בס"ד

תקשורת ומחשוב: פרויקט סיום

סתיו זילבר - 322809559

שירה רוט - 206352411

חלק א:

הסבר הקוד:

בחלק זה בנינו את הצ'אט בין השרת ללקוח.

לכל אחד פתחנו socket UDP and socket TCP על מנת להתחיל את הקשר.

הלקוח client:

בלקוח רשמנו את הפונקציה send_listen אשר מקשיבה לבקשות של הלקוח.

כל לקוח יכול לקבל הודעות ולשלוח הודעות וזה יכול לקרות במקביל ולכן עשינו שימוש בטריידים.

כל פעם שמתקבלת בקשה הטרייד שולח לפונקציה listen_for_messages והיא מטפלת בהודעה.

השרת server:

עשינו רשימה של הלקוחות אשר מחוברים. הפונקציה serve מטפלת בכל לקוח שמתחבר והשימוש בטריידים (תהליכונים) על מנת שכמה לקוחות יוכלו להתחבר במקביל. כל לקוח נשלח לפונקציה listen for client ושם הלקוח שולח הודעה ומשם ממשיכים בהתאם למילה הראשונה בהודעה.

איך להריץ את הקוד שלנו?

המילה הראשונה בהודעה תהיה אחת מהפעולות שהלקוח יכול לעשות ולבקש מהשרת, לדוגמא connect מילה זו תמיד תופיע בסוגריים משולשות <> ולאחריה ההודעה \ שם המשתמש ללא רווח.

לדוגמא:

