



רשות מחשבים

עומר רוזנבוים ■ שלומי הוד



מזה השיבור הצבאי

www.cyber.org.il

רשותות מחשבים

גרסה 2.31

עריכה

עומר רוזנבוים

שלומי הוד

כתיבה

עומר רוזנבוים - כותב ראשי

תומר גביש

מתן זינגר

רמי עמר

שלומי בוטנרו

שלומי הוד

כתיבת מצגות לפי הספר

ברק גונן

אין לשכפל, להעתיק, לצלם, להקליט, לתרגם, לאחסן במאגר מידע, לשדר או לקלוט בכל דרך או אמצעי אלקטרוני, אופטי או מכני או אחר – כל חלק שהוא מהחומר שבספר זה. שימוש מסחרי מכל סוג שהוא בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב ממטה הס"בר הца"ל.

© כל הזכויות על החומרים המקוריים ששובצו בספר זה שמורות לבעליהן. פירוט בעלי הזכויות – בסוף הספר.

© תשע"ו – 2016. כל הזכויות שמורות למיטה הס"בר הца"ל.

הודפס בישראל.

<http://www.cyber.org.il>

תוכן עניינים

3	תוכן עניינים
4	מבוא
9	תוכן עניינים מצגות
11	פרק 1 - תחילת מסע - איך עובד האינטרנט?
27	פרק 2 - תכנות ב-Sockets
48	פרק 3 - Wireshark ומודל המש השכבות
83	פרק 4 - שכבת האפליקציה
138	פרק 5 - Scapy
162	פרק 6 - שכבת התעבורה
223	פרק 7 - שכבת הרשות
289	פרק 8 - שכבת הקו
331	פרק 9 - רכיבי רשות
335	פרק 10 - השכבה הפיזית (העשרה)
361	פרק 11 - איך הכל מתחבר, ואיך עובד האינטרנט?
384	פרק 12 - תכנות Sockets מתקדם: ריבוי משתמשים (הרחבה)
405	פרק 13 - מיליון מושגים
414	פרק 14 - פקודות וכליים
416	זכויות יוצרים - מקורות חיצוניים

מבוא

רשתות תקשורת הן דבר מדהים. מАЗ ומעולם, בני אדם רצו להעביר מסרים ביניהם. בעבר הרחוק, בהיעדר אמצעים אחרים, התבססה התקשרות בעיקר על תקשורת מילולית וشفת גוף. על מנת להעביר מסרים למרחקים גדולים יותר, נעשה שימוש בשליחים. בתקופת המערות, החל האדם הקדמון לעשוט שימוש לצורכי מערות בכך לתקשר. בשלב מאוחר יותר, הופיע הכתב - מערכת סימנים מסוימת שאפשרה העברת מסרים בצורה רחבה יותר. המצאת הדפוס, במאה ה-15, אפשרה להעביר ידע ומסרים לאנשים רבים ובעלות העתקה נמוכה יחסית.

במאה ה-19 החלה מתפתחת התקשרות האלקטרונית, המאפשרת תקשורת המונים מהירה ויעילה. בני אדם, עברנו דרך ארוכה מאז שהתקשרות התבססה על תקשורת מילולית, ועד לשימוש היום יומי בגלישה באינטרנט, בדואר אלקטרוני או בתוכנות העברת מסרים כגון App WhatsApp. המהפהכה התקשורתיות משפיעה על כל תחומי החיים שלנו - על הדרך בה אנו מדברים זה עם זה, על הדרך בה אנו מփשים מידע כדי ללמידה, על מידת ההכרות שלנו עם העולם הסובב אותנו ואף על החלטות שאנו מקבלים בחינינו.

התפתחות התקשרות והתפתחות המחשב שזרום זה בזה. עולם המחשבים משתנה ומפתחת בקצב מהיר. בעוד המחשבים הראשונים היו מבודדים זה מזה, רוב מכרייע של המחשבים כיום מחוברים זה לזה דרך רשת האינטרנט, שהוא למעשה רשת של רשתות. לכל רשות יש את המאפיינים שלה: ישן רשותות קטנות (בינה מחוברים למשל שני מחשבים), רשותות בינוניות (כמו רשות של בית ספר, שיכולה לחבר כמה עשרות או מאות מחשבים) ורשתות גדולות (כמו רשות של חברת בעל אלף מחשבים). ישן רשותות קוויות ורשתות אלחוטיות, רשותות מהירות ורשתות איטיות. מטרתן של כל הרשותות הינה להעביר מידע בין מחשבים.

בספר נכנס אל תוך העולם המדהים של רשותות מחשבים. נלמד כיצד עובדת התקשרות בין מחשבים, נכיר סוגים שונים של רשותות, נבין איך הן בנויות ואייר הכל מתחבר לכך העולם הווירטואלי שאנו מכירים כיום.

קהל היעד של הספר

הספר מיועד לשני קהלי: יעד עיקריים:

1. תלמידים ומורים הלומדים במסגרת חלופת הגנת סייבר במגמת הנדסת תוכנה.
2. כל מי שמעוניין למדוד את תחום רשותות מחשבים באופן עצמאי.

שיטת הלימוד של הספר

ספר זה ככל הנראה שונה מספרי לימוד אחרים שהכרתם. הספר נועד לאפשר ללמידה עצמאית, והוא **פרקטי** מאוד וכל תרגול רב. רשותות מחשבים הינו נושא עצום ומורכב, ולא ניתן לכטוט את כלו או מרביתו בספר אחד.

מעשיות הייתה הקי שנהנכה אותו בהחלטה אלו נושאים יכולו בספר זה ואלו יישארו מחוץ לו - הספר מתמקד באוטם נושאים אשר ניתן לישם וلتתגלג בקהלות יחסית. הספר כולל מעט מאוד נושאים שהם תיאורתיים בלבד, ורובה ככלו מתעסק במושגים אמייניטיים וביטויים ברשות האינטרנט. כבר מהפרק השני של הספר, הלימוד ילווה בכתיבת קוד מצד הלומד.

במהלך הלימוד בספר, עליים לתפקיד **סטודנטים פעילים** - כלומר, לא רק לקרוא ולהבין את החומר, אלא גם לתרגל אותו. בספר משלבים תרגילים רבים, חלקם מודרכים וחילק לביצוע עצמי. כדי לרכוש שליטה בחומר הלימוד, יש לבצע את כל התרגילים ולודאו שאתם מבינים אותם. התרגילים המודרכים נבנו כך שהם מפורקים למשימות מוגבלות המתבססות זו על זו ומונחות שהלומד מבצע בפועל את ההנחיות. אל תשטפקו בקריאת התרגילים המודרכים, והקפידו לבצע אותם בשלב אחריו שלב.

אחת ממטרות הספר היא **להקנות כלים** בהם תוכלו להשתמש בעצמכם, כדי להרחיב את אופקיכם, לחזור וללמוד באופן עצמאי. אי לך, תזכו במהלך הקראיה להיחשף לתוכנות, כל תכונות ודרכי חשיבה שיאפשרו לכם להרחיב את הידע גם מעבר למה שמופיע בספר זה.

הערה: הספר מתבסס על עבודה מעל מערכת הפעלה Windows. קוראים המעוניינים לעבוד עם מערכות הפעלה מבוססות UNIX, מוזמנים לפנות לפרק [פודוט וכלים](#) כאשר מוצג בספר כל הרץ מעל מערכת הפעלה Windows, וללמוד ממנו על ההתאמות הנדרשות.

צדדים להמשך

חלק משיטת הלימוד של הספר, המעודדת במידה עצמאית, בסוף חלק מהפרקeos הוסיף סעיף "צדדים להמשך". סעיף זה נועד לתלמידים סקרנים המעוניינים להרחיב את הידע, כולל מקורות מידע נוספים ותרגילים מתקדמים.

סרטונים

לאורך הספר שובצו מספר סרטוני הדרכה שנועדו להקל על הלמידה. אתם מוזמנים להיעזר בהם לאורך הקראיה. בכל פעם שהיא קישור לסרטון, יופיע גם קוד QR שיאפשר להגיע אליו בקהלות. כמו כן, אתם מוזמנים לפנות לרשימת הסרטונים המלאה בכתבota:

<http://data.cyber.org.il/networks/videos/playlist.html>



ידע מקדים נדרש

ספר זה מניח כי לקורא היכרות בסיסית לפחות עם השפה Python. בהתאם לצורך, אתם מוזמנים להשתמש בספר הלימוד של שפת Python מאת זהר זילברמן, הזמין בכתובות:
<http://data.cyber.org.il/python/python.pdf>

התקנות נדרשות

כאמור ספר הלימוד מבוסס על שפת פיתון. ישן התקנות רבות של פיתון, لكن נרצה להמליץ על סביבת עבודה ושימוש נכון בסביבת העבודה.

מומלץ להתקין פיתון חלק מסביבת ההתקנות של גבהים, שפותחה על ידי **תומר טלגט** ושנמצאת בקישור:
<https://docs.google.com/uc?id=0B5GxVtBzVtQNNFhLSUxLZkdpekU&export=download>

מסמך הדרכה להתקינה:

<http://tinyurl.com/portableheights>

התקינה זו כוללת את כל התוכנות של גבהים, מאסmbali דרך פיתון ורשותות וכליה במערכות הפעלה.

גרסת הפיתון של התקינה היא 2.7 - בהתאם לספר הלימוד. התקינה זו כוללת את כל הכללים הנוספים שניצטרכו להשתמש בהם במהלך הלימוד (כגון wireshark, scapy).

את כל התוכנות המותקנות – wireshark, pycharm, python, ניתן למצוא בספריית ההתחלה או בkitzori הדרך בשולחן העבודה.

לאחר התקינה יהיו ברשותכם 2 שיטות שונות להרצת פיתון. נסקור אותן:

- דרך `command-line`: שיטה זו מתאימה לבדיקת דברים קטנים בפייתון, כגון פעולות מתמטיות, ביצוע `help` או `dir`. מאד קשה ולא מומלץ לתוכנת קוד פיתון של יותר משורות בזדמנות בדרכך זו.
- דרך PyCharm: סביבת עבודה שמצוירה לימוד והתנסות. היתרונות המרכזים שלה – debugging באמצעות breakpoints וחיפוי על שגיאות תכנות (כגון שכחה של נקודותים, טעות בשמות פונקציות, כמות לא נכונה של פרמטרים לפונקציה וכו'). והוא סביבת העבודה המומלצת ללימוד ולכטיבת תוכניות. מומלץ להיעזר בלימוד pycharm במצגת ההדריכה מאט ברק גון. לינקים לכל מצגות פיתון ורשותות נמצאים בהמשך.

アイكونים

בספר, אנו משתמשים באיקונים הבאים בצד הימני להדגיש נושאים ובצד הימני להקל על הקראיה:



שאלה.



הגדירה למונח.



הדגשת נקודה חשובה.



"תרגיל מודרך". עלייכם לפתור תרגילים אלו תוך כדי קריאת הספר.



תרגיל לביצוע. תרגילים אלו עלייכם לפתור בעצמכם, והפתרון בדרך כלל לא יוצג בספר.



פתרון מודרך לתרגיל אותו היה עלייכם לפתור בעצמכם.



ר堪 היסטורי, או מידע העשרתי אחר.



הפניה לסרטון.

abwechthatt midu

על מנת להבין כיצד רשותות פועלות בצורה עמוקה, נבחן גם היבטי אבטחת מידע של מערכות תקשורת לאורך הלימוד. עם זאת, בחרנו שלא לכלול את החומר הנוגע לאבטחת מידע בספר זה, והוא ניתן לכם בכיתה לאורך השנה.

תודות

אנשים רבים תרמו לתהילה יצרתו של ספר זה. אירים צור ברגורי ליוזה את הספר מטהlixir התכונן ועד לשלביו הסופיים והשפעה עליי רבות. תומר גלון מימוש את הצד השרת עבור תרגילים במהלך הספר. מיכל לשם סיעה באופן משמעותי בהבאת הספר לידי גרסה להפצה. משובים שקיבלנו על הספר לאורך הזמן שיפרו אותו ותרמו לו

מואוד. באופן מיוחד אנו מבקשים להודות ליוויי ממו וממן עבורי על העורחותם הבונות. יוחאי איזנרייך כתב פתרונות רבים לתרגילים הניתנים בספר, ובכך סייע לשפר אותם וכן סיפק פתרונות לדוגמה עבור תלמידים. ברק גונן כתב מצגות לימוד בהתאם לספר. כמו כן אנו מבקשים להודות לדניאל גולדברג, נעם ארץ, ליאור גרנטשטיין, יהודה אור ואנטולי פיימר על משוביהם. לכל המורים, התלמידים והחברים שהשפיעו וסייעו בתהילה יצירת הספר – תודה רבה.

עוֹמֵר רוזנְבַּיִם

שָׁלּוֹמֵי הָוֶד

תוכן עניינים מצגות

פיתון:

Before we start:	http://data.cyber.org.il/python/1450-3-00.pdf
Intro and CMD:	http://data.cyber.org.il/python/1450-3-01.pdf
Pycharm:	http://data.cyber.org.il/python/1450-3-02.pdf
Variables, conditions and loops:	http://data.cyber.org.il/python/1450-3-03.pdf
Strings:	http://data.cyber.org.il/python/1450-3-04.pdf
Functions:	http://data.cyber.org.il/python/1450-3-05.pdf
Assert:	http://data.cyber.org.il/python/1450-3-06.pdf
Files and script parameters:	http://data.cyber.org.il/python/1450-3-07.pdf
Lists and tuples:	http://data.cyber.org.il/python/1450-3-08.pdf
Dictionaries:	http://data.cyber.org.il/python/1450-3-09.pdf
Object Oriented Programming:	http://data.cyber.org.il/python/1450-3-10.pdf
Utilities and exceptions:	http://data.cyber.org.il/python/1450-3-11.pdf
Regular expressions:	http://data.cyber.org.il/python/1450-3-12.pdf

רשתות:

http://data.cyber.org.il/networks/1450-2-00.pdf	מבוא לשנת הלימודים:
http://data.cyber.org.il/networks/1450-2-01.pdf	פרק 1- מבוא לרשתות מחשבים:
http://data.cyber.org.il/networks/1450-2-02.pdf	פרק 2- תכונות סוקטים:
http://data.cyber.org.il/networks/1450-2-03.pdf	פרק 3א- מודל חמש השכבות:
http://data.cyber.org.il/networks/1450-2-04.pdf	פרק 3ב- wireshark :
http://data.cyber.org.il/networks/1450-2-05.pdf	פרק 4א- שכבת האפליקציה :HTTP
http://data.cyber.org.il/networks/1450-2-06.pdf	פרק 4ב- HTTP נושאים מתקדמים:
http://data.cyber.org.il/networks/1450-2-07.pdf	פרק 4ג- פרוטוקול DNS :
http://data.cyber.org.il/networks/1450-2-08.pdf	פרק 4ד- אבחון פרוטוקול SMTP :
http://data.cyber.org.il/networks/1450-2-09.pdf	פרק 5- Scapy :
http://data.cyber.org.il/networks/1450-2-10.pdf	פרק 6א- שכבת התעבורה :
http://data.cyber.org.il/networks/1450-2-11.pdf	פרק 6ב- מבוא לפרוטוקולים של שכבת התעבורה :
http://data.cyber.org.il/networks/1450-2-12.pdf	פרק 6ג- UDP :
http://data.cyber.org.il/networks/1450-2-13.pdf	פרק 6ד- TCP :
http://data.cyber.org.il/networks/1450-2-14.pdf	פרק 7א- שכבת הרשת מבוא לניטוב :
http://data.cyber.org.il/networks/1450-2-15.pdf	פרק 7ב- כתובות IP וראוטר :

<http://data.cyber.org.il/networks/1450-2-16.pdf>
<http://data.cyber.org.il/networks/1450-2-17.pdf>
<http://data.cyber.org.il/networks/1450-2-18.pdf>
<http://data.cyber.org.il/networks/1450-2-19.pdf>
<http://data.cyber.org.il/networks/1450-2-20.pdf>

פרק 7ג- פרוטוקול ICMP:
פרק 7ד- פרוטוקול DHCP:
פרק 8- שכבת הקו:
פרק 10- השכבה הפיזית:
פרק 11- איך הכל מתחבר:

פרק 1 - תחילת מסע - איך עובד האינטרנט?



ניתן לצפות בסרטון המלאו את פרק זה בכתב: <http://youtu.be/ad8EOsXFuxE>

ספר זה עוסק ברשות מחשבים. מה זה בעצם אומר? איך רשת האינטרנט עובדת?

בפרק זה נתחל לערנות על שאלות אלו באופן כללי, ונקבל תמונה כללית על איך עובד האינטרנט. בהמשך הספר, נרד לפתרים ונזכה לקבל תמונה הרבה יותר עמוקה ומדויקת. על מנת להתחיל את ההסבר, נפתח בשאלת:



מה קורא לנו גולשים לאתר Facebook?

רובנו גלשנו ל-Facebook, הרשת החברתית העצומה שמונה מעל ל-מיליארד משתמשים. אך האם ערכנו לשאול את עצמנו - מה בעצם קורא מאחורי הקלעים כshawim? איך יתכן שאנו נמצאים בבית, מקיים בדף Browser) את הכתובת "www.facebook.com", לחצים על מקש-h-Enter, ומתקבלים תמונה מצב של כל החברים שלנו?

על מנת לענות על שאלה זו, علينا להבין מה האתר Facebook צריך כדי לתפקיד.

כל אתר וכך גם האתר Facebook זקוק לאחסון (**Hosting**) - הכוונה למקום בו יימצאו דפי האתר ואלו יפנו המשתמשים. אתר Facebook ישמור גם את המידע על כל המשתמשים כגון: מי חבר שלי מי התמונות שהועלו לאתר, סטטוסים וכו'. בנוסף, אתר Facebook זקוק לעיצוב. יש לעצב לוגו להציג היקן תואג רשימת החברים, היקן יציג העדכנים שלהם, איפה יציג הפרסומות ועוד. האתר Facebook גם לאיזמות (**Authentication**) - עליו לזהות את המשתמש שפונה אליו. לשם כך, Facebook צריך לזכור את כל המשתמשים והסימאות שלהם, ולאחר מכן לשלוחו דרך דוא"ל או SMS. Facebook נדרש לתקשות. לעומת זאת, ניתן להיות דרך שתאפשר לאדם לתקשר עם האתר של Facebook בין אם מהמחשב שלו בישראל ובין אם מהסמארטפון שלו כשהוא נמצא בטיחות באיטליה.

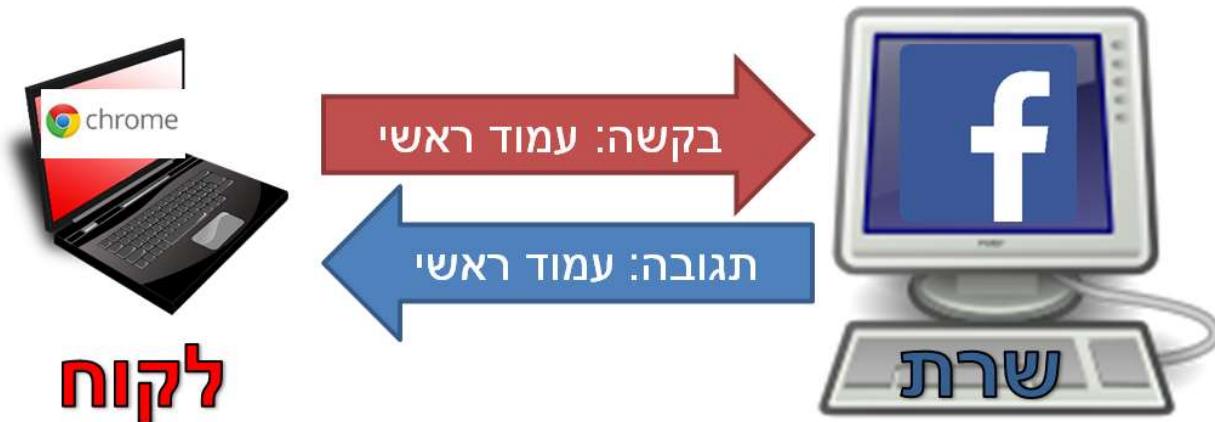
ספר זה יתמקד בתקשורת שבין רכיבים אלקטרוניים שונים ודרך העברת המידע ביניהם. בפרק הנוכחי, נתאר באופן כללי מה קורא מהרגע שאנו כותבים "www.facebook.com" בדף ולוחצים על מקש-h-Enter, ועד שמוופיע דף הבית של Facebook על המסך.

WWW – World Wide Web

כשאנו אומרים "인터넷", אנו מתכוונים בדרך כלל ל-WWW (World Wide Web). זהו אוסף עמודי האינטרנט אליהם אנחנו גולשים בדף. משמעות המילה Web היא רשת. עמודי האינטרנט מושרים אחד לשני כמו רשת

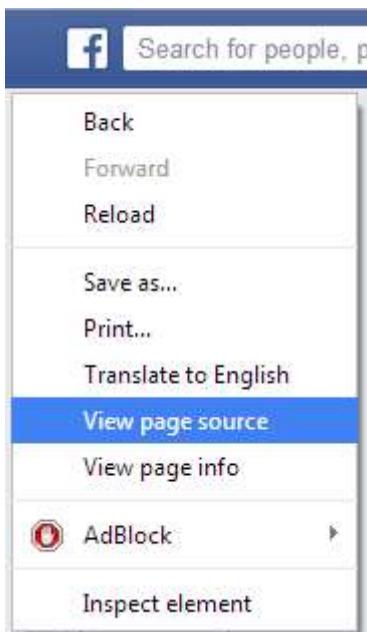
של קורי עכבי שמקשרים זה לזה. עמודי האינטרנט מפוזרים בכל רחבי העולם - World Wide - בעוד אחד נמצא בחיפה, שני יכול להיות בטוקיו ושלישי בניו-יורק. העמודים השונים מקושרים ביניהם באמצעות לינקים.

על מנת שהדף יוכל להציג את העמוד הראשי של Facebook, עליו לדעת כיצד העמוד נראה. הכוונה היא לדעת איזה טקסט קיים בעמוד, היכן כל חלק מהעמוד ממוקם, באיזה גוף הטקסט כתוב ובאיזה גודל, האם צריך להציג תמונות על המסר, אילו תמונות ועוד. את כל המידע הזה, הדף נושא משיג מאתר Facebook עצמו. ב כדי ש-Facebook ישלח לדף את המידע, על הדף להודיע שהוא מעוניין בו. כמובן, הדף צריך לשולח הודעה בקשה (Request) אל Facebook, ובה להגיד: "שלח לי את המידע הנדרש כדי להציג את העמוד הראשי של האתר www.facebook.com". כאשר Facebook מקבל את הבקשה, הוא שולח תגובה (Response) שמכלילה את המידע הדרוש.



האתר של Facebook מקבל בקשה ושולח תגובה. כמובן, הוא מספק שירות לדף. אי לך, נאמר כי האתר של Facebook הינו שרת (Server), והדף הינו לקוח (Client). באופן כללי, כאשר גולשים ב-WWW, ישנו לקוחות (דפים) ששולחים בקשות אל השירותים (אתרי האינטרנט), והשירותים מחזירים תשובות לקוחות. אותן מידע שmagיע בתשובות, משתמשת הדפים בכך להציג על המסך את אתר האינטרנט למשתמשים.

על מנת לראות את המידע שהשרת שלח ללקוח (או לפחות חלק ממנו), ניתן ללחוץ על הכפתור הימני של העבר באזורי ריק באתר האינטרנט, ולבחר באפשרות "View page source" (בעברית: "הציג מקור"):



בחילון שיפתח יוצג טקסט שמכיל את המידע ששלח השרת כתגובה לבקשת הלקוח (הטקסט שנמצא באתר, איפה כל דבר נמצא, אילו תמונות נמצאות וכו'). בשלב זה לא נתעמק במה אנחנו רואים בדיק. עם זאת, נסו לזהות בעצמכם חלקים מעמוד האינטרנט - האם אתם יכולים לזהות טקסט שמוצג לכם בדף?

 הכנסו אל האתר <http://www.ynet.co.il>. הקliquו עם הכפתור ימני של המouse, ובחרו באופציה **View page source**. מצאו בחילון שיפתחה את הכתובת הראשית שמופיעיה בעמוד של Ynet.

כתובות IP

על מנת שנוכל לשולח ולקבל הודעות באינטרנט, علينا לדעת לאן לשלוח את הבקשות ועל השרת לדעת להיכן לשלוח את התשובות. כאשר אנו שולחים מכתב דואר, אנו מציינים על המעטפה את **כתובת היעד** (厰ען) ואת **כתובת המקור** (厰ען). באופן דומה, גם כאשר נשלח מידע ברחבי האינטרנט, יש צורך בכתובות מתאימות שייזהו את השולח ואת היעד של ההודעה. למשל, בדוגמה לעיל, נרצה לדעת מה הכתובת של המחשב שלח את הבקשה (כתובת המקור), ומה הכתובת של Facebook (כתובת היעד). באינטרנט, כתובות אלו נקראות **כתובות IP** (IP Addresses).

על IP וכותבות IP בפרט, נלמד בהרחבה בפרק **שכבות הרשת**. בשלב זה רק נבין כיצד כתובות אלו נראהות. שם שלכתובות דואר יש מבנה קבוע, שכולל את שם הנמען (למשל: משה כהן), רחוב ומספר בית (הרצל 1), עיר (ירושלים) ומיקוד (1234567), כך גם לכותבות IP יש מבנה קבוע. כתובות IP מורכבות מארבעה בתים (bytes). כל בית יכול לקבל ערך בין 0 ל-255, ונוהג להפריד את הבטים בנקודה. מכאן ש-כתובת IP יכולה להיות 0.0.0.0, 0.0.0.1, ..., 255.255.255.255, והטוווח שביניהם, למשל 1.2.3.4 או 10.42.2.3.

אם נחזור לדוגמה של הדף, הרי שהודעת**הבקשה** שהדפדף שולח לאתר Facebook מכילה בכתובת המקור את כתובת ה-IP של המחשב ממנו מתבצעת הגלישה, ובכתובת**היעד** את כתובת ה-IP של האתר Facebook. כאשר נשלחת הודעה התגובה, היא נשלחת מ-**Facebook** לדפדף, ולכן **כתובת המקור** תכיל את כתובת ה-IP של האתר Facebook, בעוד **כתובת היעד** תכיל את כתובת ה-IP של המחשב ממנו מתבצעת הגלישה.

 גלו את כתובת ה-IP שלכם! לשם כך, היכנסו אל האתר <http://www.whatismyip.com>



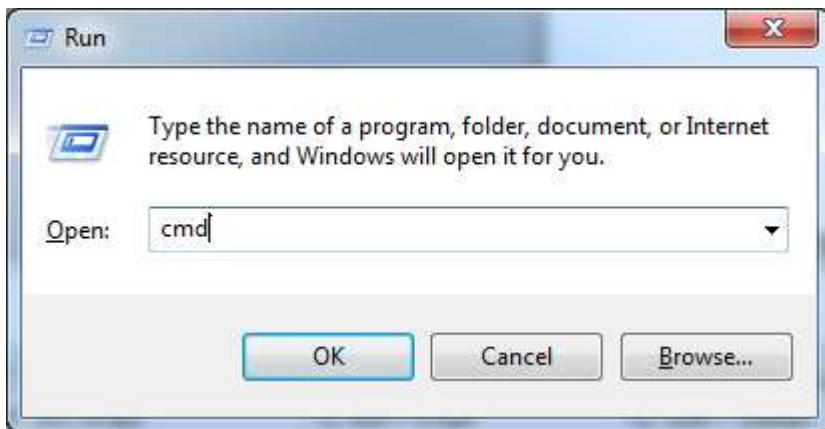
תרגיל מודרך - מציאת כתובת ה-IP של אתר אינטרנט מסוים

נאמר ונרצה לגלות את כתובת ה-IP של **Facebook**, או של **Google**. איך נוכל לעשות זאת?

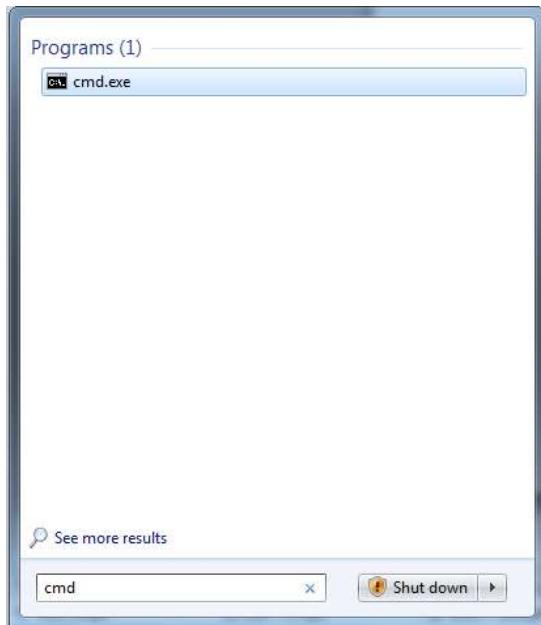
דרך אחת לגלות את כתובת ה-IP של אתר אינטרנט היא להשתמש בכלי ping. לצורך כך, הריצו את **שורת הפקודה (Command Line)**, בה נשתמש רבות לאורך הספר. לחזו על צירוף המקשיים (WinKey+R) בכדי להגיע לשורת הפעלה. Winkey הינו המקס במקלדת שנמצא בין המקס Ctrl למקש Alt ומופיע עליו הלוגו של Windows:



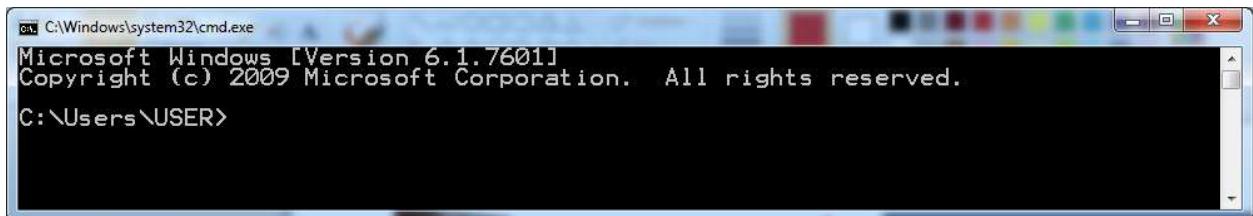
לחילופין, תוכלו לגשת אל Start (התחל) ולחזור באופציה Run (הפעל). הקישו בה את האותיות **cmd** ולחזו על :OK



לחילופין, אם אתם משתמשים ב-7 Windows ומעלה, תוכלו להקש על מקש ה-WinKey-c ולבחר בו כאשר הוא יוצג למסך:



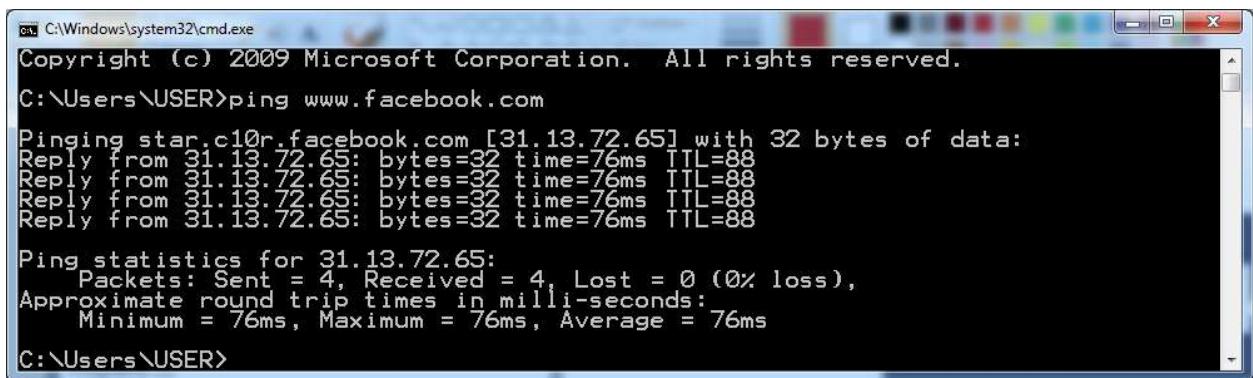
בשלב זה צפי לפתיחת חלון ה-cmd ובו שורת הפקודה:



כעת, הקישו את הפקודה הבאה:

ping www.facebook.com

והקישו Enter. על המסך ידפסו נתונים שונים. בין השאר, תוכלו למצוא את כתובת ה-IP של Facebook:

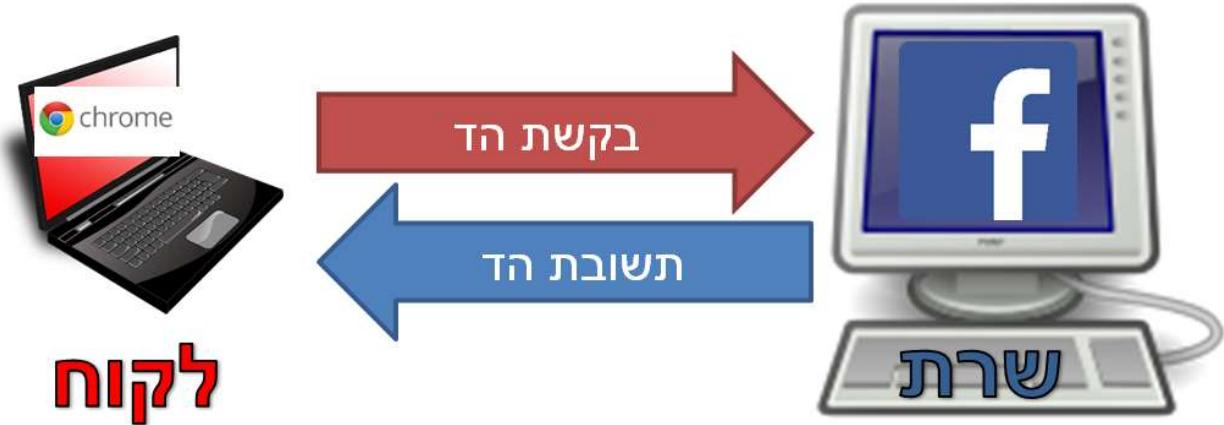


בדוגמה זו, כתובת ה-IP הינה 1.31.13.72.65

כעת, נסו בעצמכם - מה היא כתובת ה-IP של Google?



הכלי **ping** לא מועד בכדי למצוא כתובות IP של אתרים. הוא מועד בכדי לבדוק האם מחשב בעל כתובת IP מסוימת מחובר לרשת, ולמדוד כמה זמן לוקח להודעה להישלח אל אותו מחשב, ואז לחזור אליו. על מנת לעשות זאת, הכלי **ping** שולח **בקשת הד** (כמו לצעוק בערבה כדי לגנות כמה זמן לוקח להד לחזור אליו). כאשר מחשב היעד מקבל את בקשה זו, הוא משיב מיד **בתשובה הד**:



כך ניתן לבדוק האם המחשב בכתובת ה-IP המופיע קיים, ולדעת כמה מהיר החיבור בין אותו מחשב מרוחק. בהמשך הספר, בפרק שכבות הרשת, נלמד לעומק איך הכלי **ping** עובד - ואף תממשו אותו בעצמכם!

GeoIP

עד כה דיברנו על שרתים, ל��וחות וכ כתובות. כאשר גולשים לאתר אינטרנט מסוים, הودעת הבקשה צריכה להגיע בסופו של דבר אל השירות שיטפל בה. אותו שירות נמצא במקום כלשהו בעולם. האם ניתן לגנות היכן הוא נמצא?

ישנם מאגרי נתונים הcoliים מידע על המיקום הגיאוגרפי של כתובות IP. מכאן שבהינתן כתובת IP, ניתן לדעת באיזו מדינה ובאיזה עיר היא נמצאת. מאגרים אלו לא רשמיים ולא מדויקים, אך הם נותנים מענה נכון ברוב

¹ מסיבות שלא נפרט כעת, כתובת ה-IP עשויה להשתנות. לכן, יתכן שכשתריצו את הפקודה במחשב שלכם, תהיה ל- Facebook כתובת IP אחרת.

המקרים. בغالל האופן שבו בני האינטרנט, לא ניתן לייצר מاجر רשמי של מיפוי בין כתובת IP למקום גיאוגרפי, אבל על כך נלמד בהמשך הספר.

אתר לדוגמה שמאפשר למפות בין כתובת IP למיקום הגיאוגרפי שלו, הוא <http://www.geointool.com>. נכון להעתה באתר זה כדי לגלוות, למשל, את המיקום של Google:

GEO IP TOOL

language:

[View my IP information](#) | [More info about IPs](#) | [Firefox Plugin](#) | [Now online](#) | [In your Website](#)

Host / IP: [View info](#)

Host Name: iad23s07-in-f16.1e100.net
IP Address: 74.125.228.80
Country: [United States](#)

Country code: US (USA)
Region: [California](#)
City: [Mountain View](#)
Postal code: 94043
Calling code: +1
Longitude: -122.0574
Latitude: 37.4192

New tool for your Web!

Map Satellite

Map data ©2014 Google, INEGI Terms of Use

Google

[TARINGAI](#) 16 [Tweet](#) 227 [g+](#) 1 [Like](#) 1.4k [Сохранить](#) 62

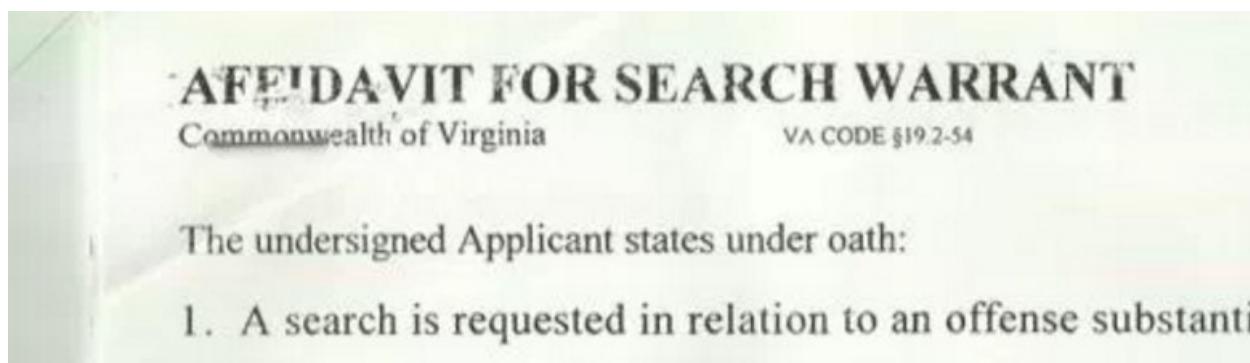
HOSTING BY

נסו זאת בעצמכם: מצאו את המיקום הגיאוגרפי של האתרים הבאים (כל אחד מהם נמצא במדינה אחרת):

- www.yahoo.com
- www.bbc.com
- www.newtoholland.nl
- www.yahoo.co.jp
- www.southafrica.co.za
- www.webawards.com.au

כאמור, המיפוי בין כתובות IP למיקום גיאוגרפי אינו מדויק. הכתובת הבאה ממחישה מדוע במקרים מסוימים זו עלולה להיות בעיה קשה: נניח שפושע מבצע פשע, ורשות החוק מגלה את כתובות ה-IP שלו ומשתמשות בה על מנת לאתר את המיקום הפיזי שלו. אם המיפוי אינו מדויק, יש אפשרות שכותבת ה-IP של הפושע תמוקם ליד ביתו של אדם תמים. במקרה זה, רשות החוק עלולות פשוט על האזרה התמימים, אשר אמן יכול להסביר שהלה טעונה, אך עדין יסבול מהטרדה: דמיינו שהמשטרה פושטה על הבית שלכם עם צו חיפוש, רק משומש שהלה טעונה, אך לא יצליחה למצוא מיקום כתובות IP של פושע.

<http://fusion.net/story/287592/internet-mapping-glitch-kansas-farm/>



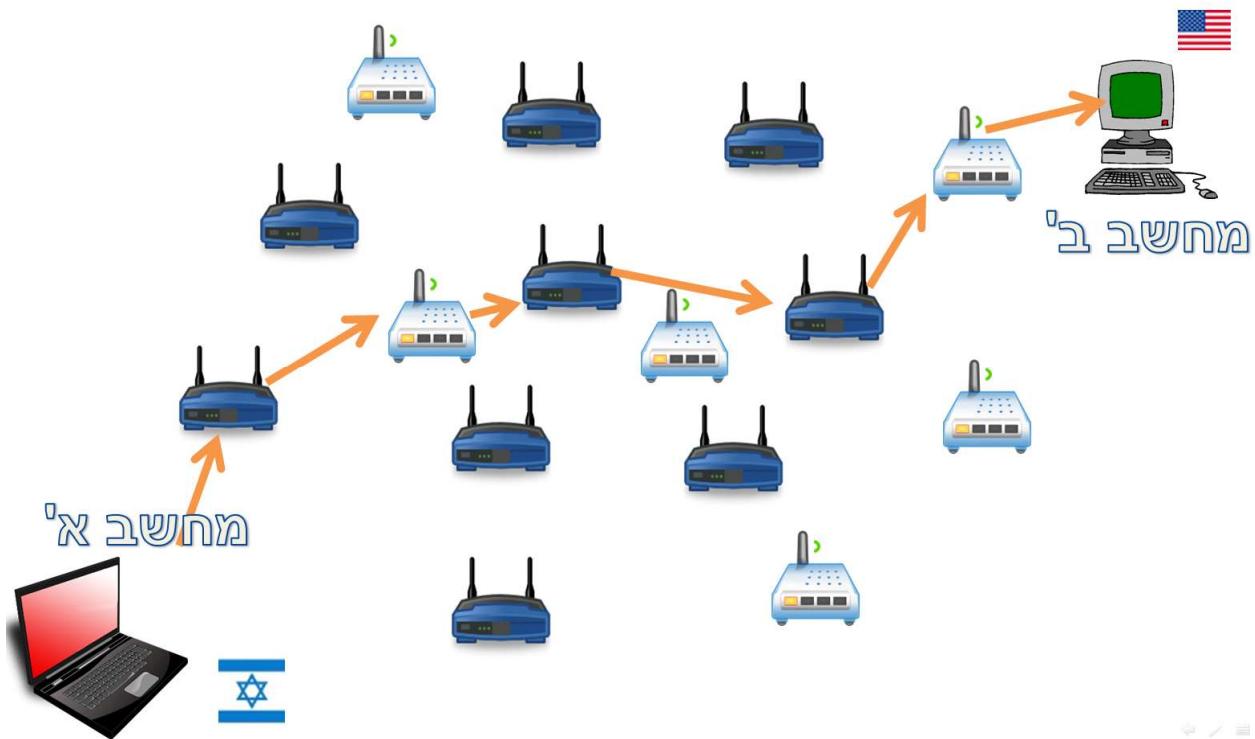
צו חיפוש משטרתי, בחשד לגנבה, שנייתן עקב מיקום IP שגוי (מתוך הכתובת)

عن האינטרנט

אם בדקתם מה המיקום הגיאוגרפי של Google, גיליתם שהוא נמצא בקליפורניה שבארצות הברית. האם זה הגיוני? איך קורה שהודעת הבקשה מהלוקה מגיעה עד לשרת שבארצות הברית, והודעת התגובה חוזרת בחזרה? ההודעות עוברות מרחוק של חצי כדור הארץ הלו ושוב! אם היינו צריכים למן טיסה כה ארוכה עברו כל הودעה שנשלחת, השימוש באינטרנט היה עשוי להיות יקר מדי גם עבור בייל גייטס². מעבר לכך, לא הגיוני שהמחשב שלנו יהיה מחוברrecv בכבלי היבר אל השירות של Google. אם כך, כיצד המידע מצליח להגיע אל Google שבארצות הברית?

האינטרנט הוא למעשה אוסף של הרבה רכיבים שמחוברים זה לזה. הודעה שנשלחת בין שני מחשבים מחוברים לרשת האינטרנט, עוברת בדרך בין הרבה רכיבים שמחוברים זה לזה באופן ישיר. כך כל רכיב מעביר את ההודעה להלאה אל הרכיב הבא, עד שהוא מגיעה אל היעד. העברה של הودעה בין רכיב אחד לרכיב אחר המוחברים ישירות נקראת **קפיצה (Hop)**.

² מייסד חברת Microsoft והאיש העשיר ביותר בעולם, נכון בזמן כתיבת ספר זה.



מה הם בדיקת הרכיבים האלה? איך הם עובדים? על שאלות אלו נענה בהרחבה בהמשך הספר. שימושם לב שלען רכיבים אלו מוטלת משימה מורכבת: עליהם למצוא את הדרך הנכונה להגעה מהמקור (מחשב א' שנמצא בישראל) אל היעד (מחשב ב' שנמצא בארצות הברית). כל אותן רכיבים משתמשים בכתבאות ה-IP של ההודעה בצד' לדעתך לאן היא צריכה להגיע.

היות שהאינטרנט הינה רשת גדולה ומורכבת, לעיתים משתמשים בעברית במונח **ענן האינטרנט** כדי לתאר אותה. ענן האינטרנט מקבל מידע מצד אחד (למשל, מחשב של אדם בישראל), ו מעביר אותו עד לצד השני (למשל, השירות של Google בארצות הברית).



תרגיל מודרך - מציאת הדרך בה עוברת הודעה

אם נוכל לגלות את המסלול שההodata עוברת? אילו רכיבים מעבירים אותה בדרך בין המקור אל היעד? הכליל **traceroute** (קיצור של traceroute - מעקב אחר מסלול) מאפשר לנו לעשות זאת. לשם כך, הריצו שוב את שורת הפקודה (Command Line) אותה פגשנו קודם לכן:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>
```

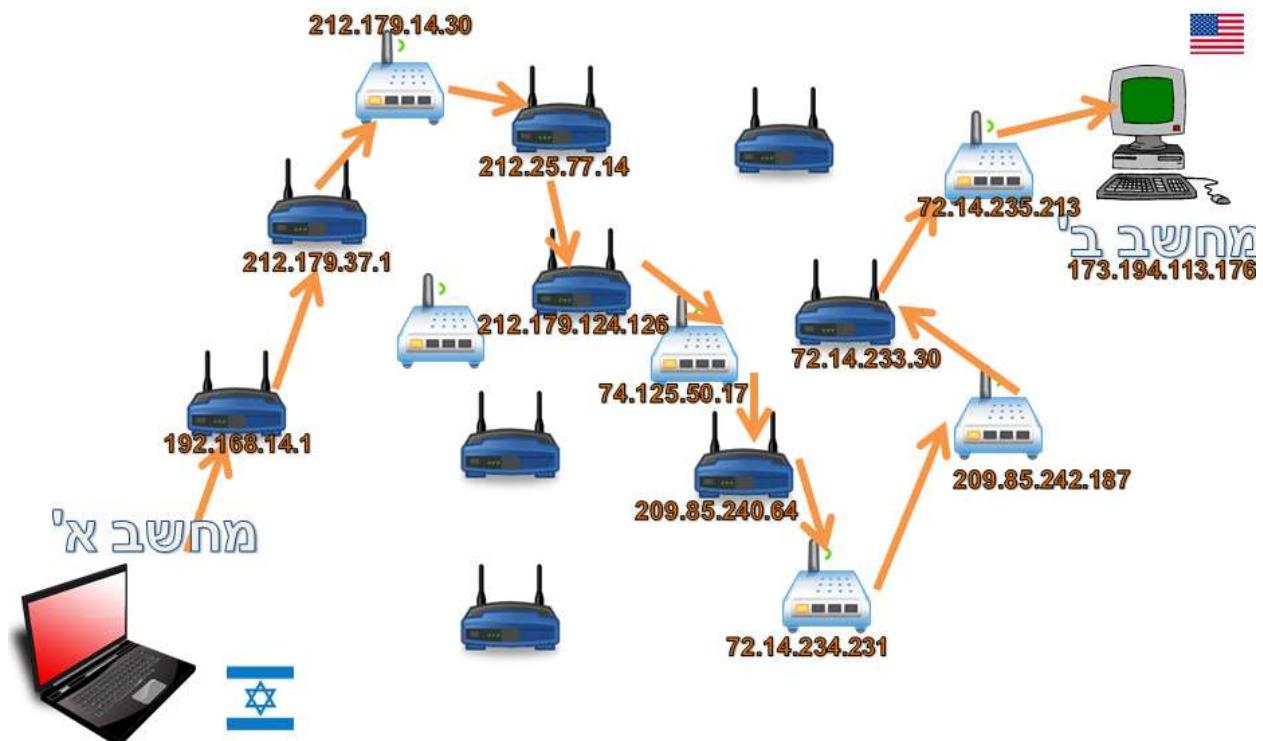
הקישו את הפקודה הבאה:

tracert -d www.google.com

```
C:\Windows\system32\cmd.exe
C:\Users\USER>tracert -d www.google.com
Tracing route to www.google.com [173.194.113.176]
over a maximum of 30 hops:
 1 <1 ms    1 ms    1 ms  192.168.14.1
 2 13 ms    13 ms   13 ms  212.179.37.1
 3 13 ms    13 ms   23 ms  212.179.14.30
 4 27 ms    25 ms   14 ms  212.25.77.14
 5 61 ms    65 ms   62 ms  212.179.124.126
 6 61 ms    61 ms   61 ms  74.125.50.17
 7 62 ms    63 ms   72 ms  209.85.240.64
 8 62 ms    62 ms   62 ms  72.14.234.231
 9 113 ms   76 ms   72 ms  209.85.242.187
10 78 ms    72 ms   72 ms  72.14.233.30
11 71 ms    72 ms   70 ms  72.14.235.213
12 73 ms    74 ms   70 ms  173.194.113.176

