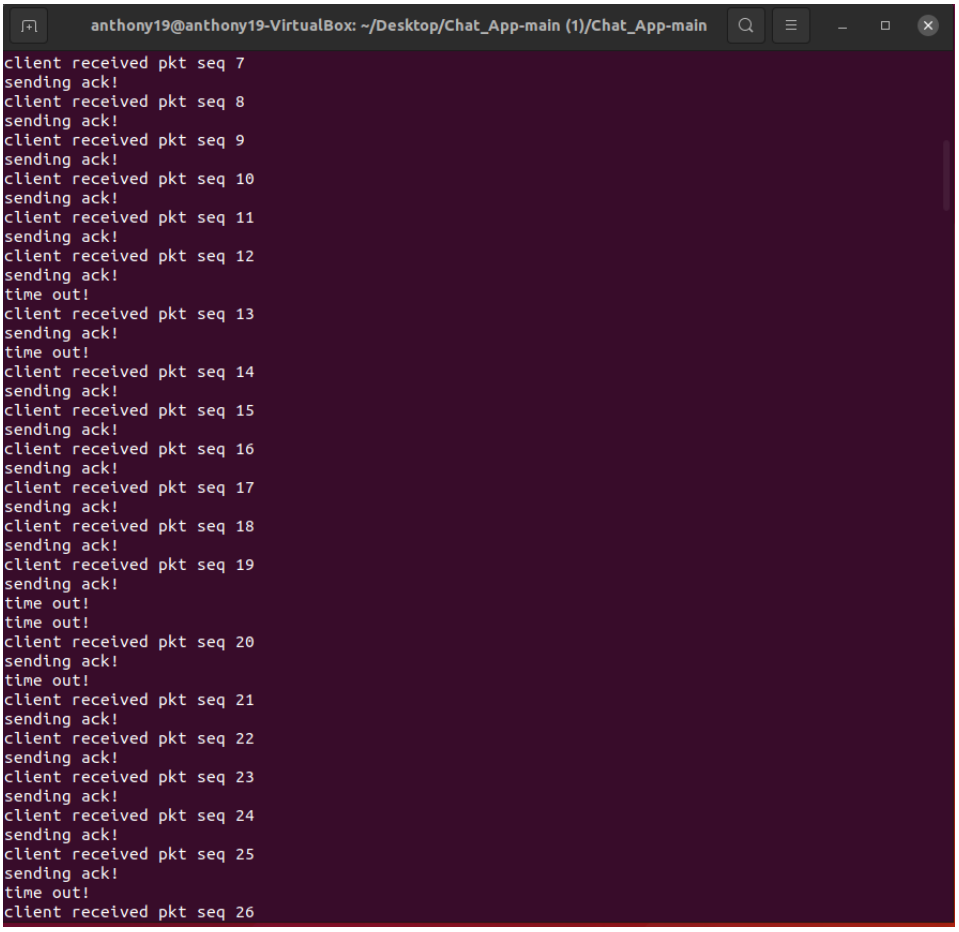
שמות: אנטוני אסייה / אנה פינצ'וק ת"ז: 206431082/209971258

פרויקט גמר – קורס תקשורת

קישור **GitHub** של הפרויקט: https://github.com/annapinchuk/Chat_App.git

.ביצד המערכת מתגברת יל אחוז איבוד פקטות:



צילום מסך של הטרמינל הלקוח

בזמן שליחה קובץ עם איבוד

10% פקטות.