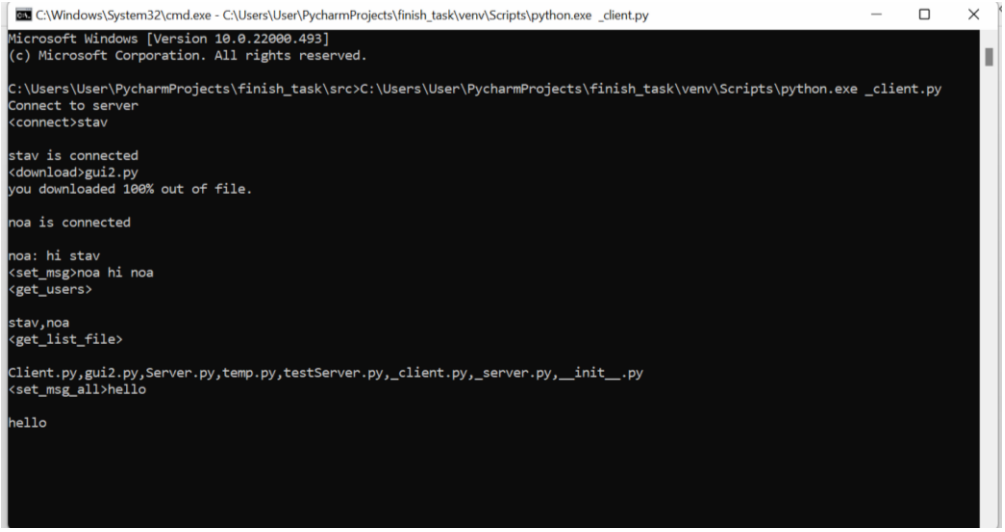
<connect>stav

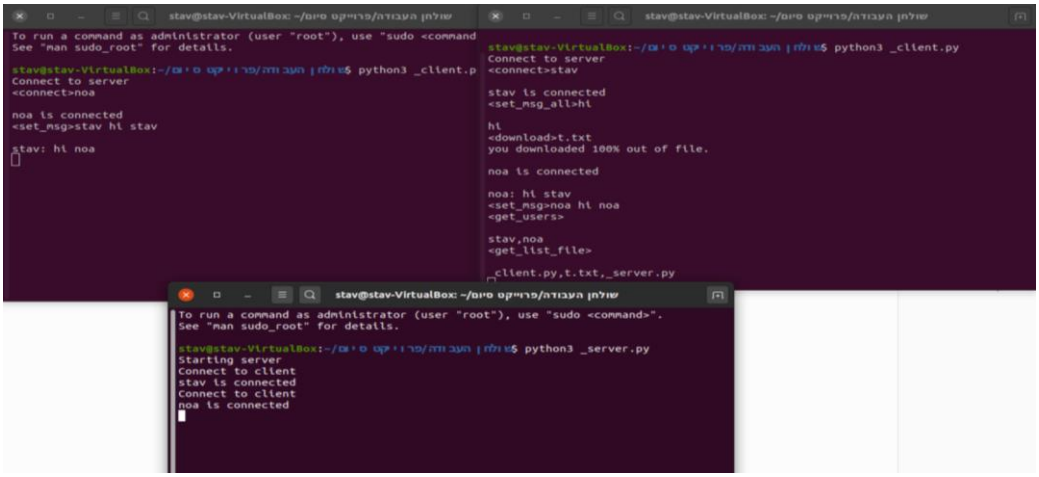
*גם הורדת הקובץ מתבצעת כך.

(ברשימת הקבצים ורשימת המשתמשים כותבים רק בתוך הסוגריים המשולשים. לדוגמא:

(< get_list_file>

לאחר מכן כל פעולה שולחת לפונקציה המתאימה ומטפלת בבקשה של הלקוח בהתאם.





חלק ב:

הסבר הקוד:

בחלק זה טיפלנו בבקשת הורדת קובץ מהשרת ע"י UDP אבל התבקשנו לעשות זאת בצורה אמינה ולכן השתמשנו בשיטת go back n – על שיטה זו הסברנו בשאלה (2) כאשר שאלו איך המערכת מתגברת על איבוד חבילות.

הגדרנו משתנים גלובלים ל: גודל החלון, מספר הבתים שנעביר בכל חבילה, וזמן לTIMEOUT.

הלקוח client:

הלקוח מבקש להוריד קובץ ורושם <download>file_name.(type) .

השרת מבין שהלקוח ביקש להוריד קובץ ולכן הולך לSOCKET UDP.

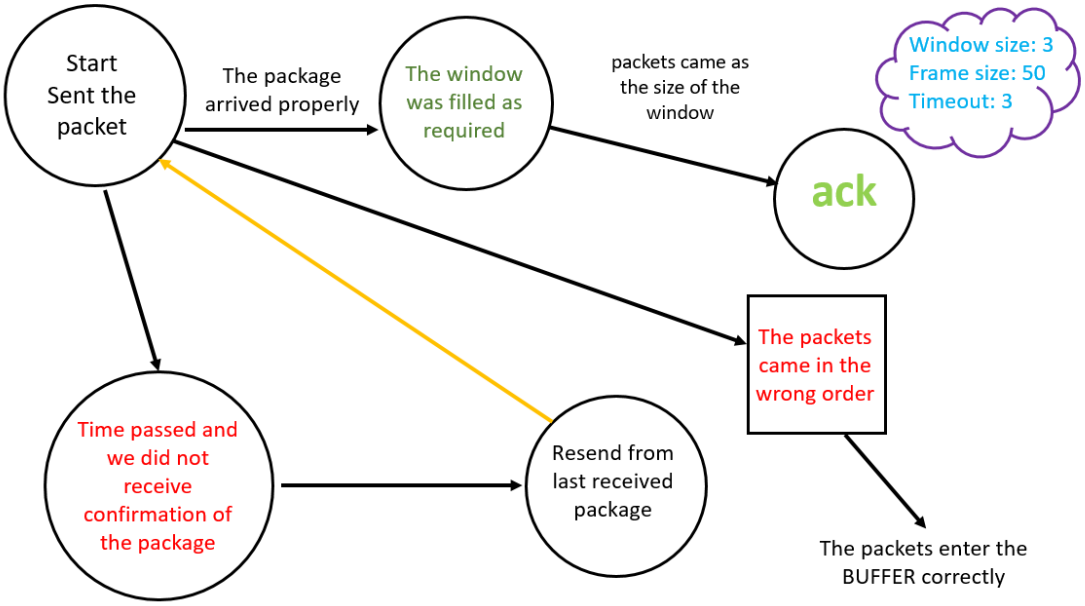
פתחנו את הקובץ לכתיבה והוספה. בדקנו שכל עוד לא הגענו לגודל הקובץ המשכנו להוסיף לBUFFER (רשימה בה שמרנו את כל המידע ולבסוף כתבנו את כולו לתוך הקובץ). הוספנו לכל FRAME מספר מעקב על מנת לדעת את סדר הגעת החבילות בצורה הנכונה ולכתוב אותם בצורה הנכונה לקובץ. הוספנו TIMEOUT שאם עבר הזמן ולא קיבלנו אישור נשלח מחדש.

