השולח שקיבל אותה מהreceiver .

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

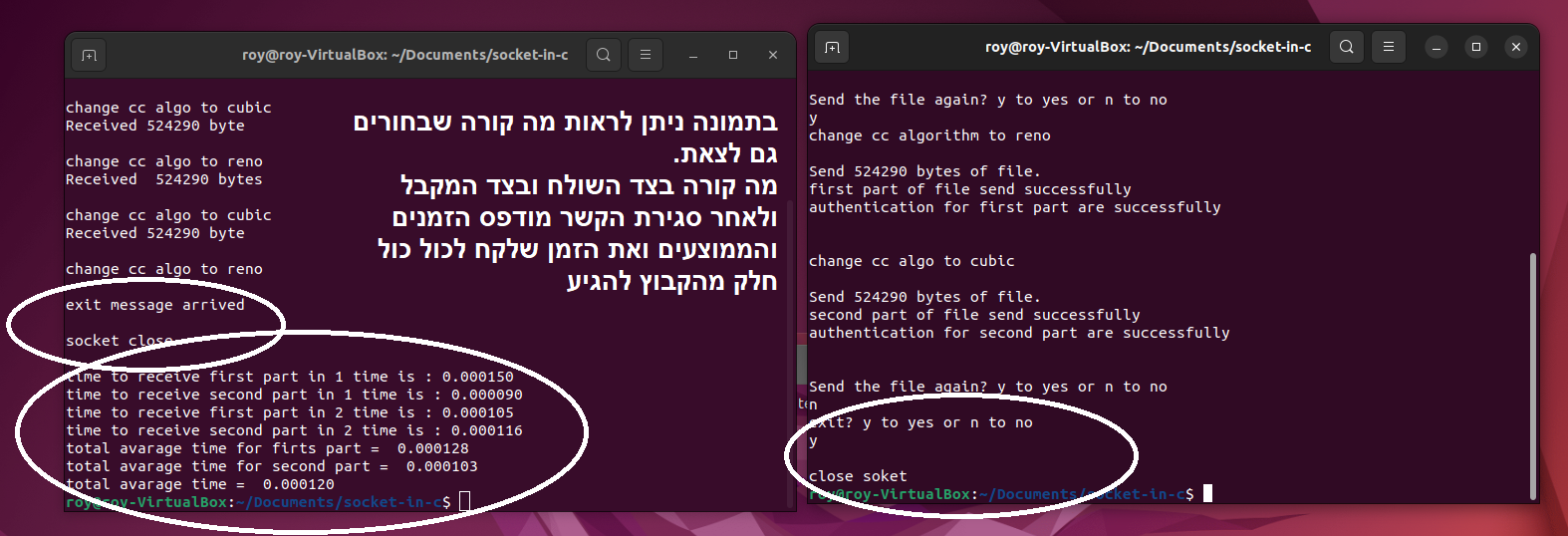


בתמונה זו ניתן לראות דוגמא להרצה של שליחת הקובץ פעם אחד ושהשולח נשאל אם לשלוח שוב הוא בוחר שכן והקובץ נשלח עוד פעם ואז יש סה"כ 4 פעמים שליחה של חלקי קבצים בין השולח למקבל.

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



בתמונה רואים מה קורה שבוחרים לא לשלוח את הקובץ שוב .