הקלטת pcap של Wireshark

מצורפת בפרויקט GitHub

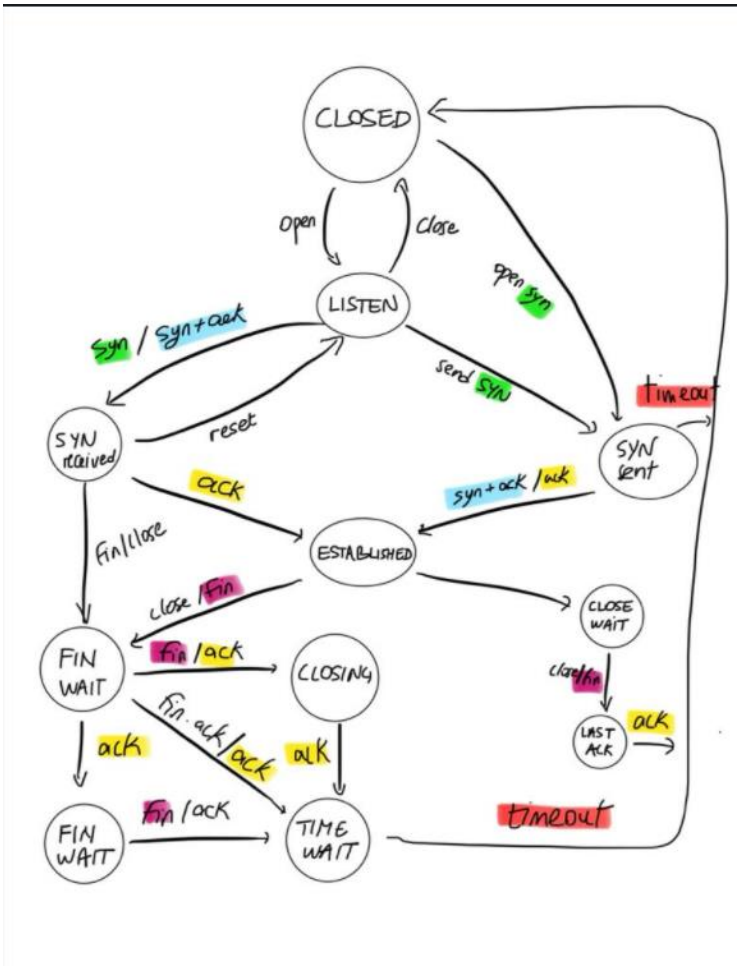
המערכת שלנו תומכך בזה שהיא ממשיכה לשלוח פקטות מהשרת ללקוח גם במקרה שהיא לא מקבלת **ACK** מהלקוח. בנוסף המערכת מאפשרת הורדת קבצים מסוגים שונים: **txt, png, gif, mp3, mp4.**

. ביצד המערכת מתגברת על בעיות **latency** :

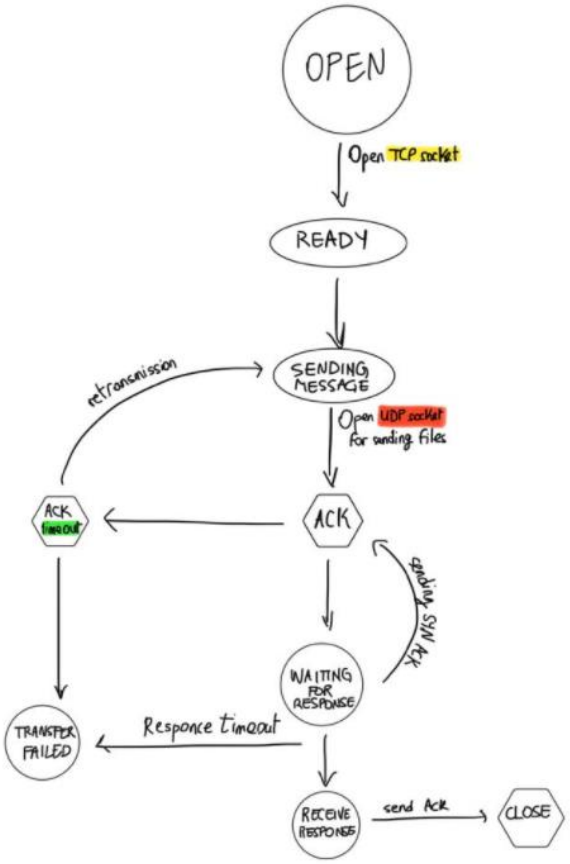
TCP הוא פרוטוקול מכוון זרם. אתה משתמש ב-**TCP** כאשר אתה רוצה להעביר הודעות מהשרת ללקוח ומלקוח לשרת, תוך מקסום התפוקה (כלומר, למזער את משך הזמן שלוקח לכל אחד לשלוח את כל הנתונים). השליחה מחדש האוטומטית של נתונים שאבדו וסידור מחדש הם אחת התכונות של **TCP**, אך **TCP** כולל גם חיבורים מצביים, בקרת זרימה ובקרת גודש. תכונות אלו הופכות את **TCP** לכלי ממש טוב להעברת כמויות גדולות של נתונים מנקודה אחת לאחרת. **UDP**, לעומת זאת, הוא פרוטוקול מכוון הודעה. השתמשנו ב-**UDP** לשלוח קבצים משרת ללקוח, תוך מזעור זמן *ההשהיה* (כלומר, משך הזמן שלוקח בין שליחת הודעה עד שהיא מתקבלת). על מנת למזער את זמן ההשהיה, **UDP** אינו מיישם שידורים חוזרים של נתונים או סידור מחדש. בפרויקט שלנו, מימשנו **Stop and Wait** עם **Timeout** שהוא משפיע על העומס ברשת. השרת מקבלת הודעות**timeout**, וכדי להוריד עומס מהשרת אנחנו מגדילים את ה**timeout** , כך שבעצם השרת מתאים את הקצב השליחה שלו.