לבסוף כתבנו את כל הBUFFER לתוך הקובץ ושלחנו ACK לשרת.

השרת server:

פתחנו את הקובץ לקריאה, בדקנו את גודל הקובץ וגם כאן בדומה ללקוח כל עוד לא סיימנו לקרוא את כל הקובץ, שלחנו את הקובץ בחלקים לפי גודל החלון וגודל הframe שהגדרנו. כאשר שלחנו את החבילות ללקוח גם כאן הוספנו בהתחלה את המספר הסידורי של כל FRAME. בדקנו גם כאן שלא עבר זמן ואם עבר שלחנו שוב. כך הלאה עד שסיימנו לשלוח את כל הקובץ.

1)דיאגרמת מצבים:



2) כיצד המערכת מתגברת על איבוד חבילות:

על מנת לעשות את הUDP אמין בחרנו בשיטת go back n .

בעזרת הפרוטוקול הזה העברת ההודעות היא אמינה.

בשליחת ההודעות יתכנו שיבושים ואובדן מידע ולכן נסמן לשולח ע"י ACK שהחבילה התקבלה, אם לא קיבלנו ACK זה אומר שהחבילה נאבדה או השתבשה בדרך. ולכן נשלח בדר"כ שוב את החבילה.

לכל חבילה יש מספר סידורי ולכל ACK יש מספר סידורי המציין על איזו חבילה (הודעה) הוא מדווח. ויש לנו זמן מקסימלי לקבלת החבילה ואם לא קיבלנו את ה ACK לאחר שעבר הזמן, נשלח אותה שוב.

בפרוטוקול זה ניתן לשלוח מספר חבילות בו זמנית ומבלי להמתין לACK עבור כל אחת בלבד, אבל השולח מוגבל בכמות החבילות שיכולות להישלח יחד - לפי גודל חלון. נסמנו בN . והחלון בעצם מתקדם בחבילות בכל פעם שמתקבל ACK.

<u>השולח מגיב לשלושה מצבים:</u>

1) <u>בקשה לשליחה:</u> אם החלון לא מלא, נשלחת חבילה, אם החלון מלא, השולח מודיע כי אין לו מקום לשלוח את החבילה כרגע.

2) <u>קבלת ACK:</u> כאשר מגיע ACK עבור חבילה X השולח מבין שכל החבילות על X הגיעו למקבל בצורה תקינה והוא מקדם את החלון בהתאם.

3)timeout: במקרה שנגמר הזמן לפי הtimeout ועדיין לא קיבלנו ACK, השולח שולח שוב את החבילות.

<u>תפקיד המקבל:</u>

כאשר מגיעה חבילה X והיא בסדר הנכון - המקבל שולח עליה ACK. בכל סיטואציה אחרת המקבל מתעלם מהחבילה ושולח ACK עבור החבילה האחרונה שהגיעה בסדר הנכון.

3) כיצד המערכת מתגברת על בעיות latency ?