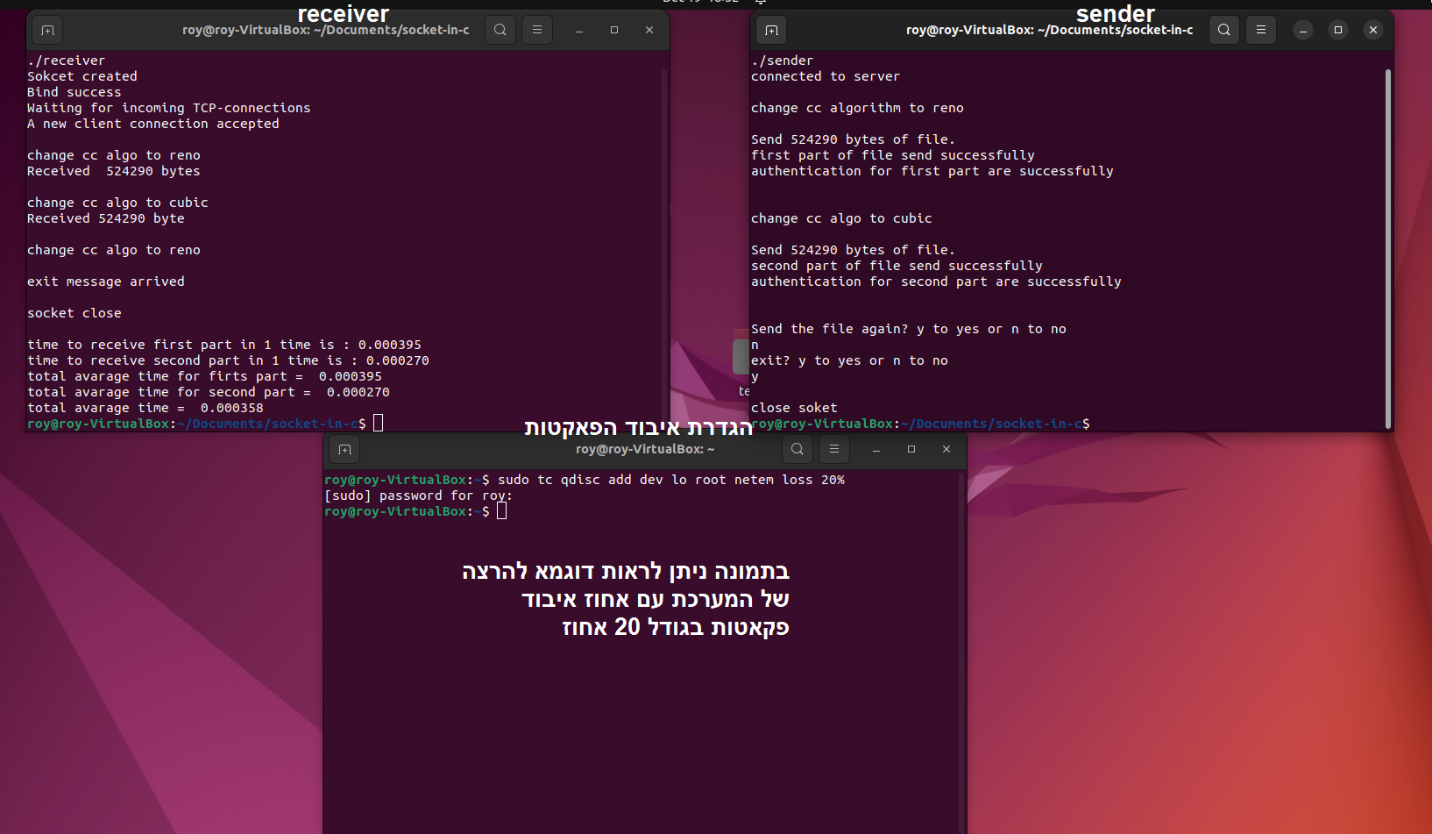
בתמונה ניתן לראות מה קורה שיוצאים מהלולאה של שליחת הקבצים ב2 הצדדים מה מודפס בכל צד , סגירת הסוקט ובצד המקבל את הזמנים שלקח לחלקים להגיע.