. דיאגרמת מצבים:

Three Way Handshake



UDP data transfer



חלק ג':

1) בהינתן מחשב חדש המתחבר לרשת אנא תארו את כל ההודעות שעוברות החל מהחיבור הראשוני ל- Switch ועד שההודעה מתקבלת בצד השני של הצאט.

מהרגע הראשון שהחשב שלנו מתחבר לרשת, הוא יודע אך ורך את כתובת MAC שלו כי הוא מחזיק את כרטיס הרשת פיזית. בנוסף, על מנת לתקשר עם גורמים אחרים ברשת הוא צריך לדעת איפה הוא נמצא ברשת, כלומר את כתובת ה-IP שלו גם את ה- *Subnet Mask* שיוכל לראות מי מחובר איתו באותה תת-רשת.

אכן, על מנת להשיג את כתובת IP שלו, נשתמש בפרוטוקול **DHCP** *Dynamic Host (Configuration Protocol)*:

- ה-Client שלוח הודעת *Discover DHCP* ב-*Broadcast* לכל המחוברים ברשת.
- השרת DHCP מחזירה הודעת *Offer DHCP* ל-Client עם כל הפרטים הרלוונטיים: כתובת IP שלו, שרת DNS ...

- ה-Client שולח לשרת **DHCP** הודעת *Request DHCP* כדי להודיע לו שהוא מקבל את ההצעה.
- ולבסוף השרת DHCP שולחת ל-Client הודעת *ACK DHCP* כדי להודיע לו שהוא יכול להשתמש בכתובת IP שהוא קיבל.

לאחר מכן, המחשב שלנו צריך לשאול לשרת ה-DNS את הכתובת IP של היעד שלו כדי שהוא יוכל לשלוח לו הודעות. נשתמש בפרוטוקול **DNS** *(Domain Name Server)* שהוא מתמקד בשכבת האפליקציה.
כתובות פיזיות שייכות לשכבת הקו *(Link layer)* ומציינות את התחנה הקרובה בכל פעם nexthop במסלול אל היעד. תהליך החיפוש של כתובות ה-MAC יתבצע ע"י פרוטוקול **ARP** *(Address Resolution Protocol)*:

- הודעת *ARP Request* נשלחה ב-*Broadcast* מהמחשב שלנו לכל המחוברים ברשת, כדי לבדוק למי הכתובת ה-MAC הזאת שייכת עבור כתובת IP מסוימת.
- בתגובה, הודעת *ARP Reply* נשלחת מהנתב אל המחשב שלנו שרשומה בה את כתובת ה-MAC שלו.

הכתובת ה-MAC של המקור היא של המחשב שלנו, כיוון שאנחנו בעצם שלוחים את ההודעה , היא ידוע לנו כי היא שייכת לכרטיס הרשת שלנו. הכתובת ה-MAC של היעד היא של הנתב *ה-nexthop* *במסלול* אל היעד.

כתובות הלוגיות הושגו בשכבת הרשת *(Network Layer)* ע"י פרוטוקול **DHCP**. כתובת ה IP-של המקור היא של המחשב שלנו, כיוון שאנחנו שולחים את ההודעה וכתובת ה IP-של היעד תהיה הכתובת של שרת DNS .

לאחר שהלוקח מצא את כתובת IP של היעד, הוא יכול להתחבר אליו. על מנת לעשות זאת, נרים קשר **TCP** *(Transmission Control Protocol)*, פרוטוקול הנמצא בשכבת התעבורה, עם השרת של היעד. נשתמש ב- "שליחת לחיצה יד משולשת" *(Three Way Handshake):*

- **SYN**: המקור שולחת את החבילה הראשונה שהיא בעצם הודעה לפתיחת קשר.
- **SYN-ACK**: היעד מקבלת את ההודעה ושולחת בתגובה הודעת אישור קבלה ואישור פתיחת קשר מצדה.
- **ACK**: המקור מיידעת את תחנת היעד על סיום שליחת חבילות הקשר בהודעת ACK.