Trace complete.
C:\Users\USER>
```

קיבלנו רשימה של כתובות ה-IP של הרכיבים שדריכם עברה הודעה שלחנו אל Google.³



³בגלל הדרכו בה עובד האינטרנט, הדרך זו נכונה עבור הודעה מסוימת אך עשויה להשתנות עבור הודעה אחרת. על כך נעמיק בהמשך הספר.

tracert מודיע על-path של ההודעה אם המסלול ארוך יותר מ-30 קפיצות (hops). המרחק בין מקור ליעד נמדד לעיתים על ידי מספר הקפיצות ביניהם (כלומר, כמה רכיבים שההודעה עברה בדרך מהמקור ליעד).

כמו כן, הכל**י tracert** מציג מדידות של זמן ב-MS (מילי שניות) שלוקח להגיע מהמקור אל כל רכיב בדרך ובחרזה ממנו. עבור כל אחד מהרכיבים, מוצגות שלוש מדידות כדי להציג את האמינות של המדידה. ההבדלים בין משל הזמן שלוקח להגיע לכל רכיב, יכולים להעיד על המרחק הגאוגרפי בין הרכיבים השונים. למשל, אם נראה לפטע הפרש גדול מאוד בין זמנים, נוכל לשער שעברנו ישתת. עבור הפרשים קטנים במיוחד, נראה שהרכיבים סמוכיםיחסית זה זה.

נסו בעצמכם להריץ את הכל**י tracert** בצד' לגלוות את המסלול שהודעה עוברת מהמחשב שלכם אל  Facebook.

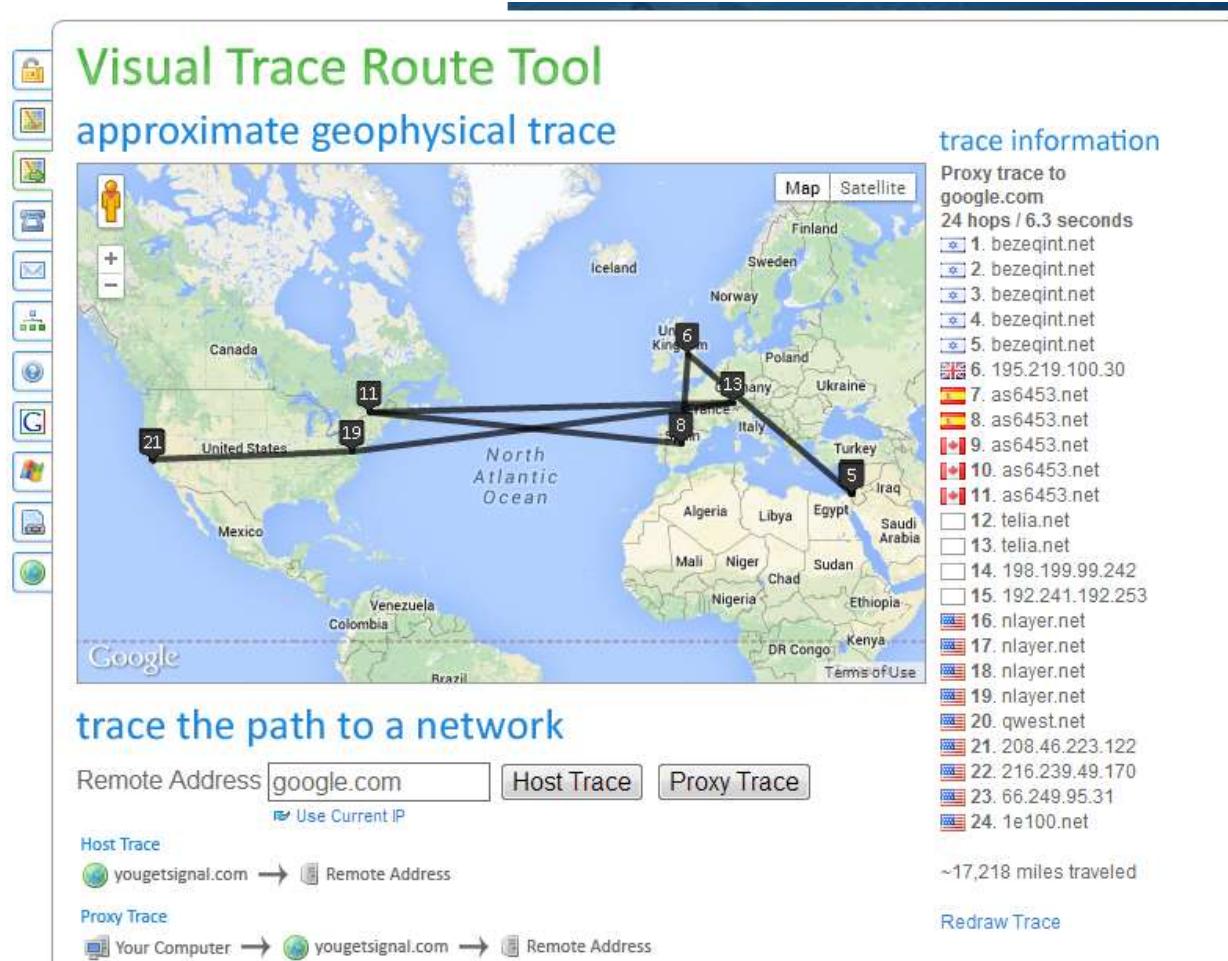
במהמשך הספר, נלמד להבין איך הכל**י tracert** פועל מאחורי הקלעים, וגם תבנו כלי זהה בעצמכם!



תרגיל מודרך - מציאת הדרכן בה עוברת הודעה על מפת העולם

הכרנו עד כה שני כלים - GeoIP ו-**tracert**. מה יקרה אם נשלב בין השניים? נוכל לראות את המסלול של הודעה על מפת העולם! פעולה זו נקראת **traceroute visual visual**. היכנסו אל האתר <http://www.yougetsignal.com/tools/visual-tracert> ובחירה באפשרות Remote Address. כתעת, צפיה להיווצר לפנייכם מפה שתראה את הדרכן שההודעה שלכם עברה ⁴:

⁴ למעשה, הודעה עוברת מהמחשב שלכם ראשית אל האתר [yougetsignal.com](http://www.yougetsignal.com), וממנו נשלחת אל Google.



לאחר שהציגם את הדרכ שעה הודעה מהמחשב שלכם אל Google, השתמשו באתר זה ומצאו



בעצמכם את הדרכ שעוברת הودעה בין המחשב שלכם לבין האתרים הבאים:

- www.berkeley.edu (California)
- www.mit.edu (Massachusetts)
- www.vu.nl (Amsterdam)
- www.ucl.ac.uk (London)
- www.usyd.edu.au (Sydney)
- www.u-tokyo.ac.jp (Tokyo)
- www.uct.ac.za (Cape Town)

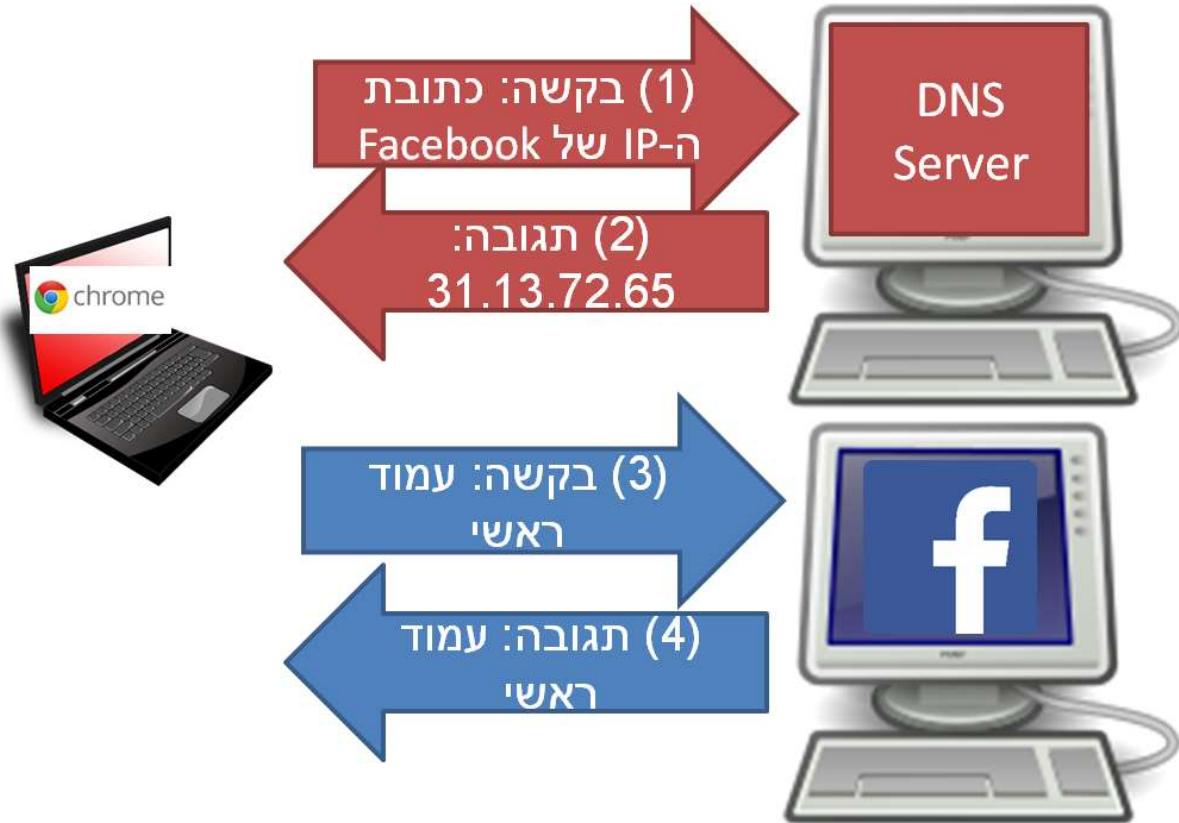
באיו מדינות עברתם בדרך לכל אתר?

DNS

קודם לכן, דיברנו על כתובות IP, ואמרנו שallow הכתובות בעזרתן הודיעות נשלחות באינטרנט. עם זאת, בטור משתמשים, כמעט ולא נתקלנו בכתובות IP. על מנת לזהות שירותי האינטרנט, השתמשנו בעיקר בכתובות מהצורה "www.facebook.com". כתובות אלו נקראות **שמות דומיין (Domain Name)**, או **שמות מתחם**. הסיבה העיקרית לכך שימושם בשמות דומיין, היא שלبني אדם קל יותר לזכור אותו מאשר שימוש כתובות IP.

הדבר דומה בדרך בה אנו שומרים מספרי טלפון: כאשר אנו מתקשרים לאדם, בדרך כלל נעדייף שהטלפון יזכיר לנו את המספר שלו, וכך נעדייף לחזיכר את המספר 054-1234567. עם זאת, שם השם שטלפון צריך בסופו של דבר להשתמש במספר כדי להתקשר למשה, וכך גם המחשב צריך לדעת את כתובת ה-IP של מחשב היעד כדי לפנות אליו. לשם כך, יש צורך בדרך לתרגם בין השניים: בין השם "משה כהן" למספר 054-1234567, או בעולם האינטרנט: בין שם הדומיין "www.facebook.com" לכתובת ה-IP הרלבנטית, למשל 31.13.72.65.

באינטרנט, הדרך לתרגם שמות דומיין לכתובות IP היא באמצעות המערכת **DNS (Domain Name System)**. במקרה זה, בין שמות דומיין אל כתובות IP, מתרחשת בכל פעם שאנו מציינים שם דומיין - בין אם כשאנו גולשים לאתר בדף, ובין אם כשאנו משתמשים בכלים כמו **tracert** או **ping**. המחשב מבין בשלב ראשון מה היא כתובת ה-IP של היעד, ורק לאחר מכן שולח אליו את הודעה. כך למשל, כאשר אנו גולשים אל Facebook, המחשב קודם צריך להבין מה היא כתובת ה-IP של האתר Facebook, ולאחר מכן לבקש ממנו להציג את העמוד:



שים לב כי הבקשה למציאת כתובת IP (הmoצגת באדום בשרטוט) נשלחת אל שרת-h-DNS, בעוד בקשה העמוד (הmoצגת בכחול בשרטוט) נשלחת אל כתובת-h-IP של Facebook. על הדרכ שבה עובדת מערכת-h-DNS, נלמד בהמשך הספר.



תרגיל מודרך - תרגום בין שמות דומיין לכתובות IP

על מנת להשתמש במערכת-h-DNS כדי לתרגם שמות דומיין לכתובות IP, נוכל להשתמש בכלי **nslookup**. הריצו את שורת הפקודה, אתם כבר יודעים כיצד לעשות זאת.Cut, הריצו את הפקודה הבאה:
nslookup www.google.com

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>nslookup www.google.com
Server: box_privatebox
Address: 192.168.14.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4001:803::1014
           173.194.112.212
           173.194.112.211
           173.194.112.208
           173.194.112.210
           173.194.112.209

C:\Users\USER>

```

על המסק מוצגות כתובות IP שונות ל-Google, למשל - 173.194.112.212.⁵ נסxo לגלוש אל כתובת ה-IP זה בדףן. כמובן, במקרה לכתוב בשורת הכתובת "www", כתבו ?Google". אם הגעתם אל 173.194.112.212

כעת, השתמשו בכל' nslookup כדי לגלות כתובות של אתרים נוספים. האם כתובות ה-IP שמחזיר זהות לכתובות שמורות כאשר אתם משתמשים בכל' ping עבור אותם אתרים?

⁵ כפי שציינו קודם לכן, כתובת ה-IP עשויה להשתנות. לכן, יתכן שכשתריצו את הפקודה במחשב שלכם, תהיה ל-Google כתובת IP שונה.

איך עובד האינטרנט - סיכום

בפרק זה התחלנו לענות על השאלה - כיצד האינטרנט עובד? ניסינו להבין מה קורה כאשר מקישים בדף את הכתובת "www.facebook.com". הבנו כי הדף הינו תוכנת **חזק**, ששולחת הודעה בקשה אל השרת של Facebook, שבturnו מחזיר **תגובה**. למדנו כי להודעות באינטרנט, בדומה להודעות דואר, יש כתובות מקור וכותבות יעד, ושלכתובות אלו קוראים באינטרנט **כתובות IP**. כמו כן, למדנו שניתן לעיתים להבין מכתובת IP מה המיקום הגיאוגרפי שלה, באמצעות **GeoIP**.

לאחר מכן, למדנו על **ענן האינטרנט**, והבנו שהמחשב שלנו לא מחובר לשירות לאתר של Facebook, אלא לרכיב המחבר לרכיבים אחרים שיצרים רשת של הרבה רכיבים שמעבירים את ההודעה שלנו מהמחשב ועד לשרת המידע. הכרנו את הכלי **traceroute** שהציג את הדרך שעוברת הודעה מהמחשב שלנו אל Facebook, וגעזרנו בcoli **visual traceroute** בצד להציג את הדרך הזה על מפת העולם. לסיום, הבנו שיש צורך בתרגום בין **שמות דומיין**, כמו com, www, אל כתובות IP, דבר הנעשה באמצעות מערכת DNS.

במהלך הפרק, הכרנו מושגים רבים וכן כמה כלים ראשונים. למדנו להשתמש בשורת הפקודה (Command Line), הכרנו את הכלים nslookup , tracert , ping ו-can את visual traceroute . עם זאת, רק התחלנו לענות על השאלות שלנו, וצירנו תמונה כללית בלבד. בעיקר, נפתחו בפנינו שאלות חדשות - איך עובדת מערכת DNS? מה זו בדיקת כתובות IP? מי הם הרכיבים הללו שמחברים את המחשבים באינטרנט, וכייזד הם פועלם? איך עובד traceroute ? איך Facebook מוחזיר לדף? תשובה? על כל שאלות אלו, כמו גם שאלות רבות אחרות, נענה בהמשך הספר.

פרק 2 - תכנות ב-Sockets

בפרק הקודם התחלנו את מסענו בעולם המופלא של התקשרות תוך הצגת אופן הגלישה שלנו באינטרנט. בפרק זה נלך בשלב אחד קדימה וננסח לתפום את המקום של כתבי האפליקציה. עד סוף הפרק, תדעו לכתוב בעצמכם צ'אט בסיסי, ואף שרת שיודע לבצע פקודות בהתאם לבקשתו שהוא מקבל.



שים לב: לאורך הפרק נציג דוגמאות קוד ב-`Python`. عليיכם להקליד ולהריץ שורות קוד אלו במקביל לתהיליך הקרייה, על מנת להבין את החומר באופן מלא.



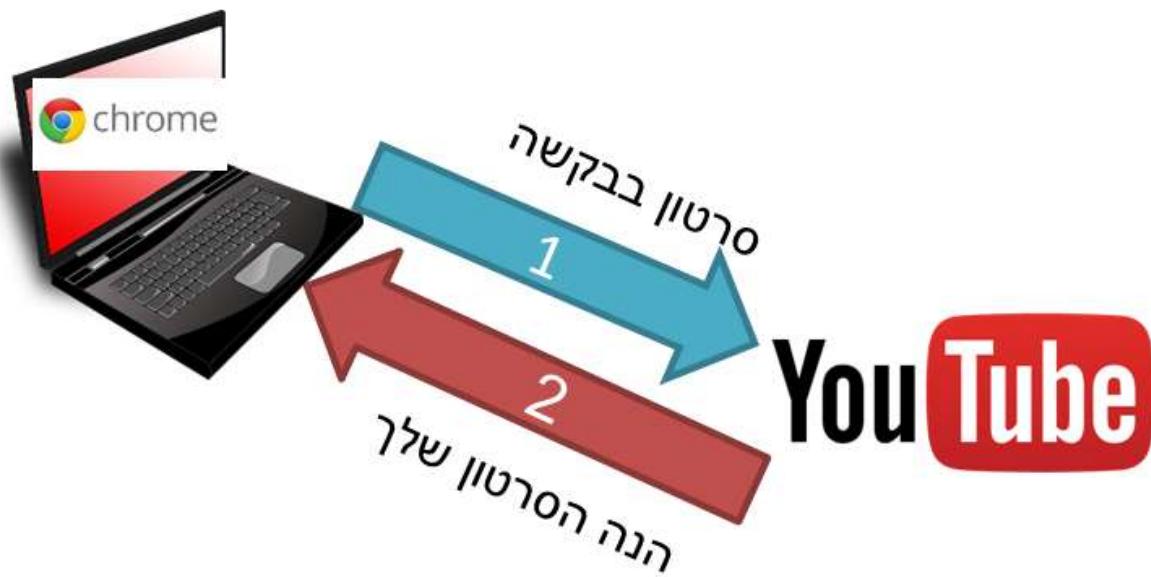
טרם תריצו את דוגמאות הקוד, מומלץ לוודא שהספרייה של `Python` נמצאת ב-`Path` של מערכת הפעלה שלכם. על מנת לעשות זאת, ניתן להיעזר בסרט ההסבר שנמצא כתובות:

<http://data.cyber.org.il/networks/videos/adding-python-to-path.html>



שרת-לקוח

צורת התקשרות הנפוצה ביותר כיום באינטרנט נקראת שרת-לקוח (Client-Server). בצורה התקשרות זו קיים רכיב כלשהו המשמש כשרת (Server), דהיינו מספק שירות כלשהו, ורכיב כלשהו הנקרא לקוח (Client) אשר משתמש בשירות המספק. לדוגמה, כאשר אתם פונים אל האתר של YouTube במקרה לצפות סרטון, הלקוח הוא המחשב שלכם (או הדפדפן שבמחשב שלכם), והשרת הוא אותו שרת של YouTube. במקרה זה, הדפדפן שלכם שולח בקשה לשרת של YouTube, והשרת מחזיר לכם תשובה, כפי שנראהشرطוט הבא:

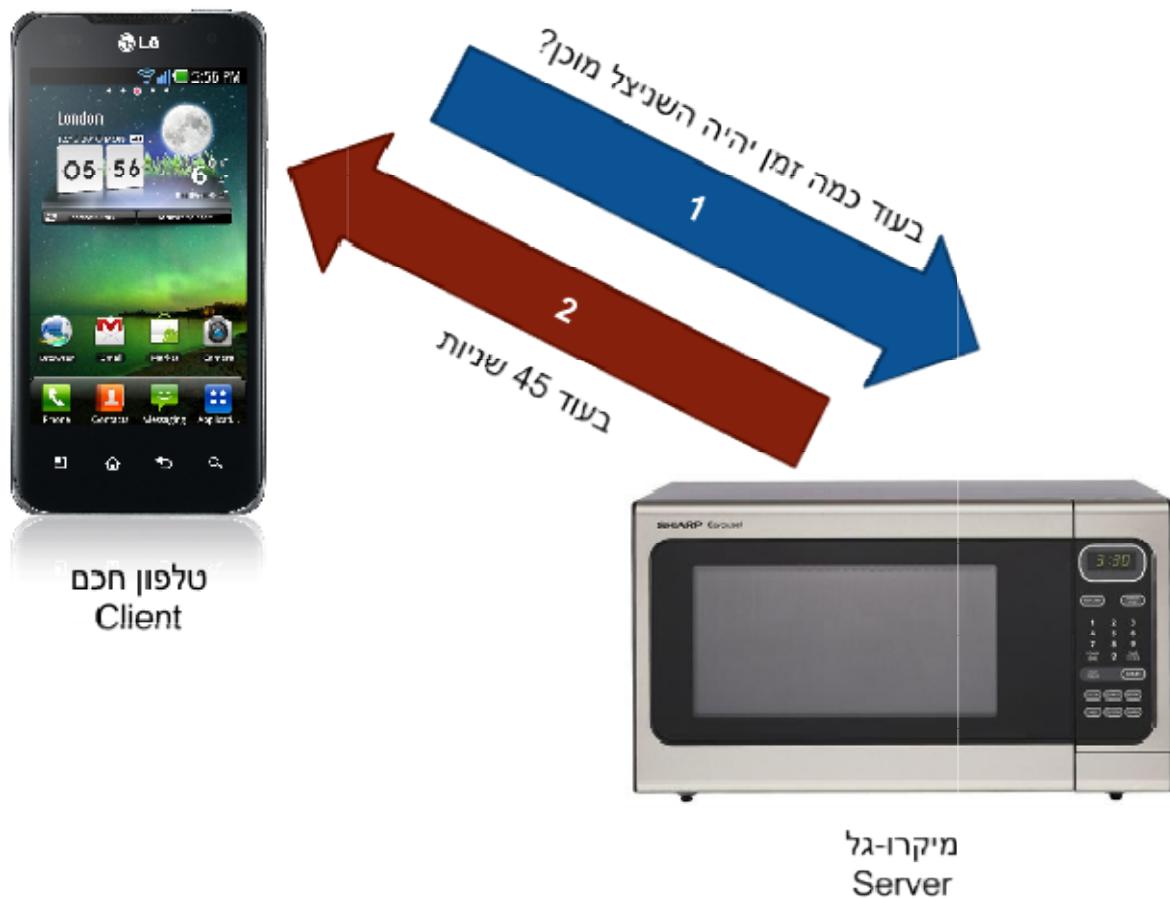


באופן דומה, במידה שאתה גולש מטלפון הנייד שלכם אל כתבה ב-[net](#), הלקוח הימנו הטלפון שלכם (או הדפדפן שבטלפון שלכם), והשרת הינו השרת של [net](#).



מי הלקוח ומי השרת בדוגמה הבאה?

בشرطוט להלן ניתן לראות דוגמה בה טלפוןכם "גולש" למייקרו-גל על מנת לדעת עוד כמה זמן נשאר לחימום של השניצל האחוב. נסו לחשב בעצמכם - מי הלקוח ומי השרת?

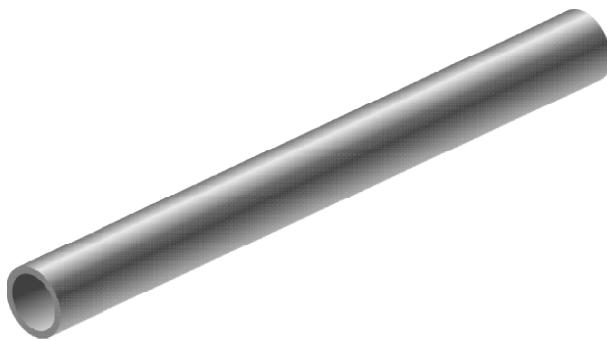


בדוגמה זו, המיקרו-גל הינו השירות, שכן הוא נותן שירות של הערכה של משך הזמן הנותר עד לתום תהליך חיים השניצל. הלוקו הינו הטלפון החכם, אשר משתמש בשירות של המיקרו-גל.

במהלך הפרק, נבנה אפליקציה בתצורת שרת-локוח וביצע זאת את תור שימוש במודול של פיתון בשם **socket**. נכתב בעצמו לkokoh, ולאחר מכן נכתב גם שרת. הלוקו והשרת שנכתבו ידעו לשלחן ולקראן מידע אחד לשני, בדומה לצ'אט בו יש שני משתתפים. לאחר מכן, הלוקו והשרת ידעו לבצע פעולות בהתאם לתקשורת ויהפכו למורכבים יותר ויוטר ככל שהפרק יתקדם.

از מה זה בעצם ? Socket

Socket הוא נקודת קצה של חיבור בין שני רכיבים. כאשר אנחנו רוצים להעביר מידע בין שני מחשבים, אנחנו צריכים לחבר ביניהם, בדומה להעברת מידע בין שני צדדים של צינור. הסתכלו על התמונה הבאה:



מידע שנכנסו מצד אחד של הצינור יצא מצד השני. כל קצה של הצינור נקרא כאמור **Socket**. הצינור יכול לחבר בין תוכנות הנמצאות על אותו רכיב (לדוגמה: Word והדפסן על המחשב שלכם), או בשני רכיבים נפרדים (למשל: הדפסן שלכם והשרת של Google). במיידת שתי תכניות מעוניינות להעביר ביניהן מידע, נקודת הכניסה של מידע למערכת ונקודת היציאה שלו ממנה יcono **Socket**.

על מנת שייהו לנו צינור, אנו זקוקים למספר דברים:

- לבנות את הצינור - את פעולה זו יבצעו השרת והלקוח שלנו.
- התחלה וסוף - לצינור המעביר מים יש נקודת התחלה ונקודת סיום. הדבר נכון גם לגבי צינור המעביר מידע בין שתי תוכנות. על מנת להגיד את נקודת התחלה והסיום, אנו זקוקים לכתובות, עליהם נפרט בעוד רגע קצר.
- גודל - לכל צינור יש קוטר ואורך, לפיהם ניתן לדעת כמה מידע יכול הצינור להכיל ברגע נתון. ה"קוטר" של הצינור שלנו, הלווא הוא **Socket**, הינו בית (byte) אחד - אנו נעביר ב-**Socket** זרם של בתים מצד לצד.

כתובות של **Socket**

כאמור, לכל צינור יש נקודת התחלה ונקודת סיום. עלינו להגיד את נקודות התחלה והסיום של ה-**Socket** שלנו. לשם כך, נצטרך להשתמש במבנה של התוכנה שאיתה נרצה לתקשר. ל-**Socket** ישנו שני מזהים:

- מזהה הרכיב - מזהה את הרכיב (מחשב, שרת וכדומה) שאיתו נרצה לתקשר.
- מזהה התכנית (התהיליך) - מזהה את התוכנה על הרכיב (למשל - שרת המיל, שרת ה-Web) שאיתה נרצה לתקשר.

ניתן לדמות זאת לשיחת מכתב דואר בין שתי משפחות הגרות בשכונה של בתים רבים קומות. מזהה הרכיב הינו מזהה הבניין של המשפחה - למשל "רחוב הרצל בעיר תל אביב, בית מספר 1". מזהה התהיליך הוא מזהה הדירה הספציפית בבניין, למשל "דירה 23".



**מזהה הבניין:
הרצל 1, תל אביב**

ה-Socket מבצע עבורנו את השירות של רשות הדואר - אמו, כמתכנתים, לא צריכים להיות מודעים לכל הפעולות שמתבצעת בראשת בצדיה להעביר את הודעה מצד אחד של Socket (תיבת הדואר של משפחחה אחת) לצד השני של Socket (תיבת הדואר של משפחחה שנייה). בעוד בעולם הדואר הקישור בין משפחחה לבין התיבה הינו כתובות הבניין ומספר הדירה, הקישור בין Socket ובין התוכנה שמנהלת אותו (כלומר – התוכנה שאתחילה את אובייקט ה-Socket הינו כתובת IP ומספר פורט (Port)). את המושג " כתובת IP" הכרנו מוקדם יותר בספר, ואת המושג "פורט" אנו פוגשים לראשונה. את שניהם נזכה להכיר בצורה עמוקה יותר בהמשך. בשלב זה, חשוב שנכיר שפורט יכול להיות מספר בטוח שבין 0 ל-65,535.

תרגיל 2.1 מודרך - הלקווח הראשון שלי

כעת נכתוב את הלקווח הראשון שלנו. לצורך התרגיל, נكتب מראש שרת עימו נוכל לתקשר. השירות נמצא באינטרנט, ושם הדומיין שלו הוא: networks.cyber.org.il. ראשית, علينا למצוא את כתובת ה-IP של השירות. לשם כך, נשתמש בכלи **lookup** כדי שעשינו בפרק [תחילת מסע - איך עובד האינטרנט?](#):

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

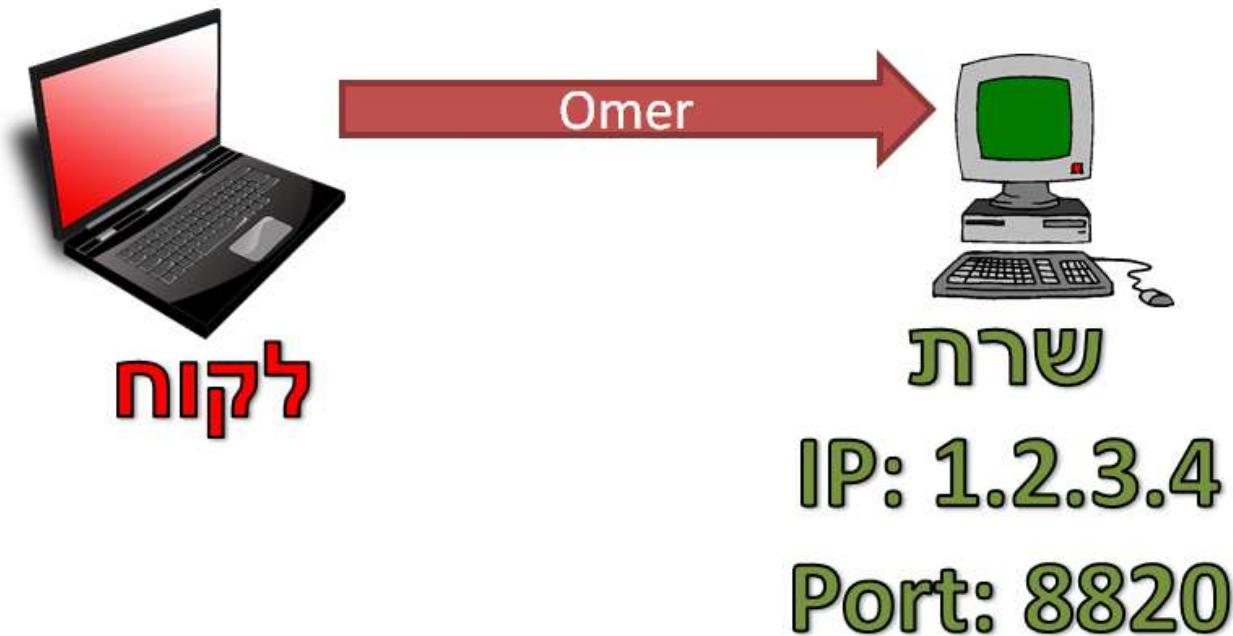
C:\Users\USER>nslookup networks.cyber.org.il
Server: box.privatebox
Address: 192.168.14.1

Non-authoritative answer:
Name: networks.cyber.org.il
Address: 1.2.3.4

C:\Users\USER>

```

לצורך ההסביר, נניח כי כתובת ה-IP של האינטרנט היא הכתובת: "1.2.3.4". שימו לב לשנות את הכתובת זו לכתובת ה-IP האמיתית של השרת, אותה מצאתם באמצעות **nslookup**. על השרת זהה, יש תוכנה שמאזינה בפורט 8820. אנו נתחבר אליו השרת זהה, ונסלח לו את השםשלם (לדוגמא: "Omer"). בהמשך, נכתוב את השרת שישתמש במידע זהה.



נתחילה מכתב ללקוח פשוט באמצעות פיתון. הדבר הראשון שעשינו לעשות לשם כך הוא ליבא את המודול של **socket**:

```
import socket
```

כעת, עליינו ליצור אובייקט מסוג **socket**. נקרא לאובייקט זה בשם `my_socket` זה בשם `my_socket = socket.socket()`

הערה: `socket()` ניתן לתת פרמטרים שכרגע אינם מרחיבים עליהם, אך נעשה זאת בהמשך.

כעת יש לנו אובייקט **socket**. בשלב הבא, נdag ליצור חיבור בין ליבן שרת אחר. לשם כך נשתמש במתודה **connect**. קודם לכן הסבכנו שהכתובות של Socket כוללות כתובת IP ומספר פורט. ובהתאם לכך, המתודה **connect** מקבלת טאפל tuple שמכיל כתובת IP ומספר פורט. לצורך הדוגמה, נתחבר לשרת שכתובת ה-IP שלו היא 1.2.3.4:8820

```
my_socket.connect(('1.2.3.4', 8820))
```

כעת יצרנו חיבור בין התוכנה שמאזינה בפורט 8820 בשרת בעל הכתובת "1.2.3.4". למעשה, יש לנו את החיבור - ועכשו אפשר לשלוח ולקבל דרכו מידע!

על מנת לשלוח מידע אל התוכנה המרוחקת, נשתמש במתודה **send**:

```
my_socket.send('Omer')
```

כאן למעשה שלחנו הודעה אל השרת. שימוש לב שההודעה היא למעשה מחוזצת - ככלומר רצף של בתים. על מנת לקבל מידע, נוכל להשתמש במתודה **recv** (קיצור של receive):

```
data = my_socket.recv(1024)
```

```
print 'The server sent: ' + data
```

ולבסוף, "נסגור" את אובייקט ה-**socket** שיצרנו בכך לחסוך במשאבים:

```
my_socket.close()
```

זהו, פשוט וקל! ראו כמה קצר כל הקוד שכתבנו:

```
import socket

my_socket = socket.socket()
my_socket.connect(('1.2.3.4', 8820))

my_socket.send('Omer')
data = my_socket.recv(1024)
print 'The server sent: ' + data

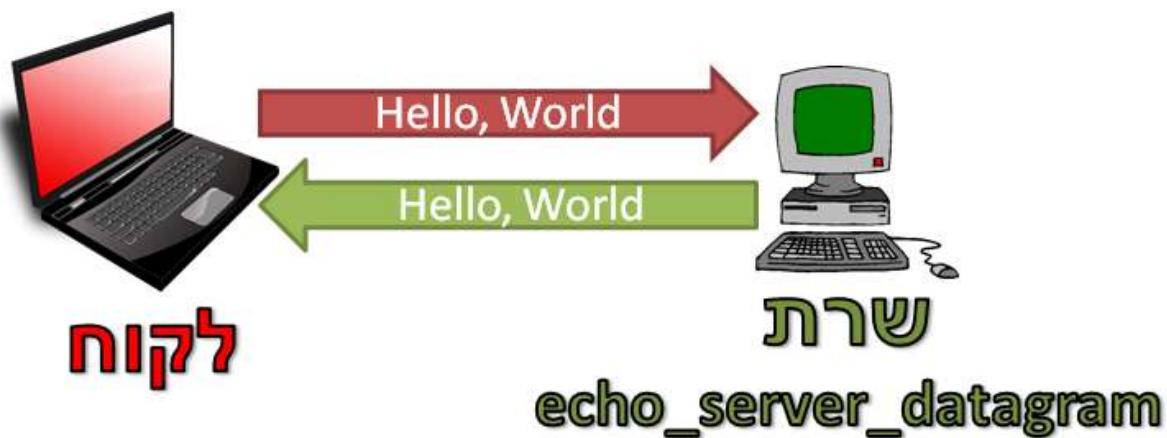
my_socket.close()
```

בשבע שורות קוד אלו יצרנו אובייקט **socket**, התחברנו לשרת מרוחק, שלחנו אליו מידע, קיבלנו ממנו מידע, הדפכנו את המידע שהתקבל וסגרנו את האובייקט. שימוש לב עד כמה נוח לכתוב קוד כזה בשפת פיתון.

תרגיל 2.2 - ל��ח לשרת הדימ: שליחה וקבלת מידע



בתרגיל זה תכתבו בעצמכם ל��ח, כאשר השרת כבר מומש עבורכם. השרת משכפל כל מידע שתשלחו לו, ושולח אותו אליכם בחזרה, כמו הד. כך למשל, אם תכתבו אל השרת את המידע: "Hello, World" (שימוש לב - הכוונה היא למחרוזת), הוא יענה: "Hello, World"



הורידו את השירות מהכתובת: ⁶http://data.cyber.org.il/networks/c02/echo_server_stream.pyc. שמרו את הקובץ **למיקום הבא:**

C:\Cyber\echo_server_stream.pyc

על מנת להריץ את השירות, הכנסו אל ה-**Command Line**, והריצו את שורת הפקודה:
python C:\Cyber\echo_server_stream.pyc

הרגע הרצתם את שירות "הדים" (echo). כאמור, שירות זה מחקה כמו הד את המידע שהוא מקבל. השירות מאמין לחיבורים בפורט קבוע - פорт 1729. כתבו קוד בפייתון המתחבר אל השירות, ולאחר מכן שלחו אליו הודעה. קבלו את תשובה השירות, והדפיסו אותה אל המסמך.

שימוש לבב:

- זוכרים שאמרנו שישיות יכולות להיות באותו המחשב? כאן יש לנו דוגמא מצוינית לכך. השירות והלקוח רצים שניהם על המחשב שלכם. ב כדי לציין זאת, עליים להתחבר אל כתובת ה-IP המינוחת **"127.0.0.1"**. כך הקוד ידע להתחבר אל המחשב שלכם.⁷
- יש להפעיל את השירות לפני שתכתבםコード את הקוד של הלקוח. אחרת, הלקוח ינסה להתחבר אל השירות ולא יצליח, מכיוון שאף תוכנה לא מזינה לפורט 1729 אליו הוא מנסה להתחבר.

⁶ קובץ בסיוםת .py הוא קובץ שמכיל את ה-**code byte** של פייתון. לצורך עניינו, הכוונה היא שלא ניתן לראות בקלות את הקוד המקורי של הסקריפט. אנו מצרפים קבצי **pyc** במקרים בהם נשתמש בעצמו את הקוד של הסקריפט בקרוב מאוד.

⁷ כתובת מיוחדת זו מציינת למשתמש שהמידע ישלח אל כרטיס הרשת הירטואלי Loopback --column העברת מידע שתישאר בצד המחשב (מערכת הפעלה) ולא תצא אל הרשת. נזכיר כתובת IP מיוחדות נוספת בהמשך הספר.

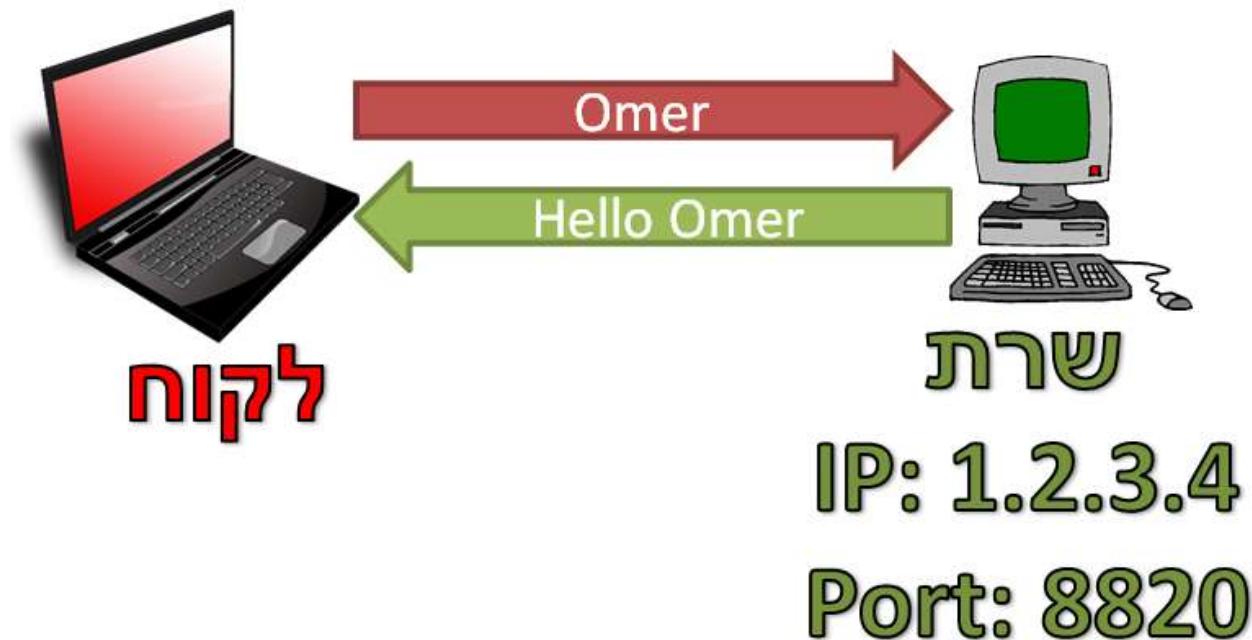
לאחר שהצלחתם לשלוח הודעה לשרת ולקבל ממנה תשובה, כתבו סקריפט פិיתון שմבוקש מהמשתמש הודעה, שלוח אותה לשרת "הדים", מקבל מהשרת את התשובה ומדפיס אותה למשתמש.

dagshim:

- על מנת לקרוא מידע מהמשתמש, תוכלו להשתמש בפונקציה `raw_input()`.
- הפונקציה `recv` היא blocking - כלומר, אם קראתם לפונקציה אך עדין לא הגיע מידע, התוכנית תעצור עד שיגיע מידע חדש.
- במידה שהחיבור הנטנטק, `recv` תחזיר מחוזת ריקה (""). יש לבדוק מקרה זה בקוד.⁸
- חשוב לסגור את ה-`socket` בסוף השימוש (כמובן - לקרוא ל-`close()`). כך מערכת הפעלה תוכל לשחרר את המשאבים שהוא הקצתה עבור ה-`Socket` שיצרנו.

תרגיל 2.3 מודרך - השרת הראשון שלי

از הצלחנו ליצור לקוח שמתחבר לשרת, שלוח אליו מידע ומתקבל ממנו מידע. כעת הגיע הזמן לכתוב גם את השירות. מוקדם יותר, יצרנו לקוח שלוח לשרת את שמו, לדוגמה: "Omer". כעת, נגרום לשרת לקבל את השם שהלקוח שלוח, ולענות לו בהתאם. לדוגמה, השירות יענה במקרה זה: "Hello Omer":



⁸ מקרה זה לא מתאר מצב שבו השירות באמת ניסה לשלוח מחוזת ריקה וקרא למתודה `(send)`. קיבלת מחוזת ריקה מעידה שהחיבור בין הלקוח אל השירות הנטנטק.

הדרך לכתיבת שרת דומה מאוד לכתיבה של לקובץ. גם הפעם, הדבר הראשון שעשינו לעשות הוא ליבא את המודול של **socket** לפניו:

```
import socket
```

כעת, עליינו ליצור אובייקט מסוג **socket**. נקרא לאובייקט זה בשם `server_socket`:
`server_socket = socket.socket()`

בשלב הבא, עליינו לבצע קישור של אובייקט ה-**socket** שיצרנו לכתובת מקומית. לשם כך נשתמש בMETHOD **bind**. מוגדרה זו, בדומה ל-**connect** אותה פגשנו קודם, מקבלת Tuple עם כתובת IP ומספר פורט. נשתמש בה, לדוגמה, כך:

```
server_socket.bind(('0.0.0.0', 8820))
```

בצורה זו יצרנו קישור בין כל מי שמנסה להתחבר אל הרכיב (במקרה זה, המחשב) שלנו (זו המשמעות של הכתובת "0.0.0.0", נרchip על כך בהמשך) לפורט מספר 8820 - אל האובייקט `server_socket`.
 אך יצירת הקישור הזה עדין אינה מספקת. על מנת לקבל חיבורים מלוקחות, נדרש להאזין לחיבורים נוכנים. לשם כך נשתמש בMETHOD **listen**:

```
server_socket.listen(1)
```

שימוש לב שהMETHOD **listen** מקבלת פרמטר מספרי.⁹

בשלב זה אנו מוכנים לחיבורים נוכנים. בעצם, עליינו להסכים לקבל חיבור חדש שmagיע. לשם כך, נשתמש בMETHOD **accept**:

```
(client_socket, client_address) = server_socket.accept()
```

הMETHOD **accept** הינה **blocking** - כלומר, הקוד "יקפא" ולא ימשיך לרווח עד אשר יתקבל בשרת חיבור חדש. כמו שניתן להבין, METHOD **accept** מחזירה Tuple שכולל שני משתנים: אובייקט **socket** שנוצר בעקבות תקשורת עם לקוח שפנה, וtuple נוסף שכולל את הכתובת של הלוקו שפנה אליו. לאחר שלקוח יפנה אל השרת שלנו, נגיע למצב שיש לנו את OBJECT `client_socket` ובאמצעותו נוכל לתקשור עם הלוקוח. לדוגמה, נוכל לקבל ממנו מידע. בהקשר זה, נרצה לקבל את השם של הלוקוח:

```
client_name = client_socket.recv(1024)
```

ונכל גם לשלוח אליו מידע:

```
client_socket.send('Hello ' + client_name)
```

המשמעותזה למשה לקבלת ושליחת מידע בצד הלוקוח, ומשתמש בMETHODS **send** ו-**recv** אשר פגשנו קודם לכן.

כשנסים את התקשרות עם הלוקוח, עליינו לסגור את החיבור איתנו:

```
client_socket.close()
```

⁹ פרמטר זה לא מציין כמה חיבורים ה-**socket** יכול לקבל, אלא כמה חיבורים יכולים לחתות בתור במערכת הפעלה מבלי שביצעו להם METHOD **accept** (למוגדרה זו נגיע בעוד רגע).

כעת, נוכל להתפנות ולתת שירות ללקוח אחר. לחולופין, נוכל לסגור את אובייקט **socket** של השרת:
`server_socket.close()`



שים לב לא להתבלבל בין שני אובייקטי **socket** השונים שיצרנו. האובייקט הראשון שיצרנו, שנקרא `server_socket`, הינו האובייקט של השרת ואליו מתחברים לקוחות חדשים. לעומתו, האובייקט שנקרא בדוגמה `client_socket`, מייצג את התקשרות של השרת שלנו עם לקוח ספציפי.

הנה, הצלחנו לכתוב שרת עובד. להלן הקוד המלא של השרת הפשט שכתבנו:

```
import socket

server_socket = socket.socket()
server_socket.bind(('0.0.0.0', 8820))

server_socket.listen(1)

(client_socket, client_address) = server_socket.accept()

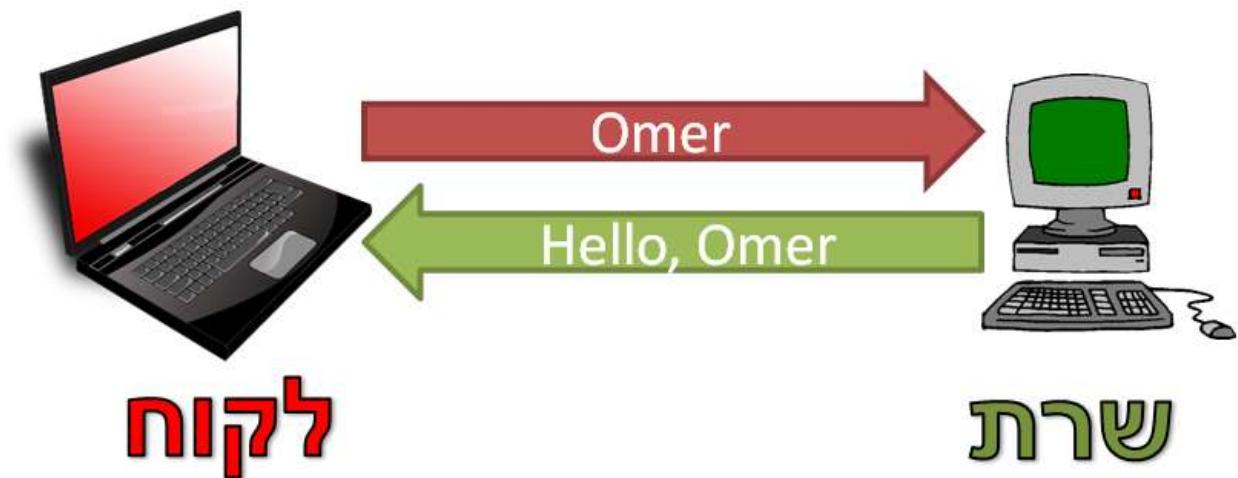
client_name = client_socket.recv(1024)
client_socket.send('Hello ' + client_name)

client_socket.close()
server_socket.close()
```



תרגיל 2.4 מודרך - הרצת שרת ולקוח על מחשבכם

כתבנו לקוח SIMPLE לשרת לנו, ושרת מקבל את השם ומחייב ללקוח תשובה בהתאם, כמפורט בشرطוט הבא:



כעת נבדוק את הליקות והשרות על ידי הריצה שלהם על המחשב שלנו.

לצורך הדוגמה, נאמר ששמרנו את הקבצים במקומות הבאים:

C:\Cyber\client.py

C:\Cyber\server.py

היכנסו ל-Command Line, והריצו את השירות:

```
C:\Windows\system32\cmd.exe - server.py
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>cd \Cyber
C:\Cyber>server.py
-
```

לפני שנריץ את הליקות, علينا לבצע בו שינוי. כזכור, כאשר כתבנו את הליקות, הניחנו שהשרות נמצא בכתובת "1.2.3.4". בהנחה שזו אינה הכתובת של המחשב שלנו, علينا לשנות את הכתובת אליה הליקות מנסה להתחבר. נוכל להשתמש בכתובת "127.0.0.1", כפי שהסבירנו קודם לכן בתרגיל 2.2 - ליקוח לשרת הדימ.

ערכו את הקובץ `client.py`. מיצאו את השורה שבה מתבצעת ההתחברות אל השירות המרחוק:

`my_socket.connect(('1.2.3.4', 8820))`

וחליפו אותה בשורה הבאה:

`my_socket.connect(('127.0.0.1', 8820))`

כעת, הלוקו ינסה להתחבר לפורט 8820 במחשב המקומי, אשר אליו מאזין השירות שכתבנו קודם לכן.
פיתחו Command Line נספף, והריצו את הלוקו:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>cd \Cyber
C:\Cyber>client.py
The server sent: Hello Omer
C:\Cyber>
```

באם הצלחתם, הפלט אמר או היה זהה לזה שבתמונה. נסו עתה לשנות את השם אשר נשלח לשרת ("Omer") לשם אחר. הריצו את הלוקו מחדש, וודאו שגםם מקבלים את הפלט המתאים.

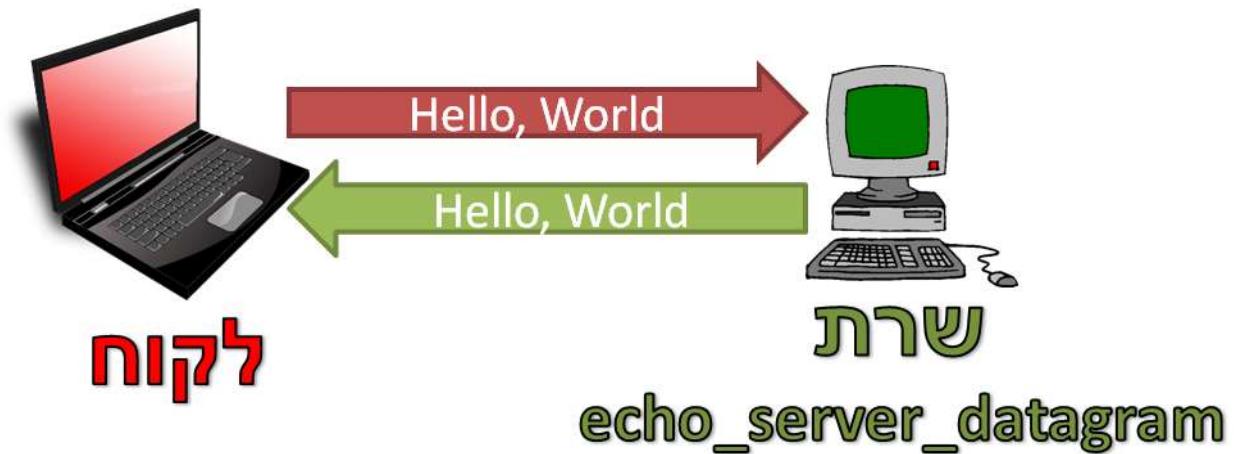
המלצת שובה – כפי שנכתב במבוא לספר זה, מומלץ להשתמש PyCharm עבור עבודה עם פיתון. באמצעותו עורך זה ניתן לעבוד בצורה נוחה יותר עם קוד, וכן לדרג אותו באמצעות breakpoints, דבר הצפוי לשיער לכם רבות בפתרון התרגילים. ניתן להיעזר במצגת על PyCharm מאות ברק גון, הזמינה כתובות הבאה:
<http://data.cyber.org.il/python/1450-3-02.pdf>

תרגיל 2.5 - מימוש שירות הדימ



תרגיל 2.2 - לוקו לשרת הדימ, כתבתם לוקו שהתחבר לשרת מוקן. כעת, באמצעות הידע שצברנו במהלך כתיבת השירות שביבינו קודם לכן, תמשכו בעצמכם את השירות בו השתמשתם בתרגיל זה.

נציר כי השירות משכפל כל מידע שתשלחו לו, ושולח אותו אליכם בחזרה, כמו היד. כך למשל, אם תכתבו אל השירות את המידע: "Hello, World", הוא יענה: "Hello, World".



שיםו לב כי את הלקוח כתבתם בתרגיל הקודם. השתמשו בו. דבר נוסף שחשוב לשים ALSO לב הוא לשימוש במתודה `close`. כשאנו קוראים למתודה זו, אנו מודיעים למערכת הפעלה שיש לנו להשתמש באובייקט **the-socket** שיצרנו, וכך מערכת הפעלה יכולה לשחרר את המשאים הקשורים אליו. בין השאר, אחד המשאים הוא מספר הפורט שמערכת הפעלה הקצתה עבור ה-`Socket`. לדוגמה, בשרת שיצרנו לעיל, מערכת הפעלה הקצתה את הפורט 1729 עבור `server_socket`. אם תוכנה אחרת תבקש להשתמש בפורט זהה, היא לא תוכל לעשות זאת, שכן הוא תפוי. רק לאחר שהפקודה `server_socket.close()` תבוצע, הפורט יחשב שוב "פנוי" ויכול להיות בשימוש ביד' תוכנה או אחר.

לעתים, תוכנה תסייע לרוץ מבל' שהיא קראה למתודה `close`. דבר זה יכול לנבוע במספר סיבות - למשל, מתכנת ש שכח | לקרוא למתודה זו. דוגמה נוספת, שיתכן ותקרו בהמשך בעבודתכם על התרגיל, היא שהתוכנה (או הסקריפט) תקרו לפניה שהקריאה ל-`close` תתרחש. במקרה זה, מערכת הפעלה לא יודעת שהפורט שוחרר, וכן מתייחסת אליו כתפוס. בשלב זה, תוכנית חדשה לא תוכל להשתמש במספר הפורט הזה. דבר זה יהיה נכון, למשל, אם תנסה להריץ שוב את הסקריפט שלכם לאחר שהוא קרט. על מנת להתגבר על בעיה זו, תוכלן לשנות את הפורט בו אתם משתמשים. זכרו לשנות את מספר הפורט גם מצד הלקוח וגם מצד השירות.

בנוסף, לאחר שמתבצעת קראה ל-`close` עשוי לעבור זמן מסוים עד שמערכת הפעלה באמת תשחרר את הפורט. יתכן שתתקלו בכך במהלך העבודה על התרגיל.

כעת אתם מצוידים בכל המידע הדרוש לכם על מנת לפתור את התרגיל. בהצלחה!

תרגיל 2.6 - שרת פקודות בסיסי



בתרגיל הראשון התבוננו לכתוב ליקוי, ובתרגיל השני התבוננו לשרת. בתרגיל זה עליים לכתוב הן את השרת, והן את הליקוי. חלק מהאתגר בתרגיל הינו כתיבת פרוטוקול למשЛОח הודעות בין השרת והליקוי.