הדפסת הזמנים

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

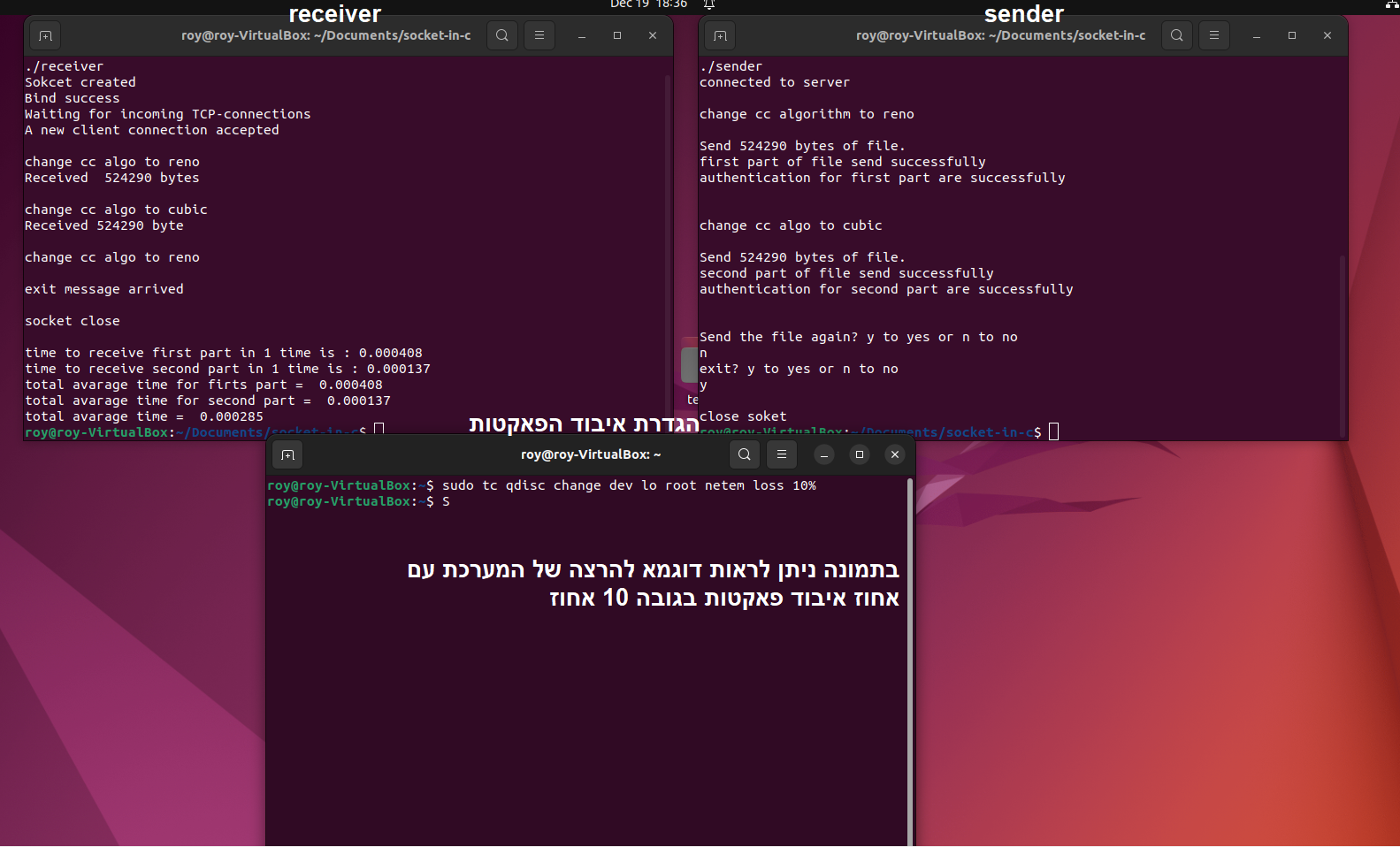
לאחר מכן מודפס הזמן הממוצע לקבל החלק הראשון של הקובץ שהאלגוריתם CC היה מוגדר על שיטה של reno .

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.

בתמונה ניתן לראות דוגמא להרצה של המערכת עם כלי של איבוד פאקטות בגודל של 20 אחוז.