באופן כללי ניתן להתגבר על בעיה זו באמצעות טכניקות שונות:

1) HTTP/2: עוזר להפחית את זמן ההשהיה ע"י מזעור של מספר השליחות או מספר התגובות מהשולח למקבל.

2) CDN: שימוש ב CDN משתמש בשמירת נתונים במטמון ואז כבקשת המשתמש צריכה לעבור רק דרך הנקודה הקרובה ביותר.

3) שמירה במטמון של הדפדפן: גם דרך זו עוזרת להפחית את זמן ההשהיה ומקטינה את מספר הבקשות חזרה לשרת.

אצלנו במערכת בשיטה שבחרנו go back n אנו מתגברים על זמן ההשהיה בעזרת timeout האומר שכאשר עבר זמן מסוים ולא קיבלנו אישור על ההודעה **נשלח שוב את הבקשה** כלומר ע"י שליחת הבקשה פעם נוספת ולא המתנה מרובה נפחית את זמן ההשהיה.

חלק ג:

1) **תהליך ההודעות מהחיבור ועד לקבלת ההודעה:**

כדי לחבר את המחשב לרשת נצטרך תחילה – כתובת IP למחשב, כתובת נתב וכתובת רשת הDNS – כלומר בבקשת DHCP . כידוע, בקשת הDHCP עטופה עם UDP שעטופה IP שעטוף ב ETHERNET . המחשב בעצם שולח BROADCAST לכל אותם הלקוחות ברשת המקומית כדי שהיא תגיע לDHCP ברשת.

כתובת המקור בהודעה זו תהיה "0.0.0.0" והיעד יהיה "255.255.255.255".

לבסוף, כאשר שרת הDHCP מקבל את ההודעה, הוא מגיב תגובת ACK העוטפת את כתובת הIP למחשב, כתובת IP לנתב הקרוב למחשב ואת כתובת הIP של שרת הDNS.

הפריימים של ההודעה נשלחים דרך הרשת המקומית. הלקוח מקבל תשובת ACK ואז יש לו את הנצרך בשביל להיות "שייך" לרשת, אך הוא עדיין לא יכול לשלוח תגובות למשתתפים אחרים ברשת כיוון שהוא לא יודע את כתובתם.

נפתור זאת כך: לפני שליחת הודעה למשתמש, המחשב ישלח הודעה לשרת הDNS עם פרוטוקול DNS כדי לקבל את כתובת הIP של הלקוח השני. המחשב יוצר שאילתת DNS, העטופה בUDP , שעטופה בIP, שעטופה בETHERNET .

אך התחנה הראשונה ברשת היא הנתב וכדי לשלוח אליו הודעה נצטרך את כתובת הMAC שלו. לכן נשתמש בשאילתת ARP. שאילתת הARP מתקבלת מהנתב המביא תשובת ARP ואת כתובת הMAC. המחשב עכשיו יודע את כתובת הMAC של הנתב הראשון אליו שולח את שאילתת הDNS.

IP DATAGRAM – מכיל שאילתת DNS המועברת ע"י LAN SWITCH מהלקוח לנתב.

השאילתה מועברת מרשת הלקוח לרשת השרת. שרת הDNS מקבל את השאילתה ושולח תגובה אל הלקוח – את כתובת הIP של הנמען. כעת למחשב יש את כתובת הIP הנדרשת. לשליחת ההודעה, נצטרך לפתוח SOCKET TCP אל השרת ולשלוח בקשת SYN לשרת, השרת מגיב בSYN ACK ונוצר חיבור TCP. כעת ההודעה נשלחת דרך הSOCKET TCP וה IP DATAGRAM המכיל את ההודעה המכיל את ההודעה מהנתב לשרת.
השרת עונה שקיבל את ההודעה ומעביר אותה לנמען.
לבסוף, כאשר הנמען מקבל את ההודעה, הוא שולח הודעת אישור לשרת שההודעה התקבלה בהצלחה.