לבסוף, לאחר שהצלחנו ליצור קשר נוכל להתחיל לשלוח הודעות באמצעות פרוטוקול TCP מהלקוח אל השרת והפוך וכך לתקשר עם היעד.

*2) <u>הסבירו מה זה **CRC**</u> ?*

CRC היא בדיקה לאיתור שגיאות בהעברת נתונים, כלומר דרך לברר האם כל המידע שנשלחה הוא גם המידע שהגיע. לפני שהמידע מועבר, מחושב ה-**CRC** ותתווסף למידע שמעבירים . שיטה הדומה ל-**checksum** שמתבססת על איזומורפיזם שבין וקטורים לפולינום מודולו 2, ומחסירים את השארית תוך שימוש ב-**XOR** במקום בחיסור רגיל. מצרפים את התוצאה שקיבלנו מימין להודעה המקורית ונשלח. הווידאו הוא מול הביטים אחרונים שנשלחו האם הם זהים לתוצאה שהתקבלה.

3) <u>מה ההבדל בין QUICK, http 2.0, http 1.1, http 1.0</u> ?

קיימים כמה הבדלים בין הפרוטוקולים הללו מכל מיני בחינות, **ההבדל העיקרי** בניהם בין ה *QUICK* ל-*http 2.0, http 1.1* ו- *http 1.0* הוא שרק http3 מבוסס על פרוטוקול QUICK בשכבת התעבורה בעוד ש- *http 2.0, http 1.1, http 1.0* משתמשים בפרוטוקול TCP.

- בנוסף לזה, קיימים הבדלים גם **באופטימיזציה חורב פס**, ב-*http 1.0* יש תופעות של בזוז רוחב פס כשאר הלקוח רוצה לשלוח רק חלק מאובייקט מסוים אבל השרת שולח הכל ולא תומך בנקודת הפסקה, בניגוד ל-*http 1.1* שכן תומך בנקודת הפסקה כבחירת מחדל וכך פחות בזבזני. לגבי *http 2.0* שהוא פרוטוקול בינארי ולן הוא הרבה יותר דחוס ופחות bits-on-wire מה- http הקודמים.
- בנוגע **לחיבור ארוך**, ב-*http 1.0* יש צורך בפרמטר keep-Alive כדי ליצור חיבור ארוך עם השרת וגם חייב ליצור חיבור TCP חדש לכל בקשה. *http 1.1* גם תומך בחיבור ארוך ובקשות ב-*pipelining* וכך ניתן להעביר הרבה בקשות ותגובות http בחיבור TCP אחד בלמד, מה שמקטין את צריכה והעיכוב של הקמה וסגירה של חיבור.
- על ידי שימוש **בריבוב** (*multiplexing*) *http 1.1* יכול להעביר בקשות אחת אחרי השנייה כך שאם אחת מהבקשות לא מגיעה ליעד שלה מסיבה כלשהי, זה חוסם את כל הבקשות האחרות אחריה. ב *http 2.0* הטכניקה הזו נותנת לו יכולת להעביר במקביל מספר רב של בקשות מבלי להסתמך על הרבה חיבורים TCP, אכן אותה שרת צריכה רק חיבור TCP אחד ומבטל את העיכוב וצריכת זיכרון הנגרמים ממספר חיבורי TCP. מצד שני ה-*QUICK* הוט מהיר יותר בנוסף איבוד פקטה לא מפיל את כל ההודעה כולה ב-*pipelining*.
- אם נסתכל גם **במנגנון אימות**, ניתן לראות ש-http 1.0 משתמש בסכמת אימות בסיסית שאינה בטוחה מכוון ששם משתמש וסיסמאות מועברים בטקסט ברור או בקידוד בבסיס 64, הגרסה http 1.1 יותר מאובטחת יחסית בגלל שהוא משתמש באימות תקציר ואומות NTLM. המנגנוני אבטחה שהותקנו על http 2.0 זהים לאלו של הגרסאות הישנות יותר אך בגרסה הזאת נוכל יותר להתמודד איתם בשל תכונות TLS חדשות. ב-QUICK לא נשתמש בTLS יותר כך שהלקוחות יכולים לקבל את המידע מוצפנת יותר מהר ויותר מובטחת.

4) למה צריך מספרי ports ?