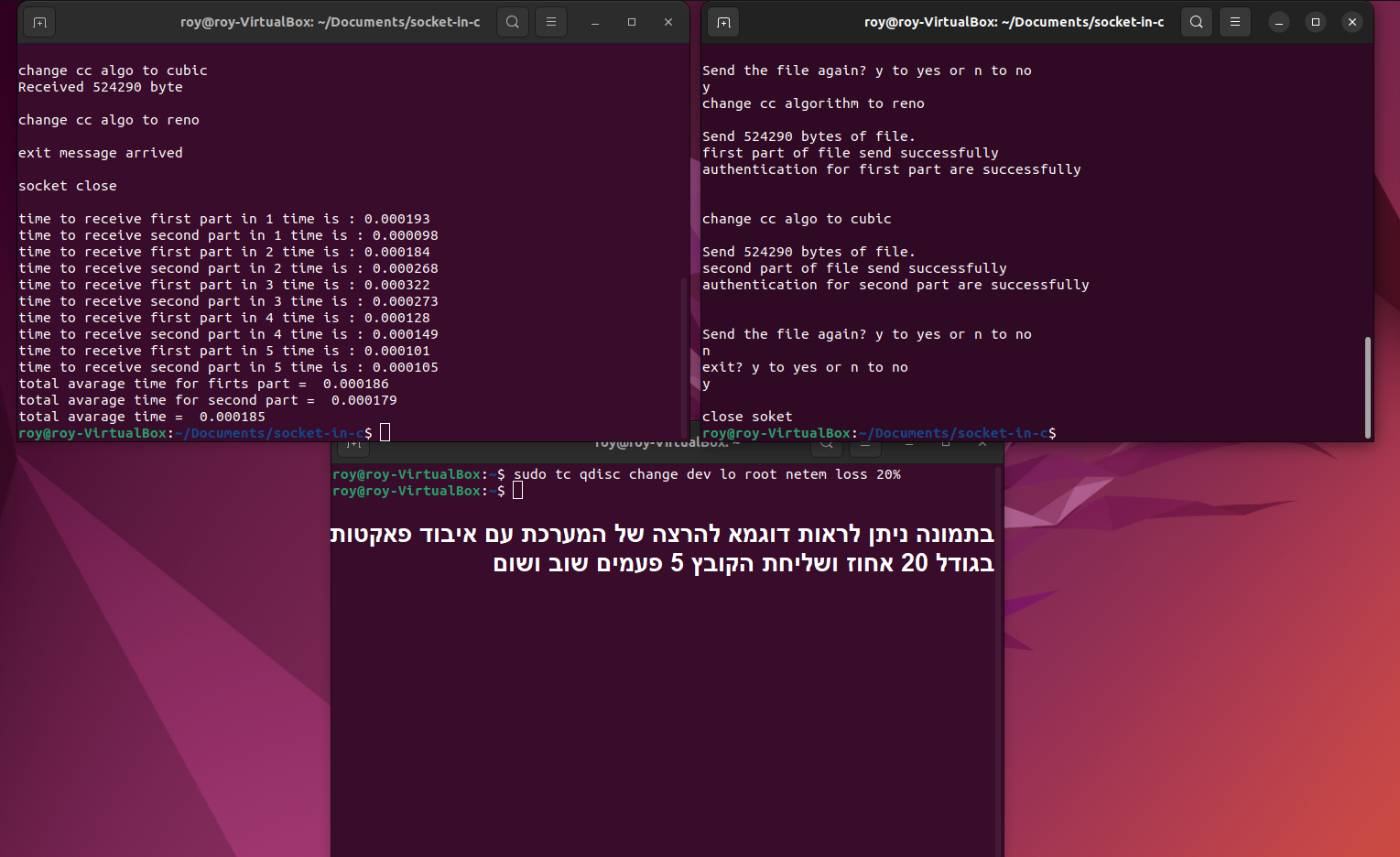
ניתן לראות את ההבדלים לעומת התמונה הקודמת שבא לא היה איבוד פאקטות כלל שהזמנים הממוצעים עלו בכל חלק מהשניים של הקובץ.

וכמובן גם עלייה בזמן הממוצע הכולל של כל 2 הקבצים.

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.



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