2) **מה זה CRC:**

CRC הוא קוד לאיתור שגיאות המשמש לאיתור שגיאות בעת העברת נתונים.
לפני כשמתחילים להעביר את המידע, מחושב הCRC ומתווסף למידע שמעבירים.
לאחר שהמידע העובר, המקבל מאשר באמצעות הCRC שהמידע אכן התקבל ללא שינויים.
השימוש בCRC נפוץ בעיקר בשל המימוש שלו בחומרה בינארית, כלומר החישוב המתמטי שלו קל. וגם היעילות שלו בגילוי שגיאות נפוצות הנובעות כתוצאה מערוצי תקשורת רועשים.

3) **מה זה QUIC:**

QUIC הוא פרוטוקול בשכבת התעבורה.
בין הבעיות הרבות ב-TCP היא שהגישה לאספקת נתונים באופן אמין בין מכונות היא מתאימה לכולם. אבל לא לכל חיבור יש את אותם צרכים, ולחלקם יש מידע נוסף שניתן להשתמש בו כדי לשפר תחבורה אמינה. חלקית כדי לאפשר חדשנות והתאמה אישית בשכבת התחבורה (וחלקית כדי לטפל בבעיות מובנות היטב עם TCP).
גוגל יצרה את הQUIC: .Quick UDP Internet Connection.
QUIC מאפשר ליצור חיבורים מאובטחים תוך שימוש בפחות בקשות הלוך ושוב, מה שמפחית את זמן האחזור של החיבור - במיוחד בעת חיבור מחדש לשרתים שהיו בשימוש בעבר. QUIC נועד גם להביא את כל המשאבים המרובים הדרושים בדרך כלל לעיבוד דף אינטרנט באמצעות חיבור יחיד תוך הימנעות מריבוי הבקשות שקו ה-TCP יגרום.

מה זה HTTP:

פרוטוקול של רמת האפליקציה בו משתמשים ב-web. הוא מדמה מודל של שרת לקוח , כאשר הלקוח הוא הדפדפן אשר מבקש, מקבל ומציג אובייקטים, והשרת הוא שרת ה web-ששולח את האובייקטים ללקוח על פי בקשתו .פרוטוקול זה משתמש תמיד ב, TCP-כאשר מספר ה port-שאליו מתחבר הלקוח הוא 80. מאחר שנתונים אלו קבועים, קל לכתוב אפליקציות חדשות שיקבלו את האובייקטים משרתי הweb.

בתחילה השתמשו ב-HTTP 0.1 וכיום משתמשים ב-HTTP .1.1 לגרסה 1.1 יש תאימות אחורה ולכן תוכנות המסתמכות עליו יכולות להבין גם הודעות מאפליקציות שנכתבו בגרסה הקודמת. בגרסה הראשונה 1.0 לא שמרו על עקביות הקשר
(persistent- non connection) אחר כך זיהו שניתן להשתמש באותו קשר למספר בקשות בגרסה 1.1 הפכו מצב זה למצב רגיל (persistent connection).
ישנם שני סוגי הודעות ה: HTTP-בקשה– request ותשובה response. שני סוגי ההודעות כתובים בפורמט ASCII שהוא פורמט קריא וקל לבדיקה.
שלוש צורות עיקריות להעברת המידע:

1) GET
2) POST
3) HEAD

<u>ההבדלים בקיצור:</u>

ב http1.0 אנחנו פותחים וסוגרים קשר על כל בקשה וקבלה.
ב http1.1 אנחנו לא סוגרים אלא ממשיכים לבקש ולקבל, אנחנו נבקש כמה וכמה אובייקטים ביחד ונקבל אותם אחד אחרי השני.
אם אנחנו ב http 2.0 יכול להיות שהserver יעשה לנו push וידחוף לנו אובייקטים אחד אחרי השני. אם אובייקט מסוים גדול יכול להיות שנפצל אותו לכמה חלקים.
QUIC זה אם אנחנו ב http3.0 ואין לנו בכלל TCP אלא הכל מתנהל אצל הQUIC דרך UDP.

http2.0 משיג טעינה מהירה יותר של דפי אינטרנט ללא אופטימיזציות של ביצועים הדורשים מאמצים אנושיים נרחבים מבחינת פיתוח. זה מפחית באופן משמעותי את המורכבות ששיש ל-http1.1 ונותן לנו פרוטוקול חזק, שלמרות שלא נטול פגמים, אולי יעמוד במבחן הזמן.