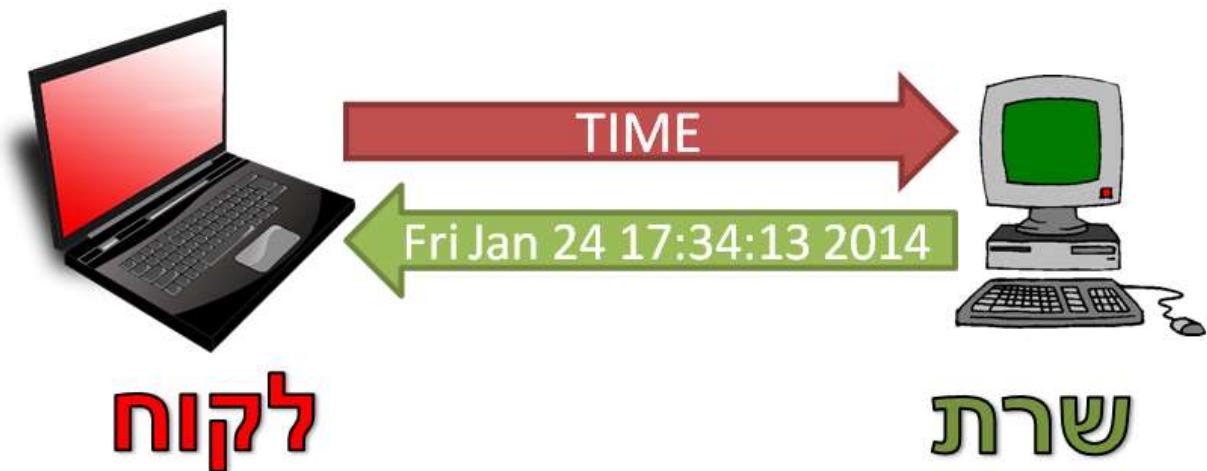
שלב 1:

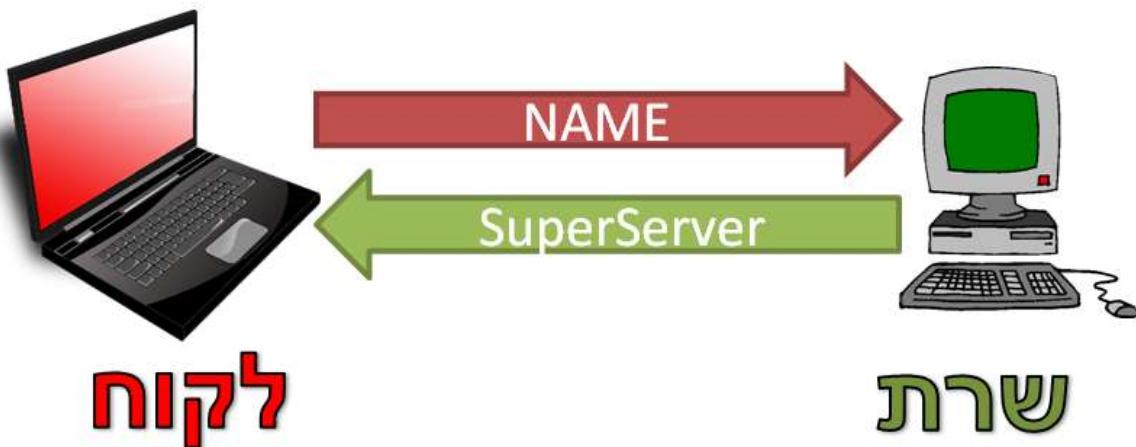
עליכם לכתוב מערכת שרת-ליקוי, כאשר השרת מבצע פקודות שליקוי שלוח אליו, ומחזיר ליקוי תשובה בהתאם.

על כל בקשה של הליקוי להיות באורך של ארבעה בתים בדיק. אורך התגובה יכול להיות שונה בהתאם לבקשתו.
להלן רשימת הביקשות שיש לתרום בהן:

- TIME - בקשה הזמן הנוכחי. על השרת להגיב עם מחרוזת ש כוללת את השעה הנוכחית אצלו.
- היעזרו במודול `time` המובנה בפייטון.
- NAME - בקשה שם השרת. על השרת להגיב עם מחרוזת שמייצגת את שמו. השם יכול להיות כל מחרוזת שתבחרו.
- RAND - בקשה מספר רנדומלי. על השרת להגיב עם מספר רנדומלי שנע בין הערכים 1 ל-10.
- היעזרו במודול `random` המובנה בפייטון.
- EXIT - בקשה ניטוק. על השרת לנתק את החיבור עם הליקוי.

להלן דוגמאות לתקשרות:





שלב 2:

תרגול כתיבת פרוטוקול - האמ בשלב 1 הלקוח שלכם ביצע `socket.recv(1024)`? כתע, צרו פרוטוקול שמאפשר לשולח מהשרת ללקוח הודעות באורך שונה. יכולת זו תשמש אתכם בתרגילים בהם לא תוכלו להניח שהאורח ההודעה מהשרת ללקוח הודעה הוא 1024 בתים או מספר קבוע כלשהו - לדוגמה בהעברת קובץ. אפשרויות לדוגמה:

- השרת ישלח ללקוח הודעה שכותוב בה מה אורך המידע שלו לעליי לקבל בתור תשובה.
- השרת ישלח ללקוח הודעה מיוחדת שמשמעותה "שליחת המידע הסתיימה".

שלב 3:

כיוון שאנחנו עוסקים בהגנת סייבר, علينا לכתוב שרת יציב, ככלומר גם אם הלקוח שלוח "זבל", לשרת שלנו אסור לקרוא. בידקו את יציבות השרת באמצעות הודעות מסוגים שונים.

הנחיות לתרגיל 2.6

לפניהם ניגשים לכתוב את הקוד, בצעו תכנון ראשוני. חשבו מה בדיקת תmansho בצד הלקוח ומה בצד השרת, נסנו לצפות מראש בעיות בהן תתקלו.

בצד הלקוח - ראשית, עליכם לבקש מהמשתמש לבחור באחת מהפקודות שצוינו לעיל (رمز: היעזרו בפונקציה `.(raw_input`).

לאחר מכן, שלחו את הבקשה לשרת, קיבלו את התשובה והציגו אותה למשתמש.
בצד השרת - עליכם לקבל חיבור מהלקוח, להבין את הבקשה שלו ולהגיב אליה בהתאם. לאחר מכן, נתכו את החיבור ותווכלו לספק שירות ללקוח חדש.

בצלחה!

תרגיל 2.7 - שירות פקודות מתקדם - טכני מרוחק (אתגר)



עד כה ביציתם שירותים ולקחוות שתקשרו עם זה עם זה וביצעו פעולות פשוטות. בתרגיל זה עלייכם לתכנן מערכת שרת-לקוח, ולאחר מכן למש אתה. המערכת תאפשר לטכני לתקשר עם מחשב מרוחק ולבצע עליו פעולות שונות. הפעם, עלייכם לתכנן כיצד יראו הפקודות והתשובות, ואין התרגיל מצין זאת עבורכם.

השרת, המחשב המרוחק, יבצע פקודות בהתאם לבקשת הלוקו (הטכני), כמו בתרגיל הקודם. כמובן: הלוקו ישלח פקודה לשרת, השירות יקבל את הפקודה, יעבד אותה, וישלח את התשובה אל הלוקו.

שים לב, לרשותכם קבצים המכילים את שלד התרגיל הן בשרת והן בלוקו. עלייכם למלא בתוכן את הפונקציות, לפי התעוד שנמצא בפונקציות ולפי הדרישה בהמשך.

data.cyber.org.il/networks/server_template.py

שלד השירות:

data.cyber.org.il/networks/client_template.py

שלד הלוקו:

להלן הפקודות שעלייכם למש:

2.7.1

כדי להבין מה מתבצע במחשב המרוחק, הטכני שלנו רוצה לקבל צלום מסך של המחשב המרוחק. עלייכם לתמוך בכך בשרת ובלוקו.

- המשמש בלוקו יכול להקליד פקודות, שהлокו יעביר לשרת
- תימכו בפקודה **TAKE_SCREENSHOT**, שתגרום לכך שהשרת יבצע צילום מסך וישמר את הקובץ במחשב השירות, במיקום לפי בחירתכם.
- מודול **PYTHON PIL** לעובדה עם תמונות – מותקן ייחד עם סביבת גבהים (להתקנה עצמאית-<http://www.pythonware.com/products/pil>).
- השתמשו בקוד הבא כדי לצלם את המסך ולשמור את התמונה לקובץ (הקוד שומר את התמונה למיקום: **C:\screen.jpg**), שנו אותו לפי הצורך:

```
from PIL import ImageGrab
im = ImageGrab.grab()
im.save(r'C:\screen.jpg')
```

2.7.2

צלום המסך נוצר בשרת, אולם כדי שהטכני יוכל להשתמש בו עלייכם לשלוח אותו ללוקו. פקודה **SEND_FILE** חד עם שם הקובץ הרצוי צריכה לגרום לשרת לשלוח את הקובץ המבוקש ללוקו. שימו לב – צילום המסך גדול מדי, חלקו אותו לחלקים ושילחו אותם ללוקו. המזיאו פרוטוקול שיאפשר ללוקו לדעת שהשרת סיים לשלוח אליו את כל התמונה.

2.7.3

לאחר שצפה בתצלום המסך של המחשב המרוחק, הטכניינו שלו נחשד שהקבצים של תוכנה כלשהי לא נמצאים במקומות או שלא כל הקבצים נמצאים. הטכניינו מעוניין להציג תוכן של תיקיה מסוימת במחשב המרוחק. לדוגמה, הצגת רשימת הקבצים שבתיקייה Cyber\Cyber. תימכו בפקודת DIR-. השרת ישלח ללקוח את התוכן של תיקיה מבוקשת. לדוגמה:

DIR C:\Cyber

לחיפוש על פי שמות קבצים, ניתן להשתמש במודול **glob**.

לדוגמה, להציג כל הקבצים בתיקייה Cyber\Cyber, ניתן להריץ:

```
import glob
files_list = glob.glob(r'C:\Cyber\*.*')
```

2.7.4

הטכניינו הגיע למסקנה שאחד הקבצים אינו צריך להיות בספריה ויש למחוק אותו. צרו בשרת וblkoch אפשרות להוראות על מחיקת קובץ כלשהו, באמצעות פקודת DELETE, לדוגמה:

DELETE C:\Cyber\blabla.txt

למחיקת קובץ ניתן להשתמש במודול OS, לדוגמה:

```
import os
os.remove(r'C:\Cyber\blabla.txt')
```

2.7.5

עתה הטכניינו שלו רוצה להעתיק קובץ כלשהו בספריה עליו הוא עבד. הקובץ המבויקש נמצא בספריה אחרת במחשב אליו מתבצעת העתתקה. הוסיף תמייה בפקודת COPY (לדוגמה: העתק את הקובץ 1.txt מ-C:\Cyber\1.txt אל 2.txt מ-C:\Cyber\2.txt). אין הכוונה לשילוח הקובץ אל הלוקוח, אלא לביצוע הפעולה על השרת בלבד. במקרה זה, השרת יחזיר ללקוח האם הפעולה הצליחה או לא. לדוגמה:

COPY C:\Cyber\1.txt C:\Cyber\2.txt

להעתתקה של קבצים, ניתן להשתמש במודול **shutil**.

לדוגמה, להעתיק הקובץ 1.txt מ-C:\Cyber\1.txt אל 2.txt מ-C:\Cyber\2.txt, ניתן להריץ:

```
import shutil
shutil.copy(r'C:\1.txt', r'C:\2.txt')
```

2.7.6

הטכניינו שלו רוצה לבדוק שהתוכנה עבדת נכון היטב. תימכו בפקודת EXECUTE אשר תגרום להפעלת תוכנה אצל השרת (לדוגמה - הרצה של תוכנת Word). במקרה צזה, על השרת להגיד ללקוח האם הפעולה הצליחה או נכשלה.

EXECUTE notepad.exe

על מנת להריץ תוכנות, נוכל להשתמש במודול **subprocess**.

לדוגמה, במקרה של הריץ את notepad, נוכל לבצע:

```
import subprocess
subprocess.call('notepad')
```

נשים לב שלעתים נדרש לתת את ה-path המלא של קובץ ההרצה שאנו רצים להריץ¹⁰, למשל כך:

```
subprocess.call(r'C:\Windows\notepad.exe')
```

2.7.7

לבסוף הטכני שלוינו רוצה להודיע לשרת לסגור את החיבור. תימכו בפקודת EXIT, שתגרום לשרת לסגור את הסוקט מול הלקוח.

טיפים לפתרון התרגיל

מספר דברים שכדי לשים לב אליהם:

1) שימושו לב, שכן מבצעים בפייתון (message)socket.send עבר message ריק, לא מתבצעת שלילה כלל. כלומר לא ניתן להסתמך על קבלת מסר ריק בלבד כדי לדעת שהשרת סיים את שליחת הקובץ.

2) שימושו לב שבפייתון, על מנת לייצג מחוזות שכוללת את התו backslash ("\"), נדרש לכתוב את התו פעמיים – כלומר "\\". זאת הייתה שהתו backslash יכול להיות חלק מנתויים מיוחדים, כגון "ח" שמסמל התחילת של שורה חדשה. לחופין, ניתן להשתמש באזot z לפני הגדרת המחרוזת. שימוש לב לדוגמה הבאה:

```
>>> not_what_we_mean = "C:\file.txt"
>>> not_what_we_mean
'C:\x0cile.txt'
>>> what_we_mean_1 = "C:\\file.txt"
>>> what_we_mean_1
'C:\\file.txt'
>>> what_we_mean_2 = r"C:\\file.txt"
>>> what_we_mean_2
'C:\\file.txt'
>>> what_we_mean_1 == what_we_mean_2
True
```

¹⁰ דבר זה נכון כאשר קובץ ההרצה אינו נמצא במשתנה הסביבה Path של מערכת הפעלה. תוכל לראות את הסרטון הבא כדי להוסיף תכניות ל-Path במידה ותרצו:
<http://cyber.org.il/networks/videos/adding-python-to-path.html>

- (3) במידה שאתם מכירים את השימוש ב-Exceptions בפייתון, מומלץ להשתמש בהם.
- (4) מידע נוסף על המודולים בתרגיל זה ועוד, תוכלו למצוא בספר ה-*Python* של מטה הס"בר הצה"ל, שנכתב על ידי זהר זילברמן ומצוא בכתובת: <http://data.cyber.org.il/python/python.pdf>

תרגיל 2.8 – קרסולת הפורטים (אתגר)



เครดיט: **איל אבני**



הגיע הזמן לעוף על גבי קרסולת הפורטים! עד כה כתבנו שרת ולקוח שעבדו מעל socket עם פורט קבוע. כעת, "נסחרר" קצת את העניינים.

עליכם לכתוב שרת ולקוח, כך שלאחר כל שליחה של מידע - הם יחליפו מספרי פורטים, וגם יתחלפו בתפקידים. דוגמה:

צד A' מתחילה בטור שרת, שמאזין בפורט כלשהו, נניח 8111.

צד B' מתחילה בטור לקוח, שמתחברת לצד A' בפורט 8111, ושולח לו הודעה. ההודעה היא מחזוזת כלשהי שהמשתמש הדין (`input_raw`). בסיום ההודעה שולח הצד B' מספר פורט, לדוגמה 8055, ומתנתק.

צד B' הופך להיות שרת ופתחת האזנה לפורט 8055.

צד A' הופך להיות לקוח, מתחבר לצד A' בפורט 8055, שולח לו הודעה תשובה, מחזוזת תשובה מאות המשמש. ביום ההודעה שולח הצד A' מספר פורט, לדוגמה 8071, ומתנתק.

וחזר חלילה על השלבים הקודמים, עד שהמשתמש מזמן הודעת "exit".

חישבו על כל מקרי הקצה שעשויים לקרות, וכיתבו קוד שמסביר מה המקרים הללו וכי怎ן התמודדותם איתם. כמובן שאת כל ההתכוותות יש להציג על המסך, כולל הפורט שבו נעשה קריאת שימוש. דוגמה:

Side A listening to port 8111

Side B connecting to port 8111

Side B: Hello

```

Side B disconnected
Side B listening to port 8055
Side A connecting to port 8055
Side A: Hi There
Side A disconnected
Side B listening to port 8071

```

הצלחتم? יפה מאד. כתת חיזרו על התהילה, אך בלי להמתין לכך שהמשתמש יזין הודעה. כמובן, צרו הודעה קבועה שעוברת בין צד א' לצד ב' וזרה, ובידקו כמה מהר אתם יכולים להעביר אותה. גירמו לתוכנה שלכם להיות מהירה, נסו לבדוק גבולות ולהבין אם הדברים מפסיקים לעבוד ומדוע.

תכנות ב-Sockets - סיכום

בזאת הגיעו לשינויו פרק תכנות ב-Sockets. במהלך הפרק למדנו מהו מודל שרת-לקוח, והסבירנו מהו Socket לאחר מכן, בנינו יחד ל��ן ושרת ראשון. בהמשך, מימשTEMם בעצמכם ל��ן ושרת "הדים", פיתחתם שרת פקודות בסיסי וכן שרת פקודות מתקדם. לאורך הפרק הצלחتم ליצור מספר מימושים במודל שרת-לקוח, ותרגלתם כיצד יכולות הפיתוח שלכם והן עבדה מול Socket.

כעת, יש ברשותכם כל עוצמתו. אתם יכולים להשתמש ב-Socket כדי לתקשר מנקודת קצה אחת לנקודת קצה שנייה, ולכתוב כל שרת-לקוח שתרצו. עם זאת, למדנו רק חלק מהאפשרויות שנותן לביצוע באמצעות Socket. לא דיברנו על סוגי שונים של Sockets, וכן ראיינו רק מודל שבו יש שרת יחיד ולקוח יחיד. עדין לא ראיינו כיצד ניתן למשר שרת שמספק שירותים למספר לקוחות במקביל. על זאת ועוד, נלמד בפרק הבאם. הידע שרכשנו בפרק זה יסייע לנו בהמשך הספר ללמידה קונספטואליים חדשים, לראות דוגמאות וכן לכתוב בעצממו קוד שימושו אותן.

פרק 3 - Wireshark ומודל חמש השכבות

בפרק הקודם השתמשנו ב-Sockets כדי לתקשר בין שני מחשבים שונים. בתור מתכנתים, היה לנו מאוד נוח להשתמש ב-Sockets על מנת לדבר עם מחשב מרוחק; חוץ מלתת את כתובתו של המחשב המרוחק, הפורט שלו הוא מזמין, להתחבר ל-Socket המרוחק ולשלוח אליו מידע - לא היינו צריכים לעשות כלום. אך עליינו לזכור ש-Socket הוא בסך הכל ממשק שמאפשר לנו לתקשר בקלות וmpshtן לנו את כל התהליך התקשורתי שקרה בפועל (כלומר - איזה מידע עבר בראשת כשאנו משתמשים ב-Sockets). אז מה קורה בפועל כשאנו מדברים עם מחשב מרוחק? בכרעון פרק זה.

בפרק הקרוב נבין טוב יותר כיצד נראה התעבורת שיצאת מכרטיס הרשת שלנו בדרך אל מחשב אחר בעולם או כניסה אליו, ותוך כדי נציג את מודל חמש השכבות (מודול לוגי שמחולק את הפעולות של מערכת תקשורת חמישה חלקים שונים) ומדווע ציריך אותו. בתחילת הפרק נראה שימוש בסיסי בכל חזק מאוד שנראה Wireshark, שיאפשר לנו לחקור ולהבין מה זה בדיקת "המידע שעובר דרך כרטיס הרשת שלנו", ולגלות דברים חדשים שלא היינו יכולים לגלו ללא התוכנה. אם יש לכם שאלות נוספות על Wireshark – שמרו אותן להמשך הפרק, שם נציג שימושים מתקדמים יותר.

פרק זה יהווה מבוא ויעזר לכם להבין את שאר הפרקים בהמשך הספר, שיסתמכו עליו. שימושם לב שעדין לא נרד לעומק של כל נושא, אלא ניתן סקירה כללית כיצד בניו כל עולם התקשורות בראשת האינטרנט, ורק בפרקים הבאים נסביר ביתר פרטים על כל נושא ונושא. בינתיים, תצאו לרכוש כלים שיאפשרו לכם לבחון את עולם תקשורת האינטרנט בצורה שלא הכרתם לפני כן!

מודל חמש השכבות

בפרק הראשון הבנו כמה ענן האינטרנט הוא גדול ומורכב, והוא מכיל אינספור **ישויות (Entities)**. ישות בראשת היא כל דבר המחבר לרשת - בין אם זה סמארטפון, מחשב נייד, שרת של Google, רכיב רשת שנמצא בדרך בין ישויות אחרות, או רכיב בקרה של תחנת כוח המחבר גם הוא לרשת לצורך שליטה מרוחק. העברת המידע בין כל הישויות הללו זו משימה כלל לא פשוטה: קציבי התקשורות הגבוהים, מספר המשתמשים הרבה בהם צריך לתרmor בו זמןנית, והכמויות העצומות של המידע העובר בכל רגע נתון באינטרנט וצריך לחצות את הגלובוס כדי להגיע לצד השני של העולם - כל אלו הם רק חלק מהאתגרים שאתם צריכים להתמודד ענן האינטרנט.



כיצד ניתן לארגן את כל המידע הזה?

נשאלת השאלה: איך אפשר לארגן את כל המידע הזה, כך שיאפשר למערכת המורכבת זו לעבוד – ולעבוד בצורה טובה?

ובכן – על השאלה זו ניסו לענות רבים וטובים, אך ברור שאם כל אחד יתן את פתרונו נגיע למצב שבו כל ישות יודעת לדבר ב"שפה שלה" ואין אף "שפה משותפת" לכל רכיבי הרשת בעולם.

אך מה קרה בפועל? נוצר מצב בו הרבה יצירניות חומרה שיזוקו מכשירים אשר תומכים בכך ורק בכך אחד ספציפי (תקן אשר החברה עצמה יצרה), מה שחייב את הלוקוחות להמשיך ולרכוש מוצרים נוספים מאותה היצרנית אם הם היו רוצים לתקשר בין שני מכשירים שונים. ברור כי המצב הזה אינו רצוי, והוא אינו מאפשר למכשירים שונים באמצעות לדבר ב"שפה אחידה" המשמשת את כל רשת האינטרנט.

כדי לפתור את בעיה זו וליצור סטנדרטיציה (תקנון) של המידע העובר על גבי רשת האינטרנט, יצר ארגון התקינה הבינלאומי (ISO = International Organization for Standardization) שאחראי על פיתוח ופרסום של תקנים בינלאומיים (internationals) את מודל שבע השכבות (המכונה גם מודל שבע הרמות). מטרתו של מודל זה, שנקרא OSI (Open Systems Intercommunications), הינה לתת קווים מנחים כיצד צריכה להיות תקשורת בין כל מערכת מחשב אחת לשניה, ללא תלות ביצן של אותה מערכת.

שימוש לב: מודל השכבות הינו מודל, ולא תקן. הוא לא מחייב את מערכות התקשות לדבר בצורה זו אחת עם השנייה, אלא מנחה את יצירניות החומרה כיצד למשם את מערכות התקשות כך שתהייה אחידות בין כלם. כפי שתראו, בספר זה לא נעשה שימוש במודל שבע השכבות אלא במודל חמש השכבות¹¹.



מה זה בעצם אומר מודל של שכבות?

לפנינו שסביר את משמעות השכבות בעולם המחשבים, השתמש בדוגמה מהחיים עליה נחיל את מודל השכבות כדי להבין על מה מדובר. ניקח למשל מערכת מורכבת כמו טיסה בשדה תעופה: כיצד תוכלו לתאר את התהילה שעובר אדם מהרגע שמנגין לשדה התעופה במדינת המקור ועד שיוצא משדה התעופה במדינת的目的? דרך אחת לעשות זאת היא לתאר את הפעולות שהוא עושה (או שעושים בשבילו) בצורה כרונולוגית: בידוק ביטחוני והפקדת

¹¹ ISO יצרו את מודל שבע השכבות באופן תיאורתי. מודל חמש השכבות (לעיתים מכונה Protocol Stack) נוצר לאחר העבודה עם רשת האינטרנט, מתוך השימוש היישומי, והוא דומה למודל שבע השכבות אולם הוא מותר על שתי שכבות (השכבה החמישית והששית) שבפועל התגלו כמיותרות ولكن הושמו מהמודל.

מצוודות, החתמת דרכון ועלייה למטוס (Boarding). מרגע שהמטוס המרייא, הוא מנוט את דרכו ונוחת במדינת היעד. לאחר מכן הנושא יורד מהמטוס, מחתים שוב את הדרכון ומתקבל בחזרה את המטען שלו.

משמעותו לב שהשתמשנו כאן באנלוגיה וכן חלק מהפרטים עשויים להיראות "מאולצים" כדי שייתאימו למודל, אך הדבר החשוב הוא להבין את הרעיון הכללי, כפי שמתואר באIOR הבא:



השלבים השונים בשדה התעופה

אם נבחן שוב את התהליך, נגלה שהוא מורכב ממספר שלבים, כאשר כל שלב מופיע הן בחלק הראשוני של התהליך (במדינת המקור) והן בחלק השני של התהליך (במדינת היעד). יש פונקציה שלמצוודות (במדינת המקור – הפקדה, במדינת היעד – קבלת), פונקציה של החתמת דרכון (במדינת המקור – חתימה יוצאה, במדינת היעד – חתימה כניסה), וכן הלאה.

בצורה זו ניתן להסתכל על התהליך כצד הבניי מספר שכבות, כפי שמתואר באIOR הבא:



השלבים השונים בשדה התעופה, הפועם בחלוקת לשכבות מוגדרות

האיור הנ"ל מספק לנו תשתיית כדי שנוכל לדבר על המבנה ממנו בנוו' התהילך המורכב של טישה. נשים לב שככל שכבה, יחד עם כל השכבות מתחתיה, מספקת שירות כלשהן.

אם נסתכל על הקשר האופקי בכל שכבה, נראה שככל משכבה שמספקת פונקציונליות כלשהי מتبוססת אך ורק על השכבות שמתוחתיה כדי להשלים "מסלול מלא אל היעד", ומוסיפה פעולות הקשורות לשכבה הספציפית. למשל:

- בשכבת השערים, החל משער העלייה למטווי ועד לשער הירידה מהמטוים, משתמשים בשירות העברת המטוים ממסלול ההמראה למסלול הנחיתה שמספק עלי-ידי השכבות שמתוחת, ובנוסף דואגים להעלות ולהורד את הנוסעים דרך השער. בכך מתבצעת העברה מלאה של האדם בלבד מנמל התעופה במדינת המקור לנמל התעופה במדינת היעד.

שימו לב - שכבה זו לא יודעת כיצד הגיעו הנוסעים אל השער, והיא אינה מודעת לעצם קיומם המזווידות או הדרוכונים. היא רלוונטית אך ורק לשעריהם, ורק בהזאה "מבנה". לעומת זאת - שכבה זו אחראית על השערים בלבד, לא מכירה את השכבות שמעליה, ולא יודעת איך השכבות מתחתיה מתפקדות. היא אינה מעוניינת להכיר דברים אחרים שהיא לא אחראית עליהם.

- בשכבת הדרוכונים, החל מהדלתק היוצא לדלפק הנכנס, מתבצעת העברה מלאה של האדם (על-ידי שימוש בשכבות שמתוחת), וכן החתמת הדרון במדינת המקור ובמדינת היעד.

• בשכבת המזווידות ומטה, החל מהפקדת המזווידות ועד קבלת המזווידות, כבר מתבצעת העברה מלאה למדינת היעד של האדם, המטען שלו, וכן החתמת הדרון בשתי המדינות.

בנוסף, ניתן לשים לב שככל שלב מסתמן על השלבים הקודמים לו: החתמת הדרון היא רק עברו מי שהפקיד כבר את המזווידות ועומד לצאת מגבולות המדינה; העלייה למטוים היא רק עברו מי שהפקיד את המזווידות וכבר החתים את הדרון; ההמראה היא רק עברו מי שהפקיד את המזווידות, החתים את הדרון, וכמוון – עלה למטוים. עובדה זו מאפשרת לנו לזרחות סדר מוגדר לתהילך: החל מהשכבה העליונה בצד השולח (דרך כל השכבות התתכנות של הצד השולח) ועד לשכבה העליונה בצד מקבל (דרך כל השכבות התתכנות של הצד מקבל).



למודל של שכבות יש יתרון חשוב שלא דיברנו עליו: **כל שכבה אינה תליה במימוש של שכבה אחרת.**

חשוב אחד יחליטו במשמעותו על מעבר לדרוכונים אלקטронיים. התנהנה של החתמת הדרוכונים עדין תמלא אחריו הייעוד שלה: רישום של אדם היוצא מדינה אחת ונכנס למדינה אחרת. היתרון הגדול הוא שלאiar התהנות בשרשרת לא אכפת כיצד היא עושה את זה - העיקר שהיא תדאג לרשות שהנוסע יצא מדינת המקור כדי שאפשר יהיה להעלות אותו למטווי דרך השער, והוא נכנס למדינת היעד כדי שיוכל להמשיך ולאסוף את

המצוודות שלו. עם זאת, הדבר היחיד שכך צריך לוודא הוא שמדינת היעד יודעת להתמודד עם תיירים בעלי דרכונים אלקטרוניים.

אם נחשוב על שכבת השערים, הרי ש מבחינה, העובדה שהאדם עבר מנגנון לנמל הייתה יכולה להתבצע לא באמצעות מטוס, אלא באמצעות מכונית, אוניה או סוא מעופף. היא אחראית על השערים בלבד. כך שכבה זו לא צריכה להיות מודעת ל **מימוש (Implementation)** של השכבות מתחתיה.

דוגמה נוספת יכולה להיות שליחת המצוודות דרך ספינה במקום בבטן המטוס – אנחנו עדין מספקים את אותו שירות, העברת מצוודות אל מדינת היעד, אולם משתמשים אותו בדרך אחרת.

במערכות מורכבות כמו רשת האינטרנט, שמתעדכנות לעיתים תכופות וمتפתחות מעט לעת, היתרון אותו הצגנו הוא חיוני. גם אם מימוש של שכבה מסוימת ישנה, שאר המערכת לא תושפע ממשינוי זה ונוכל להמשיך לדבר עם יישיות אחרות בראשת כאלו כלום לא קרה. זאת מכיוון שהשכבה עדין תספק את אותו **שירות** לרמות שמעליה (שימוש לב שמדובר על שני **במימוש השירות**, ולא בשני **השירות עצמו**). מאפיין זה של הסתרת המימוש אותו הצגנו כרגע הוא מאפיין חיוני בבניית מודולרי כגון מודול חמש השכבות.

איך זה קורה בפועל?

דיברנו מושא על מטוסים, כתע נשוב לדבר על רשותות. הבנו מדוע בניה מערכת מורכבת שכזו מודול של שכבות, ומה היתרונות שמודול זה מספק. כתע נסביר כיצד מודול השכבות מיושם בעולם הרשותות. כפי שכבר אמרנו, אנו נתעסק במודול חמש השכבות, שמחלק את מימוש מערכת התקשרות לחמש שכבות לוגיות. כל שכבה במודול השכבות מספקת שירות לרמה שמעליה, מבלי לחסוף אותה לאופן בו השירות שהוא מספקת ממומש (ובכך מאפשרת לשכבה שמעליה להתייחס אליה בתור "קופסה שחורה" שבסק הכל מציעה שירות כלשהו).

כשכבה מסוימת (נניח שכבה ח) על ישות אחת רוצה לדבר עם שכבה ח על ישות אחרת, היא עשו זאת בעזרת **פרוטוקול** ששירך לרמה ח.

פרוטוקול - הגדרה

פרוטוקול (Protocol, תקן) הינו סט מוגדר של חוקים, הקובע כלליים ברורים כיצד צריכה להיראות התקשרות בין הצדדים השונים. אם תחשבו על כך, אנחנו מכירים לא מעט תקנים בחיי היום-יום שלנו: השפה העברית, למשל, היא תקן. היא קובעת כללי תחבורה ואוצר מילים המוגדרים מראש, אשר מנחים את שני הצדדים כיצד עליהם לדבר זה עם זה על מנת שיבינו אחד את השני. היזכרו בתרגיגי השרת והלקוח אותם ביצענו

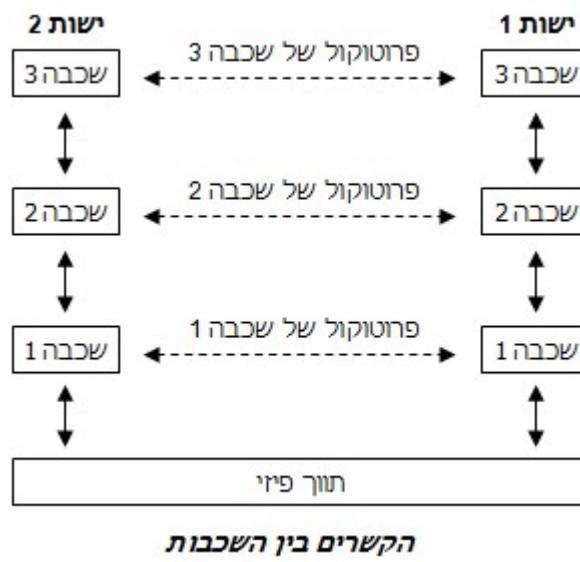
כשמדובר סוקטים: איר, לדוגמה, יודע השירות שהלקוח מבקש לבצע צילום מסך? באמצעות פרוטוקול התקשורת שהגדרתם ביניהם.

גם HTTP, הפרוטוקול המשמש להעברת דפי האינטרנט אליהם אנחנו גולשים בדפדפן, הוא תקין. הוא קבוע כיצד דבר הדפדפן עם השירות המרוחק, ובאיזה צורה יציג חזרה למשתמש את הדף שהגיש לו השירות. למעשה, בפרק הראשון בספר, שראינו שהדפדפן שלוח הודעה בקשה לשרת, והודעה זו הייתה בפועל הודעה בפרוטוקול HTTP, אליו נלמד לעומק בהמשך הספר.

בלי פרוטוקולים היינו מקבלים סיטואציה נסוח "מגדל בבל", בו כל רכיב מדבר בשפה שלו ואחד לא יכול לדבר עם השני. חשבו על שני אנשים שונים שנפגשים, האחד יודע רק סינית והשני יודע רק אנגלית. יהיה להם קשה מאוד לתקשר אחד עם השני בצורה הזאת. כדי שיצלוו לדבר אחד עם השני, עליהם להחליט מראש על "שפה משותפת" אותה שניהם יודעים.

פרוטוקול מחייב את שני הצדדים בשיחה לסת ממויינים של חוקים הקובעים כיצד יראה תהליך התקשורת ביניהם. בצורה זו הם יכולים לדבר ולהבין אחד את השני.

נחזיר לתקשורת בין השכבות. בין שכבה ח בישות אחת לשכבה ח' בישות אחרת אין אף מידע שמעבר לשירות. במקום זאת, כל שכבה מעבירה את המידע שקיבלה (ונתונים נוספים שהוא מוסף, כפי שנראה בהמשך) לשכבה שנמצאת ישירות מתחתיה, עד שmagיעים לשכבה התחתונה ביותר. מתחת לשכבה זו נמצא המידם הפיזי, ורק שם עובר המידע בפועל. ניתן לראות זאת בתרשים הבא, כאשר תקשורת וירטואלית מיוצגת על-ידי קווים מקווקווים ותקשורת פיזית מיוצגת על-ידי קוים רציפים.



אם נשים את המסקנות מהתרשים הנ"ל על הדוגמה של Sockets שראינו בפרק הקודם, נבין את הדבר הבא: בעוד שכל Socket מדבר עם ה-Socket השני בפרוטוקול של אותה שכבה, הוא "חושב" שהוא מדבר אליו שירות (על-ידי שימוש בפונקציות send וrecv) - אולם התקשורת ביניהם היא ווירטואלית, ובפועל היא משתמשת

על העברת המידע לשכבות התחתיות ושימוש בשירות שهن מספקות. המידע יורד עד לכרטיס הרשת, יצא אל הצלב (או כל תווך פיזי אחר) ומצאת דרכו אל היעד – שם הוא נקלט לכרטיס הרשת (כפי שנוכחנו לדעת על-פי מה שראינו ב-Wireshark) וועלה חזרה אל השכבה הרלוונטית.



כיצד בנויה פקטה?

הפקטה, עליה דיברנו מוקדם, היאذاكرة חבילה מידע שעוברת ברשות מקום למקום. מה שלא גענו בו קודם הוא הקשר בין הפקטה לבין מודל חמש השכבות. מה הקשר ביניהם? התשובה פשוטה היא שהפקטה מכילה בתוכה מידע של כל שכבה מודול חמש השכבות שהשתתפה בתהליך התקשרות¹², אבל מה זאת אומרת?

מוקדם יותר בפרק, הזכירנו שבתהליך השילחה כל שכבה מעבירה את הפקטה לשכבה שמתחתה. בסופו של דבר, הפקטה מורכבת ממספר פרוטוקולים הבנויים זה מעל זה, כאשר כל שכבה מוסיף את המידע שלו (הקשר לשירות אותו היא מספקת) לתחלת הפקטה של הרמה שמעליה, ובכך למעשה עוטפת אותה בעוד שכבה. חשבו על זה כמעין משחק של חבילה עוברת – בכל שלב בו בניית החבילה היא נעטפת בעוד שכבות (כאשר כל שכבה לא יודעת מה יש בפנים), ולאחר שחבילה נשלחת ועוברת בין המשתתפים – בכל שלב מקיפים אותה, שכבה אחר שכבה. לתהליך של עטיפת המידע בכל שכבה ושכבה לצד השולח קוראים (כימוס)¹³, ואילו לתהליך קילוף הפקטה מצד מקבל נקרא **Decapsulation** (קילוף).

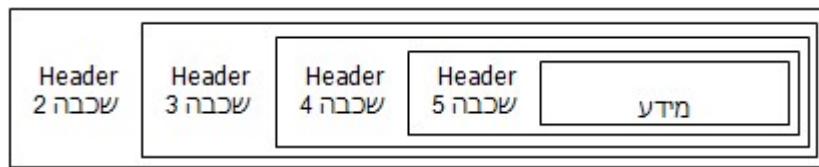
המידע שמוסיפה כל שכבה בתחלת הפקטה נקרא **Header (תחילית)**, והוא מכיל מידע שימושי לשיליטה ובקרה על הפקטה הרלוונטי לשירות שמספקת אותה שכבה (למשל: כתובת ה-IP אליה מיועדת הפקטה, בקורסות שאיות וכו').

האיורים הבאים מתארים כיצד נראה פקטה במודל חמש השכבות, ואיופה נמצא המידע של כל פרוטוקול בפקטה השלמה¹⁴.

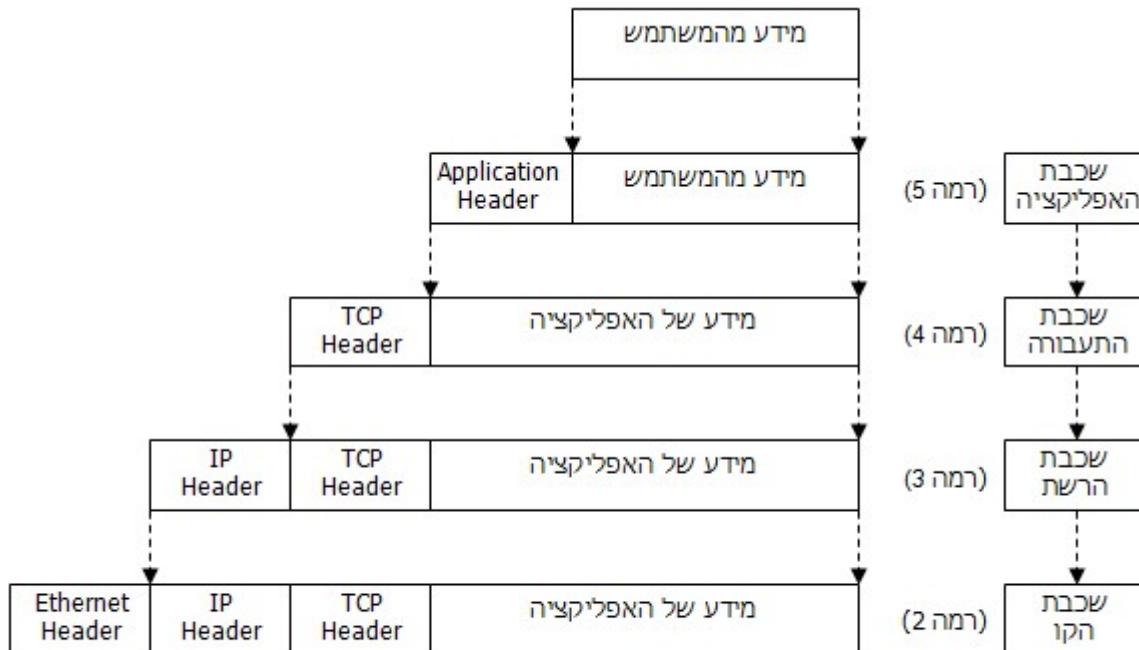
¹² לא כל השכבות חייבות להשתתף בתהליך התקשרות. במקרים מסוימים יש פקטות שמקילות רק את שכבות 3-1, למשל, זה הגיוני לחלוטן כשהשכבות מעליון כל לא הי רלוונטיות לתהליך התקשרות הספציפי בין שני הצדדים.

¹³ למונח **Encapsulation** יש משמעות נוספת בתחום מונחה עצמים: הסתרת שימוש וחשיפת משקל תכונות. שימוש לב לא להתבלבל בין השניים.

¹⁴ בשכבה השנייה בלבד מוסף מידע גם לסופם המסגרת (הוא נקרא **Trailer**), אך התעלמו ממנו במכoon ובחרנו להציג רק את ה-Header כדי לפשט את התרשים. בנוספ, לא כלמן את השכבה הראשונה (השכבה הפיזית), משומם שלרוב נתונים להתעלם ממנו בהסנה.



מבנה פקטה במודול חמם השכבות #1



מבנה פקטה במודול חמם השכבות #2

באיר אחרון, בכל שכבה נתנו דוגמה לפרוטוקול השיר לאוֹתָה השכבה (TCP בשכבה הרביעית, IP בשכבה השלישית ו-Ethernet בשכבה השנייה), אך ברור שאלן לא הפרוטוקולים הייחדים של אותה שכבה. בהמשך הספר נלמד לעומק על כל אחד מהפרוטוקולים הללו.

דבר מעניין שווה לשים עלי'ו דגש הוא שה-Header של כל שכבה (כלומר המידע עצמו, לא ה-Header'ם) זהה לפקטה של השכבה שמعلיה; בתהליך השילוח כל שכבה מקבלת מהשכבה שמعلיה את הפקטה בדיק קופי שהיא, מוסיפה לה את Header על-פי התקן (פרוטוקול) של אותה שכבה ומעבירה אותה להלאה לשכבה שמתוחת. כך למשל, בשכבה השלישית, ה-Header של הפקטה כולל בין היתר את ה-Header של השכבה הרביעית (בדוגמה לעיל, ה-Header TCP). בשכבת הקו, ה-Header של הפקטה כולל את ה-Header של השכבה השלישית, והן של השכבה הרביעית (בדוגמה זו, את ה-Header IP ואות ה-Header TCP).

פירוט חמישה השכבות

ובכן, בדומה לדוגמת המטוסים - נרצה לדעת מה עושה כל שכבה (או למעשה איזה שירות היא מספקת לرمות שמעליה). השכבות במודל חמוץ השכבות הן: שכבה הפיזית, שכבת הקרן, שכבת הרשת, שכבת התעבורה ושכנת האפליקציה. כעת נסקרוו אותן מלמטה למעלה, החל ממה שכבה התחתונה ועד לשכבה העליונה:

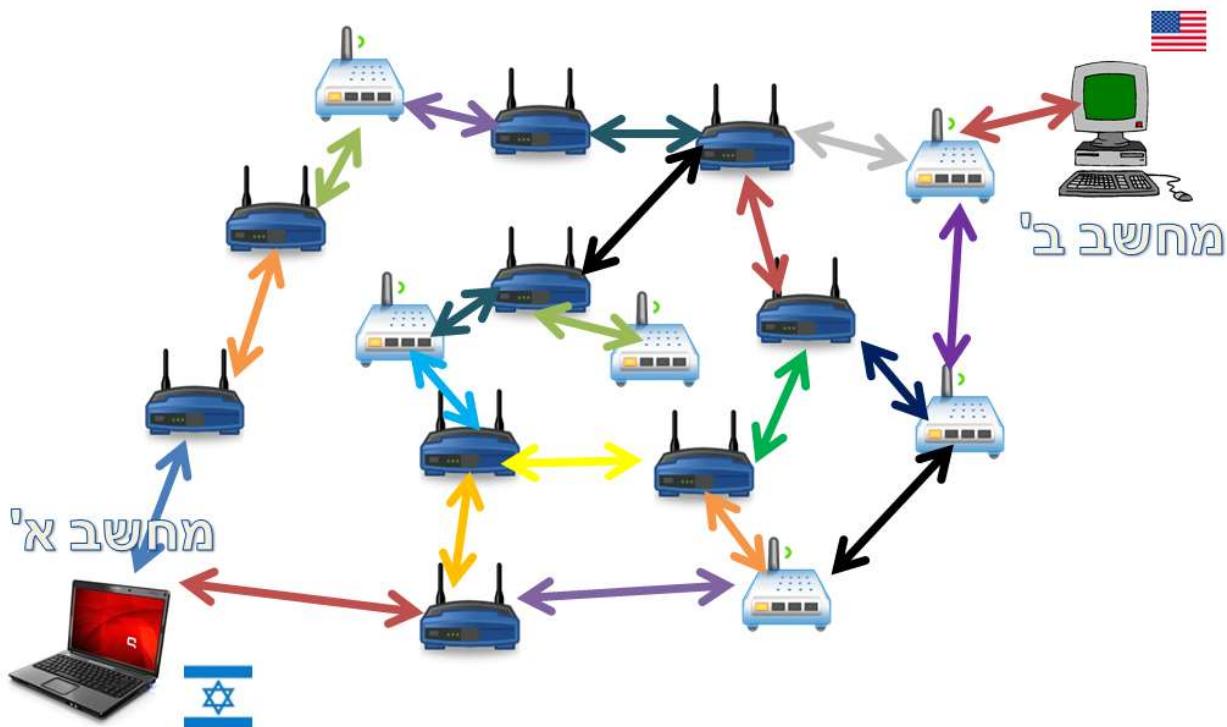
שכבה ראשונה - שכבה הפיזית

תפקידיה של שכבה זו הוא להעביר את הביטים מנקודה אחת לנקודה שנייה עם כמה שפחות שגיאות. השכבה הפיזית רק מעבירה 0 או 1 מצד לצד. שימושה לב שכבה זו אינה מודעת לריצפים של ביטים, פקטות או כל דבר זהה. מבחינתה עלייה להעביר בית אחד בלבד בכל פעם. העברה הפיזית יכולה להתבצע על גבי מגוון של תוכומים: כבלי רשת, סיבים אופטיים, באוואר (אלקטרומגנטיים, לוין) וכו' - העיקרי שהמידע יגיע לידי.

שכבה שנייה - שכבת הקרן

שכבת הקרן מסתמכת על העברה הפיזית של המידע שנעשה ברמה שמתתיתיה, ומאפשרת לנו לדבר עם ישויות אחרות הקשורות אליהן.

באיזור שלפנינו תחום האחוריות של שכבה השנייה מתבטא בכל חץ צבעוני שמקשר ישות רשת סמוכות אחת לשניה:



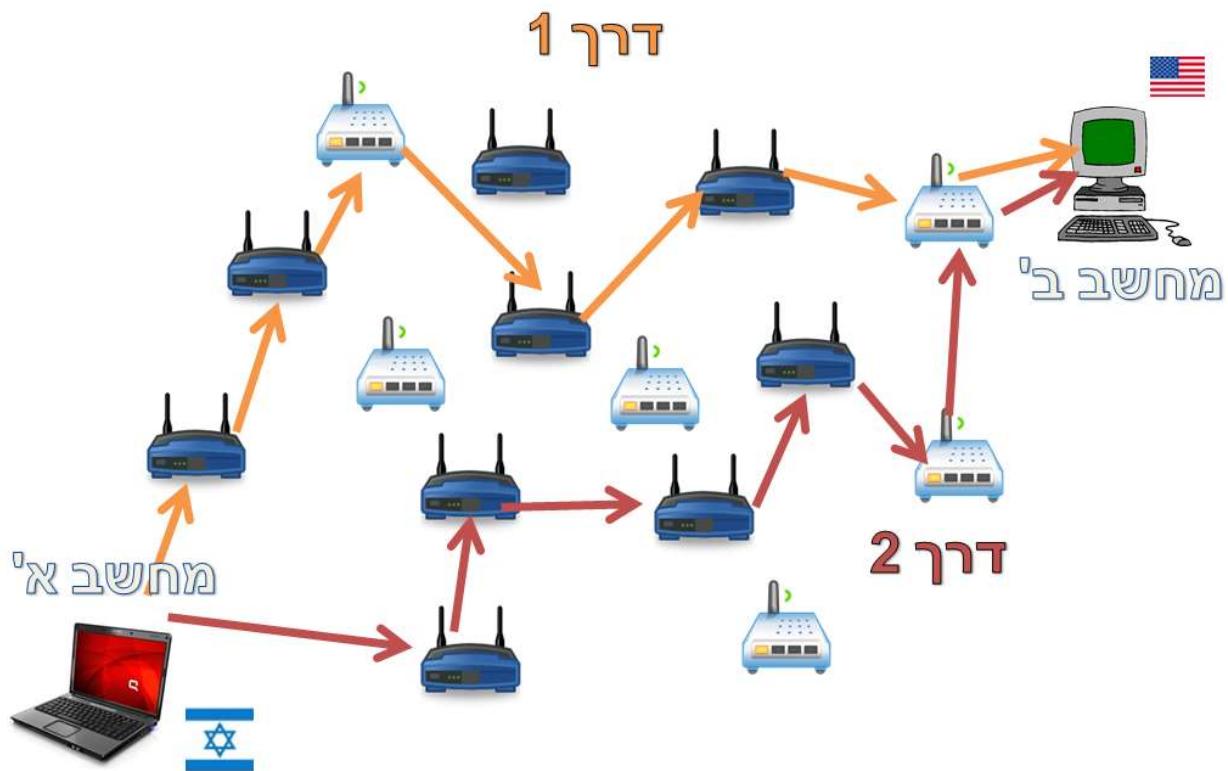
בנוסף, השכבה השנייה מוסיפה מספר יכולות חשובות:

- ארגון המידע בಗושים (המכונים **מסגרות - Frames**, כמו שראינו בфиילטר של Wireshark), בהן תוכלנה השכבות הגבוהות יותר לטפל.
- טיפול במקרים שבהם מספר ישות מנסות לשלוח מידע על אותו תווך פיזי (למשל: מספר מחשבים על אותו כבל רשת, או על אותה רשת WiFi ביתית). השכבה השנייה תמנע התנגשויות.
- טיפול ראשוני בשגיאות (או לכל הפחות זיהוי השגיאות, כדי שאפשר יהיה לשלוח את המסגרת מחדש).

שכבה שלישית - שכבת הרשות

שכבת הנקו מאפשרת לנו לדבר עם ישותות אחרות הסמכות אלינו, אך מה אם נרצה לדבר עם מישחו בקצתה השני של העולם? תפקידיה של שכבת הרשות הוא למצוא את המסלול הטוב ביותר ביוטר מאייתנו אל היעד ובוחרה. שכבה זו לא מתעסקת בתקשרות בין ישותות סמכות, אלא אחראית על המסלול המלא בין שתי נקודות קצה. ניתוב המידע מתבצע על-ידי רכיבים המכונים **רouters** (נתבים), אשר מנתחים את הפקודות בין הרשותות השונות. כך יכולה פקטה לצאת מקו אחד, לעבור דרך מספר קווים שונים ולבסוף להגיע אל הרשות אליה היא מיועדת.

באյור שלפנינו, כל מסלול חיצים בצבע מסוים מסמן מסלול שעשויה לבחור שכבת הרשות לעבור הפקטה:

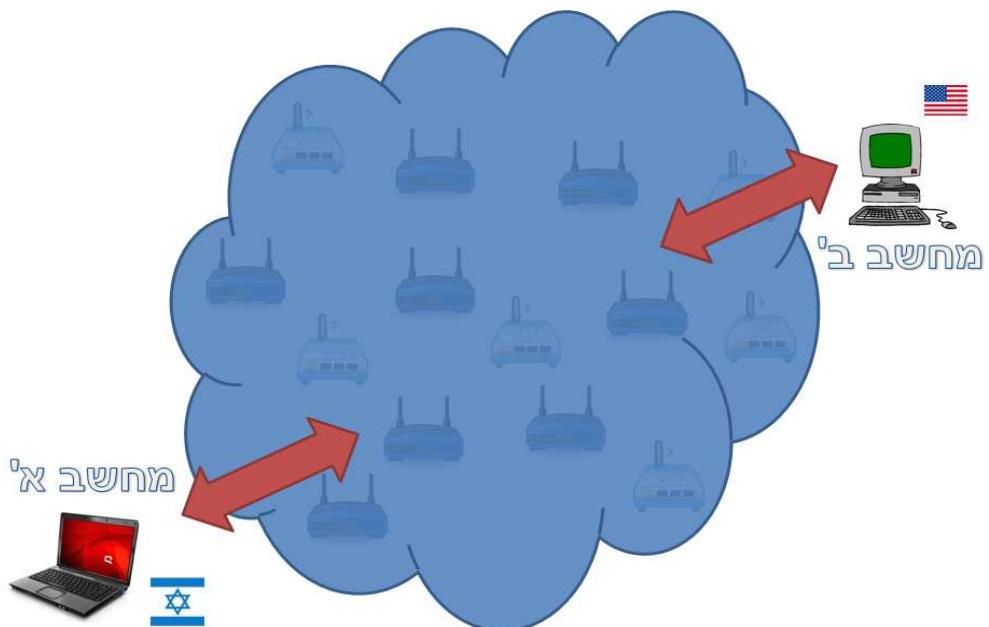


שכבה רביעית - שכבת התעבורה

שכבה התעבורה מסתמכת על שכבת הרשת שתנתב עבורה פקודות מישות כלשהי ברשות לישות אחרת, נאמר אףלו בקצה השני של העולם. אך שירות זה עדין לא מספיק. علينا לזכור שהנחה בסיסית ברשת האינטרנט היא שהחיבור עצמו אינו אמין – פקודות יכולות להישלח ולא להגיע לעדן, או להגיע לעדן באיחור רב. מה יקרה אם נשלוו שתי פקודות אחת אחרי השנייה, אך הן התחלפו והגיעו לעד בסדר הפוך? כדי להיות מסוגלים להתקיים ברשת האינטרנט אנחנו צריכים בחלק מהמקרים ליצור קישור אמין ורציף בין שתי נקודות הקצה, כזה שייתן סדר ומשמעות למידע שישלח – ולא סתם אוסף של חבילות מידע שלא בהכרח קשורות אחת לשניה.