כאשר אנחנו מתחברים למכשירים אנחנו בעצם מתחברים לכתובת ה-IP שלו. עם זאת, ייתכן שהמכשיר הזה מפעיל מספרי שירותים, כמו שרת אינטרנט, שרת אימות, שרת קבצים וכו'. הצורך בפורטים מוכרים קיים כדי לקבוע סטנדרטים בהתחברות לשרתים המספקים שירותים מסוימים. במילים אחרות, הפורט הוא דרך לזהות את השירות שאליו אתה מתכוון להתחבר. אכן, יש צורך במספרי פורטים כדי שתעבורה שמגיע מתהליכים שונים (דואר אלקטרוני, אינטרנט, מסד נתונים ...) במקורות שונים תוכל להגיע בו-זמנית לאותו מארח. לדוגמה, תעבורת *http* בפורט 80 או תעבורת *telnet* בפורט 23, וכאשר יש יותר ממקור אחד של סוג מסוים של תעבורה עבור מארח אחד, אז נוצר פורט חדש עבור התעבורה הזו שתקשר רק לאותו מארח.

5) מה זה subnet ולמה צריך את זה ?

subnet הוא בעצם רשת בתוך רשת. ממשקי המכשירים יכולים להגיע פיזית אחד לשני מבלי לעבור דרך נתב. הוא מנתק כל ממשק מה-host או מהנתב שלו ומצייר "איים" של רשתות מבודדות שנקרת תת-רשת. המטרה היא ליצור רשת מחשבים מהירה, יעילה, גמישה ומאובטחת יותר, וגורם לשיפור של ביצועי הרשת על ידי מיפוי של מסלול לנתבים. ככל שהרשתות הופכות לגדולות יותר, התעבורה העוברות דרכן זקוקה לנתבים יעילים יותר. עם זאת על ידי CIRD, ניתן להצקות כתובות IP לניתוב IP כך שנוכל לחלק את הכתובות IP ל-Classes שונים שיאפשר למיליוני מכשירים יוכלו להיות מחוברים, ומפחית את הזמן שלוקח לנתבים עד ימצאו את המכשיר המתאים. זו הסיבה שתת-רשת שימושית, היא מצמצמת את כתובת ה-IP לשימוש בטווח של מכשירים.

6) למה צריך כתובת MAC ולמה לא מספיק לעבוד עם כתובת IP ?

כתובות MAC וכתובות IP פועלות בשכבות שונות במודל השכבות. בשכבת הלינק, כתובות MAC היא בעצם הכתובת הפיזית, לכל כרטיסי רשת יש כתובת MAC ייחודית. היא מאפשרת זיהוי פיזי ישיר בטווח זהה ברשת, בעוד שכתובות IP בשכבת הרשת משומשת לזיהוי מכונות ברחבי רשתות שונות, היא הכתובת הלוגית של המכשיר ומאפשרת לתקשר בין המחשבים בתתי רשתות שונות. שני הכתובות משרטות מטרות שונות וקריטיות, כתובת MAC אומר לנו "מי אנחנו" ואינו יכול להשתנות, כתובת IP מספרת "איפה אנחנו" ויכולה להשתנות. הם עובדים יחד כדי לוודא שההודעה מועברת לאדם הנכון ברשת הנכונה. אכן, צריך את כתובת ה-IP כדי ליצור טבלת ניתוב שמאפשרת העברה של חבילות במהירות על ידי פרוטוקול DHCP.

מי שמעביר את החבילה מהמקור ליעד הוא כתובת ה-IP. אבל מה שמעביר את החבילה ממחשב המקור לנתב 1, ולאחר מכן מנתב 1 לנתב 2, ולאחר מכן מנתב 2 ליעד הוא כתובות ה-MAC. בכל שלב במסלול של שליחת ההודעה, כל פעם שחבילה מגיעה לנתב ברשת הוא משווה את כתובת MAC היעד של החבילה לכתובת ה-MAC של נתב עצמו. אם הכתובות תואמות, כלומר שהגענו ליעד שלנו החבילה מעובדת, אחרת היא נמחקת.

לכן, לכתובות MAC וIP יש תפקידים והכרחיים לעברת הודעה מנקודה אחת לנקודה אחרת.

7) מה ההבדל בין Router-ו Nat, Switch ?

קיימים לא מעט הבדלים בין ה- *Router-ו Nat, Switch*, ההבדל העיקרי בניהם בא מהגדרתם, תפקידם ומיקומם ברשת. אכן, הנתב אחראי על ניתוב ההודעות שהיא פעולה של העברת פקטות מיעד אחד ליעד אחר ברשתות שונות. בעוד שמתג הוא מכשיר רשת מחשבים המחבר מכשירים שונים יחד ברשת מחשבים אחת. נתב פועל בשכבת הרשת בניגוד למתג שהוא נצא בשכבת הלינק. לעומת זאת, ל-NAT יש תפקיד אחר לגמרה, היא בעצם טכניקה ניתוב מחשבים ש מתגרמת IP פרטי ל-IP ציבורי כדי לאפשר תקשורת עם רשתות חיצוניות.