<u>4) למה צריך מספרי port?</u>

שער שמזהה את האפליקציה באופן ייחודי על אותו המחשב, כלומר כאשר אנחנו רוצים להעביר נתונים לאפליקציה מסויימת כיצד נידע להגיע אליה?
לכן לכל דף אינטרנט יש מספר port. כל התקן המבקש לתקשר עם התקן אחר מחויב בהגדרת פורט המקור (מאיפה הוא יוצא) ופורט היעד (לאן הוא נכנס).
הפורט מזוהה לכל כתובת או פרוטוקול מסוים על ידי מספר בן 16 סיביות המאפשר 65,536 אפשרויות שונות.

5) **מה זה subnet ולמה צריך?**

Subnet היא הרשת הפנימית של רשת המחשבים המחוברים אל אותה הרשת הפנימית, למשל קבוצת המחשבים המחוברים לאותה רשת באוניברסיטה כך שלכל אחד יש את אותה תחילת הכתובת IP וסיומת שונה בין מחשב למחשב.

6) **למה צריך כתובות mac למה לא מספיק לעבוד עם כתובות ip?**

MAC היא כתובת הפיזית של המחשב אשר לא ניתנת לשינוי ונמצאת על כרטיס הרשת, וכתובת זו היא ייחודית.

כתובת הIP היא ברמה הלוגית וניתנת לשינוי, היא לא ייחודית, ניתנת ע"י מנהל הרשת או השרת.

הצורך בMAC הוא המעבר התקשורתי בין מחשבים סמוכים ברשת (כאלה שמשתמשים בהם במהלך הדרך בשביל להעביר מידע ממחשב מסוים למחשב אחר) עם כתובת ה IP- מנווטים בין 2 קצוות ברשת. כלומר כתובת ה IP היא בעצם של היעד שצריך להגיע אליו בסופו של התהליך, כדי להעזר במחשבים במהלך התהליך משתמשים בכתובת הMAC שלהם.

7) **ההבדל בין Router, Switch ,NAT:**

Router:

התקן ברמת הרשת, עובד בשיטת, Store and Forward משתמש בכתובת IP על מנת לקחת החלטות ניתוב. **מפריד בין תתי רשתות.**

Switch:

כאשר התקשורת נעשית **בתוך הרשת הביתית** אין צורך בשימוש בRouter .

נהיה זקוקים לRouter רק שנרצה לחבר את הרשת הביתית לאינטרנט או לרשת אחרת חיצונית.

הSwitch הוא בעצם מעביר תקשורת בין רכיבי התקשורת באותה רשת , תקשורת זו מבוצעת ע"י כתובת MAC address .

לכל Switch יש טבלה שבה יש שיוך בין פורט פיזי לבין כתובת MAC address , הSwitch לומד את כתובות ה MAC ושומר אותם בטבלה.

ההבדל בקיצור:

Switch עובד בתוך רשת נתונה, הוא מנתב כתובות MAC בין מחשבים שנמצאים באותה הרשת.

ברגע שאתה רוצה לצאת מהרשת שלך ולדבר עם רשתות אחרות – אז נכנס ה **Router** העובד עם כתובות IP ויכול לקבל בין רשתות שונות.

מה זה NAT:

פותר את מצוקת הכתובות.

Network Address Translation היא טכניקת ניתוב ברשת מחשבים בה נכתבות מחדש כתובות הIP של הpackets העוברות בנתב או בחומת אש, כלומר, כתובת הIP שתצא משימוש, תחזור אל מאגר הכתובות ותינתן למחשב אחר בשעת הצורך. טכניקה זו מאוד יעילה ונמצאת בשימוש נרחב.