בעוד שהמטרה הקודמת היא אופציונאלית (יש מקרים בהם לא נהיה חייבים להבטיח את סדר הפקודות שנשלחות או את הגעתן בכלל), שכבת התעבורה מספקת תמיד דבר חשוב נוספת: האפשרות לפנות אל מספר שירותים הנמצאים על אותה ישות. דמיינו שעלה שרת מסוים רצה גם תוכנה המספקת שירות מיילים וגם תוכנה המספקת שירות WEB (כלומר מגישה דפי אינטרנט). כיצד יוכל השירות להבדיל בין שתי הבקשות שמתקבלות אליו מהלקוח, האחת אל שירות המייל והאחת אל שירות ה-WEB? לשם כך, השכבה הרביעית מוסיפה לנו פרוטוקום (דיברנו על המושג כבר בפרק [תכנות ב-Socket](#)-[תכנות של Sockets](#), שם דמיינו את הפורט למזהה דירה בתווך בניין), כדי שנוכל להבדיל בין השירותים השונים ולהשתמש בכמה שירותים על אותה ישות.

באյור שלפנינו ניתן לראות שכבת הרשת "העלימה" את הצורך של שכבת התעבורה להכיר את המסלול אל היעד. מבחינת השכבה הרביעית, ישנו "ענן" המחבר בין היעד לבינה – בו היא משתמשת כדי לשלוח פקודות לישות בצד השני:



שכבה חמישית - שכבת האפליקציה

שכבה זו מסתמכת על שכבת התעבורה כדי לקבל קישור לוגי בין שתי נקודות הקצה. איך היא עשוה זאת? כפי שכבר הבנו ממודל השכבות – זה לא באמת מעניין אותה. כל עוד השכבה שמתמחתה מספקת לה את השירות של ייצור קישור שזכה, היא משתמשת בו לצרכיה השונים של האפליקציה. לשכבה זו קיימים פרוטוקולים רבים, המוכרים שבהם: HTTP (פרוטוקול הגלישה באינטרנט עליו דיברנו קודם), SMTP (פרוטוקול דואר), FTP (הعتبرת קבצים), ועוד רבים אחרים. למעשה, כמעט כל אפליקציה משתמשת בחיבור רשתி כלשהו מדברת ב프וטוקול של שכבת האפליקציה.

לסיכום סקירת השכבות, להלן טבלה המתארת את כל השכבות יחד עם מעט פרטי מידע שיאפשרו לכם להשוות ביניהן:

מספר השכבה	שם השכבה	מטרה (בקצרה)	פרוטוקול לדוגמא	שם של גוש מידע
1	השכבה הפיזית (Physical Layer)	העברת המידע בית אחר בית - 0 או 1 בכל פעם		ביט (bit, סיבית)
2	שכבת הקשר (Data Link)	תקשרות בין ישויות סמוכות זו לזו	Ethernet	מסגרת (frame)
3	שכבת הרשות (Network Layer)	השלטה על המסלול שתעביר חבילת מידע בין המקור אל היעד	IP	פרקיה (packet, חביבה)
4	שכבת התעבורה (Transport Layer)	ריבוב אפליקציות על אותה ישות (תמיד) + מתן אמינות ל קישור אופציונאלי	TCP	סגןט (segment)
5	שכבת האפליקציה (Application Layer)	שימושים שונים בהתאם לאפליקציה	HTTP	* אין שם מיוחד

שימוש לב: כמשמעותם באחד הרכיבים לגור מידע של אחת השכבות, מתקווים לרצף המידע משכבה זו ומעלה. למשל: כמשמעותם במושג "פקטה" מתקווים לפקטה בשכבה השלישי, אך גם לכל המידע של שכבה הרביעית וה חמישית (שאפשרות בטור הפקטה, כפי שהזכרנו באורח של [מבנה הפקטה](#)). המונח "מסגרת" מתאר את כל המידע השיך לשכבה השנייה, אך גם לשכבה השלישי, הרביעית וה חמישית.

בהתאם להסביר לעיל, כל מסגרת היא גם פקטה (שהרי אין חビיה בשכבה השלישי בלבד שכבה שנייה), אך לא כל פקטה היא מסגרת (שכן יש מסגרות שאין רק בשכבה השנייה).

מודל השכבות ו-Sockets

גם ה-Sockets עליהם למדנו בפרק הקודם שייכים לשכבת האפליקציה. נזכיר כי Sockets הם בסך הכל API (ממשק תכני) שספקת מערכת הפעלה כדי שאפליקציות יכולו ליצור חיבור רשמי לשויות אחרות ברשת. הם אינם שכבה במודל חמש השכבות. האפליקציות משתמשות ב-Sockets שיצרו אצלן "צינור" להעברת המידע, ודברות מעלהם בפרוטוקולים השונים של רמת האפליקציה (בדיוק כמו ה프וטוקול שאותם כתבתם בתרגיל בפרק הקודם).

עתה, כשאנחנו מכירים את מודל חמש השכבות, נוכל לשים לב לדבר הבא: כשהשתמשנו ב-Socket, בכלל לא נתנו לו פרמטרים רלוונטיים לשכבה שלו – אלא נתנו לו פרמטרים שעוזרים לו לפתח את החיבור בתבוסס על הרמות שמתוחתי! בפועל, הפרמטרים שננתנו היו רלוונטיים לשירות לשכבות שמתחת ל-Socket – שכבת הרשות (השכבה השלישי) ושכבת התעבורה (השכבה הרביעית). נמחיש זאת באמצעות דוגמה:

```
s = socket.socket()
s2 = socket.socket()
s.bind(("1.2.3.4", 80))
s2.connect(("5.6.7.8", 8820))
```

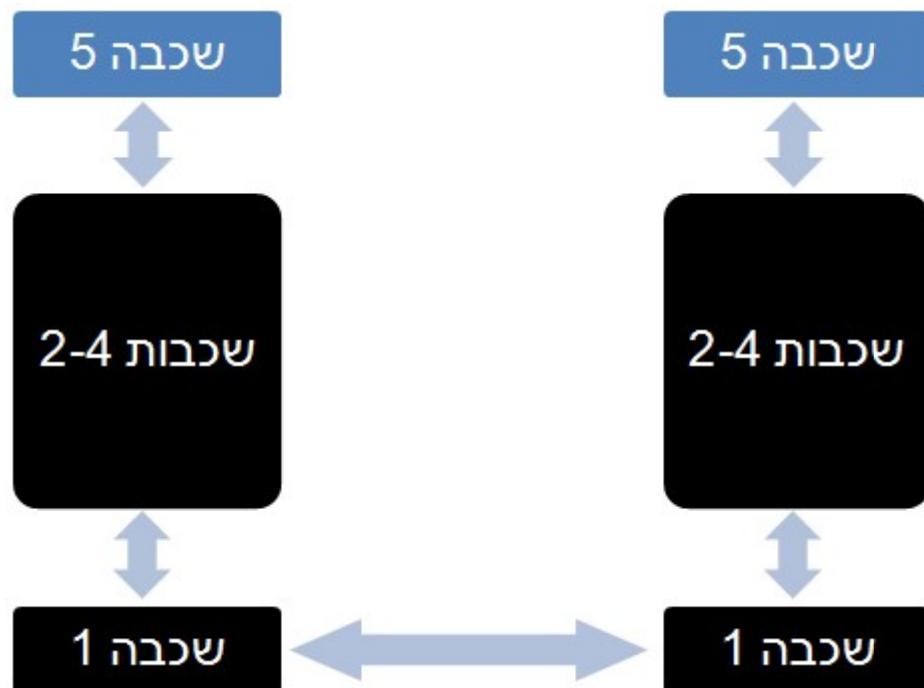
בדוגמה זו, סיפקנו את הפרמטרים של **שכבת הרשות (Network Layer)**: באיזו כתובת IP להשתמש. כמו כן, סיפקנו הפרמטרים של **שכבת התעבורה (Transport Layer)**: באיזה פורט להשתמש.

למעשה, Socket הינו ממש שמאפשר את התקשרות מהשכבה הפיזית ועד שכבת התעבורה, ומעלה מדברים בפרוטוקולים שונים בשכבת האפליקציה.¹⁵

¹⁵ למעט Raw Sockets, עליהם לא נפרט בספר זה.

בספר זה, נסקור את חמש השכבות מלמטה, כלומר החל משכבה האפליקציה של מעלה ועד לשכבה הפיזית שלמטה. שמו לב שיתכן מצב בו לא תבינו בדיקת כיצד עובר המידע בשכבות שמתוחת, ותצטרכו להתייחס אליון כל "קופסה שחורה", צו שرك מספקת שירותים ולא ברור כיצד היא פועלת (בדיקה כפי שהפרוטוקולים ברמות הגבוהות מניחים שהרמות שמתוחתין הן "קופסאות שחורות" וסגורות מספקות להם שירותים שונים). ככל שנצלול לעומק ונגיעה לرمאות התחתיונות, כך תבינו יותר כיצד עובר המידע בפועל.

כך למשל, כשןלמד על שכבת האפליקציה, הגיוני שלא יהיה ברור כיצד מובטח שהמידע עובר מישות אחת לשישות שנייה. נושא זה יתבഹר בהמשך הספר, כשןלמד על השכבות התחתיונות.



אתם מוזמנים לרשום לעצמכם לצד דף עם שאלות, כדי שתוכלו לחזור אליו אחר כך כשןלמד על השכבות הבאות ולבדוק אם שאלותיכם אכן נعمו.



למה המידע מחולק לפקודות?

למדנו על מודל השכבות, הבנו את החשיבות שלו והכרנו את התפקיד של כל שכבה ושכבה במודל - אך עדין לא שאלנו את עצמנו שאלה בסיסית, שואלי תהיתם לגביה: מדוע בכלל לחלק את המידע לפקודות? למה לא להעביר את כל המידע כרצף ארוך של Bytes, שמתחליל כשישות אחת רוצה לשЛОוח מידע לישות אחרת ומסתדרים רק כאשר כל המידע הועבר לצד השני?

לשאלת הזו קיימות מספר תשובות. נזכיר את הבולטות שבהן:

- בקרת Sağיות טובה יותר: בחלק מהשכבות נעשית בקרת Sağיות על המידע שנשלח, כדי לאלהות Sağיות ולאפשר שליחת מחודשת של המידע אם הוא לא הגיע לעדכון כראוי. חלוקת המידע לקבוצות קטנות, אותן אנחנו מכירים כפקודות, מאפשרת לזהות את Sağיות מוקדם יותר (לאחר שנשלח רק חלק קטן מרצף המידע השלם), ובמידה שקרתת Sağיה – לשЛОוח חדש אך ורק את החלק הפוגם, במקום לשЛОוח את כל המידע מחדש.
- שילוב מספר זרמי מידע (Streams) במקביל: חלוקת המידע לפקודות מאפשרת לכמה אפליקציות לשLOWוח במקביל את המידע שלהם ללא צורך להמתין קודם לשליחת תס'ים אחריהם לשLOWוח את המידע שלהם. חשוב על כך: כל כרטיס רשות יכול להוציא בכל זמן נתון אך ורק זרם נתונים אחד אל התווך אליו הוא מחובר. אם לא היו מפצלים את המידע לפקודות, כל תוכנה הייתה צריכה לחתוך עד שההקו יתפנה, ולשלוח בתורה מידע דרך כרטיס הרשות. במקרה זו לא הייתה מתאפשרה שליחת המידע במקביל בין מספר תוכנות¹⁶.
- הדבר נכון גם לגבי מספר מחשבים המשדרים על אותו הקו, שכן גם במקרה הזה לא ניתן להעביר על אותו קו יותר מרצף מידע אחד בו-זמנית. אם המידע היה עוזר באופן רציף ולא מחולק למסגרות – בכל פעם שמחשב היה שולח מידע כלשהו, שאר המחשבים שנמצאים באותו קו היו מנועים מלשלוח מידע והוא צריכים לחכות שהוא יסתיים את השליחת. חלוקת המידע למסגרות גורמת לכך שבסוף כל מסגרת ניתנת הזדמנות לשוט אחרית ברשות להתחיל לשדר מסגרת משלה, ומונעת מישות אחת להציג את הקו ברצף ארוך מאוד של ביטים¹⁷.
- מניעת בעיות סנכרון ברמת החומרה: לא ניתן לסיבת הזו לעומק, אולם נציין שברמת החומרה קל יותר לסנכרן בין מספר ישויות הקשורות על אותו קו כל עוד המידע מחולק למנוגדות קטנות, וכך יש פחות סיכוי להתנגשויות. נושא זה יורחב בהמשך הספר, בסוף הפרק על שכבת הקו.
- כאמור – לא צינו את כל הסיבות לחלוקת המידע לפקודות. חלק מהסיבות הנוספות יזכרו בהמשך הספר, ועל חלקן לא נדבר כלל.

Wireshark

כדי להבין כיצד עובר המידע בראשת, נרצה להסניף את התעבורה היוצאת והכנסת אל המחשב שלנו ולנתח אותה (הסנפה היא הפעולה בה אנו משתמשים על חבילות המידע בדיקות כפי שנשלחו או התקבלו בכרטיס הרשות). כך

¹⁶ התהילה' שבו מידע ממספר מקורות מסוולב אל תוך משותף אחד נקרא ריבוב (Multiplexing). בדוגמה זו המידע מתקובל ממספר אפליקציות, ויצא אל כרטיס הרשות (שהוא משאב יחיד המשותף לכל האפליקציות על אותה ישوت).

¹⁷ תופעה זו מכונה הרעבה (Starvation).

ונכל לגלות בדיק מה קורה ברשות ולהבין דברים שאין לנו דרך לראותם. לשם כך נשתמש בתוכנה **Wireshark**, תוכנת הסנפה מהטבות בעולם – והיא גם חינמית!

את התוכנה תימצא כМОון בהתקנת גבהים, וניתן במקרה הצורך להוריד מה קישור הבא: <http://www.wireshark.org/download.html> שימושם לב שאתם מודדים את הגרסה המתאימה למערכת הפעלה שלכם.

עקבו אחר הוראות ההתקנה (במהלך ההתקנה תישאלו אם אתם רוצים להתקין *driver* בשם *WinPcap*. אשר (התקנים גם אותו).

- קיימות מספר דרכי לפתח את התוכנה:
- לחיצה כפולה על ה-*icon* של Wireshark שנמצא על ה-*Desktop*.
 - דפודף למיקום המלא של התוכנה (בדרכ כל *exe*).
 - فاتיחה שורת הפעלה (*WinKey + R*), הקלדת המילה Wireshark ולחיצה על *Enter*.



ניתן לצפות בסרטון ההסבר לעובודה בסיסית עם Wireshark בכתבובת:

<http://data.cyber.org.il/networks/videos/wireshark-basic.html>



ניתן לצפות בסרטון ההסבר על שימוש מתקדם יותר ב-Wireshark בכתבובת:

<http://data.cyber.org.il/networks/videos/wireshark-advanced.html>



הדרך קצרה לתחלת עבודה:

ב-Wireshark יש מספר דרכי להתחיל הסנפה חדשה, דרך אפשר גם לשנות הגדרות מתקדמות. נתחיל בסקירת החלקים החשובים בדרך להטילה של הסנפה חדשה:

Interface List

תפריט זה מאפשר לנו לבחור מהו הממשק הרשמי דרכו נרצה להסניף את הרשת. ניתן להיכנס אליו במספר דרכים שונות:

- דרך הקיצור במסך הפתיחה (לחיצה על :(Interface List

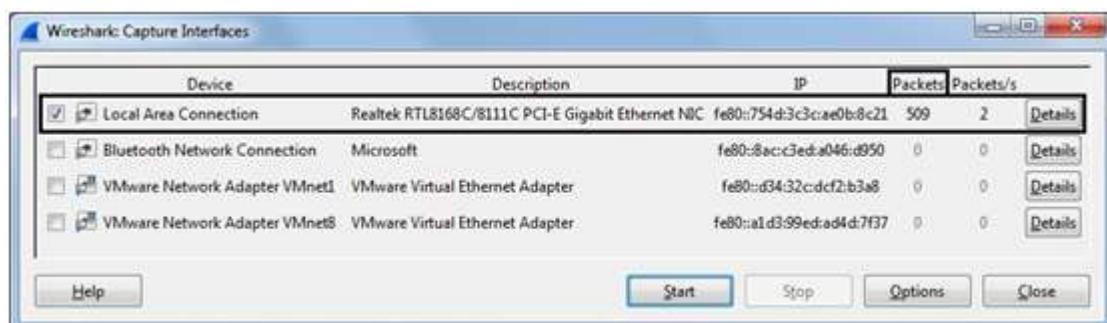


- דרך הcptor בסרגל הכלים העליון, בכל שלב בו התוכנה פתוחה (לא רק במסך הפתיחה):



- דרך התפריט (Ctrl + I ...Capture -> Interfaces •

בתפריט זה רשומים כל הכרטיסים דרכם תוכל להסניף, כתובת ה-IP שליהם, מספר הפקודות שנקלטו דרכם וקצב הפקודות שהם קולטים. כלל אכבע – אם רשומים כמה כרטיסים ואתם לא יודעים איזה מהם לבחור, לדוגמה כי אתם בכיתה ויש לכם גם רשת וגם wifi, בחרו את הכרטיס שראה את המספר הרב ביותר של פקודות. לרוב זה יהיה הכרטיס דרכו תרצו להסניף.



בדוגמה לעיל, בחרנו בכרטיס הרשות הראשון "Local Area Connection", משום שהוא רואה 509 פקודות, בעוד שבשאר הכרטיסים כלל לא נראה פקודות.

לאחר שבחרתם כרטיס, תוכלו ללחוץ על כפטור Start ולהתחליל להסניף דרכו עם הגדרות ברירת המחדל, או ללחוץ על כפטור Options ולהגיע למסך של הגדרות ההסנפה.

Capture Options

בתפריט זה ניתן לקבוע הגדרות שונות עבור ההסנפה. גם אליו ניתן להגיע במספר דרכים:

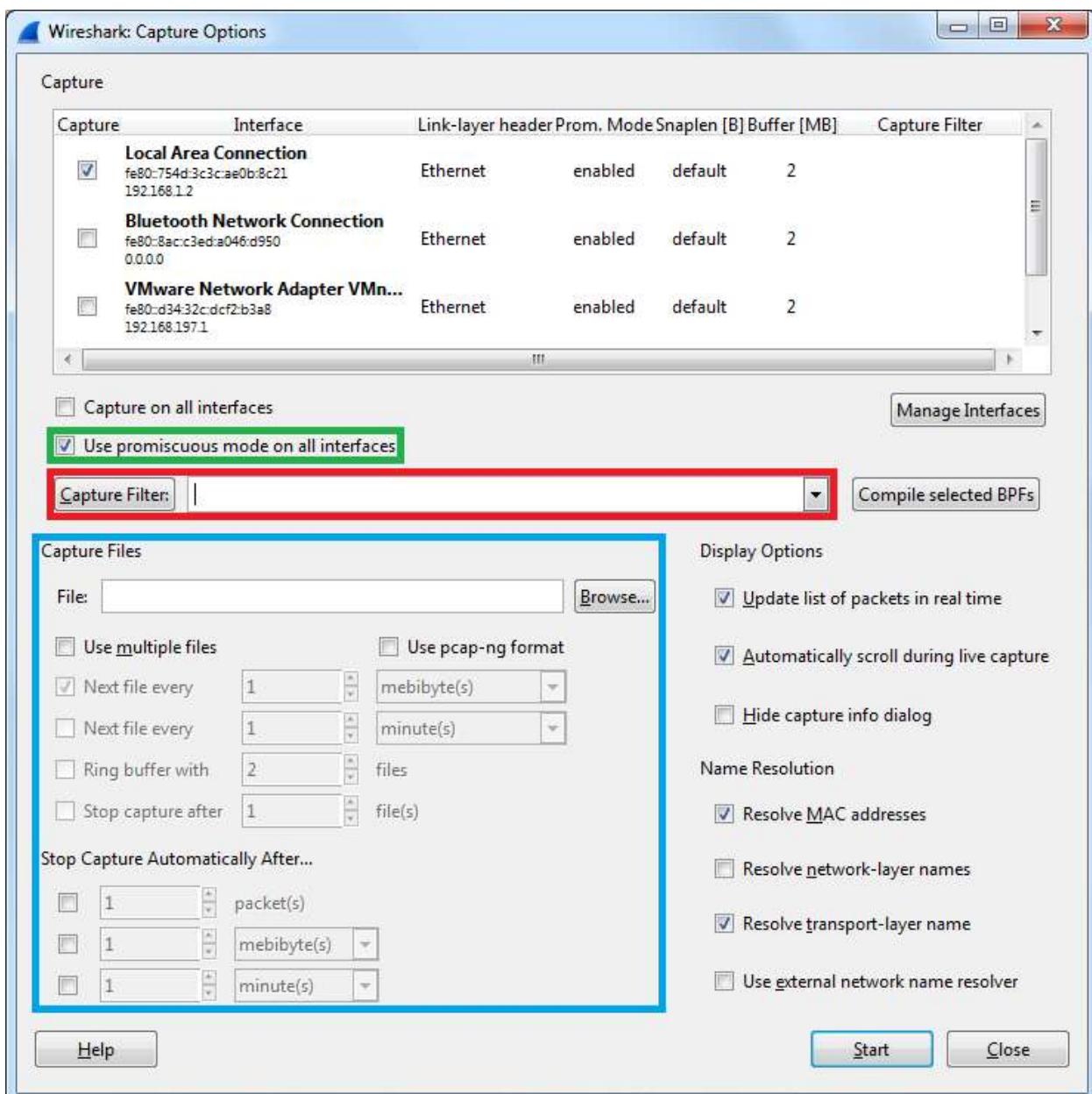
- דרך הcptor Options עבור כרטיס ספציפי במסך ה-Interface List.

- דרך הקיצור במסך הפתיחה (לחיצה על **.(Capture Options**)
- דרך הcptor בסרגל הכלים העליון, בכל שלב בו התוכנה פתוחה (לא רק במסך הפתיחה):



- דרך התפריט (Ctrl + K) (יש גם קיצור מקלדת: ...Capture -> Options)

הaggerות הבולטות במסך זה:



- **בירוק - Promiscuous Mode:** הכנסת כרטיס הרשת ל"מצב פרוץ", מה שייגרום לכך שנראה בהසנפה את כל המסגרות שרוואה כרטיס הרשת, גם כאלה שלא מיועדות אליו (את המשמעות של משפט זה נבין בהמשך הספר).
- **באודום - Capture Filter:** מגדיר מסנן של פקודות להסנפה עבור ה-Driver (נறחיב על משמעות מסנן זה בהמשך הפרק).
- **בחול - Capture Files:** מאפשר שמיירה של ההסנפה לקובץ ואף חלוקה אוטומטית שלה לקבצים עלי-פי גודל או זמן (למשל: חלוקת ההסנפה לקבצים בגודל 50 MB כל אחד, או סגירת קובץ הסנפה ופתיחה של קובץ חדש בכל דקה).

התחלת ועכירה של הסנפה

על מנת להתחיל את ההסנפה יש לבצע את אחת הפעולות הבאות:

- לחיצה על כפתור Start באחד התפריטים הקודמים שהוצעו.
- דרך הקיצור במסך הפתיחה (בחירה בכרטיס הרלוונטי מהרשימה ולחיצה על Start).
- דרך הcptutor בסרגל הכלים העליון, בכל שלב בו התוכנה פתוחה (לא רק במסך הפתיחה):



- דרך התפריט Start -> Capture (יש גם קיצור מקלדת: Ctrl + E).

שימוש לב לאחר שהסנפה פועלת, מצב הcptutors משתנה, וכעת ישן 2 פעולות נוספות שניתן לעשות:

- **עיצוב הסנפה** – דרך cptutor בסרגל הכלים העליון או דרך התפריט Stop -> Capture (יש גם קיצור מקלדת: Ctrl + E):



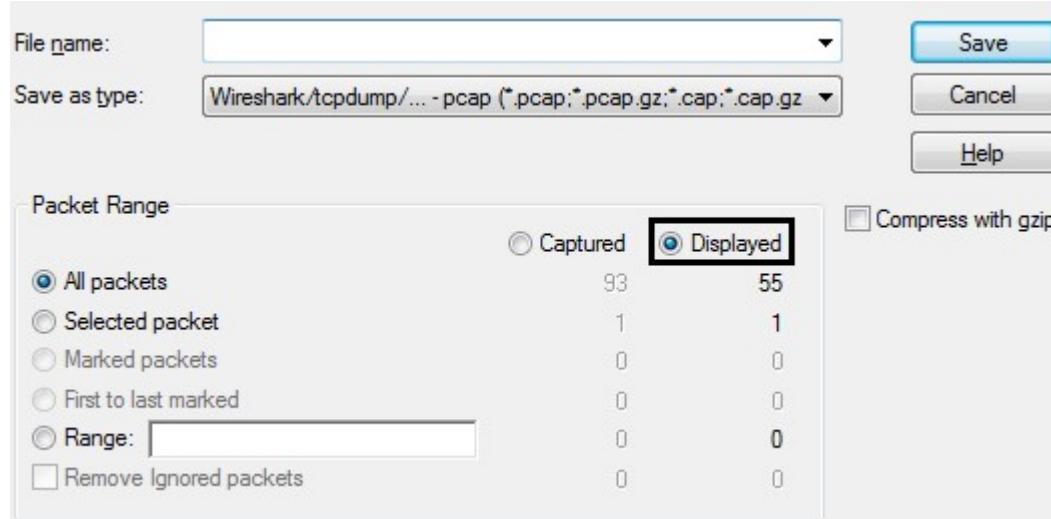
- **התחלת מחדש של ההסנפה** – ניקוי רשימת הפקודות וההתחלת הסנפה חדשה על אותו interface (כרטיס הרשת), עם אותם מאפיינים, ואוטו Display Filter. גם את האפשרות הזאת ניתן להפעיל דרך cptutor בסרגל הכלים העליון או דרך התפריט Restart -> Capture (יש גם קיצור מקלדת: Ctrl + R):



שמירה ופתיחה של קבצי הסנפה

כדי לשמר את הפקודות שנקלטו, נעצור את ההסנפה ונלחץ על File -> Save (קיצור מקלדת: Ctrl + S). הקובץ שיישמר יהיה בעל הסיומת .pcap.

אם נרצה לשמר רק חלק מהפקודות שנקלטו (שימושי בעיקר במקרים בהם רצים לשמר רק את הפקודות שענו על הפילטר הנוכחי וモציגות כתע למסך), נפתח את התפריט Export Specified Packets File -> ... File, שיציג לנו את חלון השמירה הרגיל אך יוסיף לנו את אזור ה-Range Packet שמאפשר לבחור אילו פקודות לשמרו:



במידה שנרצה לשמר רק את הפקודות שעונთ על הפילטר הנוכחי, נבחר באפשרות **Displayed**.

כדי לטעון לתוכנה קובץ הסנפה, ניתן ללחוץ לחיצה כפולה על קובץ pcap או לבחור מהתפריט File -> Open (קיצור מקלדת: O). (Ctrl + O)

מסננים (Filters)

סוגי מסננים

כפי שכבר הוזכר, ישנו שני סוגי של מסננים: מסנן תצוגה (Display Filter) ומסנן הסנפה (Capture Filter). Capture Filter, בשונה מה-Display Filter (שמייעד עבור ה-Driver), משפייע על התצוגה בלבד – כלומר, פקודות שלא עברו את הפילטר עדין קיימות בהסנפה, ואם נשנה את הפילטר יוכל להציגן לתצוגה. עם זאת, פילטר זה אינו בהרבה מהפילטר של ה-Driver.

בספר זה לא ניגע בתחריר לכתיבת Capture Filters, אולם נסקור בקצרה את ההבדלים ביניהם:

קריטריון	מסנן הסנפה (Capture Filter)	מסנן תצוגה (Display Filter)
רמה בה רץ	מערכת הפעלה (Driver)	התוכנה (Wireshark)
המקום ממנו מפעילים אותו	Capture Options	מסך ההסנפה הראשי
מתי מפעילים אותו	לפני ההסנפה	במהלך ההסנפה
אם ניתן לשנות בזמן ההסנפה	לא	כן

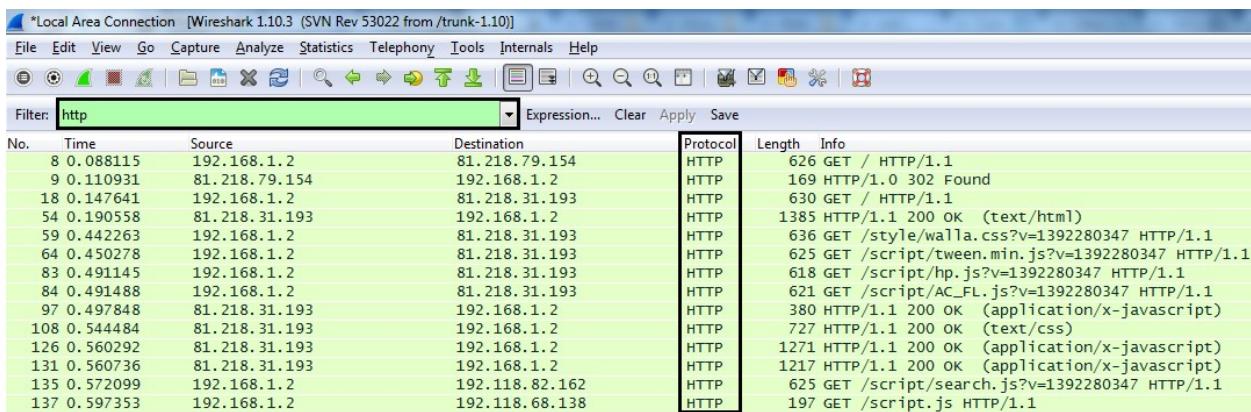
עמיר ורחב	מצומצם	תחביר
איטי	מהיר	מהירות

מידע נוסף על Capture Filters תחולו למצוא כאן: <http://wiki.wireshark.org/CaptureFilters>

דוגמאות

מסנני תצוגה מאפשרים יכולות סינון מתקדמות, ונותנים אפשרות לסתן גם על-פי השדות הפנימיים של כל פרוטוקול (יש אפילו אפשרות לסתן על-פי פרמטרים מתקדמים יותר שאינם מופיעים בפקטה המקורית, הודות לניתוח המעמיק שעושה Wireshark לכל פקטה).

- אם נרצה לפילטר על פרוטוקול מסוים, יוכל פשוט לרשום את שמו וויפיעו פקודות מפרוטוקול זה בלבד.
 - למשל: ip, arp, tcp, http



- ניתן לפילטר על שדה מסוים של פרוטוקול, כאשר אופרטור ההשוויה יכול להיות == (= שווה), != (!= שונה), > (גדול מ..), < (קטן מ..), `contains` ("...") (בדיקה אם השדה מכיל את המחרוזת "..."), ועוד.

כדי לציין איזה שדה אנו רוצים לנשום את שם הפרוטוקול, לאחריו נקודה, ולאחר מכן את שם השדה – בצורה הבאה:

<ProtocolName>.<FieldName> Operator <Value>

- למשל:

ip.src == 192.168.1.1 (192.168.1.1) (הציג כל הפקטות שכותבת המקור שלhn היא 192.168.1.1)

ip.dst != 192.168.1.1 (192.168.1.1) (הציג כל הפקטות שאינן מיועדות ל-192.168.1.1)

ip.addr == 192.168.1.1 (192.168.1.1) (הציג כל הפקטות שכותבת המקור או כתובת היעד שלhn היא 192.168.1.1)

- ניתן לשלב מספר ביטויים ביחד, ולקשר ביניהם באמצעות קשר לוגי: or / and.

- למשל:

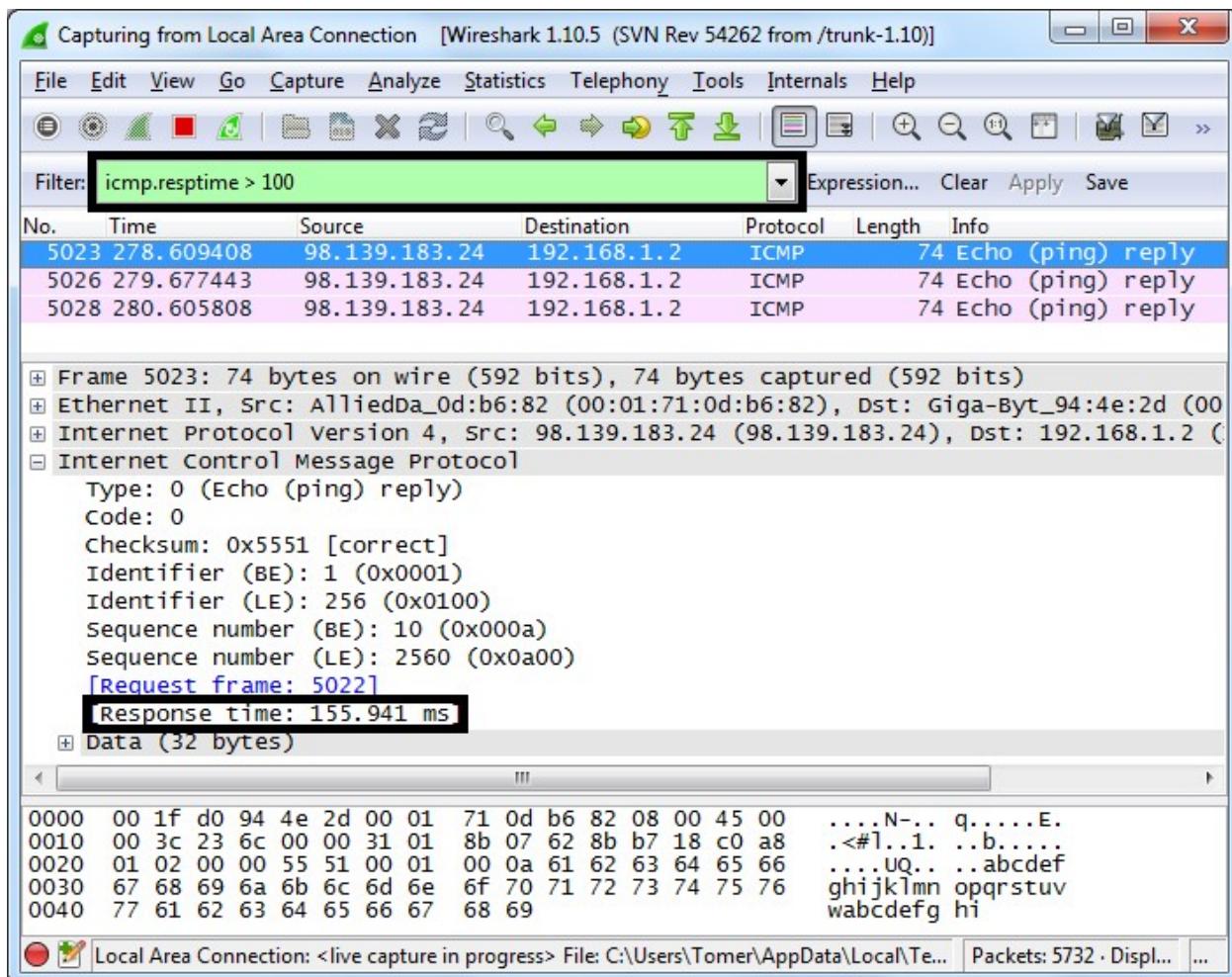
ip.addr == 192.168.1.1 or tcp.port == 22

(הצגת כל הפקות שנשלחו או התקבלו מכרטיס הרשות שכתובתו 192.168.1.1, או פקודות שנשלחו או התקבלו בפורט 22)

- הניתוח של Wireshark מאפשר לנו להשתמש בשודות שלא באמצעות קיימים בפקטה, אלא הם פרי ניתוח התוכנה עצמה:

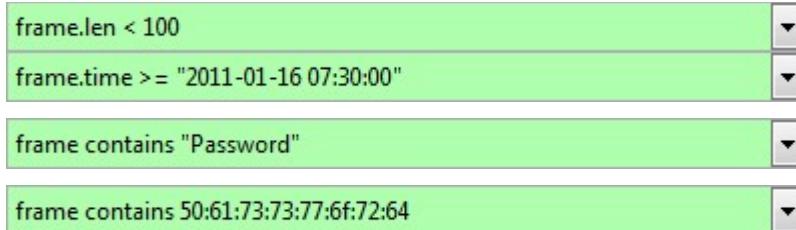
- למשל:

(פקודות ICMP שזמן התגובה שלהן היה גדול מ-100 מילישניות) icmp.resptime > 100



שימוש לב שמן התגובה הוא איננו שדה אמיתי בפקטה (הוא לא חלק מפרוטוקול ICMP), אלא מחושב בידי Wireshark על-פי ההפרש בין הזמן שבו נקלטת פקעת התשובה לבין הזמן שבו נשלחה פקעת הבקשה. במקרה Wireshark מציג שדה שנובע מהניתוח שלו ולא קיים בפרוטוקול המקורי, השדה יוקף בסוגרים מרובעים - [-].

- דבר נוסף שחייב להכיר הוא האובייקט frame (המסמן כל מסגרת שנקלטה בכרטיס הרשות, לא משנה באיזה פרוטוקול היא). כך ניתן לפilter על פרמטרים כמו אורך המסגרת (frame.len), הזמן שבו היא נקלטה ("frame contains "SomeText") או פשוט על תוכן שמופיע בה במקום מסוים ("frame.time".

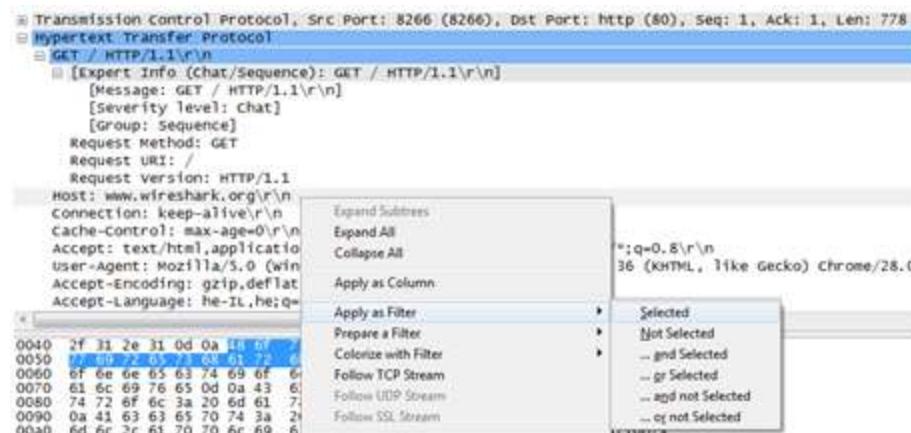


(הדוגמה الأخيرة זהה לזה שלפניהם, רק שהיא מאפשרת לתת ערכי הקוסה של הבטים במקום יציג ה- ASCII שלהם)

מגוון הфиילטרים ש-Wireshark מציע הוא כל כך רחב, כך שלא יוכל להזכיר כאן את כל הфиילטרים הקיימים. ניתן שני טיפים שווים להכיר אם אתם לא יודעים כיצד לרשום ביטוי כלשהו. הם שימושיים גם במקרה שבו שכחتم איך קוראים לфиילטר שאתם מחפשים, וגם כדי ללמוד בלבד על עוד פרוטוקולים ושדות עצמאכם:

1. ניתן להיעזר בתפריט ה-Expression, שעזר לנו לבנות ביטויים כאלו.
2. בצערת לחיצה ימנית על שדה כלשהו בחולון ניתוח הפקטה, ובבחירה ב- .Apply As Filter -> Selected.

למשל: נניח שאנו רוצים לנגן על-פי שדה ה-Host בפרוטוקול HTTP, אך איננו יודעים כיצד קוראים לфиילטר זהה. מספיק שנמצא פקעת HTTP אחת שבה מופיע השדה הזה, ונוכל להגיד אותו כ필טר באמצעות התפריט:

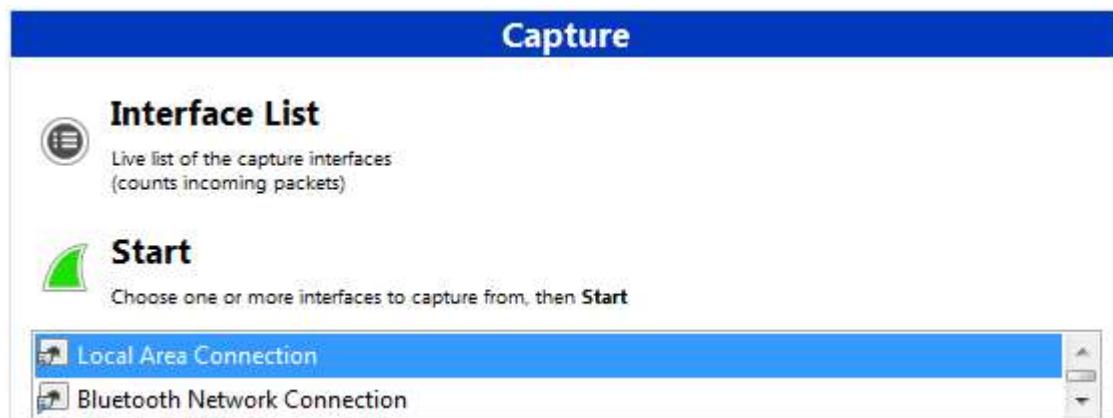


עכשו ניתן לראות את הфиילטר ש-Wireshark יצר עבור השדה הזה:

Filter: http.host == "www.wireshark.org"
שימוש לב שבאמצעות שיטות אלו אתם יכולים **ללמד את עצמכם** כיצד להשתמש ב-Wireshark, לגלוות פילטרים חדשים ואפשרויות שלא הכרתם עד כה.

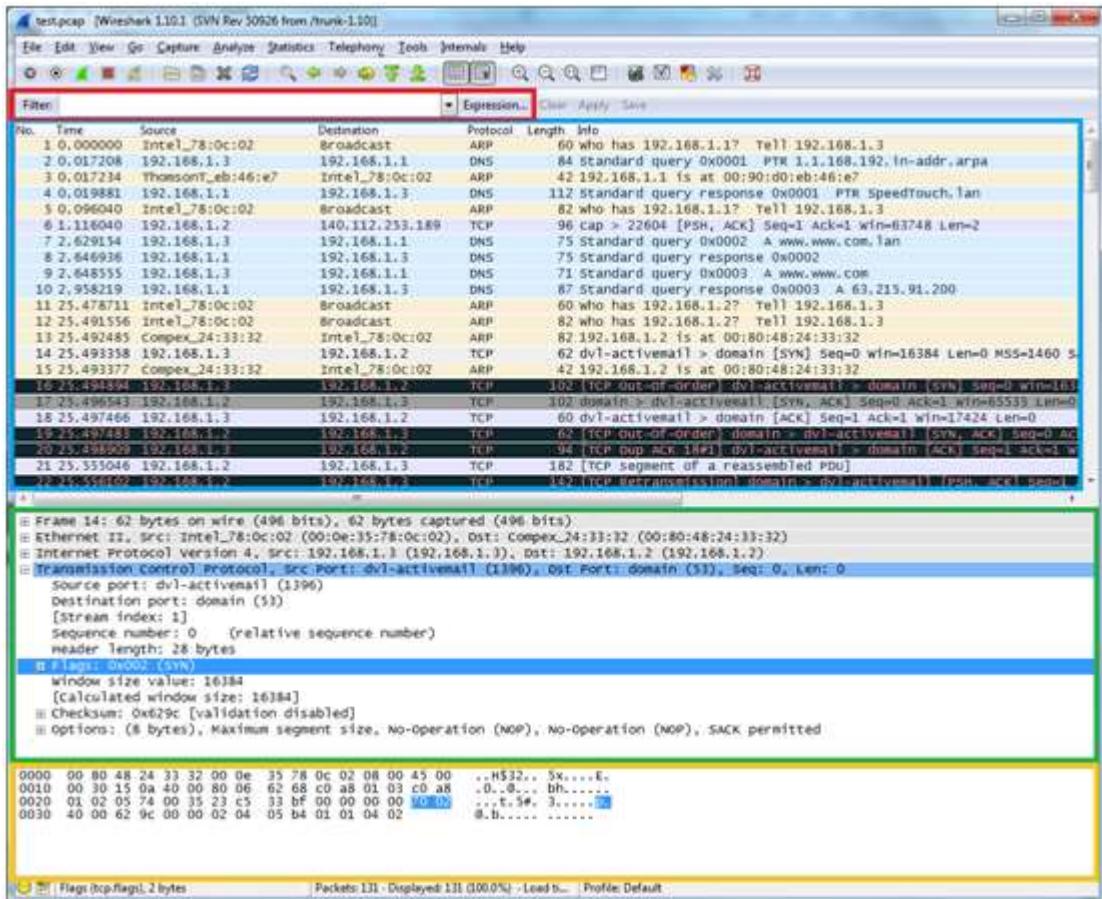
שימוש ב-WIRESHARK לניתוח מודל חמש השכבות

לאחר פתיחת התוכנה, יתקבל מסך הפתיחה, שמכיל קיצורים שימושיים כמו פתיחת קובץ הסנפה שנשמר בעבר, גישה מהירה לעזרה או תחילת הסנפה חדשה. כדי להתחיל הסנפה חדשה, בחרו את כרטיס הרשת שלכם מהרשימה שנמצאת תחת הכותרת Capture (לרוב זה יהיה Local Area Connection אם אתם מחוברים לרשת דרך כבל פיזי, או Wireless Network Connection אם אתם מחוברים דרך דלקת אלחוטי) ולחצו על .Start



כעת נפתח לכם מסך ההסנפה הראשי. אם תמתינו מספר שניות תוכלם לראות שבמרכז המסך מתמלאות להן שורות-שורות (אם לא – כנראה שבחרתם בכרטיס הרשת הלא נכון. נסו לפתיחת הסנפה חדשה על כרטיס אחר). השורות הללו מציגות **פקודות (חבילות מידע, Packets)** שכרטיס הרשת שלכם מוציא או מקבל. כזכור מההסבר אודות מודל חמש השכבות, פקודה היא מעין מעטפה ש מכילה מוען, נמען ואת תוכן ההודעה – וכך מתאפשרת העברת המידע ברשת מקום למקום. זוכרים [בפרק הראשון](#) הרינו כי נשלחת הודעה בקשה מהדף אל האתר של שרת Facebook, והודעת תגובה מהאתר של Facebook אל הדף? למעשה, הודעות אלו הן פקודות.

כעת נוכל להסתכל על החלקים מהם מורכב מסך ההסנפה הראשי של Wireshark:



1. **בAddon - מסנן תצוגה (Display Filter):** בחלון זה ניתן לסנן את הפקודות ולהציג רק את אלו שעוננות על תנאי מסוים. כרגע זו עוללה להיראות לכם כיכולת לא ממש חשובה, אבל בפעם הראשונה שתՏנינוו תראו כל כך הרבה פקודות - ותבינו שבהربבה מאוד מקרים נרצה לפלטר (לסנן) רק את אלו שמשמעותו אוטנו. דוגמא לפילטר זה יכולה להיות "רק התקשרות ביןebin הרשת של Google", כדי להסתיר את כל הגלישות שלי לשאר אתרי האינטרנט.

ניתן לכתוב ביטויים לפילטר בעצמנו, או להשתמש בחלון ה-**Expression**... שעזר לבנות פילטרים שאנו לא יודעים כיצד לכתוב.

2. **בחלון - רשימת הפקודות שנקלטו:** במרכז המסך ניתן לראות את כל הפקודות שנקלטו דורך כרטיס הרשת (ושעוננות על הפילטר שהגדכנו). נפרט על השדות המופיעים באופן ברירת המחדל עבור כל פקודה:

- No. – מספר הפקטה בהסופה (מס' סידורי).

Time – משך הזמן ש עבר מתחילת ההסופה ועד שנקלטה הפקטה.

Source – כתובת המקור של הפקטה. לפקודות IP, תוצג כתובת ה-IP, וזה הכתובת שניתן לראות בדרך כלל בשדה זה.

- Destination – כתובת היעד של הפקטה. לפקטות IP, תוצג כתובת IP, וזה הכתובת שניתן לראות בדרך כלל בשדה זה.
- Protocol – באיזה פרוטוקול הפקטה מועברת.
- Length – אורך הפקטה בת bites (bytes).
- Info – מידע נוסף על הפקטה. משתנה לפי סוג הפרוטוקול.

3. **בירוק** - ניתוח הפקטה: בחלק זה ניתן לראות ניתוח של פקטה מסומנת מרשיימת הפקטות. הניתוח מחלק את הפקטה לשכבותיה השונות, ומציג מידע על כל אחת מהשכבות (עליה נלמד בהמשך).

4. **בתוכם** - תוכן הפקטה: הצגת תוכן הפקטה בייצוג הקס-דצימלי המקורי, ובייצוג ASCII מיימי. Wireshark הוא כל' חזק, והוא מקשר לנו בין כל השדות של הפוטווקל למיקום שלהם בפקטה. שימוש לב שלחיצה על בית (byte) כלשהו של הפקטה תكشف את חלון הניתוח לחלק הרלוונטי בו נמצא הבית הזה, ולהפוך – לחיצת על נתון כלשהו בחלון הניתוח תסמן לכם היכן הוא נמצא בפקטה השלמה ותראה לכם את הייצוג שלו. בהמשך נבין טוב יותר את הקשר בין שני החלונות הללו.

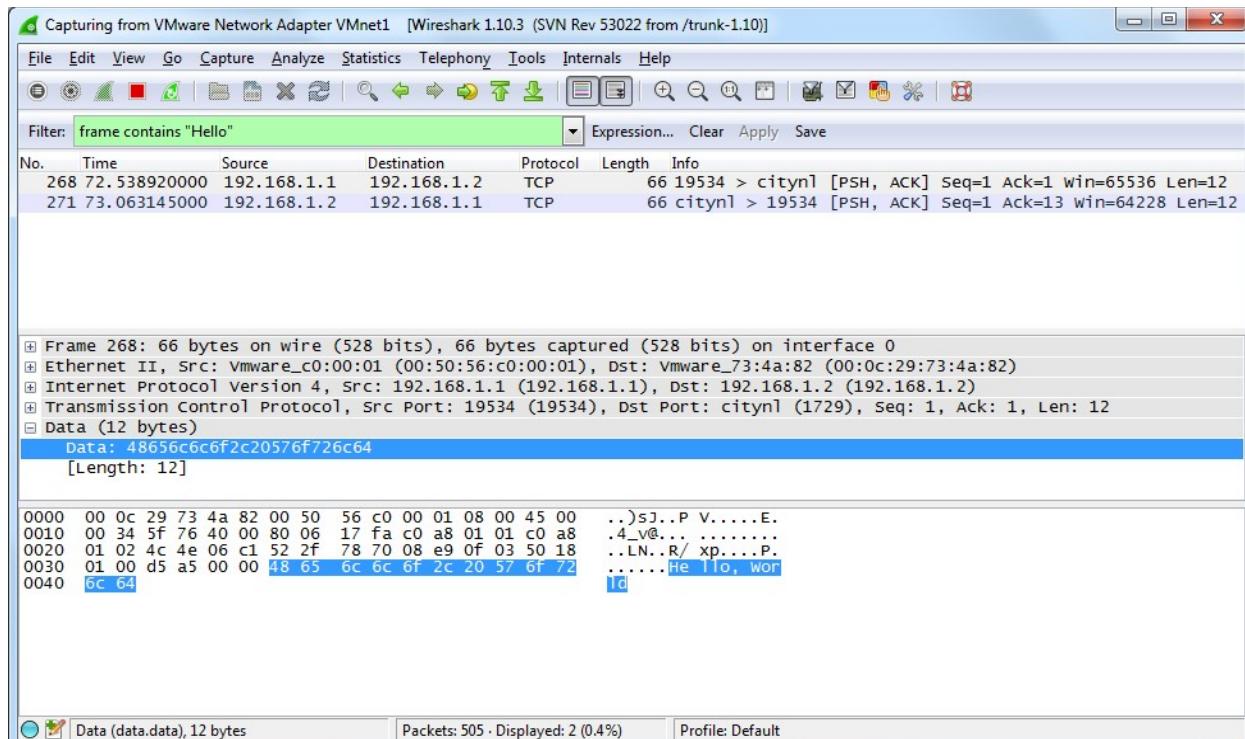
תרגיל 3.1 מודרך - הסנפה של שרת החדים מהפרק הקודם

טרם נתמקד בהסנפה, הפעילו את הסקריפטים שככובתם בתרגילים 2.2 ו-2.5 בפרק הקודם (סקריפט הלקוון וסקריפט שרת החדים) על מחשבים נפרדים. הריצו את `uk.py` על מחשב מרוחק ואת `Client.py` על המחשב הנוכחי (הסיבה לכך שאנו מרים את סקריפט הלקוון וסקריפט השרת על מחשבים נפרדים ולא על אותו מחשב נועצה בעובדה שהרבה יותר קשה להסניף את המידע שנשלח אליו המחשב דרך הכתובת 127.0.0.1 עלייה דיברנו קודם¹⁸). לאחר שיידאתם כי סקריפט השרת והלקוון מצליחם לתקשר ביניהם על גבי מחשבים נפרדים, הריצו אותם שוב בעודם פועלות ברקע.