נתב מחבר מספר רשתות ועוקב אחר תעבורת רשת ביניהן והוא תואם ל-NAT מאשר מתג שלא יכול לבצע NAT ולתרגם כתובות IP ציבוריות לכתובות פרטיות. פעולות הנתב מתבצעות סביב כתובות IP ומתגים עובדים עם כתובות MAC מכיוון שהם פועלים בגבולות רשת אחת.

8) שיטות להתגבר על מחסור IPV4 ולפרט ?

נציג כמה שיטות שמהוות פתרון על המחסור ב-**IPV4**:

- **NAT:** *(Network Address Translation)* היא טכנולוגיה אפילו פרוטוקול שהוא מתמקם בין השרת הפנימית לבין החיבור שלו לאינטרנט, ומקצה כתובת IP פנימית ייחודית, אלה כאשר גורם חיצוני רוצה לתקשר עם המחשב שלנו, הוא בעצם עושה את זה דרך כתובת IP חיצונית ובעצם התהליך הזה מאפשר לחסוך על כמות כתובות IPשמקושרות בכל העולם.
- **שרת מארח וירטואלי:** *(Virtual Hosting)* טכניקה זו יכולה להיות שימושית בעת שימוש IP ולהגיש שמות מרובים בחלק או בכל כתובות ה-IP, השרת יוכל להכיל מספר כתובות וכך לבצע אופטימיזציה בחלוקת כתובות IP.
- **DHCP:** *(Dynamic Host Configuration Protocol)* הפרוטוקול הזה מאפשר לנו להשיג כתובות IP באופן דינאמי בשרת מקומית (LAN) כאשר הוא "מצטרף" לרשת, הוא בעצם יכול לחדש את "החוזה" שלו על כתובת שלו בשימוש וכך מאפשר שימוש חוזר בכתובות.
- **IPV6:** הפרוטוקול הזה מהווה פתרון לבעיה כמות כתובות IPV4, ההבדל המהותי בניהם נובע מהגדלת מרחב הכתובות מ-32 סיביות ל-128 סיביות כך שהכמות המקסימלית האפשרית היא 2^{128} כתובות.

9) נתונה הרשת הבאה: AS2 ו-AS3 מריצים OSPF, AS1 ו-AS4 מריצים RIP, בין ה-ASS רץ BGP, אין חיבור פיזי בין AS2 ל-AS4.

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

הנתב 3c לומד על תת רשת x על ידי פרוטוקול **eBGP**, באופן יותר מדויק הנתב 3c מקבל התראת מסלול (*"path advertisement"*) x, AS4 מהנתב 4c מ-AS4.

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

הנתב *3a* לומד על תת רשת x על ידי פרוטוקול **iBGP** , באופן יותר מדויק הנתב *3a* מקבל את כל המידע מהנתב 3c שהוא בעצמו מקבל התראת המסלול x, AS4 (דרך פרוטוקול eBGP) מהנתב 4c מ-AS4.

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

הנתב *1c* לומד על תת רשת x על ידי פרוטוקול **eBGP**, באופן יותר מדויק הנתב *1c* מקבל התראת המסלול x, AS3,AS4 (inter-domain) מהנתב *3a* ב- AS3 , שהוא בעצמו לומד על ידי פרוטוקול iBGP (intra-domain) מהנתב 3c , שהוא בעמו לומד על x דרך פרוטוקול eBGP מהנתב 4c מ-AS4.

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

כיוון שאין חיבור פיזי בין AS2 ל-AS3, אז הנתב*2c* לומד על תת רשת x על ידי פרוטוקול **iBGP,** באותו אופן כמו בסעיפים קודמים. המידע יוצא מ-AS4 עובר ב-AS3, AS1 ומגיע בסוף ל-AS2 דרך נתבים שמקשרים בניהם. במידע בתוך אותו AS מועבר דרך פרוטוקול iBGP ובין ASS שונים עם הפרוטוקול eBGP. בסוף במידע מגיע לנתב *2a* ע"י פרוטוקול eBGP מהנתב 1b ומועבר ל-*c 2* ע"י iBGP.

.