נותנים לכל מחשב IP פנימי ולכולם באותה הרשת יש אותו IP חיצוני. כאשר רוצים להעביר פקטה ממחשב למחשב הוא יוצא מהIP הפנימי של מחשב המקור לIP החיצוני של מחשב היעד ואז לIP הפנימי שלו. וזה נקרא תהליך הNAT.

8) **שיטות להתגבר על המחסור בIPv4:**

המחסור בIPv4 הוא מצוקת הכתובות. המשמעות היא שלא נותרו עוד כתובות IPv4 פנויות להקצאה. בIPv4 כתובת IP היא בת 32 סיביות.

הפתרון:

על-מנת להתמודד עם מצוקת כתובות ה IP-הוגדר תקן חדש - IPv6 שבה כל כתובת IP מורכבת מ-8 קבוצות של 16 סיביות, כלומר 128 סיביות. תקן זה מאפשר מרחב עצום של 2 בחזקת 128 (מספר בעל 38 ספרות) כתובות שונות, ופותר את המחסור בכתובות.

המטרה של IPv6 היא הגדלה משמעותית בכמות הכתובות הIP האפשריות וגם להתאים לטכנולוגיות הרשת האחרונות.

פתרון נוסף להתגבר על המחסור הוא שימוש בNAT שאותו הסברנו בשאלה הקודמת.

9) a,b,c,d – נתונים על הרשת

OSFP: פרוטוקול ניתוב להעברת נתונים בין ראוטרים שונים הנמצאים באותה רשת.

BGP: הוא פרוטוקול שרץ על TCP . מקשר בין שני gateways אם אין בעיות מדיניות הם מדווחים אחד לשני. כל gateways יעביר לכל הרשת אוטונומית את כל ה prefix שאפשר להגיע ממנו והלאה.

נענה על הסעיפים:

e) בעזרת איזה פרוטוקול לומד הנתב 3c על תת רשת x ?

תחילה נצטרך לעבור מרשת AS3 לרשת AS4 זאת נעשה בעזרת **BGP** אשר הפרוטוקול השייך אליו הוא **TCP** לאחר מכן כדי להגיע לx אנחנו נצטרך להשתמש ב**RIP** אשר שייך לפרוטוקול **UDP**.

f) בעזרת איזה פרוטוקול לומד הנתב 3a על תת רשת x ?

תחילה נצטרך "לזוז" בתוך הרשת AS3 מ3a ל3c וזאת נעשה בעזרת **OSFP** לפי הנתונים.

ומכאן נוכל לחזור על התהליך מהסעיף הקודם.

כלומר ראינו שימוש בפרוטוקולים:**OSFP**, **BGP, RIP**

g) בעזרת איזה פרוטוקול לומד הנתב 1c על תת רשת x ?

תחילה נצטרך לעבור מרשת AS1 לרשת AS3 זאת נעשה בעזרת **BGP**, נגיע ל3a ומכאן נמשיך כמו הסעיף הקודם.

כלומר גם כאן נראה שימוש בפרוטוקולים:**OSFP**, **BGP, RIP**

h) בעזרת איזה פרוטוקול לומד הנתב 2c על תת רשת x ?

תחילה נצטרך "לזוז" בתוך הרשת AS2 מ2c ל2a בעזרת **OSFP**, לאחר מכן נגיע בעזרת **BGP** ל1b ברשת AS1 , נעבור ל1c בעזרת **RIP** מכאן נמשיך לפי הסעיף הקודם.

כלומר גם כאן נראה שימוש בפרוטוקולים:**OSFP**, **BGP, RIP** .

הסבר על הקלטת WIRESHARK:

בתמונה זו אנו רואים חלק מהורדת הקובץ ללקוח, ניתן לראות למטה את מספר הFRAME ונתונים נוספים.