עכשו הגיעו לך חלק המעניין: כבר ראיינו כי המידע שנשלח דרך ה-`Sockets` מודפס למסך ("Hello, World"), אך האם נוכל למצוא אותו בהסנפה? התשובה היא, כמובן, חיובית – משום שהמידע נשלח דרך כרטיס הרשות שלנו ולכן נקלט בהסנפה. אם הייתם זרים, אולי הצלחתם לזהות את הפקודות הרלוונטיות מבין כל הפקודות שהציגו במסך ההסנפה, אך גם אם לא – אנו נשתמש באופציית `sinon` הפקות כדי להציג רק את הפקודות הרלוונטיות אלין. רשמו בשדה `Filter` את `Display Filter` הבא:

(ה필טר הזה גורם לכך שיוצג רק הפקות שמופיעות בהן המילה Hello (Hello Socks))

¹⁸ במערכות הפעלה `Linux` (בשונה מ-`Windows`), ההסנפה על התעבורת שנשלחת למחשב עצמו דרך 127.0.0.1 היא פשוטה וזהה להסנפה על כל כרטיס רשת אחר. גם ב-`Windows` ניתן להסניף תעבורת שנשלחת אל 127.0.0.1, אולם התהליך מסובך יותר ודורש התקינה של תוכנות מיוחדות. אי כך, לא נעשה זאת בספר זה.

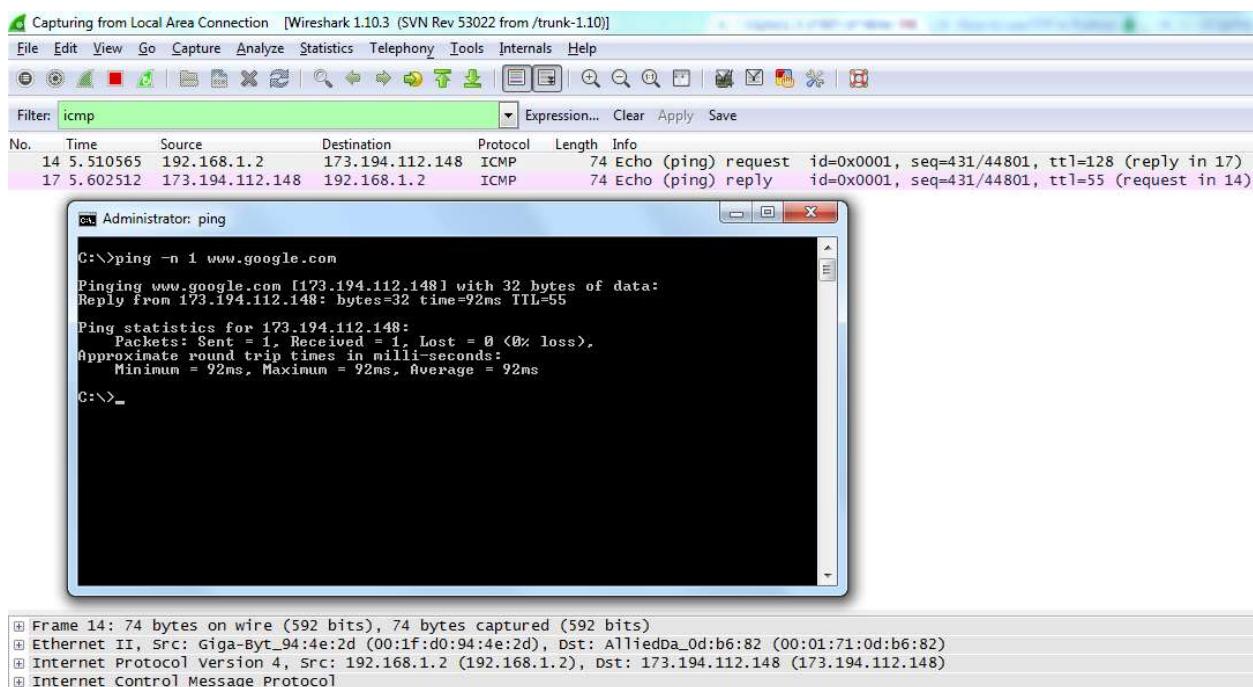


אפשר לראות שמוינעה גם הפקטה שהכילה את המידע מה-client (בעל כתובת ה-IP 192.168.1.1 במקורה שלנו) ל-server (192.168.1.2), וגם הפקטה שחזרה מה-server ל-client!

דוגמה נוספת יכולה להיות הסנפה של פקעת ping, אותו סיוק עלי' דיברנו בפרק הראשון. פתחו חלון cmd והריצו את הפקודה הבאה (וודאו, כמובן, שיש לכם הסנפה ברקע):

ping -n 1 www.google.com

ה-flag בשם -n קובע כי תישלח רק בקשה ping אחת, ולא ארבע בקשות כמו שנשלחות בדרך כלל.

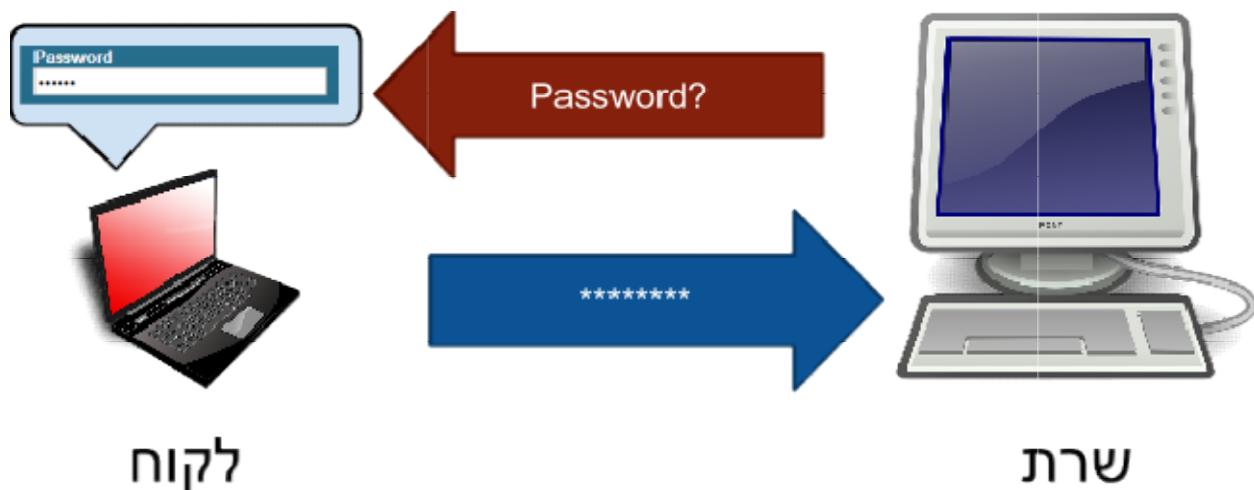


שיםו לב לתוצאה שהתקבלה: שלחנו בקשה אחת המכונה "Echo (ping) request", ניתן לראות זאת בעמודת ה-Info) אל השרת, וקיבלנו ממנה תשובה (Echo (ping) reply) מיד לאחר מכן. שימו לב גם אל כתובות ה-IP) של הפקטות – הפקטה הראשונה ובها הבקשה נשלחה מכתובת ה-IP שלם (192.168.1.2) אל השרת של Google 173.194.112.148), ואילו הפקטה השנייה ובها התשובה נשלחה בדיק בכיוון ההפור (מ-173.194.112.148). אל 192.168.1.2).

דרך אגב, הפילטר שהשתמשנו בו כדי להציג אר וرك את פקודות ה-פוק הוא הפילטר **icmp**, שהוא ה프וטוקול בו עוברות בקשות ותשובות pock. בהמשך הספר נרחיב את הדיבור על פרוטוקול זה ונלמד לעומק איך ping עובד. בנוסף, איפילו נכתב כל דמי pock בעצמו!

תרגילים מודרך 3.2 - הסנפת סיסמא גלויה

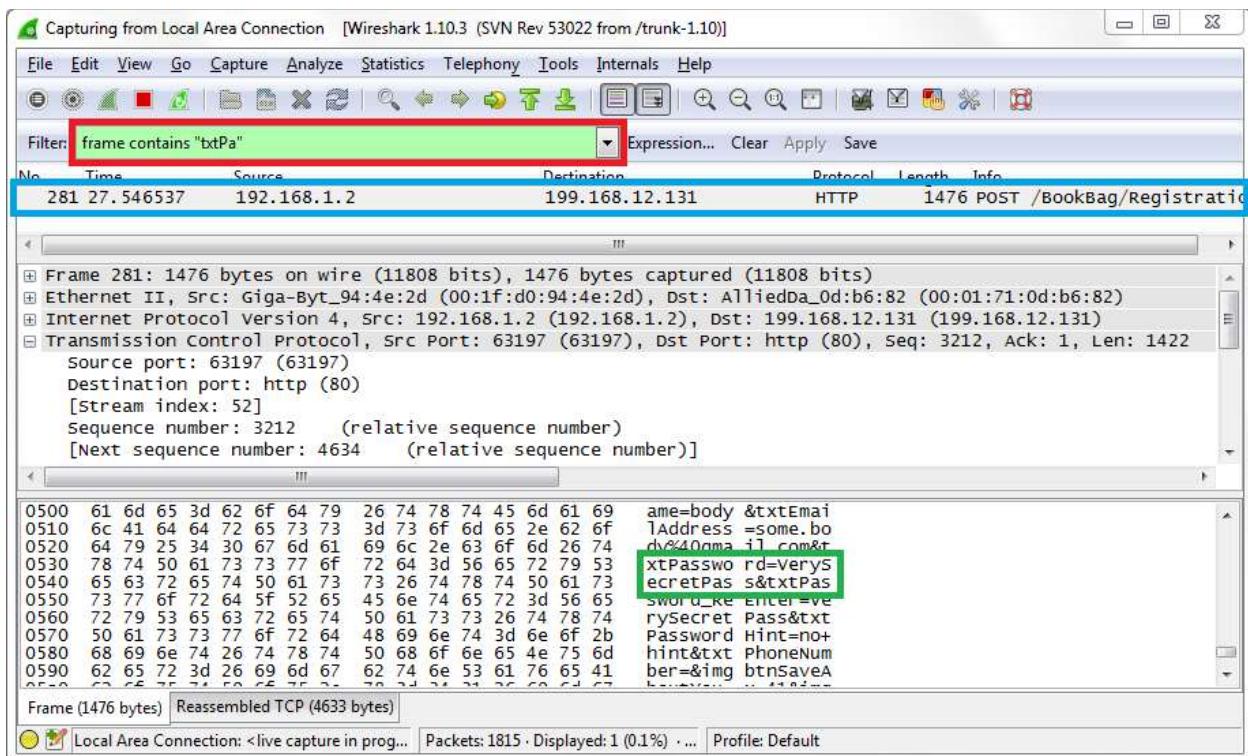
cutet נסתכל על דוגמה מגניבה יותר: חשבתם פעם מה קורה כשאתם מקלידים סיסמת כניסה לעמוד כלשהו באינטרנט (למשל, עמוד הכניסה אל תיבת המייל שלכם)? על המסר מוצגות כוכבות, אך כיצד השרת מאמת אתכם ומבודא שהסיסמה שלכם נכונה? האם נשלחות אליו הוכבות שאתם רואים על המסר?



כדי לענות על השאלה הזאת, היכנסו לעמוד <http://data.cyber.org.il/networks/links/plain-password.html> והכניסו את כל הפרטים הדרושים כולל סיסמה (אל תשים את הסיסמה האמיתית שלכם!) והירשמו לאתר. כמובן שבזמן הלחיצה על כפתור Register וודאו כי פועלת הסנפה ברקע. כתע, הכניסו את הפילטר הבא למסנן התצוגה:

frame contains "txtPa"

לחצו על Enter. צפיה להתקבל פקטה אחת שננתה על הפילטר. הקliquו עליה ברשימת הפקטות, ועכשו תוכלו – להסתכל על התוכן שלה בתחתית המסך. נסו לזרות משהו מפתח בחילוק האחרון של הפקטה, ורק אחריו שתגלו – עברו לפסקה הבאה.



נו, זיהיתם את הסיסמה שלכם שהועברה לשרת? מדהים, לא?

הדוגמה הקצרה זו הייתה אמורה להמחיש לכם דבר שallow' העליתם על דעתכם כבר קודם – כל דבר שנשלח לשוטר כלשהו ברשת י יצא מהמחשב שלכם ועובד דרך כרטיס הרשת, בין אם הוא גלוי לעיניכם (הטיקסט שבחרתם לשלחן דרך **Socket** בפייתון, פווט שאתם מפורטים בעמוד הפיסוק שלכם) או סמי מהן (הסיסמה שמוצגת בתור כוכיות באתר אך נשלחת בגלוי, או תזוזה של השחקן שלכם במשחק רשות אל מול שחקנים אחרים בעולם).

דרך אגב, אם הדוגמה האחורונה גרמה לכם לחוש שהסיסמה שלכם גלויה לכל מי שהיה נגיש למידע שיוצא מהמחשב שלכם ויכול להסניף אליו – הדאגה שלכם היגיונית, אולם אין לכם ממה לחושש. כיון, הרוב המוחלט של האתרים משתמש בשיטות אבטחה שונות כדי להבטיח שהסיסמה שלכם לא תעבור בצורה גלויה ברשת האינטרנט, אלא תוצפן אצל הלוק והפוענה רק כשהתגיע לשרת.

תרגיל 3.3 - שימוש בסיסי ב-Wireshark בעזרת שילוח בקשת ping

1. פתחו חלון cmd ורשמו **ping 8.8.8.8**. הסתכלו על הרסנפה ונסו למצוא את הפקודות שנשלחו (תזכורת: היעזרו בפילטר "icmp" בפינט שbowo עבורות הפוטוקול בקשות ה-ping).

כעת, חשבו את ה-round trip (משך הזמן החל מרגע שליחת הבקשה ועד קבלת התשובה) על-פי שדה ה-time Wireshark (שצורך מציג את הזמן בו עברו הפקטות בכרטיס הרשות, בפורמט של שניות מתחילה ההסנהה).
בדקו את עצמכם – האם הגיעם לאותה תוצאה שהופיעה בחלון cmd ?

2. השתמשו ב-flag בשם -l (שקובע את גודל המידע שיישלח בפקחת ping), והריצו את השורה הבאה:
ping -l 500 8.8.8.8 -c 1

גודל הפקטה שנשלחה כעת שונה מגודל הפקטה שלחנו קודם (כאשר לא השתמשנו בflag שמצין את גודל המידע שנשלח). נסו למצאו לפחות שני מקומות Wireshark בהם אפשר לראות שגודל הבקשה זהו שונה מגודל הבקשה הקודמת.

3. הסתכלו על תוכן הפקטה שנשלחה בסעיף 2, בחילון המציג את תוכן הפקטה. בקשת ping מכילה בתוכה מידע, והוא מzystה לקבל את תשובה ping עם אותו מידע בדיק (בדומה למה שעשיתם בפרק תכנות ב-Sockets כשימושם שרת הדימ). ב-Wireshark, ניתן לראות את המידע הזה בשדה "Data" של פקחת ping. תוכלן להזות מהו המידע שנשלח בבקשת ping אל השירות?

Follow TCP/UDP Stream

אפשריה נוספת לך להפלייה היא Follow TCP Stream או Follow UDP Stream (השימוש הראשון הוא הרבה יותר נפוץ). אפשרות זו שופעלת על פקטה, מפלתרת את כל הפקטות השייכות אותה "Stream" (כלומר נשלחו והתקבלו דרך אותו Socket¹⁹), ובכך מאפשרת לראות את התעבורה "דרך העניינים של ה-Socket באפליקציה". נדגים את השימוש באפשרות זו בעזרת התרגיל שכתבתם בפרק [תכנות ב-Sockets](#) / [תרגיל 2.6](#):

[שרות פקודות בסיסי](#):

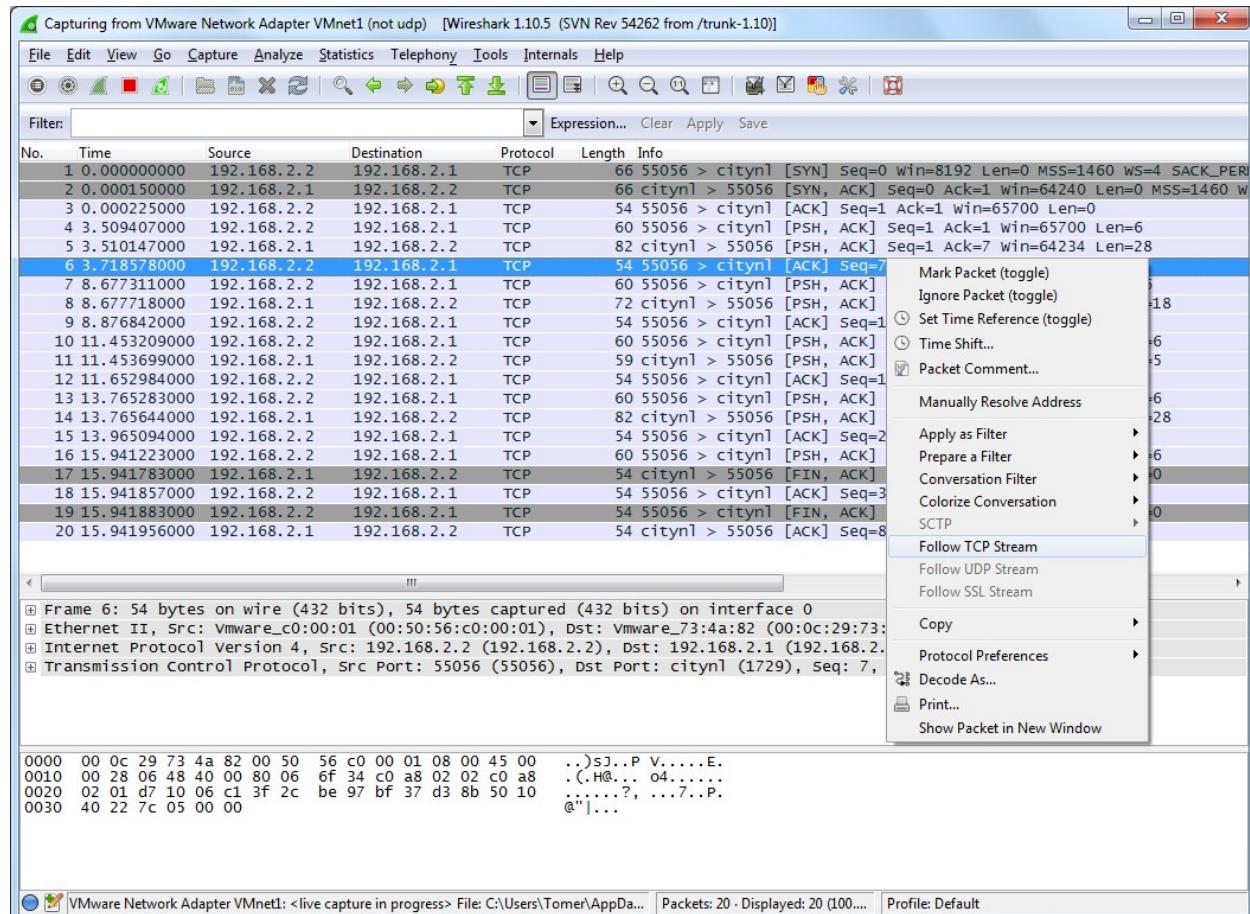


תרגיל 3.4 מודרך - צפייה במידע של Socket בעדרת Stream

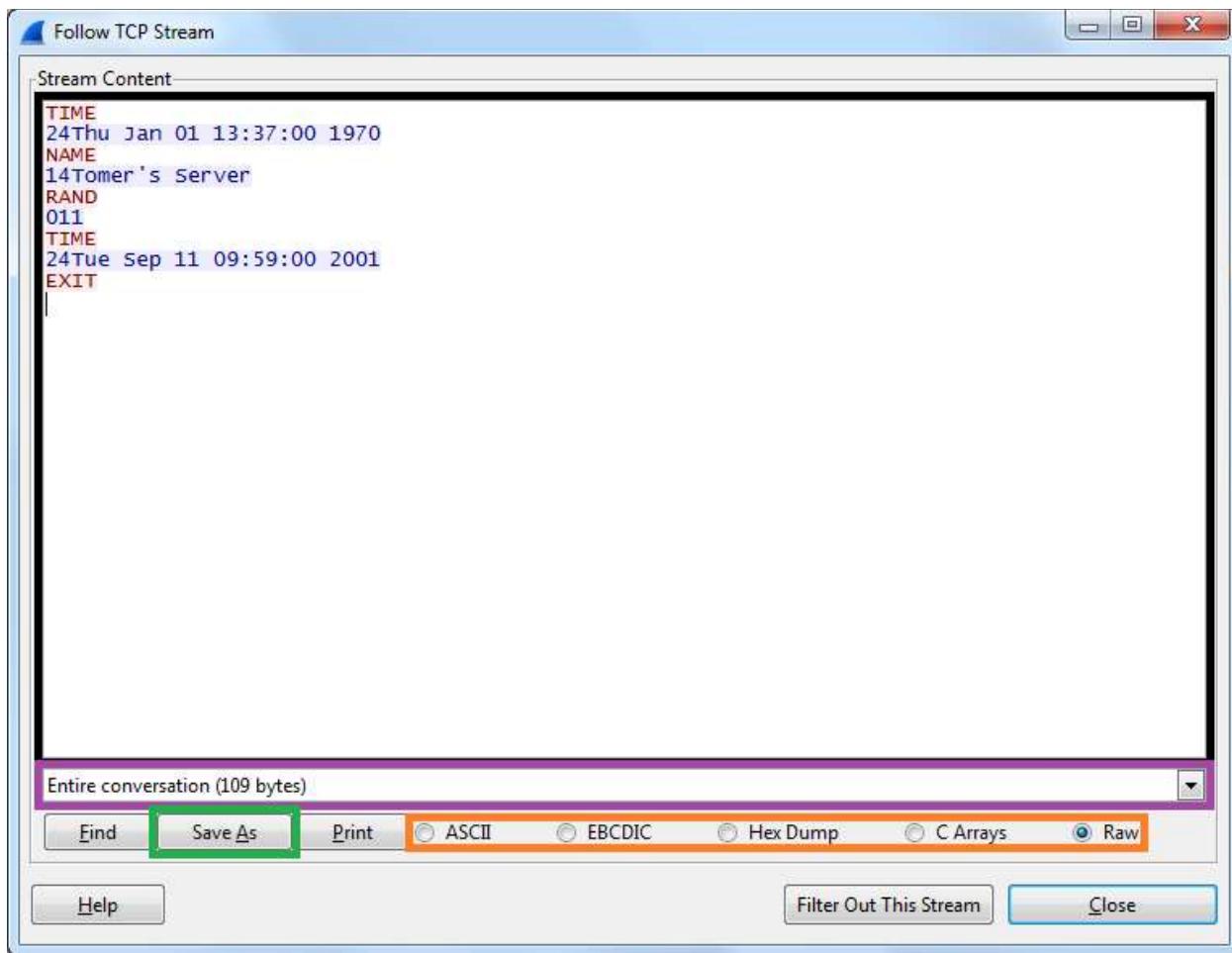
- השתמשו בלקוח שכתבתם בתרגיל שרת פקודות בסיסי (TIME, RAND, NAME, EXIT).
- שנו את קוד הלקוח כך שהוא יפנה אל שרת גבהים שנמצא כתובות בכתובת networks.cyber.org.il בפורט 8850 (מה כתובות ה-IP אליה יש לפנות? מיצאו אותה בעצמכם בעזרת הכלים שלנו).
- התחילה הנסנהה חדשה והריצו את הלקוח. פיתחו את ההסנהה.

¹⁹ למעשה, הכוונה היא לכל הפקטות שהן חלק מה-stream בשכבת התעבורה. על המשמעות של מושג זה, וכייז Wireshark מצליח להבין אילו פקטות שייכות לאיזה Stream - נלמד בפרק [שכבת התעבורה](#).

- בחרו את אחת הפקודות שהועברו בין שני המחשבים כחלק מאותו ה-Socket, לחצו על הכפתור הימני של העכבר ובחרו באופציה ".Follow TCP Stream"



- בעת נפתח חלון המציג את כל המידע ש עבר על גבי ה-Socket



שים לב, כי מופיע כאן רק ה-Data השיר לשכבות האפליקציה, ללא כל ה-Header'ים והפרמטרים ששימושו לחיבור בין שתי נקודות הヅפה. אם תסתכלו על המסגרת השחורה, תראו שככל צד בתקשרות מופיע במצב אחר - **באדוֹן** התקשרות בין הלקוח לשרת, ובכחוֹל התקשרות בין השרת ללקוח. ניתן להזיהות את הפרוטוקול בו בחרנו להעביר את המידע חזקה מהשרת ללקוח: שליחה של 2 ספירות המסמלות את אורך התשובה, ומיד לאחריהן נשלחת התשובה עצמה.

בנוסף, ניתן להגביל את הצפיה רק לאחד הצדדים של התקשרות (המסגרת **סגוליה**), לבחור את הייצוג בו נרצה לצפות במידע (המסגרת **כתומה**) וכן לשמור את המידע לקובץ ובכך לקבל את כל המידע שעבר ב-**Socket** (המסגרת **ירוקה**).

סטטיסטיקות

Wireshark היא תוכנה חכמה - היא מבצעת ניתוח לכל הפקודות שmag'יות ומסיקה מהן מסקנות. כתוצאה לכך, אנו יכולים לקבל תוצאות סטטיסטיות נחמדות שנמצאות תחת תפריט Statistics: אורך הפקודות בהסופה (Packet Lengths), גוף שמתעדכן בזמן אמת ומציג את כמות התעבורה (IO) וישיות רשותן שנקלטו בהסופה (Endpoints).

Endpoints: test.pcap

IPv4 Endpoints								
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Latitude	Longitude
192.168.1.3	110	16 581	60	4 727	50	11 854	-	-
192.168.1.1	6	504	3	274	3	230	-	-
192.168.1.2	118	21 931	55	13 479	63	8 452	-	-
140.112.253.189	1	96	0	0	1	96	-	-
205.227.136.203	3	1 169	1	240	2	929	-	-
83.170.75.178	10	4 589	5	3 715	5	874	-	-

Name resolution Limit to display filter

Help Copy Map Close

בנוסף, יש את חלון ה-**Conversations**, שמציג את כל השיחות (תקשורת דו-צדדית בין שתי ישויות ברשת) שעלו בהסופה.

Conversations: test.pcap

IPv4 Conversations													
Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start	Duration	bps A→B	bps A←B		
192.168.1.1	192.168.1.3	6	504	3	274	3	230	0.017208000	2.9410	745.32	625.64		
140.112.253.189	192.168.1.2	1	96	0	0	1	96	1.116040000	0.0000	N/A	N/A		
192.168.1.2	192.168.1.3	104	16 077	47	11 580	57	4 497	25.493358000	74.2394	1247.86	484.59		
192.168.1.2	205.227.136.203	3	1 169	2	929	1	240	48.951858000	0.0046	1629467.22	N/A		
83.170.75.178	192.168.1.2	10	4 589	5	3 715	5	874	73.102744000	0.0741	400965.99	94332.24		

Name resolution Limit to display filter

Help Copy Follow Stream Graph A→B Graph B→A Close

תרגיל 3.5 - דיהוי שרתיים בחילון ה-Endpoints וה-Conversations



השתמשו בכלи **ping** שהכרתם בפרקם הקודמים, על מנת לבדוק קישוריות אל ynet.co.il, google.com, il-il.it, walla.co.il. נסו למצוא את השיחות שלכם עם אותן האטרים בחילון ה-**Conversations** (כבר למדתם כיצד להמיר בין ה-**Domain** לבין IP, כך שתctrco לזרות שיחות בין ה-IP שלכם ל-IP של השירות המרוחק). בנוסף, נסו לזרות אותן גם בחילון ה-**Endpoints**.

מודל חמש השכבות - סיכון Wireshark

בפרק זה נחשפנו לראשונה לכלי **Wireshark**, שהוא כלי רב-עוצמה שמאפשר לנו לבדוק את המידע שיוצא ומתקבל בכרטיס הרשת שלנו. בהמשך הספר נעשה שימוש נרחב ב-Wireshark, כדי למדוד עוד על ה프וטוקולים השונים ולהבין כיצד הם עובדים.

התחלנו מהסנהפה של שירות החדים שכתבתנו בפרק הקודם, הסנונו גם שימוש בכלים **ping** ו-**arping** ועוד. שעובדות בכלל ברשת האינטרנט – הכל כדי שתיווכחו כמה כוח יש ל-Wireshark ומה אפשר לעשות באמצעותו.

לאחר מכן דיברנו על ענן האינטרנט, ועל הצורך לארגן את המידע שעובר בו בצורה טובה. הצגנו את **מודל חמש השכבות**, מדוע צריך אותו ואילו יתרונות הוא מספק. הראינו כיצד השכבות מדברות האחת עם השנייה ואיך המידע עובר בפועל (כלומר כיצד בנייה פקטה במודל חמש השכבות).

סיימנו את החלק בהציג **קצירה של כל אחת מהשכבות** (איזה שירות היא מספקת לשכבות שמعلיה, ואיזה שירות היא מקבלת מהשכבות שמתחילה), באופן שהוואה מפת דרכים להמשך הלימוד בספר. כתה, שנצללו עמוק ונלמד על כל שכבה, תוכלו להבין היכן היא ממוקמת במודל חמש השכבות ומה תפקידה באופן כללי.

לסיום, הצגנו **אפשרויות מתקרבות של Wireshark** כמו התעסקות עם קבצים, מסננים וսטטיסטיות. אלו מקווים שספגתם קצת מההתלהבות לגבי התוכנה, ושאתם רק מוכנים להשתמש בה כדי לחזור ולמדוד עוד על נושאים שאתם לא מכירים ברשותם.

בהמשך הספר נשתמש בידע שרכשנו בפרק זה ב כדי למדוד לעומק על שכבות, פרוטוקולים ורכיבי רשת שונים.

פרק 4 - שכבה האפליקציה

מצויים בכלים שלנו - הסנפת תקשורת בין מחשבים ותכנות Sockets בפייתון, ולאחר מכן על מודל חמש השכבות, אנחנו מוכנים להתחיל ולחזור את השכבה הראשונה שלנו - שכבה האפליקציה.



ניתן לצפות סרטון "מבוא לשכבה האפליקציה" בכתבוב:
<http://youtu.be/yftdGTiEP8A>

אפליקציות (ישומים, בעברית) - כינוי לתוכנות שבן אנחנו עושים שימוש במחשב, וזה מושג שנעשה הרבה יותר נפוץ ומוכר מאז שה咍 השים הנרחב בסמארטfony ובטאבלטים. גם ישומים שרצים על המחשב שלנו (דוגמא נפוצה - דפדףים), וגם אפליקציות ב-iPhone או-Android (כמו Whatsapp או האפליקציה של Facebook), עושים שימוש בתקשורת דרך האינטרנט כדי לשלוח ולקבל הודעות (Whatsapp), להעלות תמונות (Facebook / Instagram), לקבל מיילים (Gmail) ועוד.



שכבה האפליקציה היא אוסף הпротокולים בהם עושים שימוש באופן ישיר, והיא מספקת הפשתה מעלה תקשורת הנתונים בראש האינטרנט.

עד סוף הפרק, נבון בבדיקה מה המשפט הזה אומר.
מעבר לכך, נכיר לעומק איך עובד פרוטוקול HTTP ואת יכולות שהוא מספק, ונתבונן כיצד אתרים ואפליקציות כמו Facebook או המפות של Google משתמשות בו. בנוסף - נಮש בעצמנו שרת אינטרנט, ולקוח שמתקשר אליו. בהמשך, נלמד על פרוטוקול DNS ודרך הפעולה שלו.

פרוטוקול HTTP - בקשה ותגובה

נתחיל עם הפרוטוקול הנפוץ ביותר של שכבה האפליקציה - HTTP - המשמש לגלישה באינטרנט.
לפני שנכבר במיילים, נתחיל בהסתכלות על הפרוטוקול.



ניתן לצפות סרטון "מבוא ל-HTTP" בכתבוב:
<http://youtu.be/BS46e9GYHNI>



מהו מושב רשות?

פרוטוקול HTTP מיועד לאפשר לאפליקציות לגשת ולעשות שימוש במשאבי-רשות באינטרנט. בשלב זה יש לנו הינה של מהי אפליקציה.icut ננסה להבין מהו מושב-רשות. למשל: שירות המפות של Google, הוא מושב רשות. כמו כן, עמוד ה-*Facebook* (כינוי נקרא *timeline*) שלו הוא מושב ברשות. כך גם חשבון Twitter שלו, וכן גם כתבה בפייסבוק.



תרגיל 4.1 מודרך - התנסות מעשית בתקשרות HTTP

לצורך כך הפעילו את Wireshark ותחווilo הסנפה עם "http" בתור פילטר.

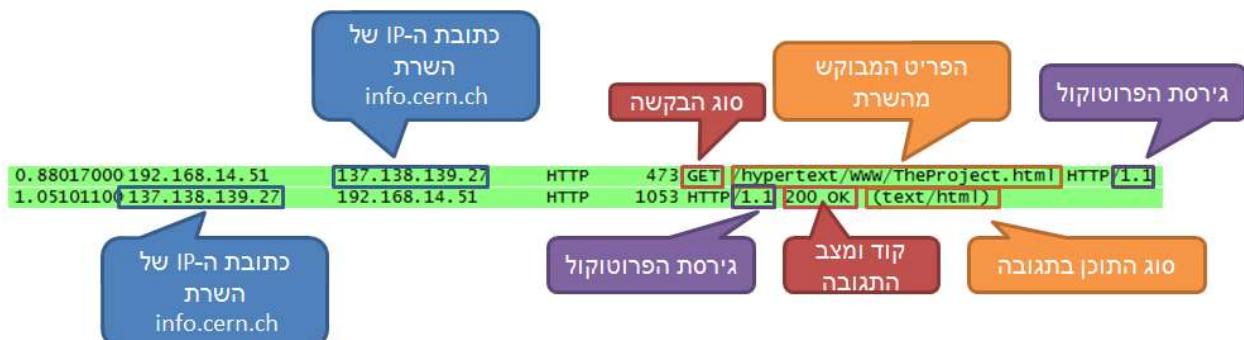


icut פתחו את הדף החביב עליהם, והכינוו את הכתובת הבאה:
 http://info.cern.ch/hypertext/WWW/TheProject.html
 לאחר שהדף יטען, תראו עמוד קצר ובו מספר שורות.



איך הדף נטען בדף שלנו? מה בעצם קרה כאן?

נחזיר אל Wireshark כדי למצוא את התשובה - עצרו את ההסנפה, ושימו לב שנוצרו שם שתי שורות:



עיקרון "בקשה-תגובה"

סוג התקשרות שראינו כאן משתמש בשיטת תקשורת שנקראה "בקשה-תגובה", ובה מחשב מבקש מידע מסוים ממחשב אחר על ידי שליחת בקשה מסוימת, ובתגובה המחשב השני מודיע לו את המידע הרלוונטי.

במקרה שלנו, הלוקה מעוניין בדף אינטרנט בעל כתובת כלשהי, ועל כן שולח בקשה אל השירות (שאלהיה אפשר להתייחס כ-”אנו השב לי את העמוד בכתב הזו”). לאחר מכן השירות ישלח תגובה (שאלהיה אפשר להתייחס כ-”הנה עמוד האינטרנט”).



כאשר דיברנו בפרק הראשון על שליחת בקשה ל-Facebook וקבלת התשובה ממנו, הפעולה שתארנו הייתה למעשה בקשה ותגובה ב프וטוקול HTTP.

שורות הבקשה של HTTP

הפקטה בשורה הראשונה היא הבקשה שנשלחה מהלוקה (במקרה זה - מהדפן שלכם) אל שירות באינטרנט. בשורה השנייה אנחנו רואים את התשובה שקיבל הדפן שלכם מאותו שירות, ואוותה ננתה בהמשך. נתבונן בivid ובין איך הפרטוקול בני - שימושם לב שפaketת הבקשה ניתנה הכוורת:

GET /hypertext/WWW/TheProject.html

המילה GET מצינית לנו בקשה HTTP מסוג GET (בהמשך גם נלמד על סוגים נוספים ב-HTTP), שנועדה להביא פריט מידע כלשהו מהשירות באינטרנט שמסתתר מאחורי הכתובת info.cern.ch.

שורות התגובה של HTTP

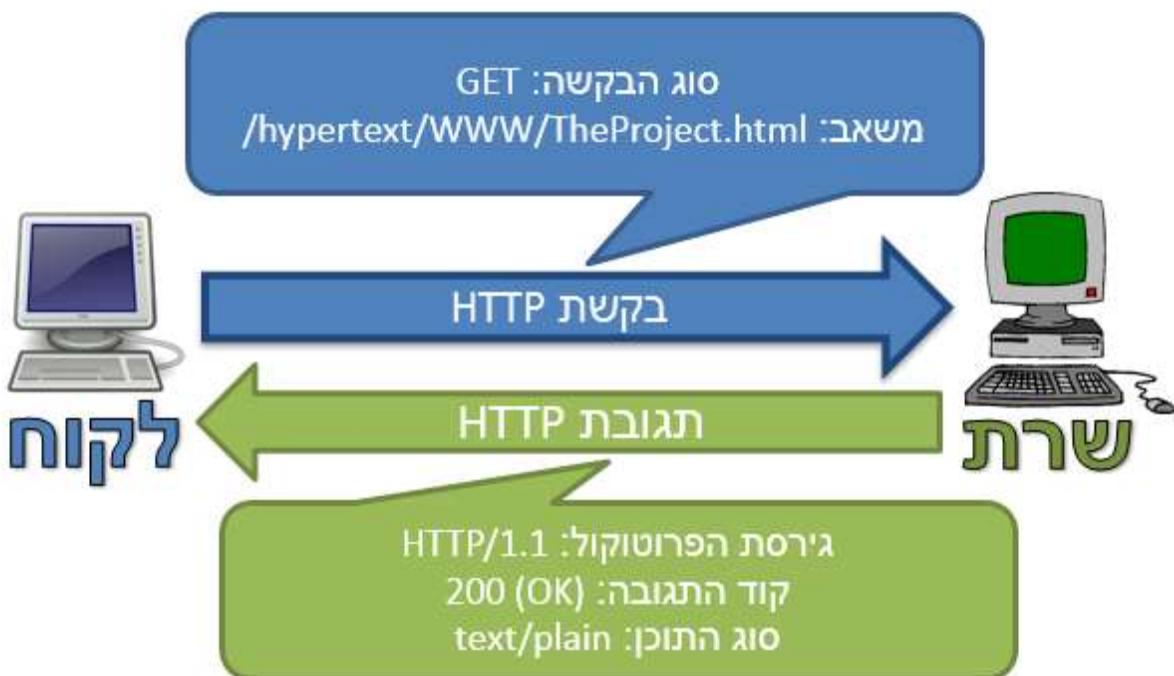
בשורות התגובה מופיעו "HTTP/1.1 200 OK" (text/plain) - למעשה יש בשורה הזו שלושה נתונים:

1. גירסת פרוטוקול HTTP שבה השתמשנו (1.1)
2. מצב התשובה - בפרטוקול יש מספר קודים מוגדרים מראש כדי לתאר את מצב התשובה; כשמדובר בתשובה תקינה, מקבל תשובה עם קוד 200 שאומרים שהכל בסדר (OK), אם הייתה בעיה כלשהי קיבל קוד אחר שאמור לרמז על סוג הבעיה (לדוגמא, קוד 404 המפօרט שהמושא המבוקש לא נמצא).

את הרשימה המלאה ניתן למצוא בעמודו: <http://goo.gl/COC4J7>

3. סוג התוכן שבתשובה - במקרה זה מקבל טקסט. במקרים אחרים ניתן לקבל בתגובה תמונה, סרטיון יידאו או קוד.

סיכום ביניים של התקשרות שריאנו



תרגיל 4.2



הסניף באמצעות Wireshark עם פילטר "http" בזמן שתכניותו בדף את הכתובת:

<http://www.lilmodtikshoret.com/notfound>

מצאו את פקודות הבקשות והתשובות, ובדקו בפקחת התגובה מהו הקוד וסוג התוכן שהתקבל.

מה מסתור בתוך הבקשה/תגובה? Headers ("מבוא" ב-HTTP)

בנוסף לשולשות הפרמטרים שראינו בשורת הבקשה, תקשורת HTTP לרוב מכילה שדות מידע נוספים, מעבר לתוכן שעובד. שדות אלה נשמרים בשורות שמויפות אחריו שורת הבקשה/תגובה, ולפניהם (Data-Header, ונקראות HTTP Headers (בעברית - "שורות כותרת" או "מבוא"). למעשה, כל שורת Header מכילה שם של שדה והערך שלו, כשהם מופרדים על ידי נקודות (:).

למשל, בדוגמה הבאה ניתן למצוא, מיד לאחר שורת הבקשה, שדה Header בשם "Accept", לאחריו שדה Header בשם "Accept-Language" (שהערך שלו הוא 'he', כלומר - עברית), ושדות נוספים. שדה ה-Header האחרון בבקשת זו נקרא "Connection":

```

Frame 440: 615 bytes on wire (4920 bits), 615 bytes captured (4920 bits) on interface
Ethernet II, Src: Elitegro_28:2d:e9 (10:78:d2:2d:e9), Dst: Tp-LinkT_eb:cf:7a (94:0c:11:eb:cf:7a)
Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 137.138.139.27 (137.138.139.27)
Transmission Control Protocol, Src Port: 64951 (64951), Dst Port: http (80), Seq: 1, A
Hypertext Transfer Protocol
GET /hypertext/www/TheProject.html HTTP/1.1\r\n
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, i
Accept-Language: he\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; windows NT 6.1; WOW64; Trident/4.0; S
Accept-Encoding: gzip, deflate\r\n
Host: info.cern.ch\r\n
Connection: Keep-Alive\r\n
\r\n
[Full request URI: http://info.cern.ch/hypertext/www/TheProject.html]

```

חלק משדות ה-Header יכולים להופיע גם בבקשתו וגם בתגובה (למשל: אורך התוכן), חלק יופיע רק בבקשתו (למשל: סוג התוכן שהליך מוכן לקבל בחזרה, "סוג" הלקוח - לדוגמה: דפדפן כרום) וחלק יופיע רק בתגובה (למשל: "סוג" השירות).

רשימה של שדות Header ניתן למצוא בכתובות:

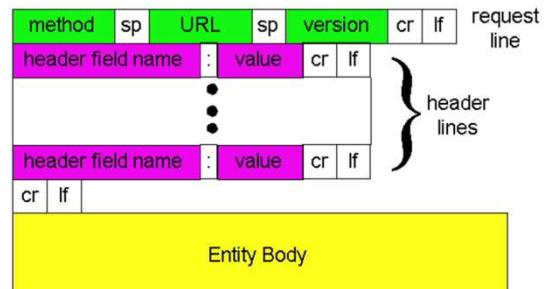
http://en.wikipedia.org/wiki/List_of_HTTP_header_fields

מבנה פורמלי של בקשת HTTP

```

GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n

```



הבקשה היא מחוץ טקסטואלית, שומרכבות משלושה חלקים: שורת הבקשה, שדות ה-Header ותוכן הבקשה (בשלב זה, נתעלם מהחלק השלישי, שכן בקשות GET לא מכילות תוכן).

כדי להפריד בין שורה לשורה, נהוג להשתמש ברכף של שני תווים - ז' ומיד אחריו ²⁰.

²⁰משמעות התו ז' הוא carriage return, לרוב מיוצג על ידי הסמל ↵, ומקורו במכונות הכתיבה. תפקידו של מקש זה הוא להחזיר את ראש הכתיבה לתחלת השורה. משמעותו התו ח' הוא line feed - ירידת שורה. צירוף שני המKeySpecים הללו מאפשר להתחיל שורת כתיבה חדשה.

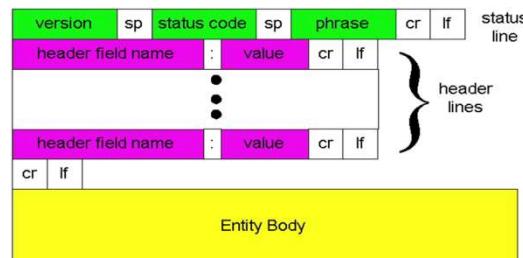
מכיוון ששורת הבקשה היא שורה אחת בלבד, ניתן לצפות שמיד עם סיום השורה (הופעת התווים ח'ז') יתחלו שדות ה-Header. כל שורה בחלק זה מכילה שדה אחד ואת הערך שלו. לאחר שדה ה-Header האחרון, יופיע כרגעיל "סיום שורה" (ח'ז'), והשורה שלאחר מכן תהייה ריקה - כלומר מיד יופיעו שוב ח'ז'. זהו סימן לכך שלב ה-Header הסטיים, וכל שאר המחרוזת תכיל את תוכן הבקשה, בוណון בשלב מאוחר יותר בפרק זה.

כדי להבין טוב יותר את הشرطוט שמצוג בצד ימין: הקיצור `sp` משמעתו התו רווח (space), הקיצור `z` הוא התו ז', והקיצור `If` הוא התו ח'ז.

בצד שמאל נתונה דוגמה לבקשת GET - נסו לזהות את החלקים השונים בבקשת (תזכורת: חלק התוכן יהיה ריק), את התווים המפרידים ביניהם, ונסו לזהות שדות Header שנדרשו בהם בפסקה הקודמת.

מבנה פורמלי של תשובה HTTP

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```



בדיקן כמו הבקשה, גם תגובה HTTP היא מחרוזת טקסטואלית שבניה בבדיקה באמצעות צורה - אוטם שלושה חלקים, ואחתה דרך המפריד ביניהם. עיברו על הדוגמה בצד שמאל, וודאו שגםם מזהים את החלקים והמפרידים ביניהם, חפשו שדות Header מוכרים ונסו להבין את המשמעות שלהם.

בדוגמאות שראינו עד כה, יש משמעות גם לחלק השלישי - התוכן בתגובה ה-HTTP (שם מועבר תוכן דף האינטראקט) - ועל כך בפיסקה הבאה.

תוכן (מידע - Data - ב-HTTP)

از ראיינו תקשורת בסיסית של בקשה ותגובה, אבל למרות שקוד התגובה, סוג התוכן ושדות ה-Header משמשים כמידע די מעניין, אנחנו (בטעור משתמשים) מעוניינים בתוכן עצמו. בנוסף של דבר, האפליקציה צריכה לקבל את התוכן כדי להציג לנו תמונה, הודעה או מסמך.

לבסוף, האפליקציה תציג את התוכן שנמצא בתגובה - למשל תenga סרטון וידאו או תציג הودעת טקסט. צפתה לקבל תמונה וחזר תוכן מסווג `text`, כנראה שיש בעיה.

לאחר מכן, האפליקציה תבדוק את סוג התוכן כדי לוודא שהוא תואם את מה שהיא ציפתה לקבל - למשל, אם היא צפתה לוודא שהיא מושא מסויים, היא תבדוק את קוד התגובה כדי לוודא שהיא תקינה - למשל, תזودה שהתקבל קוד OK 200 (כמו שראינו בדוגמה הקודמת). אם למשל התקבל קוד 404 (משאב לא נמצא), היא תציג הודעה שגיאה מתאימה.

התקבצות מודרנת בתגובה HTTP



הגיע הזמן שנתבונן בתוכן של תשובות HTTP - נחזר לדוגמה שראינו ב-Wireshark, ועכשו "נפתח" את פקעת התגובה (ונראה את מה שקרהנו לו בתרשים "תגובה HTTP" - [HTTP Response](#)):

The screenshot shows an HTTP session in Wireshark. The request (row 528) is an HTTP GET for the homepage. The response (row 531) is an HTTP 200 OK with the content of the page. Red arrows point from the labels 'header' and 'body' to their respective parts in the captured data.

No.	Time	Source	Destination	Protocol	Length	Info
528	21.277030000	192.168.1.100	137.138.139.27	HTTP	618	GET /hypertext/www/TheProject.html HTTP/1.1
531	21.352697000	137.138.139.27	192.168.1.100	HTTP	1077	HTTP/1.1 200 OK (Text/html)
542	21.352777000	192.168.1.100	137.138.139.27	HTTP	388	GET /favicon.ico HTTP/1.1
545	21.354683000	137.138.139.27	192.168.1.100	HTTP	267	HTTP/1.1 200 OK (Text/html)

```

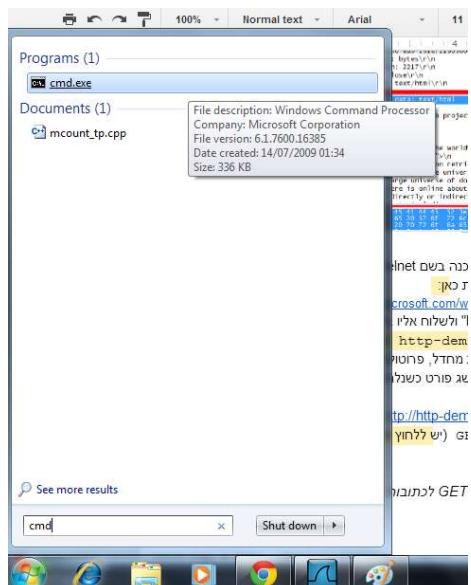
# Transmission Control Protocol, Src Port: http (80), Dst Port: 53187 (53187), Seq#: 1448, ACK#: 553, Len: 1023
[2] Reassembled TCP Segments (2458 bytes): #530(1448), #531(1011)
[2] Hypertext Transfer Protocol
[2] HTTP/1.1 Transfer Protocol
[2] HTTP/1.1, 200 OK[1]
Date: Tue, 17 Dec 2013 10:46:08 GMT\n
Server: Apache/2.4.1\n
Last-Modified: Thu, 03 Dec 1992 08:37:20 GMT\n
Etag: "40521e0c-8a0-291e721005000"\n
Accept-Ranges: bytes\n
Content-Length: 2217\n
Connection: close\n
Content-Type: text/html\n

[2] Line-based text data: text/html
<HEADER>\n<TITLE>the world wide web project</TITLE>\n<NEXTID N="55">\n</HEADER>\n<BODY>\n<H1>a1d wide web</H1><P>the world wide web (w3) is a wide-area<A>\nNAME=<A HREF="whatis.html">\nhypermedia</A> information retrieval<br>\ninitiative aiming to give universal<br>access to a large universe of documents.<P>\nEverything there is online about<br>w3 is linked directly or indirectly<br>to w3</P>

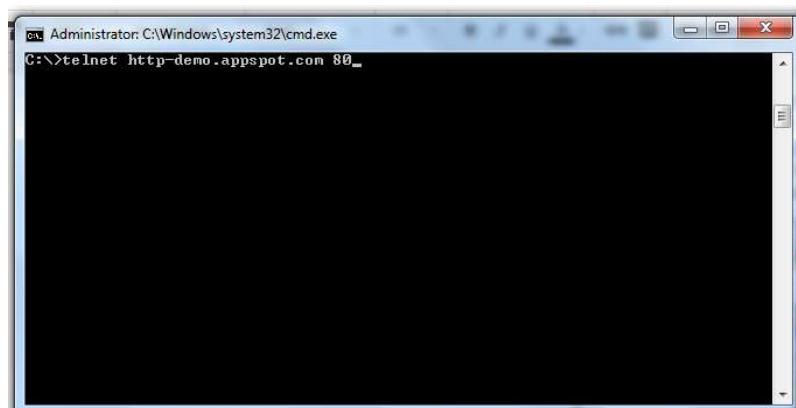
```

כדי לבדוק את התקשרות הzo بصورة נוחה יותר, ניתן להשתמש בתוכנה בשם **telnet** (במקום הדפדפן). הנחיות כיצד להפעיל אותה על מערכת הפעלה Windows 7 נמצאות בכתבoted: <http://goo.gl/Oo9DqK>

על מנת ליצור חיבור לשרת של האתר "http://http-demo.appspot.com" ולשלוח אליו בקשות GET, נפתח את שורת



.telnet http://http-demo.appspot.com 80 (Command line (ה-CLI), ונ裏ץ את הפקודה הבאה:



משמעות המספר "80" היא הפורט בשרת אליו נתחבר - כבירה מחדל, פרוטוקול HTTP העושה שימוש בפורט 80 (ענין הפורט לא נדרש לצורך ההבנה של פרק זה, וילמד עמוק בפרק שכבות התעבורה. לעומת זאת, היזכרו בשרת שבנו בפרק תכנות ב-Sockets 2.3 מודרן - השרת הראשון שלו, ולצורך מה שimes מס' הפורט).

הבנו קודם לכן שכאר שולחים בקשת GET, מבקשים משאב ספציפי מן השרת. גם למשל, כאשר מבקשים את המשאב `a.html`, אנו שולחים GET בצורה הבאה:²¹

`GET /a.html HTTP/1.1`

במידה שלא נבקש אף משאב בצורה ספציפית, אנו נפנה למשאב שנמצא בכתובת "/", כר:

²¹ יתכן שנצרך להזיף Header Host מסוג Host כדי שהשרת יסכים לקבל את הבקשה. לשם כך, יש להקיש ב-`telnet GET /a.html HTTP/1.1`
Host: http://http-demo.appspot.com

כאשר נסביר על Headers בהמשך הפרק, נבין את המשמעות של שורה זו.

GET / HTTP/1.1

או בציון כתובת מלאה:

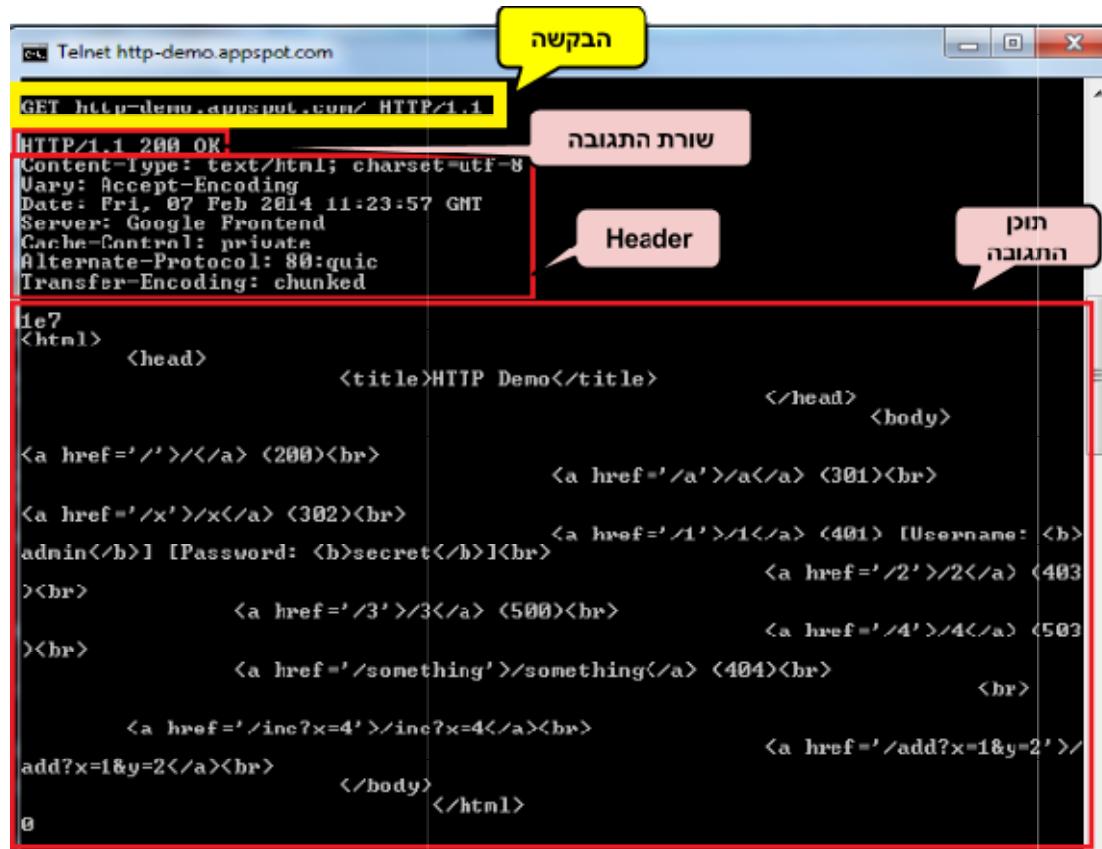
GET www.google.com/ HTTP/1.1

בעת גליהא של משתמש לאוֹר. כתע נבקש את המשאב שסקול לגלישה לכתחובות:

<http://http-demo.appspot.com> – אותו המשאב שנמצא בכתובת "/".

GET http://demo.appspot.com/ HTTP/1.1

(יש ללחוץ פעמיים על Enter). שימו לב לתשובה שהתקבלה:



4.3 תרגיל

כתבו מהו התוכן ואילו שדות Header מתקבלים בתגובה לבקשת GET לכתובות הבאות:

1. <http://http-demo.appspot.com/a>

2. <http://http-demo.appspot.com/x>

3. <http://http-demo.appspot.com/2>

4. <http://http-demo.appspot.com/3>

5. <http://http-demo.appspot.com/4>

6. <http://http-demo.appspot.com/something>

פרוטוקול HTTP - תכונות צד שרת בפייתון, הגשת קבצים בתגובה לבקשת GET

אם נזכיר לרגע בפרק [תכנות/ Shart-Lkutn](#), נבין שני סוגי רכיבים משתתפים בכל תקשורת צדו - צד השירות, שמספק גישה למשאים (גנוח, מיילים), וצד הלקווח, שהוא למעשה אפליקציית הקצה שבה אנחנו משתמשים, וմבקש משאים מן השירות. בדוגמה שראינו זה עתה, המשאים הוי דפי אינטרנט טקסטואליים, אותם סיפק השירות בתגובה לבקשת של הלקווח (הדף או הכל [telnet](#)). למעשה, השירות עונה לבקשתות ש מגיעות מרבבה לקוחות, ובינתיים אנחנו מפשיטים ומדברים על לקוח בודד. נדבר בהמשך על ההשלכות של "טיפול" בלקוחות רבים.



מעט רקע היסטורי - בתחילת דרכה של השימוש הביתי בראשת האינטרנט, בסביבות שנות ה-90', השירות אינטרנט שימשו בעיקר ל"הגשה" של עמודי תוכן סטטיים. הכוונה ב"הגשה" היא להעביר את התוכן של הקובץ שנמצא בכתובת שביקשה האפליקציה. למשל, בדוגמה שראינו בחלק הקודם, השירות הגיש לדףון קובץ טקסט.

נתעכט לרגע כדי להבין איך בנוiot הכתובות הללו: לכל אתר בראש יש כתובת ייחודית, שמכונה URL - ראשי תיבות של Universal Resource Locator (בעברית: "כתובת משאב אוניברסלית"). למעשה, ה"כתובות" שהכנסנו לדףון בסעיף הקודם הוי URLs. על משמעות המילה "משאב" נרחיב עוד בהמשך. URL בסיסי בניו באופן הבא:



במקרה זהה, השרת שמאחוריו הכתובת `www.site.com` יחפש בתיקיה `a` תיקיה בשם `b` ובתוכה יחפש קובץ בשם `file_name.txt`. הפניה מתבצעת בסכמה של HTTP²². אם אכן קיים קובץ זהה, השרת יגיש אותו בתור תגובה לבקשתו.

כדי להבין טוב יותר את הרעיון, תමמשו כתע בעצמכם שרת זהה, שmagish קבצים בתגובה לבקשתו. אל דאגה, התרגיל מפורט, ומפורק למשימות קטנות מאד.

לפני כן, נסביר מעט יותר על המימוש של **root directory**. כפי שציינו קודם, הפניה למשאב מתבצעת כמו במערכת קבצים – פניה ל-`"\folder_a\folder_b\file_name.txt"` למשא פונה לקובץ `file_name.txt` בתיקיה `folder_b` שבתיקיה `folder_a`. אך היכן נמצאת תיקיה `folder_a`? היא נמצאת בתיקיית השורש, `root directory`, של האתר. בפועל, התיקיה הזו היא פשוט תיקיה כלשהי במחשב שהמتنנת הגדר אותה בתור `root directory`. בחירה נפוצה היא להגדיר את `root directory` כעת, כאשר הליקוח פונה ושולח בקשה מסווג:

GET /folder_a/folder_b/file_name.txt HTTP/1.1

הבקשה תתבצע למשא לקובץ הבא אצל השרת²³:

`C:\wwwroot\folder_a\folder_b\file_name.txt`

תרגיל 4.4 – כתיבת שרת HTTP

לאחר כל אחד מהשלבים הבאים, הקפידו לבדוק את השרת שלכם על ידי הרצה של התוכנית, ושימוש בדף קלוקה; היזכרו במשמעות של הכתובת `127.0.0.1` אותה הזכרנו בפרק תכונות ב-Sockets – הכתובת שאליה נתחבר באמצעות הדף דן תהיה `http://127.0.0.1:80` (אשר 80 הוא הפורט בו נשתמש). על מנת לבדוק את הפתרון שלכם, אנו ממליצים להוריד אתר לדוגמה מהכתובת: data.cyber.org.il/networks/webroot.zip.

העתיקו את תוכן קובץ ZIP אל סדרה כלשהי (כਮון שיש לפתח את הקובץ) והשתמשו בה בתור ה-`root`

²² ברוב המקרים בסכמה יצון פרוטוקול – כגון HTTP או FTP. עם זאת, במקרים מסוימים, היא לא תכלול פרוטוקול, כמו הסכמה "file".

²³ זאת בהנחה שהשרת מרים מערכת הפעלה Windows. כאמור, במקרים הפעלה שונות ה-path עשוי להיות שונה בזורה שונה.

1) כתבו שרת המחברת לתקשורת מהלך ללקוח את index.html ויתמוך באפשרויות השונות שיש בעמוד directory אינטרנט זה.

1) כתבו שרת המחברת לתקשורת מהלך בפרוטוקול TCP בפורט 80. לאחר סגירת החיבור על ידי הלקוח, התוכנית נסגרת.

2) הוסיף תמייה בחיבורים עוקבים של לקוחות. כלומר, לאחר שהחיבור מול לקוח נסגר, השרת יוכל לקבל חיבור חדש לקוחות.

3) גרמו לשרת לוודא כי הפקטה שהוא מקבל היא GET HTTP, כלומר - ההודעה שהתקבלה היא מחרוזת מהצורה שראינו עד כה: מתחילה במילה GET, רווח, URL כלשהו, רווח, גירסת ה프וטוקול (HTTP/1.1), ולבסוף התווים \o - \d.

- אם הפקטה שהתקבלה אינה HTTP GET - סגורו את החיבור.

4) בהנחה שהשרת מקבל בקשה HTTP GET תקינה ובה שם קובץ, החזירו את שם הקובץ המבוקש אל הלקוח. בשלב זה, החזירו את שם הקובץ בלבד, ולא את התוכן שלו.

• שימוש לב שבד-Windows משתמשים ב-\".COM פרט בצוון מיקום קובץ, בעוד שב인터넷 וגם בלינוקו משתמשים ב-\"/".

- הערה: את שם הקובץ יש להעביר בתור שם משאב מבוקש, ולא ב-Header נפרד.

• הערה נוספת: בשלב זה, אל תעבירו Headerם של HTTP כגון הגירה או קוד התגובה.

5) כתעת החזירו את הקובץ עצמו (כלומר, את התוכן שלו).

- אם מתבקש קובץ שלא קיים - פשוט סגורו את החיבור (היעזרו ב- os.path.isfile).

הערה: בגיןוד לכמה מהתרגילים הקודמים, כאן יש לשולח את כל הקובץ מיד, ולא לחלק אותו למקטעים בגודל קבוע (כפי שעשינו, למשל, בתרגיל 2.7).

6) הוסיף את שורת התגובה ו-Headerם של HTTP:

- גרסה 1.0 HTTP.

• קוד תגובה: 200 (OK).

• השורה Content-Length: (מלאו בה את גודל הקובץ שמוחדר).

7) במקרה שבו לא קיים קובץ בשם שהתקבל בבקשת, החזירו קוד תגובה 404 (Not Found).

8) אם השרת מקבל בקשה GET ל-root (כלומר למיקום "/") - החזירו את הקובץ index.html (כמובן, וידאו שקיים קובץ זה; תוכלו ליצור קובץ בשם index.html שמכיל מחרוזת קצרה, רק לשם הבדיקה).

9) אם השרת מקבל בקשות לקבצים מסוימים שונים, הוסיף ל-Header של התשובה את השדה Content Type בהתאם לסוג הקובץ שהתקבש. תוכלו להעזר בתנונים הבאים:

- קבצים בסויומת txt או html:

Content-Type: text/html; charset=utf-8

- קבצים בסויומת jpg:

Content-Type: image/jpeg

- קבצים בסוימת **js**:

Content-Type: text/javascript; charset=UTF-8

- קבצים בסוימת **css**:

Content-Type: text/css

(10) כתה הוסיף תמייה במספר Status Codes נוספים (היעזרו בויקיפדיה):

.1. 403 Forbidden – הוסיף מספר קבציםשאליהם למשתמש אין הרשה לגשת.

.2. 302 Moved Temporarily – הוסיף מספר קבצים שהמיקום שלהם "zz". כר למשל, משתמש שיבקש

את המאוב **page1.html**, יקבל תשובה 302 שתגרום לו לפנות אל המאוב **page2.html**. לאחר מכן,

הלקוח יבקש את המאוב **page2.html**, ועבורי יקבל תשובה 200 OK.

.3. Internal Server Error – במקרה שהשרת קיבל בקשה שהוא לא מבין, במקומ לסגור את החיבור,

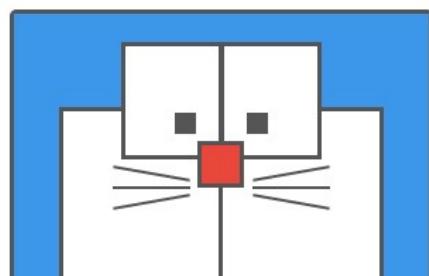
החזיר קוד תגובה 500.

נסו את השרת שלכם באמצעות הדפסן- גירמו לשרת לשלוח את המידע שנמצא ב-**webroot** ותוכלו

לצפות באתר הבא:



4.4 File Type



אתר בדיקת תרגיל שרת HTTP – קרדיט תומר טגם

מדריך לכתיבה ובדיקה של תרגיל כתיבת שרת HTTP

קוד של שרת HTTP הוא קוד מורכב יחסית לתרגילים קודמים ולכן מומלץ לתכנן אותו מראש ולחולק אותו לפונקציות. יש דרכים רבות לכתוב את קוד השרת, תוכלו להתבסס על שלד התוכנית הבא:

http://data.cyber.org.il/networks/HTTP_server_shell.py

המקומות שעליים להשלים בעצמכם נמצאים תחת העלה "DO". שימוש לב ש כדי שהתוכנית תעבור תצרכו להוסיף לה קבועים ופונקציות נוספות, השלד אמר רק לסייע לכם להתמקד.

טיפים לכתיבה

1. שימושו לב שההתשובות שאתה מוחזרים הם לפני כל השודות של HTTP. אל תפספסו אף סימן רווח או ירידת שורה...
2. לעיתים עלול להיות מצב שבו הן השרת והן הדפדפן מצפים לקבל מידע. כדי לצאת במצב זה, ניתן להגדיר SOCKET_TIMEOUT. שימושו לב שם הזמן שהגדרתם עבר, תקבלו exception. עלייכם לטפל בו כדי שהשרת לא יסימן את הריצה.

איקון

מרבית האתרי האינטרנט כוללים איקון מעוצב שמופיע בלשונית של הדפדפן. גם הדפדפן שלכם יבקש את האיקון מהשרת. כדי למצוא הינק נמצוא האיקון, תוכלו לפתוח את העמוד index.html בתוכנת +notepad++ ולחפש את .favicon.ico



רוצים ליצור לעצמכם איקון אישי? תוכלו להשתמש באתר <http://www.favicon.cc> תוכלו להעלות לתמונה כלשהו או להמציא איקון משלכם. לאחר שהסיטםת לשב איקון, לחצו על כפתור download icon ושמרו אותו במקום המתאים.

כלי דיבוג – breakpoints

שימוש breakpoints לдобוט דיבוג קוד השרת שלכם הוא הכרחי. צרו breakpoints במקומות שבו הקוד עדין מבוצע באופן תקין ועיקבו אחרי הערכים שמקבלים משתנים וփונקציות מוחזירות כדי לבדוק בעיות בקוד שלכם. באמצעות בדיקת ערכי משתנים תוכלו לבדוק, לדוגמה, מה הבקשה שלוקה.

```

203     def main():
204         # Open a socket and loop forever while we:
205         server_socket = socket.socket(socket.AF_INET)
206         server_socket.bind((IP, PORT))
207         server_socket.listen(10)
208         print "Listening for connection on port %d" % PORT
209
210         while True:
211             client_socket, client_address = server_socket.accept()
212             print 'New connection received'
213             client_socket.settimeout(SOCKET_TIMEOUT)
214             handle_client(client_socket)

```

כל דיבוג - דף דפן כרום

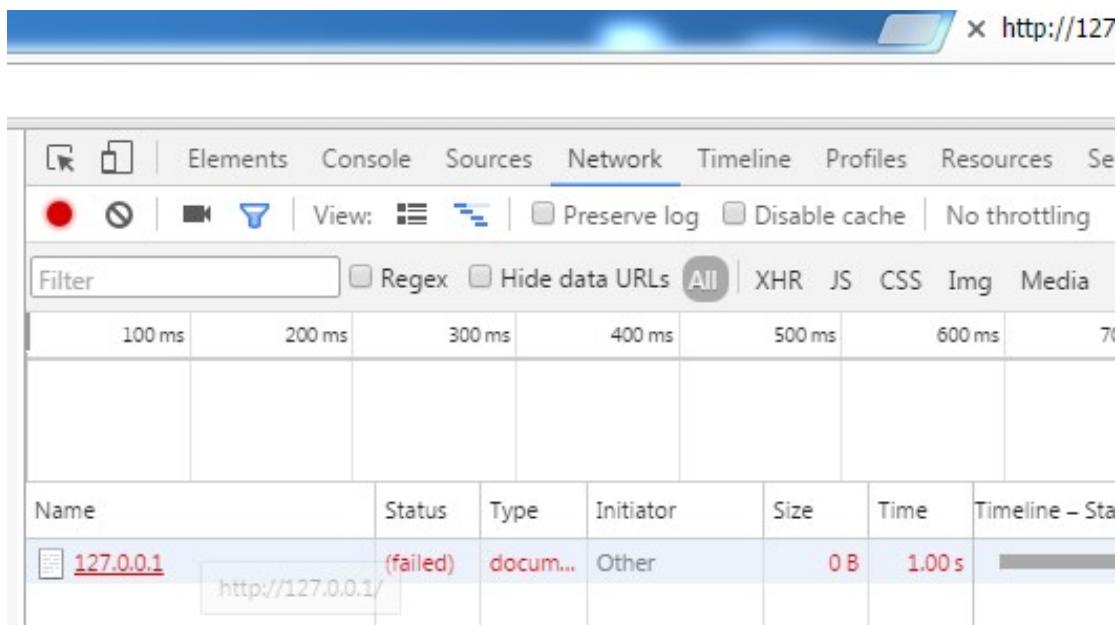
דף כרום כולל אפשרות לעקוב אחרי כל התגובה בין הדפן לשרת. איך עושים את זה?
בutor הדפן לחצו על F12 ולאחר מכן על טאב network. יפתח לכם המסך הבא:

The screenshot shows the Network tab of the Chrome DevTools. At the top, there are tabs for Elements, Console, Sources, Network, Timeline, Profiles, Resources, Security, and Audits. The Network tab is selected. Below the tabs, there are buttons for View (grid, list, tree), Preserve log, Disable cache, and No throttling. A filter bar with 'Filter' and 'All' selected is present. The main area shows a timeline with several requests. One request from '127.0.0.1' is highlighted, showing a status of 200 OK, type document, initiator Other, size 25.1 KB, and time 5 ms.

לחצו על הנקודה האדומה כדי להתחילה הקלטה חדשה.
אם תumedו על שם של קובץ כלשהו תוכלו לקבל פרטים נוספים בטאים החדשמים שמופיעים לצדויו - Headers, Preview, Response, Timing
המידע המעניין ביותר לטובות דיבוג השירות שלכם מופיע ב-s-headers, שם ניתן לראות את הביקשות שנשלחו לשרת ואת התשובות של השירות.

This screenshot shows the detailed view for the '127.0.0.1' request. It includes tabs for Headers, Preview, Response, and Timing. The Headers tab is selected. The General section displays the Request URL (http://127.0.0.1/), Request Method (GET), Status Code (200 OK), and Remote Address (127.0.0.1:80). The Response Headers section shows HTTP/1.1 200 OK, Content-Length: 25587, and Content-Type: text/html; charset=utf-8. The Request Headers section shows GET / HTTP/1.1, Host: 127.0.0.1, and Connection: keep-alive.

אם מסיבה כלשהי לא התקבלו מהשרת כל הקבצים, תוכלו לראות מה לא התקבל באזרור מיוחד שמקורו לתייעוד שגיאות. לדוגמה, כאשר השרת לא מקבל את בקשת ההתחברות של הלוקו:



כלי דיבוג - Wireshark

כלי נפלא זה, אליו עשינו היכרות בעבר, יכול לספק לכם את כל המידע שעבר בין השרת ללוקו שלכם - בתנאי שתristolו את השרת ואת הלוקו על שני מחשבים נפרדים, שמחוברים ע"י ראוטר או switch כפוי שווודי יש לכם בבית.

בשלב ראשון, וודאו שהשרת שלכם מאמין לכתובת IP 0.0.0.0 (ולא 0.0.0.1). כתובות 0.0.0.0 אומrette למכירת הפעלה שלכם לשולח לשרת שבניתם לא רק פקודות שמגיעות מתוך המחשב עצמו (127.0.0.1) אלא גם פקודות שמגיעות מחכבים אחרים - בתנאי שהם פונים לפורט הנכון. לאחר מכן בדיקו באמצעות ipconfig מה כתובת ה-IP של השרת שלכם (שם לו, כתובת IP ברשות הפנימית שלכם, בהמשך נלמד מה היא כתובת פנימית). בדוגמה הבאה הכתובת היא 10.0.0.1:

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix .: Home
    Description: Intel PRO/100 MT Desktop
    Physical Address: 00-0C-29-00-00-0A
    DHCP Enabled: Yes
    Status: Connected
    Connection Type: Shared Internet Connection
    Link Layer Protocol: TCP/IPv4
    IP Address.1: 10.0.0.1
    Subnet Mask .1: 255.255.255.0
    Default Gateway .1: 10.0.0.1
    DNS Servers .1: 10.0.0.1
    IP Address.2: 169.254.10.125
    Subnet Mask .2: 255.255.0.0
    Default Gateway .2: 169.254.1.1
    DNS Servers .2: 169.254.1.1
    IP Address.3: 169.254.10.125
    Subnet Mask .3: 255.255.0.0
    Default Gateway .3: 169.254.1.1
    DNS Servers .3: 169.254.1.1
```

לפני שנתקדם לשלב הבא, הפעילו wireshark.

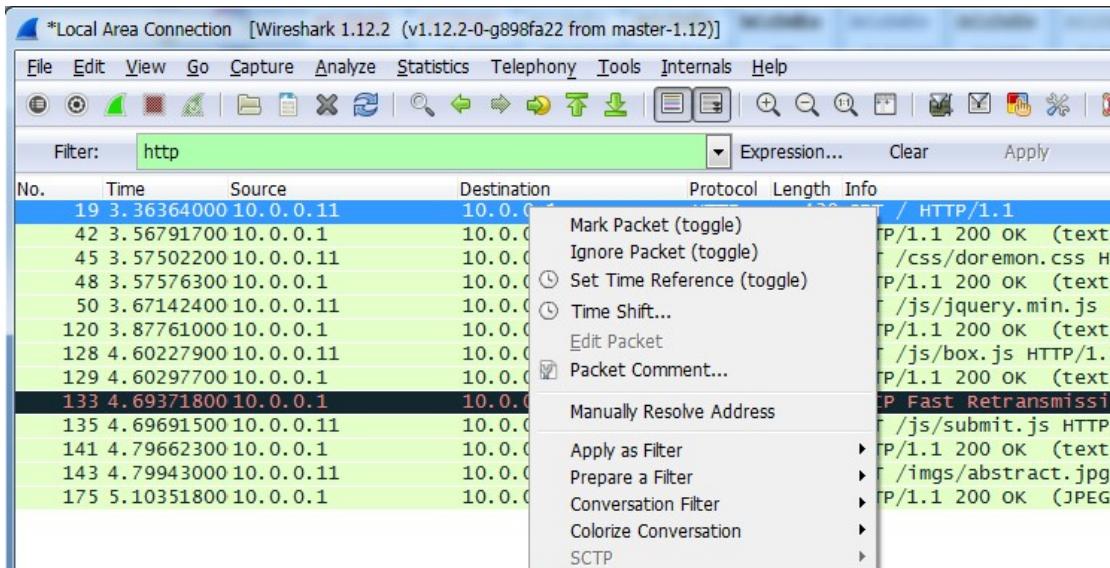
כעת עלייכם לקבוע בהתאם את כתובת ה-IP שהליך פונה אליה:



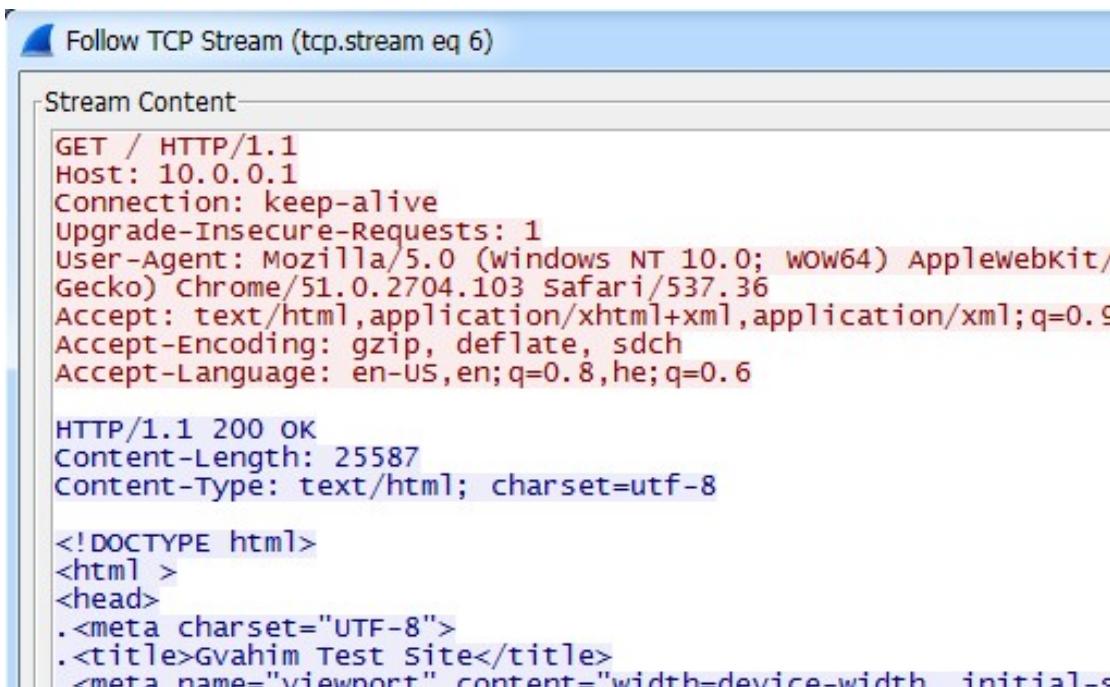
תוכלו למצאו בתוך wireshark את התעבורה בין הדפסן בלקוח לבין שרת ה-HTTP שלכם. נfiltrar את התעבורה לפי http:

*Local Area Connection [Wireshark 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)]						
No.	Time	Source	Destination	Protocol	Length	Info
19	3.36364000	10.0.0.11	10.0.0.1	HTTP	430	GET / HTTP/1.1
42	3.56791700	10.0.0.1	10.0.0.11	HTTP	903	HTTP/1.1 200 OK (text/html)
45	3.57502200	10.0.0.11	10.0.0.1	HTTP	386	GET /css/doremon.css HTTP/1.1
48	3.57576300	10.0.0.1	10.0.0.11	HTTP	208	HTTP/1.1 200 OK (text/html)
50	3.67142400	10.0.0.11	10.0.0.1	HTTP	372	GET /js/jquery.min.js HTTP/1.1
120	3.87761000	10.0.0.1	10.0.0.11	HTTP	163	HTTP/1.1 200 OK (text/html)

nocll לראות את כל הבקשות של הלקוח ואת תשובות השרת. בצלום מסך זה ניתן לראות שחלק מהמידע (השורה הצבועה על ידי wireshark בשחור) שודר פעמיים מהשרת ללקוח - Retransmission - שמעיד על כך שהשרת לא קיבל אישור על המידע שהשלח ללקוח (האישור הוא ברמת שכבת התעבורה, פרוטוקול TCP, עליו נלמד בהמשך). אם נרצה לעקוב אחרי כל הפקודות שעברו בין השרת והלקוח שלנו, כולל הודעות הקמת וסיום קשר ושליחה מחודשת של פקודות, nocll להקליק על שורה כלה ולבחרTCP stream follow.



ויצגו בפנינו מסכים שמראים את כל המידע בעבר, הן בשכבה האפליקציה (השרת והלקוח במצבים שונים) והן בשכבה התעבורה. מכך ניתן להשתמש בфиילטרים נטולים לארח בעיות שאולי גרמו לכך שהשרת שלנו לא עבד כפי שתיכנו.



Filter: tcp.stream eq 6				Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info
15	3.17640900	10.0.0.11	10.0.0.1	TCP	62	50279->80 [SYN] Seq=0 Win=64
17	3.36110200	10.0.0.1	10.0.0.11	TCP	62	80->50279 [SYN, ACK] Seq=1 Win=64
18	3.36343700	10.0.0.11	10.0.0.1	TCP	60	50279->80 [ACK] Seq=1 Win=64
19	3.36364000	10.0.0.11	10.0.0.1	HTTP	430	GET / HTTP/1.1
20	3.36426100	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 0 bytes]
21	3.36426500	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 1 bytes]
22	3.37566600	10.0.0.11	10.0.0.1	TCP	60	50279->80 [ACK] Seq=377 Win=64
23	3.37570300	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 2 bytes]
24	3.37571000	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 3 bytes]
25	3.46497200	10.0.0.11	10.0.0.1	TCP	60	50279->80 [ACK] Seq=377 Win=64
26	3.46501300	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 4 bytes]
27	3.46502100	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 5 bytes]
28	3.46502400	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 6 bytes]
29	3.46502700	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 7 bytes]
30	3.46503000	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 8 bytes]
31	3.46503600	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 9 bytes]
32	3.46768100	10.0.0.11	10.0.0.1	TCP	60	50279->80 [ACK] Seq=377 Win=64
33	3.46771500	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembly, offset 10 bytes]

זהו, בהצלחה!

פרוטוקול HTTP - תשובה "תכנותית" לבקשת GET, ופרמטרים של בקשת GET



במהלך עשורים האחרונים האחرونנות, "אפליקציות web" (כינוי לאפליקציות ושרותים שעושים שימוש ב프וטוקול אינטרנט) צברו יותר ויותר פופולריות, ובמקביל התפתחו באופן מואץ היכולות, רמת התהוכן והמורכבות של מערכות אלו. אתרים אינטרנט כבר לא הסתפקו בהגשה של דפי html ותמונות ערכיים מראש, אלא עשו שימוש בקוד ותכונות כדי ליצור דפי תוכן "динמיים".

לצורך הממחשה, חשבו מה קורה כשאתם מתחברים ל-Facebook - אתם רואים דף אינטרנט בו דברים קבועים - סרגל כלים, לינקים - והרבה דברים שנקבעים מחדש בכל כניסה - למשל הודעות על פעילות של החברים שלכם ופרסומות. למעשה מאחוריו הקלעים מסתתר שרת, שמקבל החלטות איך להרכיב את דף התוכן עבור כל בקשה של כל משתמש.

כך כאשר אתם פונים ל-Facebook, השירות צריך להחליט כיצד לבנות עבורכם את הדף. לאחר שהוא מבין מי המשתמש שפנה ומה אותו משתמש צריך לראות, הוא בונה עבורו את הדף ומגיש אותו ללקוח.

השירות sh-Facebook מספק מתחכם בהרבה מעמודי אינטרנט בראשית הדרך, שבהם היה תוכן סטטי בלבד - קבוע מראש, והשירות רק היה מגיש את העמודים הללו ללקוחות.

cut נמש שרת שמגיב בתשובות תכנותיות לבקשת GET. כמובן, השירות שלנו יבנה תשובה באופן דינامي בהתאם לבקשת שהגעה מהלקוח. בינהים, נמש את הצד השירות, ולצורך הצד הלקו נמשיך להשתמש בדף.

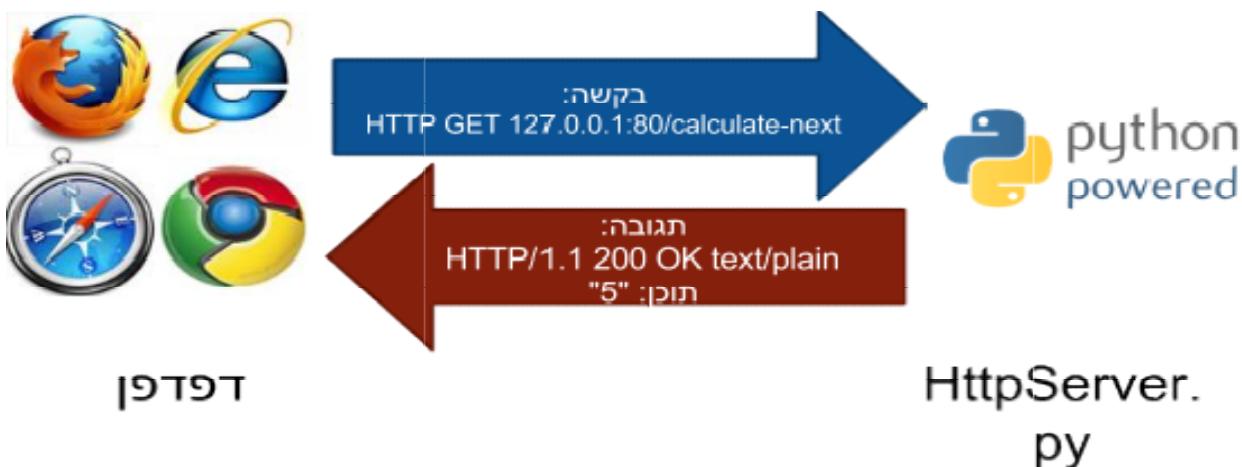
תרגיל 4.5



עדכנו את קוד השירות שתכתבם, ככה שפנוייה לכתובת "calculate-next/"/ לא תחפש קובץ בשם זהה, אלא תענה על השאלה "מה המספר ש McGuי אחרי 4". כמובן, השירות פשוט יחזיר תמיד "5" בתור תוכן התגובה לבקשת זו. הריצו את השירות ובדקו שהמספר 5 מתקבל כתגובה לפניה לכתובת זו.

בתרגילים הבאים, שימו לב לכלול Header'ים רלבנטיים בתשובותיכם, ולא "סתם" Header'ים שאתם רואים בהסנפות. כדי שהතגובה תהיה תקינה, ציינו קוד תשובה (כגון OK 200). עליכם לכלול גם Content-Length (נדרש להיות גודל התשובה, בביטחון, ללא גודל ה-Header'ים של HTTP). ציינו גם את ה-Type (במקרה שתחזרו קובץ – סוג הקובץ שאתם מחזירים, שאפשר להסיק לפי סיומת הקובץ). תוכלו להוסיף גם Header'ים נוספים, אך עשו זאת בצורה שתואמת את הבקשת והתשובה הספציפיות.

אם ביצעתם את התרגיל נכון, זה מה שבעצם קרה כאן:



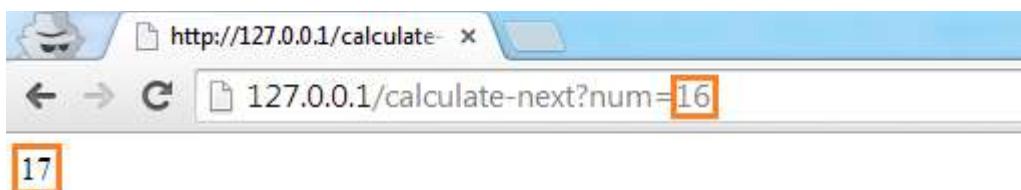
בטח כבר עלייתם בעצמכם על כך שהשרת במקורה הזה הוא מודרני ואמנויותי. בתקינות שכתבנו בפייטון, בדרך כלל עשינו שימוש בקלט מהמשתמש כדי לעשות איתה חישוב כלשהו במהלך התוכנית. השרת שלנו היה הופך להרבה יותר שימושי, אם האפליקציה הייתה יכולה לבקש את המספר שבא אחריו מספר כלשהו; למעשה, אפשר לחשב על השרת שלנו כמו על כל תוכנית בפייטון שכתבנו עד היום, למעט העובדה שלא ניתן להعبر לתוכנית קולט, ולכן היא מחזירה תמיד את אותו הפלט. אם נוכל להעביר לה בכל בקשה קולט שונה, נקבל את הפלט המתאים לכל קולט.

אם כך,icut נעביר קולט לשרת באמצעות פרוטוקול ה-HTTP - למעשה משתמש במה שנקרא "Parameters" - פרמטרים בבקשת ה-GET. הדרך שבה מעבירים פרמטרים בבקשת GET היא באמצעות תוספת התו "?" בסיום הכתובת (זה אותו שמשמש בפרוטוקול HTTP כמפריד בין הכתובת של המחשב לבין משתני הבקשה), ולאחר מכן את שם הפרמטר והערך שלו.

תרגיל 4.6



עדכנו את קוד השרת שכתבתם, כך שפניה לכתובת "/calculate-next?num=4" תתרפרש כפניה לכתובת calculate-next עם השמה של הערך '4' במשנה num, וכן תחזיר את המספר 5. אבל, "/calculate-next?num=16" תחזיר את התשובה '17', "/calculate-next?num=10000" תחזיר את התשובה '10001', והבקשה "/calculate-next?num=-200" תחזיר את התשובה '-199'.



זהו למעשה הדרך שבה יכולה אפליקציה להעביר פרמטרים כחלק מהבקשה שלה.

כדי להמחיש את העניין זהה, נתבונן בדוגמה מענית יותר, ולאחר מכן נבחן להבין את קטע הקוד של השרת שבו השתמשנו:

הכניסו את השורה הבאה בדף:

זהו למעשה בבקשת GET שנשלחת אל שרת החיפוש של Google. אבל, אתם עדין לא רואים תוצאות חיפוש, כי ה"אפליקציה" (במקרה זה, האתר של Google בדף שלכם), לא יודעת מה אתם רוצים לחפש. ברגע שתתחלו להכניס מילת חיפוש (נניח: "application"), נתחיל לראות תוצאות חיפוש במסך.

A screenshot of a Google search results page. The search query is `application`. The results show approximately 514,000,000 results found in 0.30 seconds. The top result is a link to the Wikipedia article on Application software. Below it, there are other links related to the term, such as `Application - Wikipedia, the free encyclopedia` and `Web application - Wikipedia, the free encyclopedia`.

AIR הגינו התוצאות? הדף שהוחל בבקשת GET אל השרת של גוגל.

AIR נראית הבקשה? טוב ששאלתם, היא נראית בדיקת CRC.²⁴

GET `www.google.com/search?q=application` HTTP/1.1

נסו בעצמכם להכניס את ה-URL בדף בלבד, ללא המילה GET וללא גירסת הпрוטוקול, ותראו בעצמכם מה קורה.

²⁴ ברוב המקרים ה-`Host` (בדוגמה זו "www.google.com") לא יכול בשורת ה-`GET`, אלא ב-`HTTP Header` נפרד. עם זאת, על מנת שהדוגמאות תהיוינה ברורות, אנו נציג אותם באופן מפושט.

לאחר שבדקתם בדףן - פיתחו **telnet** מול הכתובת com **www.google.com** בפורט 80 (כמו שעשינו בתרגיל 4.3), וביצעו את הבקשה זו. נסו להבין את התגובה.



תרגיל 4.7

בכל אחת מהשורות בטבלה הבאה, תמצאו כתובות של מושב באינטרנט (אםן עדין לא הסבכנו את המשמעות הפורמלית של המילה "מושב", אך ראייתם דוגמאות למשאים, למשל www.google.com/search), שם של פרט, ורשימת ערכים. מה שעליכם לעשות הוא להרכיב בקשה GET בכל אחד מהmarker'ים (כפי שראינו שבונים בקשה ממושב, שם של פרט וערך שלו), לנסוט אותה בדףן שלכם, ולכתוב בקצרה מה קיבלתם.

הסבר	בקשת GET	ערכים	פרט	מושב
		jerusalem, new-york, egypt	q	google.com/maps
		madonna, obama	q	twitter.com/search
		israel, http, application	search	wikipedia.org/w/index.php
		obama, gangam, wolf	search_query	youtube.com/results



נקודה למחשה: שימוש לב להבדל המרכז בין ה-URLים הללו לבין URLים שעלייהם התבוססTEM בתרגיל שרת `http` שמייחדים בסעיף הקודם - כו"ם "נעלו" החלקים מה-URL שמתיארים מיקום במערכת הקבצים, ואת מקומם החליפו הפורטרים. זהה תוצאה של השינוי שהתרחש באינטרנט בעשורים האחרונים ה先后ות - מעבר מהגשה של דפי תוכן שנוצרו מראש, לבניה של תשובות על סמך פורטרים וקוד תוכנה שמרכזיב את התשובה.

כעת נחזור לאחד החיפושים שעשינו ב-Twitter בתרגיל הקודם; שימוש לב שהتوزאות שקיבלנו מכילות ציוצים (tweets) שהכילו את המילה `obama`. מה אם הייתי רוצה לחפש את הפרופיל של אובמה? (שימוש לב שיש בצד שמאל כפטור שעושה את זה). האפן שבו ה"אפקטיבציה" (במקורה שלנו, האתר) של Twitter עושה את זה, הוא שהוא שולח פורטר נוסף לשרת.

פורטר אחד יכול לצל את מילת החיפוש (`obama`), והפורטר השני יכול את סוג החיפוש - חיפוש משתמשים (users). בין שני הפורטרים יופיע המפ прид &. נראה ביחיד איך זה עובד.

הכניסו את השורה הבאה בדף: <https://twitter.com/search?q=obama&mode=users>

באופן דומה, ניתן לחפש תמונות של אובמה, על ידי הבחירה הערך של הפורטר mode בערך photos - בנו את הבקשה ונסו זאת בעצמכם.

תרגיל 4.8



כמו שאותם וודאו מתארים לעצמכם, ניתן להעביר בבקשת GET גם יותר שני פרמטרים, באוטה הצורה. הריכיבו את בקשת GET לשרת המפות של Google (בדומה לדוגמה שעשיתם בתרגיל הקודם), אך במקום חיפוש כתובת, בקשו את הוראות הנסיעה מטל אביב לירושלים בתחבורת ציבורית, על ידי שימוש בפרמטרים הבאים:

- saddr - כתובת המוצא (למשל: 'Tel+Aviv').
- daddr - כתובת היעד.
- dirflg - אמצעי התחבורה. תוכלו להעביר 'ז' עבור תחבורה ציבורית (או 'א' אם אתם מתכוונים לרכבת).

בנו את בקשת GET, הכניסו אותה לדף שלכם, וודאו שאתם מקבלים את הוראות הנסעה.

תרגיל 4.9



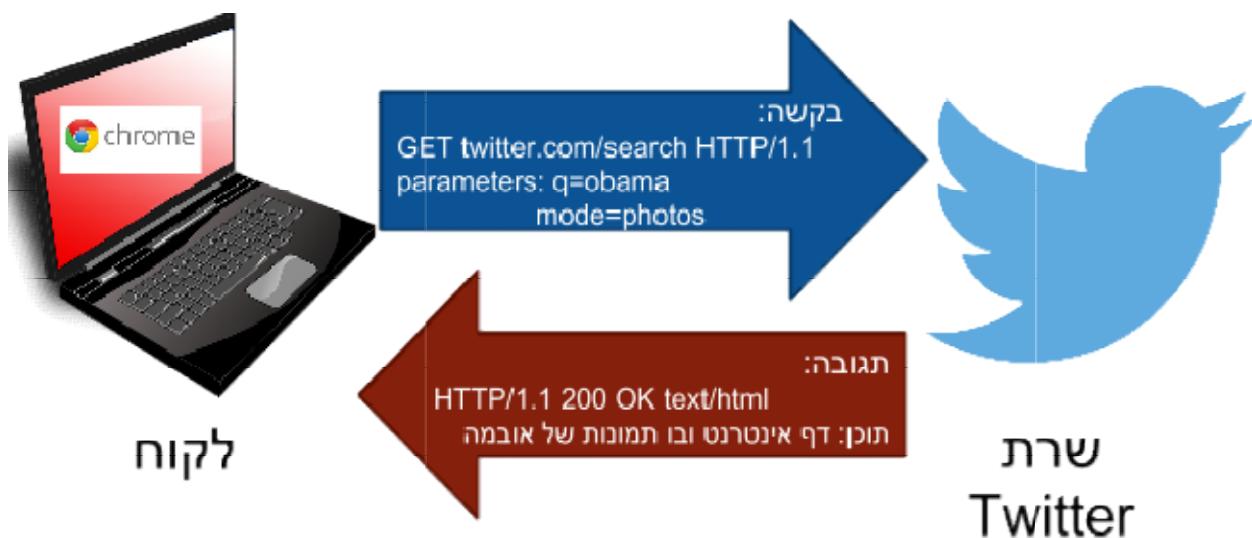
כתבו שרת HTTP פשוט, שמחשב את השטח של משולש, על סמך שני פרמטרים: גובה המשולש והרוחב שלו. למשל, עבור שורת הבקשה הבאה: `http://127.0.0.1:80/calculate-area?height=3&width=4`, יחזיר "6.0".
בדקו אותו גם עבור קלטים נוספים.

פרוטוקול HTTP - בקשות POST ותוכנות צד לקוח בפייתון (הרחבה)

נפתח בשאלת - האם העלייתם פעם תמונה ל-Facebook? קל להניח שכן (ואם תתעקשו שמדובר לא עשייתם זאת, ניתן להניח במקומות כי שלחתם קובץ PDF או מסמך Word למשהו במיל, או לפחות מילאתם טופס "יצירת קשר" באתר כלשהו). המשותף לכל המקרים שתיארנו הוא - הצורך להעביר כמה נתונים מהאפליקציה אל השרת, כמוות של מידע שלא ניתן להעביר בבקשת GET.

נתעכבר כדי להבין טוב יותר את ההבדל בין בקשת GET עם פרמטרים לשרת של Twitter, שמחזירה דף אינטרנט עם תמונות של אובמה (כמו שראינו בסעיף הקודם), לבין בקשה לשרת של Facebook, שתאחסן שם תמונה.

במקרה שכבר ראיינו - שליחת בקשה לתמונות באתר Twitter - האפליקציה (הדף שמציג את עמוד האינטרנט של Twitter) רוצה להציג למשתמש את התמונות שמתאימות לחיפוש "אובמה":



כדי לעשות זאת, נשלחת לשרת בקשה די קצרה, שמכילה בסך הכל את הכתובת "twitter.com/search", ושני פרמטרים – obama, photos. כל זה נכנס בפקטה אחת. התשובה, לעומת זאת, מכילה הרבה מידע, ואם נסניף את התקשרות, נראה שההתשובה מורכבת ממספר גדול של פקודות. עשו זאת, ומצאו את פקחת הבקשת, וכמה פקודות היה צריך כדי להעביר את כל התשובה.

מה ההבדל לעומת המקרה שבו נעה תמונה ל-Facebook?

במקרה זה, המידע נמצא לצד האפליקציה (ספקטיף – התמונה). טלפון ממוצע תמונה יכולה להיות בגודל של יותר מ-1MB), והוא מעוניינית להעביר את כלו לשרת.

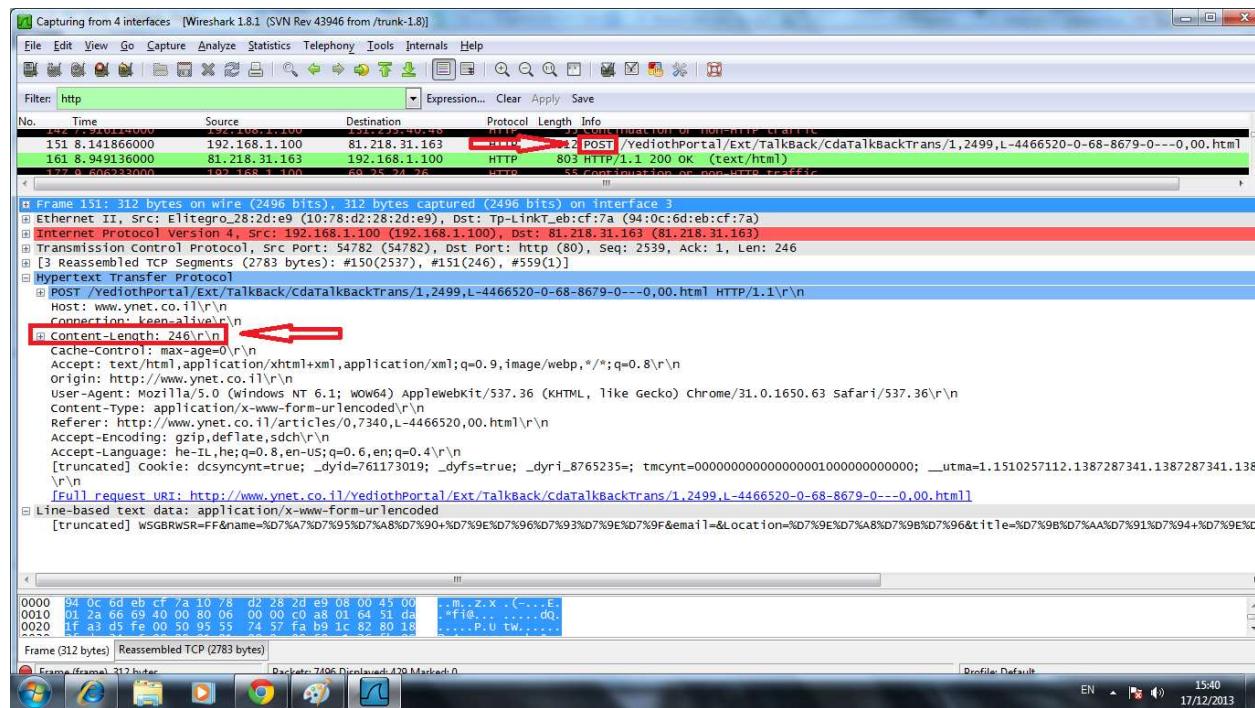


אם ניתן להשתמש בפרמטרים של בקשת GET כדי להעביר את כל התמונה?

אם ניקח בחשבון את העובדה שהאורך המקסימלי של URL בו תומכים רוב הדפדפנים הוא 2,000 תווים, וה- URL הוא כל מה שניתן להעביר בבקשת GET, די ברור שלא יוכל להעביר את התמונה לשרת באמצעות בקשת GET.

לכן, נעשה שימוש בסוג חדש של בקשה שקיים בפרטוקול HTTP – בקשה POST.

כדי להתנסות בשימוש בבקשת POST, ניכנס לאתר www.ynet.co.il, נבחר את כתבה שמצאנו בה עניין, ונוסיף לה בתגובה - "כתבת מענית, תודה רבה". - רגע לפני שנלחץ על הכפתור "שלח תגובה", נפעיל הסנהפה ב-Wireshark, ולאחר שנשלח את התגובה, נוכל למצוא פקעת HTTP עם בקשה מסוג POST, שנשלחה לשרת :Ynet של



נודא שאנו מבינים מה קרה כאן - בתגובה ששלחנו, מילאנו את השם, את מקום המגורים, ואת כתובות המייל שלנו. בנוסף, מילאנו כתובת לתגובה, ואת תוכן התגובה עצמו (שיכול להיות גם ארוך הרבה יותר מאשר "כתבת נחמדה"). האם היה ניתן לשלוח את כל הנתונים האלה לשרת באמצעות בקשה GET ?
www.ynet.co.il/articles/17773?name=Moshe&address=Rehovot&mail=moshe&rehovot.co.il&title=nice_article&body=thank_you_this_was_a_great_article_...

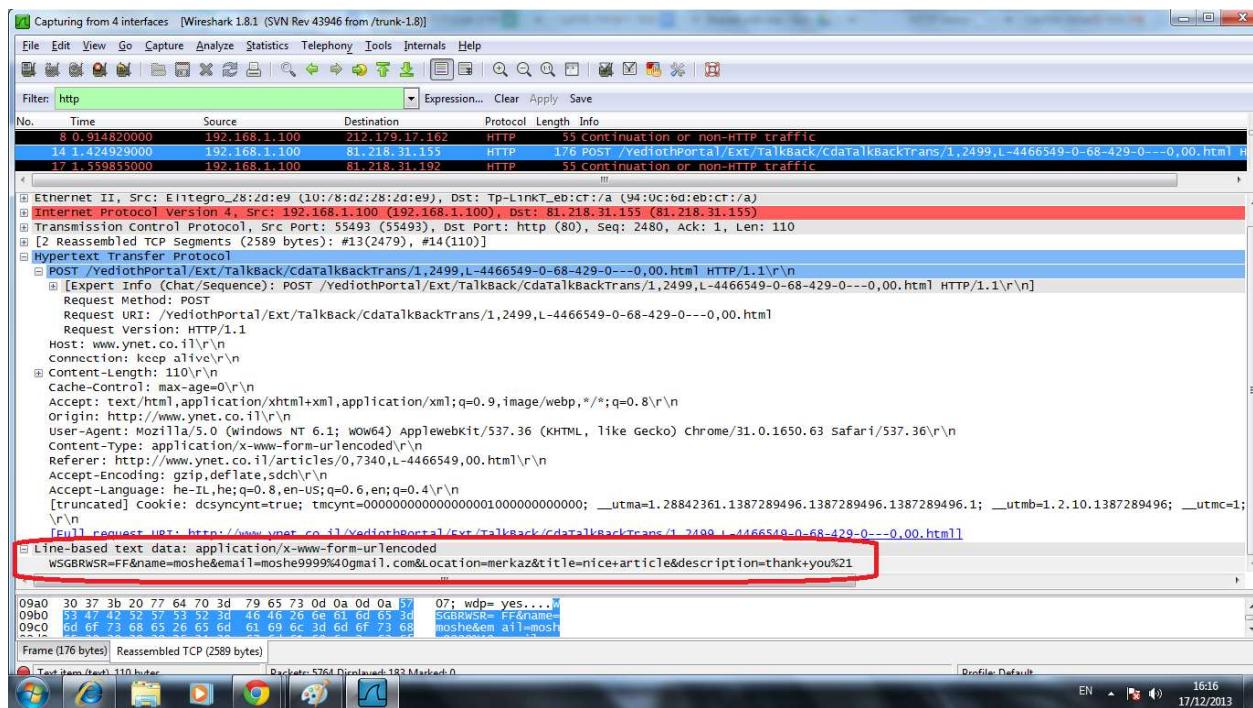
זהו אינה אופציה מציאותית - האורך המקסימלי של URL הוא 2,000 תווים, בעוד שרק התגובה עצמה עשויה להיות יותר ארוכה מכ...
 לכן, בנוסף לפרמטרים שעוברים חלק מהכתובת (URL), בבקשת POST ניתן לזרף תוכן, שם יועבר תוכן התגובה.

نبיט בתהיל'ך המלא שבוצע מול השירות של Ynet:



באופן כללי, אחד השימושים הנפוצים בבקשת POST הוא בטפסים - הכוונה היא לטופס שמכיל מספר שדות, שאוותם המשתמש מלא ואז לוחץ על כפתור כדי לשלוח את הטופס - זהו בדיקת המקירה עם תשובות באתר Ynet. דוגמה נוספת - אם תלחצו על "צור קשר" באתר של חברת הסולורי שלכם, ותملאו שם את הטופס, התוכן ישלח לשרת של האתר האינטרנט שלהם באופן דומה.

אם תרצו "לראות" איך עובר התוכן עצמו בבקשת POST - לטפסים יש דרך דרי סטנדרטית להעביר את הנתונים האלה. הדרך הזאת קצת מזכירה את האופן שבו משתמשים בפרמטרים ב-URL, רק שבמקרה זה, אין מגבלה של מקום. כדי לראות זאת - ב-Wireshark התבוננו בתוכן של פקעת POST, וחפשו את text-data..
נסו למצוא שם את השדות השונים שמילאים בתגובה (שםכם, מקום המגורים, הכתובת וכו').



תרגיל 4.10



מכיוון שבקשות POST לא נוכל לייצר באמצעות שורת הפקודה בדפסן, נצטרך לכתוב תוכנית בפייתון שתדמה גם את אז הלcoil בתקשורת - כולם את האפליקציה.

כתבו תוכנית חדשה שתתפקיד בתור ל��ח HTTP. התוכנית תשלח בקשה POST אל השרת (תוכלו לבחור את הכתובת בעצמכם, למשל /upload). ה-body של בקשה ה-POST יוכל את התוכן של התמונה שיש לשמר בתיקיית התמונות. את שם הקובץ לשמירה ציינו בתור פרמטר בשם "file-name".
בנוסף, משזו בשרת את קבלת בקשה ה-POST ושמירת התמונה בתיקיית התמונות לפי שם הקובץ שהגיע.

בטח טענו (ובצדקי!) שאין שם טום בשרת שרק ניתן לשמור אליו תמונות, אם לא ניתן לבקש אותן בחזרה.

תרגיל 4.11

הויספו לשרת שכתבתם תמייה בבקשת GET תחת המשאב `image`, שמקבלת פרמטר בשם `image-name`, ומחזירה את התמונה שנשמרה בשם זהה (או קוד תגובה 404, במידה שלא קיימת תמונה בשם זהה).

הristol את השרת החדש, וביצעו פקודת POST (אחת או יותר) כדי להעלות תמונות לשרת. קראו לאחת התמונות בשם "test-image".

כעת, פתחו את הדפדן (בו בדרך כלל השתמשנו עד כה בתור צד ל��וח ליצירת בקשות GET), והכינו את השורה הבאה:

```
http://127.0.0.1:80/image?image-name=test-image
```

אחרי שראיתם את התמונה שהעליתם מוקדם יותר חוזרת מהשרת, וודאו שהבנתם עד הסוף מה בעצם קרה כאן.

HTTP - סיכון קצר

עד כה, למדנו כיצד להתנהל עם משאבי באמצעות פרוטוקול HTTP. בתחילת, למדנו לבקש משאבי על סמך השם שלהם באמצעות GET; לאחר מכן למדנו לצרף פרמטרים נוספים כדי "לחידד" את הבקשה שאיתה אנו פונים אל משאב הרשת - לדוגמה, כשפנינו אל שירות המפות עם חיפוש המסלול מ"א לירושלים, הוספנו פרמטר נוסף לבקשה שמסמן שהמסלול צריך להיות בתחום ציבורית.

הבנו כי בקשה של משאב לא אמורה לשנות דבר בצד השירות - רק להחזיר תוצאה מסוימת. ניתן לחזור על אותה הבקשה מספר רב של פעמים, והתוצאה אמורה להיות זהה.

לעומת זאת, בהמשך למדנו להעלות של מידע מצד האפליקציה אל השירות באמצעות POST. פעולה זו בחלט גורמת לשינוי בmäßigו שנמצא בשרת, כפי שראינו בשרת אחיזור התמונות שכתבנו בסעיף הקודם. אני יכול לבקש תמונה בשם "21-december" ולקבל הודעה שגיאה שאנו משאב אינו נמצא (404), וביום שלאחר מכן לבצע שוב את אותה הבקשה, אלא שהפעם קיבל תמונה אחרת. מה ההסבר לכך? נראה שבינתיים מישו ביצעו פקודת POST והעלה תמונה בשם זהה.

הבנו כי מאחורי משאבי האינטרנט נמצאים שירותי שמריים קוד - בדומה לשרת שכתבנו בפרק זה - ומשתמשים בו כדי לייצר תשובות לבקשת GET ו-POST. אבל, שירותי אלה הם לרוב מורכבים הרבה יותר מהשרת שכתבנו, ולרבות ישמשו גם בסיס נתונים (database) - יכתבו אליו בבקשת POST, ויקראו ממנו בבקשת GET. הם גם יידעו איך לתמוך במספר משתמשים (לקחוות) שմבקשים בבקשת בו זמן, יפעלו מנוגנוני אבטחה והרשאות, ויתמכו בסוגי בקשות נוספים (כמו למשל בקשות למחיקה).

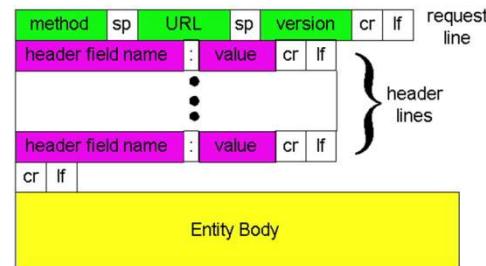
בגלל שרת HTTP ה-[HTTPServer](#) המ כל כך נפוצים וופולריים, רוב האתרי האינטרנט לא ממשיכים את פרוטוקול HTTP בעצםם, אלא עושים שימוש במימושים נפוצים של שירותי-Calala (למשל, בפייתון ניתן להשתמש במימוש [SimpleHTTPServer](#)).
נושאים אלו לא מכוסים בפרק זה לעת עתה, אך אנחנו מעודדים קוראים סקרנים לחקור וללמוד בעצמם, ב[חלקן](#).
[צעדים להמשך של פרק זה](#).

HTTP - נושאים מתקדמים ותרגילי מחקר

בחלק זה נלמד על יכולות מתקדמות יותר של פרוטוקול HTTP, שיינו מבוססות על מנגנון ה-Header שלמנו בחלק הבסיסי. בשלב זה של הלימוד, ננצל את יכולות שרכשנו ב--wireshark על מנת לחקור בעצמנו חלק מהנושאים.

כפתיב, נזכיר במבנה המלא של בקשת HTTP - ה-Header מופיע לאחר שורת הבקשה ולפני המידע (data) עצמו:

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```



(מתומו) Cache

דרך אוחת "להאץ" את חווית השימוש באינטרנט היא באמצעות ההבנה שיש לשאבם שאוותם מבקשיםשוב ושוב, וכך למרות שאין בהם כל שינוי, המידע שלהם עבר מספר רב של פעמים על גבי רשת האינטרנט.

לצורך הדוגמה, חישבו על כך שבכל פעם שאתה גולשים לאתר [eth0](#) כדי להתקען בחדשנות, תמונה הלוגו של [Ynet](#) תעבור מהשרת אל הדפדפן שלכם (הליך). הדבר נכון גם לגבי שאר הכותרות ומציע העיצוב שקבועים בדף - כל זה גורם לתעבורה "מיותרת", שהרי המידע היה כבר בדף שלכם בעבר, ומazel הוא לא השתנה! התעבורה המיותרת גורמת גם ל贋זז של עליות (רוחב הפס וכמות המידע שעוברת על גבי), וגם של זמן (בהתנה לשאב שיגיע).

הweeney של מנגנון ה-[Cache](#) (מתומו) הוא לשומר משאבים كالה על המחשב של הלקוח, וכל עוד הם לא משתנים, לטען אותם מהדיסק המקומי ולא על גבי הרשת.

המנגנון שמאפשר לבצע זאת בפרוטוקול HTTP נקרא Conditional-Get (בקשת GET מותנית).משמעות השם: בקשה-GET תבצע רק **בתנאי** שלא קיים עותק מקומי ועכני של המחשב.

איך ידע הלקוח האם העותק שקיים אצלו עדכני, או שבשרת כבר יש גרסה חדשה יותר? באמצעות קר שיעביר לשרת (יחד עם בקשה-GET) את הזמן בו שמר את המחשב. השרת יחזיר את המחשב רק אם יש לו גרסה חדשה יותר (כלומר, שנוצרה לאחר הזמן שמצוי בבקשת).

לצורך ההמחשה - נחשוב על המקירה בו נג露ש באמצעות דף-פנ לאתר `net.yz`. הדף-פנ יבקש את הלוגו של `net.yz`. אבל רק אם הוא השתנה מאז הפעם האחרונה שהדף הציג את האתר ושמר עצמו אצלו. ברוב המוחלט של הפעמים, הלוגו לא השתנה, והשרת פשוט יורה לדף-פנ: "תשמש בעותק שקיים אצלך". וכך נחסכה העברת הלוגו של `net.yz` שוב ושוב ברשות.

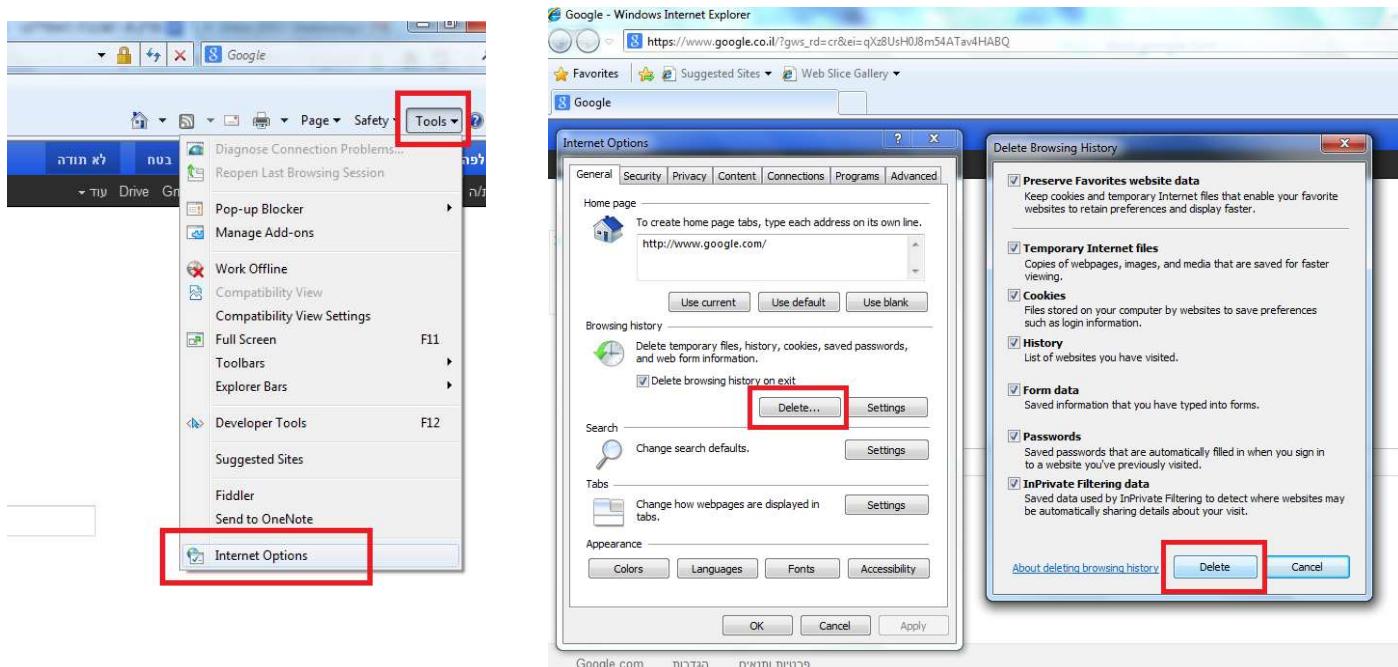
ננסה להבין את המנגנון זהה באמצעות תרגיל מעשי:

תרגיל 4.12 מודרך - הבנת Caching באמצעות Wireshark

שיםו לב שרוב הדף-פנים כולם שומרם ב"מטען" דפים שהם הורידו ולא מורידים את הדף מחדש בכל פעם, אלא רק כשהם חשובים שיש בהזורה.

1. היכנסו ל-Internet Explorer ורוקנו את המתמן:

tools->Internet Options->Delete Browsing History -> Delete

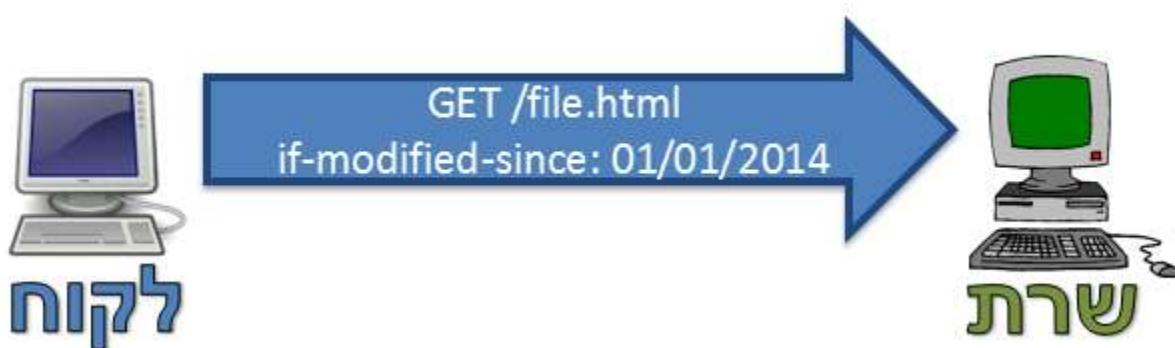


- .2. סיגרו את הדף. פיתחו אותו מחדש.
 - .3. מה, לא פתחתxs הנספה עדין? קדימה!
 - .4. היכנסו לעמוד הבא: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
 - .5. חכו עד שהדף יסימן לטען את העמוד.
 - .6. כעת היכנסו שוב לעמוד זהה (או לחצו על **(Refresh / F5)** .).
 - .7. הפסיקו את ההספנה.
 - .8. הסתכלו על הודעת-GET הראשונה ששלחנו.
- אם היא כוללת שדה Header בשם **?"IF-MODIFIED-SINCE"**
אם כן, מה הערך שמופיע שם? מודיעו הערך זהה? אם לא – למה?
- .9. בידקו את תשובת השרת. האם הוא החזיר את התוכן של העמוד? אם לא – מדוע? כיצד ידעתם זאת?
 - .10. הסתכלו על הודעת-GET השנייה ששלחנו.
אם היא כוללת שדה Header בשם **?"IF-MODIFIED-SINCE"**
אם כן, מה הערך שמופיע שם? מודיעו הערך זהה? אם לא – מה?
 - .11. בידקו את תשובת השרת.
אם הוא החזיר את התוכן של העמוד? אם לא – מדוע? כיצד ידעתם זאת?



כיצד עובד מנגנון **Cache** וביקשת-**Conditional-GET** ?Conditional-GET

בשלב הראשון, הלוקו פונה בבקשת משאב מסוים. הלוקו מציין שברצונו לקבל את המשאב, רק אם זה שונה מתאריך מסוים. בדוגמה הבאה:



הלוקו מבקש את המשאב file.html. הלוקו מבקש שהשרת ישלח לו את הקובץ זה, אך ורק אם הוא השתנה מאז ה-01.01.2014. הסיבה היא, שהлокו שומר בתאריך זה את הקובץ file.html ל-cache שלו. لكن, אם הקובץ לא

השתנה מאז ה-01.01.2014, הרי שהעוטק של השרת זהה לעותק שנמצא ב-Cache של הלוקו. במקרה שבו הקובץ לא השתנה, השרת עונה בתשובה **HTTP 304 Not Modified**:



בשלב זה, הלוקו יכול לטעון את המשאב מtower ה-Cache, ולא לקבל אותו מהשרת.

במקרה אחר, יתכן שהקובץ התעדכן, נאמר ב-01/03/2014. אי לכך, השרת יחזיר את הקובץ המעודכן בתשובה:



במקרה זה, הלוקו מבין שהעוטק השמור אצליו אינו עדכני. הוא טוען את הקובץ מהשרת, ומחליף את הקובץ המקורי把他 Cache החדש.

Cookies ("עוגיות")

במהותו, HTTP הוא פרוטוקול "חסר-מצב" (באנגלית: stateless) או "חסר-זיכרון". המשמעות היא שכל בקשה מטופלת בפני עצמה, ללא קשר לבקשת הקודמות - בבקשת מסווג GET אפשרות גישה למשאבי אינטרנט (תמונות, מסמכים וכו'), ובקשות POST מאפשרת מידע משליך ללקוח אל השירות (טפסים, תמונות, מיילים וכו').

עם זאת, בהרבה אפליקציות ואטרוי אינטרנט נפוץ הרעיון של session (פעילות ממושכת) - למשל באתר Amazon, ניתן להוסיף עוד ועוד מוצרים לעגלת הקניות, וכן לקבל הצעות למוצרים על סמך המוצרים שבגלאה, ולאחר מכן לבצע תשלום כרשימת המוצרים שבחרנו לאורך ה-session כבר סוכמה.

 **תרגיל לחשיבה עצמית:** כיצד ניתן ממש מגננון של עגלת קניות? מה נדרש לצורך כך בשרת? מה נדרש באפליקציה הלוקה?

השרת צריך "לזכור" עבור כל session של משתמש אילו מוצרים כבר הוספו לעגלת הקניות ואילו מוצרים הוציאו לו; בכל תשובה לבקשת GET של הלוקה, על השירות לצרף את רשימת המוצרים שכרגע בעגלת הקניות, וכן מוצרים רלוונטיים על סמך המוצרים שכבר בഗלאה; בכל בקשה POST (להוספה נוספת לעגלת), על השירות להוסיף את המוצר לרשימת המוצרים שבגלאת הקניות של ה-session.



שימוש לב שהמידע אודות ההזמנה מגיע אל השירות בשלבים, למשל, באמצעות מספר בקשות שונות; ראשית מגיעה הבקשה להוסיף את מוצר'A' לרכישה, לאחר מכן הבקשה להוסיף את מוצר'B', ולבסוף הבקשה לביצוע ההזמנה, המספקת את פרטי אמצעי התשלום וכתוות המשלוח.

תקשורת מהצורה זו מחייבת את השירות "לזכור" מידע בין טיפול בבקשתות שונות. חשוב גם לשים לב שהשתר ציריך "לזכור" בנפרד את עגלת הקניות לכל session של כל משתמש; אסור שיבלב בין עגלות של sessions שונים.



תרגיל לחשיבה עצמית: כיצד ניתן לעשות זאת? שימוש בREQUEST HTTP, כיצד יידע השירות לאיזה session היא שייכת?

הפתרון לכך הוא שימוש במנגנון **cookies** ("עוגיות") היא מחורזת שמשותפת לשרת וללקוח; השירות קובע את המחרוזת הזו בתחילת session, ולאחר מכן יצרף את המחרוזת הזו לכל בקשה שלו (בשני המקרים עושים שימוש ב-**HTTP Header fields** כדי להעביר את המחרוזת הזו).

את ה-**cookie** ישמר הלוקה בדיסק המקומי, וכן היא תישמר בסיס הנתונים של השירות. ל-**cookie** נקבע אורך חיים, כך שללאחר זמן מסוים פג התוקף שלה, והשירות יקבע מחדש session חדש.

על מנת להבין את האופן בו עובד מנגנון **cookies**, נשתמש בדוגמה. נאמר שיש לנו לקוח בשם client123 שניגש אל שירות קניות באינטרנט:



כעת, האתר מעוניין לשמור מידע של הלוקה עצמו. לכן, הוא מייצר עבורו מידע. נאמר שהשירות בחר בمخזה 24601. כעת, השירות יאמר ללקוח להשתמש מעתה בمخזה זה:



עתה ואילך, כל עוד זמן התוקף של העוגיה לא פג, בכל פניה שהלקוח יבצע אל האתר הקניות זהה, הוא ישלח גם את המזהה שלו:



באופן זה, בכל פעם שהלקוח יפנה לשרת, השרת יוכל לראות את המזהה ולחזות שמדובר בלקוח(client123)



- אוטנטיקציה (אימות וזיהוי) Authentication

הבעיה שאיתה מתמודד מנגנון אוטנטיקציה היא "זיהוי" של משתמש - חישבו על כך לשירות המיל שלכם (לדוגמה: Gmail או Yahoo! Mail) יש כנראה מיליוןים או אפילו מאות מיליוןים של משתמשים.

כשאתם נכנסים אליו, השירות צריך לוודא שתתקבלו רק הودעות שנשלחו אליכם.

בשביל לעשות זאת, השירות צריך לבצע שני דברים:

1. **זיהוי** - לידע מהו המשתמש.
2. **אימות** - הרי לא מספיק שהמשתמש "igyid" מהי הזהות שלו; כל אחד יכול לגלוש ל-Twitter ולטעון שהוא אשטן קוצר. זה עדין לא יספיק כדי לקבל גישה לحسابו Twitter של אשטן קוצר ולפרסם הודעות בשמו.

על מנת לקבל גישה לحسابו של זהות מסוימת, המשתמש צריך להוכיח שהוא אכן בעל זהות. בדרך הנפוצה ביותר לבצע את ההוכחה זו היא באמצעות סיסמה, שכוראה רק בעל הזהות האמתי אמרור לדעת, ושתמנע משתמשים להתחזות לזהות שאינה שלהם.

בפרוטוקול HTTP מוגדר מנגנון אוטנטיקציה, שמשמש לזיהוי ואימות של משתמשים. הוא די "חלש" בהשוואה למנגנונים נפוצים אחרים (כמו SSL, שלא נלמד במסגרת הפרק זהה), אבל ראוי ללמידה כדי להבין את ההתפתחות של המנגנונים האלה.

לפני שניכנס לפרטים לגבי איך בדיק עובד מנגנון האוטנטיקציה ב-HTTP, נבחן בצורה בסיסית מאיו כמה עקרונות חשובים של אוטנטיקציה ושימוש בהצפנה, ומה קובע את מידת ה"חוזק" (או ה"חולשה") של מנגנון.

הדבר הראשון שחייב להבין הוא שהפרטיות המשתמשים היא נושא רגש מאיו - אם יתפרנסו דרכי פשוטות לפוץ לאחד שירותי הדואר האלקטרוני הפופולרי, או לאחת הרשותות החברתיות הגדולות, אפשר רק לדמיין את הבלהה שתיזכר²⁵.

אם ככה, על מנת להגן על פרטיות של משתמשים, ולמנוע פריצה לحسابות שלהם, חשוב להקפיד על כך שלב האימות (שמוזכר לעיל), בו מציג מעביר המשתמש סיסמה לשרת כדי להוכיח את זהותו שלו, יבוצע בצורה מוצפנת שמנעת מגורם שלishi "להאזין" לתקשורת זו ו"לගונב" את הסיסמה של המשתמש.

²⁵ מספיק להזכיר בבהלה שהיא הייתה בשנת 2013, כשהתפרנסו ידיעות לגבי זה שסוכנות המודיעין האמריקאית NSA יכולה לפצח את מנגנוני האבטחה של חברות האינטרנט הגדולות כדי לעקוב אחריו התכתיויות של משתמשים. במשך החודשים שלאחר מכן, כל חברות האינטרנט הגדולות (Google, Facebook, Yahoo, ... ואחרות) הגיעו בהכחשה של הידיעות האלה, ופירסמו את השיפורים שעשו במנגנוני האבטחה שלהם, כדי לוודא שלאף גוף אין גישה למידע פרטי של משתמשים. למւניינים - ניתן לקרוא עוד בעמוד: <http://goo.gl/iRjICp>

כל שהצפונה של הסיסמה נעשית בצורה מתחכמת יותר וקשה לפריצה, נאמר שמנגנון האותנטיקציה "חזק" יותר. לקריאה נוספת על מגנוני אוטנטיקציה, הנכם מוזמנים לפנות [לפרק "צעדים להmarsר" של פרק זה](#).



שים לב: חשוב להבין שבטחה באינטרנט מהו גודל ומורכב מאד. ההסבר שנייתן כאן הוא פשוטי ביותר, ויתכן שלא ברור עד הסוף לחלק מהקוראים. מכיוון שעיקר העיסוק בפרק זה הוא שכבת האפליקציה, ולא בטחה, מנגנון האותנטיקציה מוצג כאן רק כדי להמחיש "על קצה המזלג" אוסף של בטחה בשכבה האפליקציה. הבנה עמוקה של נושאים בטחה והצפונה אינה הכרחית כדי למדוד את ה프וטוקול HTTP, וקוראים חלק האותנטיקציה לא ברור להם, לא יפגעו אם ידלו לחלק הבא.

כאמור, מנגנון האותנטיקציה של HTTP נחשב למנגנון "חלש", כפי שנראה בתרגיל הבא:



תרגיל 4.13 מודרך - הבנת אוטנטיקציה מעל HTTP באמצעות Wireshark

1. רגע, אנחנו שוב לא עם הסנפה עובדת? קידמה, קידמה...
 2. כעת גשו לאתר המוגן באמצעות סיסמה; היכנסו לאתר הבא: <http://http-demo.appspot.com/1>. שם משתמש: admin, סיסמה: secret.
 3. הפסיקו את הסנפה והסתכלו בתכנית.
 4. קראו על אוטנטיקציה ב-HTTP בעמוד: <http://goo.gl/8UvnN>
 5. כאשר שלחנו את פקעת ה-GET הראשונה, מה הייתה תגובת השרת? מה היה Status Code שלה?
 6. כשלחנו פקעת GET נוספת, לאחר הזנת שם המשתמש והסיסמה, איזה שדה נוסף נוסף ב-HTTP?
 7. שם המשתמש והסיסמה שהזנו מקודדים (encoded) במחוזת התווים שמופיעות אחרי "Authorization:Basic". שימו לב כי הנתונים אינם מוצפנים, אלא מקודדים בלבד באמצעות קידוד בשם base64, שהוא ניתן לפענה בקלות!
 8. היכנסו לאתר הבא: <http://opinionatedgeek.com/dotnet/tools/Base64Decode>, וביצעו decode למחוזת. האם הצליחם לפענה את שם המשתמש והסיסמה?
 9. האם יש בעיה באבטחה במודול ה-HTTP Authentication ?Basic Authentication
- אם כן – הסבירו מדוע ותנו דוגמא ל蹶ה בעיה. אם לא – הסבירו מדוע.

אל דאגה! ישנן דרכים לעשות את השימוש באינטרנט מאובטח הרבה יותר, אבל כדי למש אתן נדרשים כלים יותר מקיפים מאשר HTTP Basic Authentication. הקוראים הסקרים מוזמנים לפנות [לפרק "צעדים להmarsר" של פרק זה](#).

פרוטוקול DNS - הסבר כללי

DNS (ראשי תיבות של Domain Name System) הוא פרוטוקול נפוץ נוסף בשכבה האפליקציה. תפקידו העיקרי הינו לתרגם שמות דומיינים (כגון www.google.com, www.facebook.com או http://demo.appspot.com) לכתובת IP הרלוונטית. לבני אדם נוח יותר לזכור שמות טקסטואליים כגון "www.google.com" מאשר כתובות IP כמו "173.194.39.19". לשם כך מועד פרוטוקול ה-DNS. למעשה, ניתן לחשב על DNS כמוין "ספר טלפוני" - כמו שמספר הטלפונים מתרגם בין שם של אדם או עסק (שאותו יותר קל לבני אדם לזכור) לבין מספר טלפון, כך ה-DNS מאפשר לאתר כתובת IP באמצעות שם של האתר.

על מנת להבין את הצורך ב프וטוקול זה, ננסה להבין כיצד הדפדפן פועל כאשר מנוטים לגלוש לאתר מסוים. נזכיר בדוגמה הראשונה שריאנו בתחילת פרק זה, בה הכנסנו את הכתובת Wireshark://info.cern.ch/hypertext/WWW/TheProject.html

528 21.277073000	192.168.1.100	137.138.139.27	HTTP	618 GET /hypertext/www/TheProject.html	HTTP/1.1
531 21.352697000	137.138.139.27	192.168.1.100	HTTP	1077 HTTP/1.1 200 OK	(text/html)

שיםו לב שהדומיין בכתובת זהו הינו info.cern.ch, אך אם נזכיר בפרק [תכנות ב-Socket-Sockets](#) נבין שבתקורת אינטרנט לא ניתן להתחבר אל דומיין, אלא דרושה כתובת IP. בהسنפה לעיל ניתן לראות שהדפדפן פתר את הבעיה הזה בדרך כלשהי, ופנה אל הכתובת 137.138.139.27. בשלב שעליינו דילגנו בהסביר שמוופיע בתחילת הפרק, והוא שלב התרגום - תרגום שם של דומיין (במקרה זה - info.cern.ch) לכתובת IP (במקרה זה - 137.138.139.27). תרגום זה נעשה באמצעות פרוטוקול DNS.

בדומה לפרטוקול HTTP, גם פרוטוקול DNS פועל באמצעות בקשה (Request), שנקראת גם **שאילתא** (Query), ותשובה (Response). לפני הדפדפן ניתן לארח המבוקש (info.cern.ch), על מערכת הפעלה למצוא את כתובת ה-IP הרלוונטית. לשם כך, המחשב שלנו מתקשר לשרת DNS:



כעת, כאשר ללקוח יש את כתובת ה-IP של שרת המידע, הוא יכול לפנות אליו באמצעות פרוטוקול HTTP.

על מנת להבין את דרך הפעולה של שירות DNS כדי להמיר את שם הדומיין לכתובת IP, علينا להכיר כיצד שמות דומיין מורכבים. לשם כך, علينا להכיר את היררכיה השמות של DNS.

היררכיות שמות

DNS משתמש במבנה היררכי של **אזורים (Zones)**. התו המפריד שיוצר את ההיררכיה הוא התו נקודה (.). כך למשל, הדומיין com מTARGET שרת בשם "www.facebook.com" בתוקן האזור "www" שבתוך האזור ".com". הדומיין "he.wikipedia.org" מTARGET לשרת בשם "he" בתוקן האזור "wikipedia" שבתוך האזור "org".



חלוקת-h-DNS לאזורים מאפשרת חלוקת אחראיות ומשאבים. במצב זה, אף שרת DNS לא צריך לטפל בכל הדומיינים באינטרנט. לכל אזור יכול להיות שרת DNS שידאג אך ורק לאזורים והדומיינים שנמצאים תחתו. כך למשל, השירות שאחראי על כל הדומיינים ותת-הdomיין (subdomains) של google.com (mail.google.com ו-mail.google.com) לא צריך להכיר את www.facebook.com, www.google.com ו-he.wikipedia.org. השירות זה מכיר רק את הדומיינים ותת הדומיינים שתוחת com.google.com.he.wikipedia.org.

האזור הראשי בראשו הינו האזור Root המוצג בידי התו נקודה (.). למעשה, האזור com, כמו גם האזור org, מוכלים בתחום האזור Root. כך שם הדומיין המלא עבור "www.google.com" הינו למעשה "www.google.com." (שים לב לנקודה שמופיעה בסוף הכתובת).

תרגיל 4.14 מודרך - התבוננות בשאלית DNS



על מנת לראות שאלית DNS, פיתחו את Wireshark והתחילה הסנפה. אתם יכולים להשתמש במסנן הטעינה "dns". כעת, הייעזרו בכל nslookup אותו פגשנו לראשונה בפרק תחילת מסע - איך עובד האינטרנט? ?.

הריצו את שורת הפקודה (Command Line), ולאחר מכן, הריצו את הפקודה הבאה:

nslookup www.google.com

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

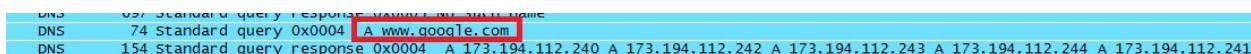
C:\Users\USER>nslookup www.google.com
Server: box.privatebox
Address: 192.168.14.1

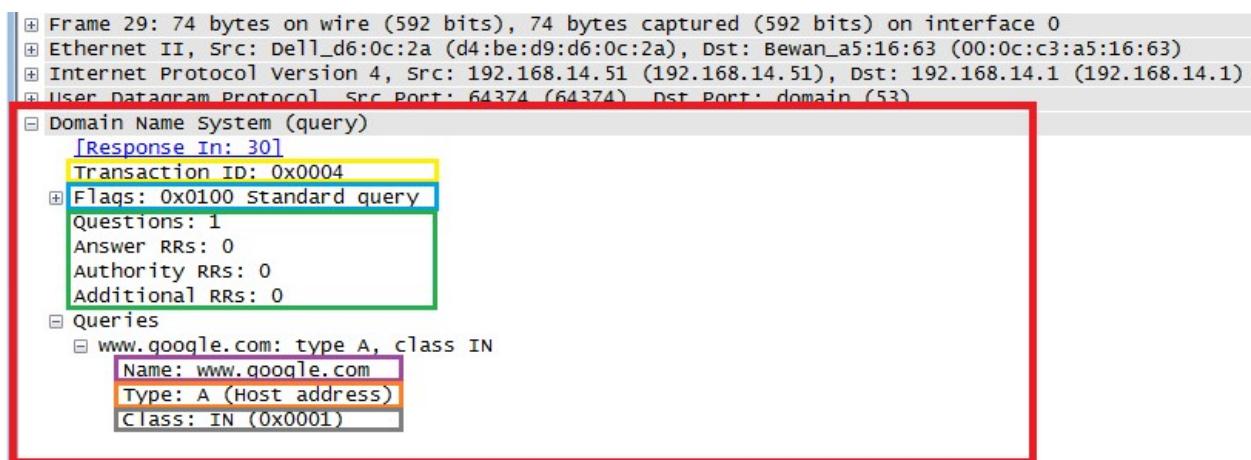
Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4017:801::1010
          173.194.112.240
          173.194.112.242
          173.194.112.243
          173.194.112.244
          173.194.112.241

C:\Users\USER>

```

הערה: יתכן שתראו יותר מואסף אחד של שאלתא ותשובות. רק את החבילה שכוללת את השאלה עבור שם הדומיין "www.google.com"

Cutet נטמוך בחבילת השאלתא (Query)



כפי שניתן לראות, פרוטוקול DNS (שהחלק שלו בחבילה מסומן באדום) נשלח בשכבה החמשית, מעל פרוטוקול UDP בשכבה הרביעית. Cutet נטמוך בשדות של פרוטוקול זה.

השדה הראשון, המסומן בצהוב, הוא שדה ה-ID Transaction. שדה זה כולל מידע של השאלה הנוכחית, על מנת להפריד אותה משאלות אחרות. כך למשל, השאלה זו קיבלת את המזהה 4. יתכן והשאלת הבאה מקבלת את המזהה 5. דבר זה מקל על המחשב להפריד בין התשובות שיקבל מהשרת, ולדעת איזו מהן שייכת לאיוז שאלת.

השדה השני, המסומן ב**כחול**, הינו שדה הדגלים (Flags). שדה זה מורכב משמונה דגליים בעלי משמעותות שונות. בדוגמה לעיל, הדגלים מצינים כי מדובר בשאלתא סטנדרטית. עבור תשובה, למשל, יהיו דגליים שונים.

השדות הבאים, המסומנים ב**ירוק**, מတאים כמה רשותות מכילה חבילת DNS. בשאלות ותשובות של DNS ישן רשותות, הנקראות **Resource Records** (או בקיצור **RR**, כפי ש-Wireshark מצין). כל רשותה כזו מכילה מספר פרטיים, כפי שתcanf נראה. בחבילת השאלה שלפנינו, ישנה רשותה שאלה אחת, ואין רשותות נוספות.

לבסוף, אנו רואים את רשותה השאלה. ב**סגול**, שדה השם (Name), שכולל את שם הדומיין המלא. בדוגמה זו, השאלה היא עבר השם `www.google.com`. ב**כתום**, אנו רואים את סוג (Type) הרשותה שעליה שואלים. כאן מדובר בסוג A, המתאר רשותה המפה בין שם דומיין לכתובת IP. ישן גם סוג רשותות נוספים, כמו הרשותה PTR העוסка בדיקת הדבר ההפוך - מפה בין כתובת IP לבין שם הדומיין הרלבנטי. באפור, אנו רואים את סוג הרשת (Class). בכל המקרים שאתה צפויים לראות, הסוג יהיה תמיד IN, וכך לא נתעכב על שדה זה.



תרגיל 4.15 - תשאול רשותות מסווגים שונים

בתרגיל הקודם, השתמשנו ב-**nslookup** על מנת לתשאול מה כתובת ה-IP של הדומיין `www.google.com`. כעת, אנו מבינים שלמעשה שלחנו שאלתא עברו רשותה מסווג A על השם `www.google.com`. ניתן לציין בפני **nslookup** במפורש עברו איזה סוג רשותה לתשאול, בצורה הבאה:

nslookup -type=<TYPE> <host/address>

כך לדוגמה, עברו תשאול רשותה מסווג A, ניתן לכתוב:

nslookup -type=A www.google.com

כפי שציינו קודם, קיימים גם סוגים נוספים, כגון PTR - שמתאר רשותה שטיפה בין כתובת IP לבין שם הדומיין שלו.

השתמשו ב-**nslookup** וגלו מהו שם ה-DNS עבור כתובת ה-IP הבאה: 8.8.8.8. מהו שם הדומיין שמצאתם?



תרגיל 4.16 מודרך - התבוננות בתשובה DNS

בתרגיל המודרך הקודם, שלחנו שאלתא מסווג A עבור הדומיין `www.google.com`, צפינו בחבילת השאלה שנשלחה. כעת, נתמקד בחבילת התשובה אותה הحسنפה:

```

Frame 30: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface 0
Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
Internet Protocol Version 4, Src: 192.168.14.1 (192.168.14.1), Dst: 192.168.14.51 (192.168.14.51)
User Datagram Protocol, src Port: domain (53), Dst Port: 64374 (64374)
Domain Name System (response)

[Request In: 29]
[Time: 0.001354000 seconds]
Transaction ID: 0x0004
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 5
Authority RRs: 0
Additional RRs: 0
Queries
www.google.com: type A, class IN
    Name: www.google.com
    Type: A (Host address)
    Class: IN (0x0001)
Answers
www.google.com: type A, class IN, addr 173.194.112.240
www.google.com: type A, class IN, addr 173.194.112.242
www.google.com: type A, class IN, addr 173.194.112.243
www.google.com: type A, class IN, addr 173.194.112.244
www.google.com: type A, class IN, addr 173.194.112.241

```

השדה הראשון, המסומן בצהוב, הוא שדה ה-ID. Transaction ID. באופן הגיוני, הערך הינו 4, זהה לערך עבור השאלה הראשית שראינו קודם. כך המחשב יכול לדעת שהתשובה זו שייכת לשאלתא שראינו קודם.

השדה השני, המסומן ב**כחול**, הינו שדה הדגלים (Flags). ניתן לראות שכעת הדגלים שונים מבמקרה של שאלתא. במקרה זה, הדגלים מעידים על תשובה חוזרת ללא שגיאות.

השדות הבאים, המסומנים ב**ירוק**, מתארים כמה רשומות מכילה חבילת DNS. במקרה זה ניתן לראות שינוי רשותת שאלתא אחת, ועוד חמש שאלות תשובה.

לאחר מכן, ב**אדום**, אנו רואים את רשותת השאלתא שראינו קודם לכן. בחבילות תשובה, שירות ה-DNS משכפלים את השאלתא שנשלחה אליהם ושולחים אותה חוזרת אל השולח.

לסיום, ב**סגול**, ניתן לראות את חמישה רשותות התשובה. אפשר לראות שישנן חמישה רשותות שונות, כאשר כל אחת מכילה כתובות IP שונות. הסיבה לכך היא שלעתים רשותת DNS מצביעה על יותר מכתובת IP אחת. במקרה זה, למשל, ניתן ואחד השירותים של Google לא יהיה זמין. במקרה זה, מערכת הפעלה תוכל לפנות אל כתובות IP אחרות, שאולי תקשר לשרת שכן זמין. כאן אנו לומדים על יתרון נוסף של מערכת DNS - היכולת לקשר כתובות IP שונות אחת ליותר מכתובת IP אחת.

נתמקד באחת הרשותות התשובה:

Answers

www.google.com: type A, class IN, addr 173.194.112.240	
Name: www.google.com	
Type: A (Host address)	
Class: IN (0x0001)	
Time to live: 3 minutes, 30 seconds	
Data length: 4	
Addr: 173.194.112.240 (173.194.112.240)	
+	www.google.com: type A, class IN, addr 173.194.112.242
+	www.google.com: type A, class IN, addr 173.194.112.243

המבנה דומה מאוד למבנה רשותה של שאלתא. אלו הם הרשותות:

- **בסגול** - שדה השם (Name), שכולל את שם הדומיין המלא. בדוגמה זו, התשובה היא עבור השם

www.google.com

- **כתום** - שדה סוג (Type) הרשותה. כאמור, מדובר בדבר מסווג A.

- **בպור** - סוג הרשת (Class). הערך הוא IN.

- **אדום** - Time To Live. שדה זה קובע כמה זמן יש לשמור את הרשותה ב-cache של הלקוח. בדומה ל-HTTP, גם עבור שאלות DNS לא נרצה לשאול שאלות סטם. במקרה זה, על הלקוח לזכור את כתובת ה-IP של Google במשך שלוש וחצי הדקות הקרובות, ורק לאחר מכן - לשאול שוב.

- **ירוק** - אורך (Length) המידע. שדה זה משתנה בהתאם לסוג השאלתא. כאן, הגודל הוא 4 - מכיוון שכתובת IP היא באורך של ארבעה בתים (bytes).

- **בכחול** - המידע עצמו (Data). במקרה של רשומה A, מדובר בכתובת ה-IP הRELNETית לשם הדומיין עליון נשלחה השאלה.

כעת, ברשות הלוקה יש את כתובת ה-IP של www.google.com, והוא יכול להשתמש בה כדי לתקשר עם השרת.



תרגיל 4.17 – תשאול DNS רקורסיבי (אתגר)

לאחר שהבנתם כיצד עובד תשאול רקורסיבי של DNS - הגיע הזמן למשוך זאת בעצמכם כדי לאלו מהי כתובת ה-IP של הדומיין maps.google.com. העזרו בכלים **nslookup** ובטייעוד שקיים אודוטויו ברחבי האינטרנט. בצעו את השלבים הבאים:

- בחרו בשרת Root כלשהו. איך מצאתם את כתובת ה-IP שלו? מה היא כתובת ה-IP?
- באמצעות תשאול שרotaNS, גלו מי הוא שרotaNS האחראי על ה-zone של com. מיהו שרotaNS? מה היא כתובת ה-IP שלו? מה הרצטם בכך לגלות זאת?
- באמצעות שרotaNS האחראי על ה-zone com, גלו מי הוא שרotaNS האחראי על ה-zone של google. מיהו שרotaNS? מה היא כתובת ה-IP שלו? מה הרצטם בכך לגלות זאת?
- באמצעות שרotaNS האחראי על ה-zone google.com, גלו מה היא כתובת ה-IP של maps.google.com. מה היא הכתובת? מה הרצטם בכך לגלות זאת?



תרגיל 4.18 – goog elgoog (אתגר)

เครดיט: איל אבני



הבוט הביס הצל שי הארכנ, אל פאו - החיתוף טפשטב ליחתטן ליגרט לכ ישנים דברים שעושים בסדר הנהוג, וישנים דברים שנעשים לחלוטין הפוך. אנחנו נתחיל מהסוף, ובתקווה – נצליח לשנות אותו!

בתרגיל זה ניצור שרotaDNS, שיגרום לך שככל פניה שלנו ל-www.google.co.il תנותב לכתובת אחרת. טרם תתחילה לעבוד על התרגיל, שימו לב לקרוא היטב את ההוראות ולודא שאתם מבינים אותן.

הכוונות:

בחלק זה נציג הסוגה של DNS QUERY ו-DNS RESPONSE עבור www.google.co.il, כדי שנוכל בהמשך לzechot את ה-QUERY המבוקש ולגרום לשרת שלנו להחזיר תשובה תקינה. ראשית, וודאו שאתם מחוברים דרך כבל הרשת. כתע, הריצו את שורת הפקודה הבאה (חשוב: פתחו CMD בהרשאות Administrator). השורה תגרום למחיקת ה-cache של ה-DNS שלכם, כיון שסביר שכותבת ה-IP של גугл כבר נמצאת שם ולפיכך לא תבוצע DNS QUERY אותה אנחנו רוצים למצוא:

```
ipconfig /flushdns && taskkill /F /IM iexplore.exe
```

פתחו את Internet Explorer.

פתחו Wireshark והפעלו הסוגה עם ה-filter הבא:

```
udp.port == 53
```

אלשו ל: www.google.co.il באמצעות Internet Explorer

עצרו ושימרו את ההסוגה ב-Wireshark.

הורידו את הקובץ

data.cyber.org.il/networks/gvahimchallenge/elgoog.rar

חלצו את הקבצים מתוך rar שהורדתם לתיקית התרגיל והריצו את הקובץ elgoog_set.bat בהרשאות Administrator.

הסקריפט יגידיר את שרת ה-DNS הראשי שלכם C-1.0.0.127, ואת המשני בתור השרת הקודם שהוא לכם. השרת המשני נכנס לפעולה במידה שתרת הראשי אינו מוחזיר תשובה, וכן תוכלן להמשיך להשתמש באינטרנט.

התרגיל:

עליכם לכתוב שרת שיקבל את בקשות-h-DNS עבור כתובת ה-IP של il.google.co.il, ויחזר כתובת IP של שרת אחר.

מצאו בהסוגה את בקשת-h-A עבור שם הדומיין google.co.il וו-ww ואת התשובה לה.

* מומלץ להיעזר באופציה של Follow UDP/TCP Stream על הבקשה שמצאתם.

בשלב הבא, צרו שרת UDP שייזין בבקשתות-h-DNS. כיון שטרם למדנו לכתוב שרת UDP, תוכלן להשתמש בשולץ התוכנית הבאה. עליכם לגרום לשרת להאזין על הפורט של UDP ולכתוב את הפונקציה dns_handler. בפרק הבא, תלמדו לעומק על פרוטוקול UDP ובבינו כיצד הקוד פועל.

```

import socket

DNS_SERVER_IP = '0.0.0.0'
DNS_SERVER_PORT = ???
DEFAULT_BUFFER_SIZE = 1024

def dns_udp_server(ip, port):
    """
        Starts a UDP server on a given IP:PORT, and calls
        dns_handler(data, client_address)
        prototyped function on any client request data.
    """
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((ip, port))
    print "Server started successfully! Waiting for data.."
    while True:
        try:
            data, addr = server_socket.recvfrom(DEFAULT_BUFFER_SIZE)
            dns_handler(data, addr)
        except Exception, ex:
            print "Client exception! %s" % (str(ex), )

def main():
    """
        Main execution point of the program
    """
    print "Starting UDP server.."
    dns_udp_server(DNS_SERVER_IP, DNS_SERVER_PORT)

if __name__ == '__main__':
    main()

```

בפעם הבאה שהבקשה הספציפית הזאת מופיעה, החזירו את הכתובת 212.143.70.40 במקום את כתובות ה-IP
שחזרו במקור.

ברגע שאתם חושבים שסימתם והכל רץ כמו שצרי, גלשו מחדש לכתובת. מה קרה?

הערות וטיפים:

כשסימתם את העבודה, לפני שהולכים הביתה, עליים להריץ את הקובץ המצורף: elgooG_revert.bat
בהרשות Administrator.

שיםו לב שהשורה של nslookup עשויה להחזיר שגיאה במידה ו-Internet Explorer כבר סגור – זה בסדר גמור
וכפוי.

מומלץ להריץ את כל התרגיל בתור administrator, גם כבודדים את הסקריפט שלהם דרך cmd, וגם כאשר
שעובדים עם PyCharm

לפניהם עונים לבקשת הספציפית שאתם רוצים לענות לה עם התשובה שלכם – מומלץ לראות שאותם מקבלים מידע כמו שציריך.

חשוב טוב לפניהם שאתם רצים לכתוב – איך אתם רוצים שהקוד יראה, ומה בסופו של דבר התוכנה שלכם אמורה לעשות.

בכל פעם שתרצו לבדוק את התוכנה שלכם, מומלץ שתבצעו את ההכנות מחדש עד לשלב פתיחת הדפסן.

כדי להכניס מידע בינהו למחוזת בפייתון, עליכם להוסיף א' לפני כל בית – אז את ערכו הhexadicלימי. לדוגמה, במידה ונרצה להכניס את הערכים 37hex 0x13, נכתב בפייתון:

```
my_binary_string = '\x13\x37'
```

שימוש לב פרוטוקול DNS הוא לא פרוטוקול טקסטואלי. הייעזרו בפונקציה `hex` בפייתון. לדוגמה – =
`hex(213)` .0xD5

בתרגיל אין צורך לבצע פעולות על שכבת התעבורה, שכן פרוטוקול DNS עובר בשכבה האפליקציה. לכן הדבר היחיד שצריך לבצע בקשר לשכבת התעבורה הוא פתיחת socket UDP.

אתגרים נוספים:

1: גרמו לבקשת הקודמת להפנות ל-127.0.0.1, פתחו את שרת ה-HTTP מהתרגילים הקודמים שלכם. מה קרה?

2: גרמו לתוכנה שלכם להפנות ל-IP לפי כתובות מוגדרת מראש – בצורה גנרטית. כך שלדוגמא גלישה ל-
`nana10.co.il` תפנה לכתובת `google.co.il` של גוגל, וגלישה ל-`o.co.il` תפנה אותנו אל `o.co.il`. נוסף, יוכל
 ללא מאמץ רב להוסיף כתובות נוספות בעתיד.

מבחן פרוטוקול - SMTP

עד כה למדנו על שני פרוטוקולים נפוצים בשכבה האפליקציה. כעת יש בידיכם את הכלים להבין בעצמכם איך פועלים פרוטוקולים בשכבה האפליקציה. כדי לתרגל זאת נבצע מבחן של פרוטוקול שנקרא SMTP. כל מה שעלייכם לדעת בשלב זה, הוא ש-SMTP הוא פרוטוקול שהוא נפוץ מאוד בעבר ושימש לשילוח דואר אלקטרוני. את יתר הפרטים אודות הפרוטוקול תסיקו בעצמכם.

בטור התחלת, **תאבחן** את הprotokol, כלומר תנתחו בעצמכם, מתוך דוגמה של שימוש בprotokol, את הפוקודות, השלבים והפרמטרים שלו. לאחר מכן, **תמשו** ל Koho SMTP בפייתון שיישלח בעצמו דואר אלקטרוני.

הורידו את קבצי ההסנפה pcap מהכתובת [smpt1.pcap](http://data.cyber.org.il/networks/c04/smtp1.pcap) ו- [smtp2.pcap](http://data.cyber.org.il/networks/c04/smtp2.pcap).

קבצים אלו מכילים הסנפות של מחשב מסוים. בכל הסנפה ישנו משלוח מייל באמצעות protokol SMTP. מטרתכם היא להבין את protokol SMTP. כמובן – להבין את מבנה protokol, הדרך שבה הוא עובד ואייר יש לרשום בו הודעות על מנת להצליח לשלווה אימיילים באמצעותו. תעשו זאת באמצעות שימוש Wireshark, ניסוי וטעייה, ולא שימוש במקורות חיצוניים. פעולה זו, של הבנת דרך הפעולה המתובנות בהסנפות, נקראת "אבחן" והיא שימוש במקירים בהם או צרכיים להבין איך עובדת מערכת מביי לנו תיעוד מלא שלה.

התרכזו ושימושו דגש בצד הלקוח של protokol (הצד **שלו**ת את המייל), אין צורך להיכנס לפרטים עבור צד הרשות.



הנחה חשובה: אין להשתמש או להיעזר באינטרנט עבור תרגיל זה!

הנחיות נוספות:

- ראשית, הבינו מה היא כתובת ה-IP של המחשב שמננו מתבצעת ההסנפה.
- סננו את הפקודות כך שתראו רק את פקודות של protokol SMTP.
- שימוש לב: לאחר הסינון יתכן ותראו גם פקודות מסווג protokol IMF. התעלמו מהן במהלך התרגיל.
- היעזרו ב- TCP Stream Follow, אותו הכרתם בפרק Wireshark ומודל חמש השכבות Follow /TCP Stream TCP/UDP Stream, כדי לראות את מהלך protokol בצורה נוחה.
- נסו להבין – מה הן הפקודות שבprotokol? מה הן הבקשות והתשובות? אילו פרמטרים יש לכל פקודה?
- לאחר התרגיל, זכרו את מטרת protokol SMTP – לשלווה מיילים. עצרו וחשבו כיצד אתם שולחים דואר אלקטרוני – איזה מידע אתם נדרש לספק לשם קר? איזה מידע משתנה בין מייל ואיזה מידע לא משתנה? חפשו זאת בקבצי ההסנפה.

- נסו להבין את התמונה הכללת לפני שאתם צולמים לפרטיהם. למשל, לו הייתם מדרשים לאבחן את פרוטוקול HTTP, חשוב קודם להבין שהמבנה הבסיסי הוא בקשה-תגובה, וכי יש סוגים שונים כמו GET ו-POST, הרבה לפני שצללים לסוגי Headers ה-HTTP השניים.

- במהלך התרגיל, תצטרכו להשתמש בקידוד Base64. את הקידוד הזה פגשנו באוטנטיקציה של HTTP, ואז השתמשנו באתר אינטרנט כדי להמיר אל הקידוד ובחרורה. הפעם נשתמש בפייטון.

על מנת לקודד מחרוזת לקידוד Base64 באמצעות פייטון, משתמש בפונקציית **encode**:

```
>>> my_string = 'This is a string...'
>>> my_string.encode('base64')
'VGhpcyBpcyBhIHN0cmliZW==\n'
```

משמעותו לב Ci פייטון מוסיף למחרוזת את '\n' (הטו של ירידת שורה) שאינו חלק מקידוד Base64, ולכן עליהם להסירו. סימני השווה (=) הם כן חלק מקידוד Base64.

על מנת לבצע את הפונקציה הפוכה, משתמש בפונקציית **decode**:

```
>>> 'VGhpcyBpcyBhIHN0cmliZW==\' .decode('base64')
'This is a string...'
```



תרגיל 4.19 - שימוש בקוח SMTP

בתרגיל זה תמשכו בעזרת פייטון לkish שילוח דואר אלקטרוני באמצעות פרוטוקול SMTP.



הנחה חשובה: יש להשתמש בספריית **socket** בלבד. אין להשתמש בספריות עזר כגון **smtplib**.

על מנת לעשות זאת, תמשכו במדמה שרת SMTP, שנמצא בכתבota: networks.cyber.org.il. השרת מאזין

על פורט סטנדרטי של SMTP, שמספרו 587.

מדמה השרת של גברים מקבל את המידע שנשלח מהלקוח שלכם. אם המידע התואם את דרישות הפרוטוקול, השרת יחזיר לכם תשובה לפי הפרוטוקול וימתין לבקשת הבאה. אם הלקוח שלכם שגה בימוש הפרוטוקול, השרת יחזיר תשובה "הבקשה אינה לפי הפרוטוקול" וינתק את ההתקשרות.

אם מימשتم את הפרוטוקול נכון וуд סופו, השרת גברים יחייב לכם על סיום התרגיל בהצלחה.

dagshim limish:

- השרת מקפיד על כך שכל הודעה שנשלחת אליו תתאים להודעה שבפקודותشبוקובץ ההסנהה.
- שימוש לביצורי תוכים מיוחדים, אם צריכים להיות כאלה, בסוף כל הודעה.
- השרת ניתק אתכם? תקנו את הבעה והתחברו שוב.

- השרת אינו בודק שהסיסמה ושם המשתמש שלכם מתאימים. במילויים אחרות - אפשר להתחבר עם כל שם משתמש וסיסמה.
 - השרת אינו מתיימר למש את כל הפקודות של פרוטוקול SMTP, אלא רק לוודא שהצלחתם לאבחן את הפרוטוקול בקורס מוצלחת.
 - כמובן שהשרת גם אינו שולח את המייל ששלחתם ☺ זהו תרגיל אבחוני בלבד.
- הצלחה והנאה בתרגיל זה!

שכבה האפליקציה - סיכום

בפרק זה סקרנו לעומק את שכבה האפליקציה. לאחר שהבנו את המטרות של שכבה זו, התמקדו בפרוטוקול HTTP. הבנו את מבנה הבקשה והtagובה של ה프וטוקול, וכן הכרנו את Headers השונים ואילו ערכיהם יש בהם. לאחר מכן, כתבנו שירות HTTP שהלך והפתחה. בתחילת הגשו קבצים בתגובה לבקשת GET. לאחר מכן, תמכנו בסוגים שונים של תשובות, ובהמשך סיפקנו תשובה תכניתית לבקשת, בהתאם לנוטונים שהלכו העבר בפרמטרים של בקשת GET.

כשהסתכלנו על המשמעות של פרמטרים שונים בבקשת GET, ראיינו שירותים אמיתיים כגון Google, Twitter, YouTube או Wikipedia. הצלחנו לבנות בעצמנו בקשות עם פרמטרים שהשפיעו על השירות, כגון בקשה לקבלת הוראות נסעה מכתובת אחת לכתובת אחרת ב-Google Maps. לאחר מכן, למדנו גם על בקשות POST ותוכנתנו תוכנת צד לקוח. לבסוף הכרנו גם נושאים מתקדמים ב-HTTP, כגון מתומן (Cache), עוגיות (Cookies) ואוטנטיקציה (Authentication).

בהמשך הפרק, הכרנו גם את פרוטוקול DNS והבנו שהוא משמש בעיקר לתרגום בין שמות דומיין לכתובות IP. ראיינו כיצד בניה שאלת DNS, וכן כיצד בניה תשובה. למדנו להשתמש בכל nslookup כדי לחשאל רשומות מסווגים שונים, הכרנו את מושגי Zone ו-hs-Records.

לסיום, הבנו את פרוטוקול SMTP באמצעות תרגיל. בתחילת אבחן את הprotokol, והבנו מתחם הסופות את דרך הפעולה שלו. לאחר מכן, הצלחנו לכתוב בפייתון לקוח שלוח מייל באמצעות פרוטוקול SMTP. כך, יחד עם HTTP ו-DNS, הכרנו שלושה מימושים שונים בשכבה האפליקציה.

בפרק הבא, נמשיך להתקדם במודל השכבות ולסקור שכבות נוספות - החל משכבה התעבורה, ועד לשכבה הפיזית. לפני שנוכל לעשות זאת, נלמד כללי חשוב נוספת לנו לתרגל את החומר שנלמד בשכבות הנמוכות יותר - Scapy.

שכבה האפליקציה - צעדים להמשך

על אף שהעמוקנו רבות מידע שלנו בשכבה האפליקציה, נותרו nonetheless רבים בהם לא נגענו. שכבה זו מתאפסית במספר רב מאוד של שימושים ופרוטוקולים אותם לא הכרנו כלל. בנוסף, ישנו nonetheless רבים ב-HTTP וב-DNS בהם לא העמוקנו את המידע שלנו.

אלו מכם שמעוניינים להעמק את הידע שלהם בשכבה האפליקציה, מוזמנים לבצע את הצעדים הבאים:

קריאה נוספת

נושאים מתקדמים בפרוטוקול HTTP

- הিירות עם שימושים נפוצים של שרת HTTP - לדוגמה SimpleHTTPServer, בכתב:
- <http://goo.gl/z2DtZf>
- מנגנוני אוטנטיקציה - ניתן לקרוא בדף: <http://en.wikipedia.org/wiki/Authentication>
 - באופן ספציפי, על המנגנון של Basic Authentication של HTTP, ניתן לקרוא עוד בדף: http://en.wikipedia.org/wiki/Basic_access_authentication
 - על המנגנון SSL/TLS, ניתן לקרוא בדף: http://en.wikipedia.org/wiki/Transport_Layer_Security (עדיף לעשות זאת לאחר לימוד פרק שכבת התעבורת).

נושאים מתקדמים בפרוטוקול DNS

- על מנת להבין את הדרך שבה באמצעות שאילתא, מומלץ לקרוא אחד מהמקורות הבאים:
- פרק 2.5.2 (Overview of how DNS works) בספרו Computer Networking: A Top-Down Approach James F. Kurose (מהדורה ששית) מאות.
 - פרק 2.2.6 (DNS Queries) בספר Pro DNS and BIND, בכתב:
- <http://www.zytrax.com/books/dns>

פרוטוקולים נוספים בשכבת האפליקציה

בשכבת האפליקציה יש אינספור פרוטוקולים, והמספר רק הולך וגדל. עם זאת, מומלץ לקרוא ולהכיר פרוטוקולים מסווגים שונים, כגון:

- קבלת דואר - פרוטוקול POP. ניתן לקרוא כאן:
http://en.wikipedia.org/wiki/Post_Office_Protocol
- העברת קבצים - פרוטוקול FTP. ניתן לקרוא כאן:
http://en.wikipedia.org/wiki/File_Transfer_Protocol

פרק - 5 Scapy

מבוא ל-Scapy

בפרק [תכנות ל-Sockets](#) למדנו כיצד ניתן להשתמש בפייתון ובספריה **sockets** כדי להצליח לפתח בעצמנו אפליקציות. בעוד Sockets הוא כלי מצויין בכך לכתוב קוד לשכבה האפליקציה, הם לא עוזרים לנו לבצע פעולות בשכבות נמוכות יותר, עליו נלמד בפרק הבאים. לשם כך - נזכיר את Scapy.

Scapy היא ספרייה חיצונית ל-python שמאפשרת שימוש נוח במשקי הרשות, הסנפה, שליחה של חבילות, ייצור חבילות וניתוח השדות של החבילות. עם זאת, שימוש לבו-sh-[Scapy](#) רצה "מעל" python, ולכן כל הfonקציות המוכרות לנו – כגון print או dir, עדין יעבדו ונוכל להשתמש בהן.



מה למשל אפשר לעשות עם Scapy?

באופן כללי - כל מה שעולה בדמיונכם שנייתן לעשות ברשות.

באמצעות Scapy, נוכל להסニアף ברשות ולבצע פעולות על החבילות שנקלבל. על אף Wireshark הינו כלי שימושי בעבודה שלנו עם רשותות, הוא לא מאפשר לנו דרך לבצע פעולות מורכבות. למשל, מה יקרה אם נרצה להסニアף תובורת HTTP, כפי שעשינו בפרק [שכבה האפליקציה / פרוטוקול - HTTP בקשה ותגובה](#), ולשמור לקובץ את כל הכתובות אליהן הטענה גלישה? מה אם נרצה לשמור רק את הכתובות שאליהן הטענה גלישה והכתובות עונთ על תנאי שהגדכנו מראש? מה נעשה אם נרצה לראות את כל התמונות שעברו באותה הסנפה? מה אם נרצה לשלוח בעצמנו חבילות, כאשרנו שולטים בדיוק במבנה שליהן?

את פעולות אלו ועוד נוכל לבצע באמצעות Scapy, ותאפשרו אותנו בעצמכם עד סוף פרק זה. במהלך הפרק נלך יחד, צעד אחר צעד, וביצע לראשונה חלק קטן אך מכובד מט הפעולות sh-[Scapy](#) מציע לנו. בהמשך הספר, נשתמש ב-sh-[Scapy](#) על מנת לבדוק מרוחק איזה תוכנות רצות על מחשב מרוחק, נכתב בעצמנו כדי שדומה לו-Ping שפגשנו קודם לכן, נגלה מה הדרך עברה חבילת מידע בין שתי נקודות קצרה ועוד.



פרק זה נכתב כמדריך אינטראקטיבי, ובמהלכו נתקדם בהדרגה בעבודה עם Scapy. על מנת ללמידה ממנה בדרך היילה ביותר, **פתחו את Scapy לצליכם והקישו את הפקודות ייחד עם הפרק.** וודאו כי אתם מצלחים לעקוב אחר הצעדים ומビינים את משמעותם.

שימוש לב Ci במהלך הפרק ניגע במושגים שעדין לא הסבכנו לעומק - כגון IP, Ethernet ו-TCP. אל דאגה, את הידע על מושגים אלו עמוקיק בהמשך הספר. בינתיים, ניעזר בהם בצד' להבין את השימוש ב-Scapy.



בתום פרק זה תכירו את אחד הכלים המשמעותיים ביותר לעבודה עם רשתות בכלל, ובמספר זה בפרט. ילווה אותנו בהמשך הספר כלו.

התקנה

התקנת scapy מבוצעת אוטומטית באמצעות התקנת סביבת העבודה של גבהים. אם מסיבה כלשהי תרצה להתקין בעצמכם (לא מומלץ), ראו [נספח א' - התקנת Scapy](#) בסוף פרק זה.

בואו נתחיל - הרצת Scapy

בהתחלת והתקנתם את Scapy בהצלחה, פתחו את ה-Command Line, Command Line, הכנסו למספרה scripts תחת ספריית ההתקנה של Python בצורה הבאה:

cd C:\Python26\scripts

ולאחר מכן, הריצו בשורת הפקודה את השורה הבאה:

scapy

כעת המסך אמור להיראות כך:

```

C:\Windows\system32\cmd.exe - scapy
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>cd c:\Python26\Scripts

c:\Python26\Scripts>scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: No IPv6 support in kernel
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python Crypto lib. Won't be able to decrypt WEP.
INFO: Can't import python Crypto lib. Disabled certificate manipulation tools
Welcome to Scapy (2.2.0-dev)
>>> -

```

אל תדאגו לגבי כל הערות Scapy היקר מודיע למסך כשהוא עולה. מדובר בחבילות שעדין לא התקנתם כי
הן לא נחוצות לכם, או למשל חוסר תמיכה ב-IPv6, אשר לא רלבנטית לתרגום הנמצאים בספר²⁶.
נ hereby, עכשו Scapy פועל אפשר להתחיל לעשות דברים מגניבים.

קבלת של פקודות (או – הסנפה)

ממש שם שאנו מסניפים באמצעות Wireshark, יוכל לבצע הסנפה ב-Scapy, וכך לטפל בחבילות שהגיעו אלינו
באופן תוכנתי.

שם כך, Scapy מספק לנו את הפונקציה **sniff** (מלשון הסנפה). נתחיל בהסניף שתי פקודות כלשהן, ונבקש מה-
Scapy להציג לנו אותן:

```
>>> packets = sniff(count=2)
>>> packets
<Sniffed:      TCP:2 UDP:0 ICMP:0 Other:0>
```

Scapy הציג לנו סיכום של הפקודות שהוא קיבל: שתי פקודות מסוג TCP. ניתן לבקש ממנו להציג לנו פירוט גדול
יותר באמצעות המתוודה **summary**:

```
C:\Windows\system32\cmd.exe - scapy
>>> packets = sniff(count=2)
>>> packets
<Sniffed: TCP:2 UDP:0 ICMP:0 Other:0>
>>> packets.summary()
Ether / IP / TCP 81.218.31.137:http > 192.168.14.51:54045 A / Raw
Ether / IP / TCP 192.168.14.51:54035 > 81.218.31.155:http PA / Raw
>>>
```

ניתן גם להתייחס לאובייקט **packets** בתור רשימה:

```
>>> packets[0]
>>> packets[1]
>>> len(packets)
```

²⁶ עוד על IPv6 – [בנספח ג' של פרק שכבת הרשת](#).

```
C:\Windows\system32\cmd.exe - scapy

>>> packets[0]
<Ether dst=d4:be:d9:d6:0c:2a src=00:0c:c3:a5:16:63 type=0x800 | <IP version=4L
ihl=5L tos=0x88 len=99 id=24328 flags= frag=0L ttl=47 proto=tcp checksum=0x68aa sr
c=173.194.70.189 dst=192.168.14.51 options=[] | <TCP sport=https dport=50996 seq
=4023089030L ack=2861541357L dataofs=5L reserved=0L flags=PA window=3240 checksum=
0x6c71 urgptr=0 options=[] | <Raw load='\x17\x03\x03\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x05\x80w\x15\xe4\x03\xf2WVM\xb2\x01v\x03\x1a\x85\xd9\x11\xe0\xeaHx;\xb4\x98\x0
c\xb1\x8a0\xc0-\x90\xd6\x16K\x845!\xc5#\xa\xdd\x16DS\x7f' |>>>
>>> packets[1]
<Ether dst=d4:be:d9:d6:0c:2a src=00:0c:c3:a5:16:63 type=0x800 | <IP version=4L
ihl=5L tos=0x88 len=81 id=24329 flags= frag=0L ttl=47 proto=tcp checksum=0x68bb sr
c=173.194.70.189 dst=192.168.14.51 options=[] | <TCP sport=https dport=50996 seq
=4023089089L ack=2861541357L dataofs=5L reserved=0L flags=PA window=3240 checksum=
0xa0d5 urgptr=0 options=[] | <Raw load="\x17\x03\x03\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x05\x81"\x89\xfb\xcd\x1e7\xc9\xe1``\xd5\x10\r0-\x1e7'\xf4\xcb&\xd6g\x18\x9eq\x
02\x15r" |>>>
>>> len(packets)
2
>>> _
```



תרגיל 5.1 מודרך - הסנפה של DNS

בutor התחלת, נסזה ליצור הסנפה Scapy שתציג לנו אך ורך כבויות DNS. ראשית, הריצו את Wireshark שילוחו אותו במהלך העבודה על מנת להזכיר כיצד פרוטוקול DNS נראה.
פיתחו את Command Line, והשתמשו בכלи **nslookup**, כפי שלמדו בפרק תחילת מען - איך עובד האינטרנט? ברכי למצוות הכתובת של www.google.com:

```
C:\Windows\system32\cmd.exe - nslookup
C:\Users\USER>nslookup
Default Server: box.privatebox
Address: 192.168.14.1

> www.google.com
Server: box.privatebox
Address: 192.168.14.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4005:808::1011
          173.194.113.144
          173.194.113.147
          173.194.113.145
          173.194.113.146
          173.194.113.148

> www.google.com
Server: box.privatebox
Address: 192.168.14.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4001:804::1013
          173.194.113.147
```

על מנת לסנן על חבילות DNS, ניתן להשתמש בפקודת התצוגה "nsps". מצאו את החבילה הרלוונטי ב-Wireshark. היא אמורה להיראות כך:

Frame 64: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
 Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
 Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 192.168.14.1 (192.168.14.1)
 User Datagram Protocol, Src Port: 59581 (59581), Dst Port: domain (53)
 Source port: 59581 (59581)
 Destination port: domain (53)
 Length: 40
 Checksum: 0x9dbe [validation disabled]
 Domain Name System (query)
 Response In: 65
 Transaction ID: 0x0008
 Flags: 0x0100 Standard query
 0... = Response: Message is a query
 .000 0.... = Opcode: Standard query (0)
0. = Truncated: Message is not truncated
1 = Recursion desired: Do query recursively
 0.... = Z: reserved (0)
 0.... = Non-authenticated data: Unacceptable
 Questions: 1
 Answer RRs: 0
 Authority RRs: 0
 Additional RRs: 0
 Queries
 www.google.com: type A, class IN

0000	00	0c	c3	a5	16	63	d4	be	d9	d6	0c	2a	08	00	45	00c.. ...*..E.
0010	00	3c	01	0e	00	00	80	11	9c	1e	c0	a8	0e	33	c0	a8	:<.....3..
0020	0e	01	e8	bd	00	35	00	28	9d	be	00	08	01	00	00	015.(.....
0030	00	00	00	00	00	00	03	77	77	77	06	67	6f	6f	67	6cw ww.goog
0040	65	03	63	6f	6d	00	00	01	00	01							e.com... ..

נשים לב למספר דברים:

- **בכחול** - שדה הדגמים של DNS, שמראה על שאלתא.
- **בירוק** - השאלה עצמה, מסוג A (מיופיע שם כתובות), על הדומין www.google.com

כעת נלמד כיצד להשתמש במסנן (filter) בסיסי במהלך ההסנפה. ממש כמו שהשתמשנו במסנן בצד' לסן חבילות לא רלבנטיות כאשר השתמשנו ב-Wireshark, נרצה לעשות זאת גם כשאנו משתמשים ב-Scapy. נבצע זאת באמצעות הפקודה **ifilter** של הפונקציה **sniff**.

בשלב ראשון, עלינו להגדיר את הפונקציה שבה נרצה להשתמש כדי לסתן את החבילות. הפונקציה תקבל בכל פעם חבילה אחת, ותחליט האם היא צריכה לעבור את הסינון או לא. בדוגמה זו, נבדוק האם החבילה מכילה שכבת DNS.

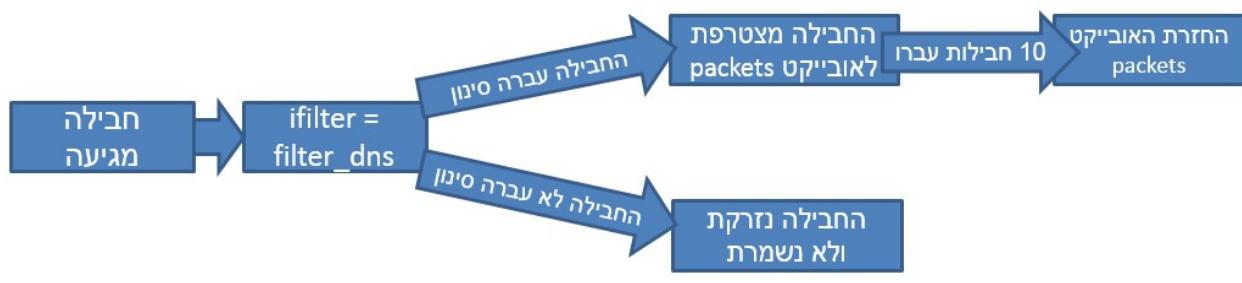
להלן הקוד הרלכנטי:

```
>>> def filter_dns(packet):
...     return DNS in packet
```

כלומר, הפונקציה תחזיר True אם קיימת שכבת DNS נחלק מהחbillה, ו-False אם לא קיימת שכבה כזו. כעת, נשתמש בפונקציה זו כפרמטר ל-**sniff**, בצורה הבאה:

```
>>> packets = sniff(count=10, lfilter=filter_dns)
```

כלומר, בשלב זה, כל חבילה שתתקבל תשלוח אל הפונקציה `filter_dns`. החבילה היא אובייקט של Scapy. אם החבילה עוברת את הסינון (True) - החבילה תשמר, ותווך לבסוף אל האובייקט `packets` כאשר יהיו 10 חבילות שעברו את הסינון. שימוש לבשה הפונקציה `sniff` היא blocking, כלומר - הפונקציה לא תסימן לרגע עד אשר ימצאו 10 חבילות שעברו את הסינון. אם החבילה לא עוברת את הסינון (`filter_dns` תחזיר False) - החבילה תיזרק:



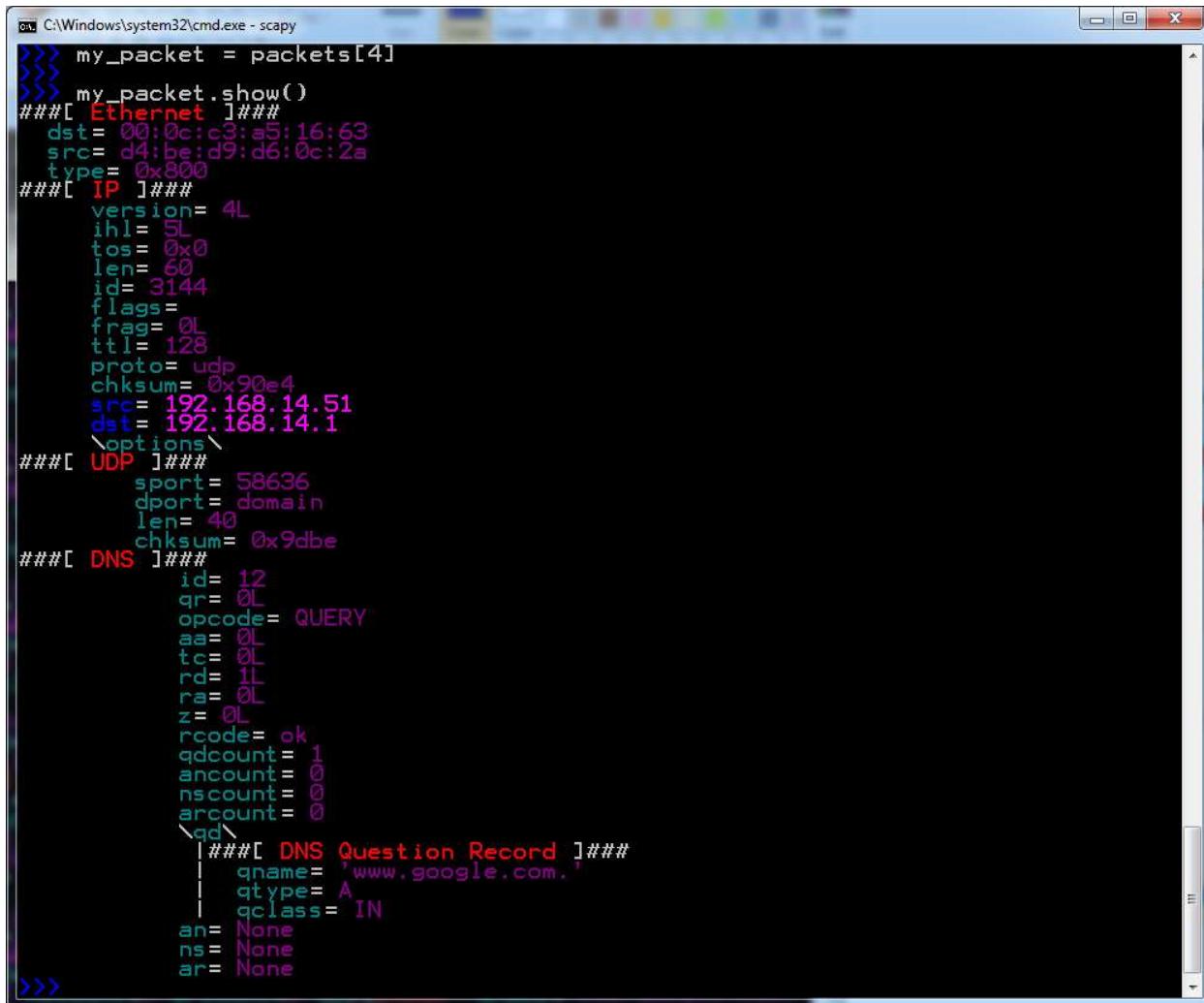
הריצו שוב את הקוד כמו שעשיתם קודם. כשהפונקציה **sniff** מסיים לרווח, צפויות להיות בידינו 10 חבילות DNS:

```

C:\Windows\system32\cmd.exe - scapy
>>> def filter_dns(packet):
...     return DNS in packet
>>> packets = sniff(count=10, lfilter=filter_dns)
>>> packets
<Sniffed: TCP:0 UDP:10 ICMP:0 Other:0>

```

כעת נשמרות כל אחת החבילות שהסנפנו, ונוכל להביט ולראות איך משתמש Scapy מסתכל על חבילה. לשם כך נשתמש במתודה **show()**:



```

C:\Windows\system32\cmd.exe - scapy
>>> my_packet = packets[4]
>>> my_packet.show()
###[ Ethernet ]###
dst= 00:0c:c3:a5:16:63
src= d4:be:d9:d6:0c:2a
type= 0x800
###[ IP ]###
version= 4L
ihl= 5L
tos= 0x0
len= 60
id= 3144
flags=
frag= 0L
ttl= 128
proto= udp
checksum= 0x90e4
src= 192.168.14.51
dst= 192.168.14.1
\options\
###[ UDP ]###
sport= 58636
dport= domain
len= 40
checksum= 0x9dbe
###[ DNS ]###
id= 12
qr= 0L
opcode= QUERY
aa= 0L
tc= 0L
rd= 1L
ra= 0L
z= 0L
rcode= ok
qdcount= 1
ancount= 0
nscount= 0
arcount= 0
\qd\
|###[ DNS Question Record ]###
| qname= www.google.com.
| qtype= A
| qclass= IN
an= None
ns= None
ar= None
>>>

```

ראו איזה יופי! Scapy מראה לנו את כל הפרטים על החבילה, במבט שדומה מאוד ל-Wireshark. אנו רואים את ה הפרדה לשכבות (שכבה הロー - Ethernet, שכבה הרשות - IP, שכבה התעבורה - UDP, שכבה האפליקציה - DNS), ואת השינויים השונים בכל שכבה.

כעת, נתמקד בשכבה DNS של החבילה. ניתן לעשות זאת באמצעות גישה לשכבה DNS בלבד, בצורה הבאה:

```

>>> my_packet[DNS]
>>> my_packet[DNS]
<DNS id=12 qr=0L opcode=QUERY aa=0L tc=0L rd=1L ra=0L z=0L rcode=ok qdcount=1 a
ncount=0 nscount=0 arcount=0 qd=<DNSQR qname='www.google.com.' qtype=A qclass=I
N> an=None ns=None ar=None >
>>> =

```

מכיוון שלא נוח להסתכל על חבילת DNS (או חלק מחבילה) בצורה זו, נוכל להשתמש שוב במתודה `show()`:

```
>>> my_packet[DNS].show()
###[ DNS ]###
  id= 12
  qr= 0L
  opcode= QUERY
  aa= 0L
  tc= 0L
  rd= 1L
  ra= 0L
  z= 0L
  rcode= ok
  qdcount= 1
  ancount= 0
  nscount= 0
  arcount= 0
  \qd\
    |###[ DNS Question Record ]###
    |  qname= 'www.google.com.'
    |  qtype= A
    |  qclass= IN
    an= None
    ns= None
    ar= None
>>> _
```

אנו רואים כאן כל שדה ושדה של שכבה ה-DNS. נוכל גם לגשת לשדה מסוים. למשל, אם נרצה לבדוק את שדה `Opcode`, ולדעת האם מדובר בשאלת DNS או בתשובה DNS, נוכל לעשות זאת כך:

```
>>> my_packet[DNS].opcode
0L
>>> _
```

השדה שווה ל-0. ניתן לראות שלמרות שכאשר ביצענו `show`, נעזרנו ב-Scapy שביצע לנו את ה"תרגום" ואמר לנו ש-0 משמעו QUERY (שאליתא), בדומה לכך שבה Wireshark "סביר" לנו את המשמעות של שדות שונים, כשהאנו ניגשים לערך בעצמו אנו מקבלים את הערך המספרי. אם נחזור ל-Wireshark, נוכל לראות שאכן 0 משמעו שאליתא.

כעת, נוכל לשפר את פונקציית ה-`filter` שכתבנו קודם לכן, ולסנן על שאלות DNS בלבד:

```
>>> def filter_dns(packet):
...     return (DNS in packet and packet[DNS].opcode == 0)
```

נביט שוב בחבילת DNS שלנו. למעשה, ניתן לראות שככנת DNS מוחלקת מבחינה Scapy לשני חלקים: החלק הכללי של DNS, והחלק שכולל את השאלתא עצמה (ומופיע בתור **DNS Question Record**). אל החלק השני ניתן לגשת שירות בצורה הבאה:

```
>>> my_packet [DNSQR]
<DNSQR qname='www.google.com.' qtype=A qclass=IN |>
>>> my_packet [DNSQR].show()
###[ DNS Question Record ]###
  qname= 'www.google.com.'
  qtype= A
  qclass= IN
>>>
```

כעת, נוכל גם לגשת לשם שעליו הtcpצעה השאלתא:

```
>>> my_packet [DNSQR].qname
'www.google.com.'
>>>
```

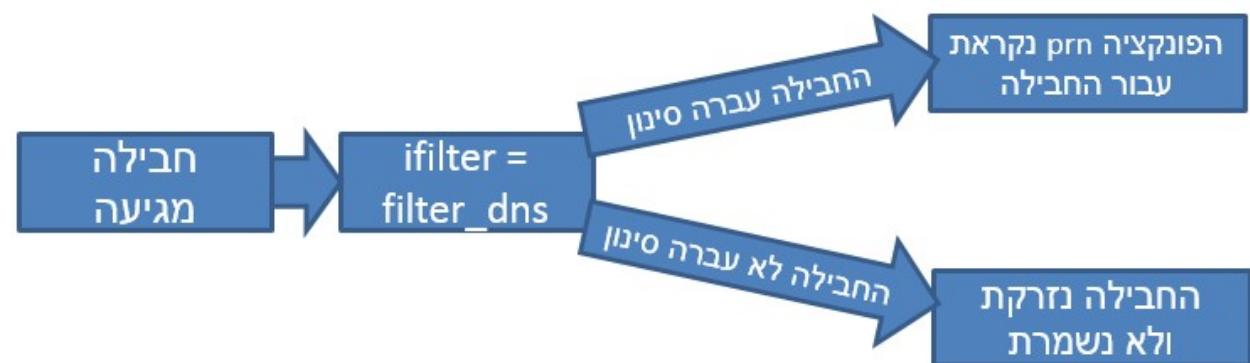
כמו כן, נרצה שהסינון שלנו יבצע רק על שאלות מסווג A. נבדוק מה הערך שנמצא בשדה `qtype` ומשמעותו שאלתת A:

```
>>> my_packet [DNSQR].qtype
1
>>> _
```

עתה נוכל להשתמש במצוזה זה בכך לספק את פונקציית `filter` שלנו:

```
>>> def filter_dns(packet):
...     return (DNS in packet and packet[DNS].opcode == 0 and packet[DNSQR].qtype == 1)
```

עכשו אנו מסננים אך ורק על שאלות DNS מסווג A. בשלב הבא, נרצה להדפיס למסך את השם שעליו הtcpצעה השאלתא. לשם כך, נשתמש בפרמטר של הפונקציה `sniff` שנקרא `prn`. פרמטר זה מקבל פונקציה שמבצעת פעולה על כל פקטה שעוברת את ה-`filter`. כמובן, כל חבילה שהצליחה לעבור את הסינון שהתבצע קודם לכך באמצעות `ifilter` תשלח אל הפונקציה שניתנה ל-`prn`. מכאן שרשרת הפעולות שלנו תראה כך:



ראשית, עלינו להגיד את הפונקציה שתרצה. ברכינו להציג את שם הדומין שעליו התבצעה השאלה. לשם כך, נגידר את הפונקציה بصورة הבאה:

```
>>> def print_query_name(dns_packet):
...     print dns_packet[DNSQR].qname
```

לפני ביצוע ההסנהה, נאפס את המטמון של DNS באמצעות הפקודה ipconfig /flushdns

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>ipconfig /flushdns

Windows IP Configuration

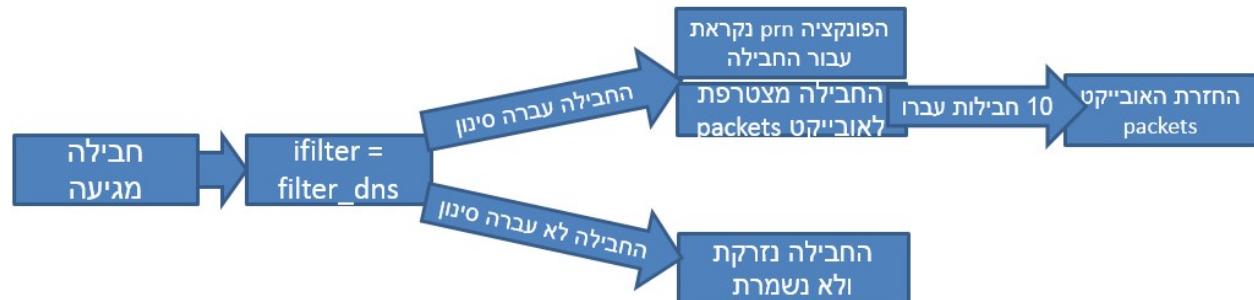
Successfully flushed the DNS Resolver Cache.

C:\Users\USER>
```

כעת, נוכל לקרוא לפונקציה **sniffos**, כאשר אנו מסננים על שאלות DNS מסוג A בלבד באמצעות **ifilter**, ומדפיסים את שמות הדומיינים שעלייהם הتبיעה השאלה באמצעות **grep**:

```
>>> sniff(count=10, lfilter=filter_dns, prn=print_query_name)
```

זיהי שרשרת הפעולות המלאה:



בגוי לגלוש בזגדפו ולראות אילו תוצאות אתם מתקבלים. אתם אמורים לראות פלט הדומה לפלט הבא:

```
C:\Windows\system32\cmd.exe - scapy
>>> def print_query_name(dns_packet):
...     print dns_packet[DNSQR].qname
>>> def filter_dns(packet):
...     return (DNS in packet and packet[DNS].opcode == 0 and packet[DNSQR].qtype ==
... = 1)
>>> sniff(count=10, lfilter=filter_dns, prn=print_query_name)
www.google.co.il.
www.google.co.il.
www.google.com.
www.google.com.
apis.google.com.
lh4.googleusercontent.com.
plus.google.com.
apis.google.com.
lh4.googleusercontent.com.
plus.google.com.
<Sniffed: TCP:0 UDP:10 ICMP:0 Other:0>
>>> _
```

באם לא נבצע פעולה זו, מערכת הפעלה עשויה "לזכור" את התשובות לשאלות קודמות ששאלנו, ולא לשאול עליהן. כך למשל, אם נגלוש ל-www.google.com, מערכת הפעלה עשויה "לזכור" את כתובת ה-IP שנמצאה עבורו קודם לכן, ולתת אותה כתשובה מוביל ב암ת לשאול את שרת ה-DNS.

מכיוון שהפונקציה שנותנת ל-`h2k` היא קוד פיתרון לכל דבר, נוכל לבצע כל פעולה שנרצה. וכך, למשל, לשמר את הדומינום הלו לקובץ. שימוש לב כמו קול לעשות זאת באמצעות Scapy.

תרגיל 5.2 מודרך - הסנפה של HTTP

פרק שכבת האפליקציה למדנו על פרוטוקול HTTP. לצורך הפעלה של Scapy, לא מכיר את פרוטוקול HTTP באופן מובנה, והוא מתיחס אליו פשוט כאל מידע (כלומר - מחזורת של תווים). אם נסניף חבילת HTTP, היא תיראה כך:

למעשה, כל שכבת ה-HTTP שלנו הובנה על ידי Scapy בתור מידע Raw, והוא מוצג כרכץ מיידי ותו לא.

ברצוננו להציג את כל בקשות ה-HTTP שפונות לעמוד מסויים. ככלומר בקשות מסוג GET.



כיצד נוכל לسان חבילות לא מזוהות?

העבירורה שלה היא TCP²⁸ והמידע שהיינו אנו חbijת GET. הינה אכן חbijת GET.

כדי לבדוק האם המידע מתחילה ב-'GET', עלינו ראשית להפוך אותו למחוזצת:

```
>>> data_string = str(my_packet[Raw])
```

cut נוכל לבדוק האם המידע מתחילה במחוזת הידועה הראשית:

```
>>> data_string.startswith('GET')
```

בשלב זה, נוכל לכתוב פונקציית filter באמצעותה נסנן חבילות כאלו בלבד:

```
>>> def http_get_filter(packet):
```

```
...     return (TCP in packet and Raw in packet and str(packet[Raw]).startswith('GET'))
```

תרגיל 5.3 - הסנפה



השתמשו בMSN שהגדכנו קודם לכן, והדפiso באופן מסודר את כל ה-URLים אליו ממבצע בקשה GET במהלך ריצת הסкриיפט. שימו לב לכתוב URL מלא, כולל גם את ה-Host. לדוגמה, במקרה my_packet שהציגנו קודם, תודפס למסך כתובת הבאה:

www.ynet.co.il/home/0.7340.L-8.00.html

²⁸ על פרטיזן זה, והוא בפה שמתהמשים בו כפרטיזן שכבת המערבה של HTTPS. גלמוד בפרק שכבת המערבה.

הסניף - סיכום

ובכן, למדנו להסニアב באמצעות Scapy. הצלחנו להסニアב חבילות DNS ו宦別 HTTP, בינו מסקן באמצעות **ifilter** והצלחנו לבצע פעולות על החבילות באמצעות **chk**. ראיינו גם איך נראות חבילות ב-**Scapy** ולמדנו קצת איך לעבוד איתן. עם זאת, הכרנו חלק קטן מאוד מהיכולות של הפונקציה **sniff**. נלמד יכולות נוספות בהמשך, אך קודם נעשה זאת - נכיר פעולות נוספות. הרו להסニアב חבילות זה טוב ויפה, אך בהחלט לא מספיק.

יצירת חבילות

לעתים נרצה פשוט ליצור חבילה מסוימת, אולי להסתמך על חבילה קיימת שהסニアב מהרשות. נתחיל ביצירת פקטה של ²⁹IP. הקישו את הפקודה הבאה:

```
>>> my_packet = IP()
```

לא קרה הרבה. בואו ננסה לראות את הפקטה. הקישו את הפקודה הבאה:

```
>>> my_packet
<IP | >
```

לא הודפס פלט רב למסך. אם הדבר אומר שנוצרה לנו פקטה ללא אף שדה?

בואו ננסה לראות את כל הפרטים על הפקטה, באמצעות המתודה **()show**. עשו זאת כך:

```
>>> my_packet.show()
```

```
C:\Windows\system32\cmd.exe - scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: No IPv6 support in kernel
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python Crypto lib. Won't be able to decrypt WEP.
INFO: Can't import python Crypto lib. Disabled certificate manipulation tools
Welcome to Scapy (2.2.0-dev)
>>> my_packet = IP()
>>> my_packet
<IP |>
>>> my_packet.show()
###[ IP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= ip
cksum= None
src= 127.0.0.1
dst= 127.0.0.1
\options\
```

אייה יופי! אנחנו רואים את כל השדות. Scapy בנה אותם עבורנו, וגם נתן להם ערכים היגיוניים (למשל הגירה היא 4, שכן מדובר ב-IPv4). ttl הוא 64, וכך הלאה³⁰). מכאן ש-**Scapy** יודע ליצור עבורנו חבילות עם ערכים היגיוניים, ואנחנו יכולים לדאוג אך ורק לערכים המעניינים אותנו.

²⁹ את פרוטוקול IP פגשנו כבר בפרקיהם הקודמים, ונרחיב את ההכרות איתו בפרק שכבת הרשת.

³⁰ על משמעות מושגים אלו (IPv4 ו-ttl) - נלמד בפרק של שכבת הרשת.

האם יש צורך במתודה `show` בכל פעם שברצוננו לדעת את ערכו של שדה ייחיד? ממש לא! לדוגמה, לכל חבילה של IP יש כתובות מקור המציגות מי שלח את החבילה, וכתובות היעד - המציגות למי החבילה מיועדת. למשל, חבילה שנשלחה מהמחשב שלנו אל השירות של Google, תכלול בשדה כתובות המקור את כתובות ה-IP של המחשב שלנו, ובשדה כתובות היעד את כתובות ה-IP של השירות של Google.

בואו ננסה לגלוות רק מה ערך כתובות המקור (ה-Source Address) של הפקטה:

```
>>> my_packet.src  
'127.0.0.1'
```

פשוט וקל.

כעת ננסה לשנות את אחד השדות. נאמר ונרצה שכתובות היעד (Destination Address) של הפקטה תהיה '10.1.1.1'. נוכל לעשות זאת כך:

```
>>> my_packet.dst = '10.1.1.1'
```

כעת אם נסתכל שוב בפקטה, נראה ש-Scapy מצין בפנינו רק את הפרמטר ששיםינו:

```
>>> my_packet.dst='10.1.1.1'  
>>> my_packet  
<IP dst=10.1.1.1 |>
```

מכאן אנו למדים ש-Scapy מציג לנו רק את הפרמטרים השונים, המעניינים. קודם לכן, כניסינו להציג את `my_packet`, הוא בחר שלא להראות לנו את שדה כתובות היעד, שכן כתובות היעד לא השתנתה מערך ביריתת המוחלט שלו.

כਮון ששיםינו כתובות היעד יבוא לידי ביטוי גם כ`show_packet.show_my`. בצעו זאת בעצמכם כעת. אם ננסה לשנות יותר משדה אחד, תקלה תופעה דומה:

```
>>> my_packet.ttl = 5  
>>> my_packet
```

כאן שניסנו את הערך של השדה `ttl`, וכעת scapy הראה גם את כתובות היעד כסוגה, וגם את שדה ה-`ttl`:

```
>>> my_packet.dst='10.1.1.1'  
>>> my_packet  
<IP dst=10.1.1.1 |>  
>>> my_packet.ttl = 5  
>>> my_packet  
<IP ttl=5 dst=10.1.1.1 |>
```

לחילופין, ניתן גם ליצור כך את הפקטה מראש, בצורה הבאה:

```
>>> my_packet = IP(dst = '10.1.1.2', ttl = 6)
```

```
>>> my_packet
>>> my_packet = IP(dst = '10.1.1.2', ttl = 6)
<IP ttl=6 dst=10.1.1.2 |>
```

שכבות

עד כה הצלחנו לבנות פקטה עם שכבה אחת יחידה (IP). כמו שוזדיי הבנתם מהפרקים הקודמים בספר, פקטות IP שלעצמן לא מאד מעניינות. הן בדרך כלל מגיעות עם שכבה נוספת מעליה. על מנת להוסיף שכבות נוספות מעל השכבה IP, נבצע לדוגמה את הפעולה הבאה:

```
>>> my_packet = IP() / TCP()
```

השתמשנו באופרטור / על מנת "להעניק" שכבה אחת מעל שכבה אחרת. כאן, יוצרים פקטה עם שכבת IP ומעליה שכבה של TCP³¹. בואו נראה כיצד Scapy מציג את הפקטה:

```
>>> my_packet
my_packet
<IP frag=0 proto=tcp |<TCP |>>
```

(הערה: הסוגרים המשולשים נקבעו והודגשו על ידי כותב הספר ולא על ידי Scapy)

שיםו לב לכך הייצוג כאן. שכבת TCP תחומה בידי הסוגרים המשולשים האדומים, בעוד שכבת IP תחומה בסוגרים המשולשים הכתולים. ניתן לראות כי שכבת TCP מוכלת בשכבת IP!! היזכרו במושג ה- Encapsulation (או כינויו) עליו דיברנו בפרק מודל השכבות.

IP הוא פרוטוקול שכבה שלישית ו-TCP הוא פרוטוקול שכבה רביעית. בואו ננסה עתה ליצור גם שכבה שנייה, ולשם כך נשתמש בפרוטוקול ה-Ethernet³²:

```
>>> my_packet = Ether() / IP() / TCP()
my_packet
<Ether type=0x800 | <IP frag=0 proto=tcp |<TCP |>>>
```

cut הפקטה my_packet הינה פקטה IP וכן שכבת TCP. נוכל גם לשנות את הפרמטרים של השכבות השונות בזמן ייצור הפקטה:

³¹ על פרוטוקול זה נלמד לעומק בפרק [שכבת התעבורת](#).

³² על פרוטוקול זה נלמד בהרחבה בפרק [שכבת הקו](#).

```
>>> my_packet = Ether() / IP ttl=4 / TCP dport=80
>>> my_packet
<Ether type=0x800 |<IP frag=0 ttl=4 proto=tcp |<TCP dport=http>>>
```

שימוש לבכמה דברים:

1. בשכבה ה-Ethernet, מוצג לנו ה-type (שווה ל-0x800) זאת מכיוון שה-type מצביע על כך שהשכבה הבאה היא אכן שכבה IP.
2. בשכבה ה-IP, עצת מוצג גם ה-ttl. זאת מכיוון שהוא בו ערך לא ברירת מחדל בשורה הקודמת.
3. בשכבה ה-TCP, צינו שפורט היעד (port) או בקיצור (dport) יהיה 80. Scapy יודע לבצע את התרגום ולהציג אותו כפורט הייעודי של HTTP³³ - ממש כמו ש-Wireshark יודע לעשות.

דבר נוסף שאפשר לבצע ב-Scapy הוא להוסיף מידע "Raw", שיישמש אותנו כ-payload לשכבה הנוכחית. Payload של שכבה הוא המידע ש"מעליה". כך למשל, ה-payload של שכבה IP יכול להיות שכבה ה-TCP וכל המידע שלה, והוא-payload של TCP עשוי להיות, למשל, HTTP. ראיינו שניתן להוסיף מידע "Raw" באמצעות Scapy כאשר הסופנו חבילת HTTP קודם לכך. על אף Scapy, כאמור, לא מכיר את פרוטוקול HTTP, יוכל לייצר חבילה HTTP בצורה הבאה:

```
>>> Ether()/IP()/TCP()/Raw("GET / HTTP/1.0\r\n\r\n")
```

ראו מה מתרחש:

```
>>> my_packet = Ether() / IP() / TCP() / Raw("GET / HTTP/1.0\r\n\r\n")
>>> my_packet
<Ether type=0x800 |<IP frag=0 proto=tcp |<TCP |<Raw load='GET / HTTP/1.0\r\n\r\n'>>>
```

ונכל לבצע זאת גם מבלי לכתוב Raw, אלא לכתוב את השורה בצורה פשוטה:

```
>>> Ether()/IP()/TCP()/"GET / HTTP/1.0\r\n\r\n"
```

ניתן גם לבדוק ייזוג Hexdump (הצגת המידע בפורמט הקסדיימלי) של הפקטה, כך שהיא תיראה בדומה לדרכו Wireshark: בה היא מוצגת ב-

```
my_packet = Ether() / IP() / TCP() / Raw("Cyber rulez")
hexdump(my_packet)
0000  FF FF FF FF FF FF 00 00 00 00 00 00 00 00 45 00 :3.....@.i.....E.
0010  00 33 00 01 00 00 40 06 7C C2 7F 00 00 01 7F 00 :....P. ....P.
0020  00 01 00 14 00 50 00 00 00 00 00 00 00 00 50 02 :....P. ....P.
0030  20 00 20 97 00 00 43 79 62 65 72 20 72 75 6C 65 z ...Cyber rule
0040  7A
```

³³ מדובר בנושא ה포רטים בפרק שכבת התעבורה.

Resolving

בפרקים הקודמים הזכרנו את התרגום של שמות דומיין (כגון "www.google.com") לכתובות IP (כגון "172.15.23.49"). תהיליך זה נקרא Resolving, ו-Scapy יבצע אותו בשביבתו. לכן, אם נכתבת את השורה הבאה:

```
>>> my_packet = IP(dst = "www.google.com")
```

Scapy יציב בשדה כתובת הריעד של ה-IP את כתובת ה-IP של Google. נסו זאת בעצמכם!

שיםו לב שכאשר תסתכלו על החבילה שיצרתם (באמצעות המתוודה **show** למשל), תראו כי Scapy לא רושם את כתובת ה-IP אלא את הדומיין שציינתם. עם זאת, כאשר נשלח את החבילה (כמו שנלמד לעשות בהמשך הפרק), Scapy יdag להבין את כתובת ה-IP הרלבנטית ולהציב אותה בשדה הרלבנטי.

שליחת פקודות

עד כהן למדנו ליצור בעצמנו פקודות בשכבות שונות, וכן להציג אותן על המסך. כעת נלך צעד אחד קידימה, ונלמד גם **לשלוח** את הפקודות שיצרנו.

Scapy מציע, בגדול, שתי דרכים לשולוח פקודות: שליחה בשכבה שלישית ושליחה בשכבה שנייה. בשלב זה נתעלם מהאופציה לבצע שליחה ברמה שנייה, אך נחזור אליה בהמשך הספר.

ניצור חבילה שמתחליה בשכבה שלישית, למשל חבילת IP ומעלה מידע טקסטואלי בסיסי (זהו לא חבילה תקינה, ו-Google לא באמת צפוי להגיב אליה):

```
>>> my_packet = IP(dst = "www.google.com") / Raw("Hello")
```

פתחו את Wireshark והריצו הנספה.

כעת נוכל לשולוח את החבילה:

```
>>> send(my_packet)
```

Sent 1 packets.

נסו לזרוח את החבילה בהסנהה שלכם ב-Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	Bewan_a5:16:63	Spanning-tree-(for-STP)	60 Conf.	Root = 32768/0/0:0:c:c3:a5:16:63 Cost = 0 Port = 0x8001	
658	1.58358300	192.168.14.1	224.0.0.251	IGMPV2	60	Membership Query, specific for group 224.0.0.251
853	1.63714700	192.168.14.51	224.0.0.251	IGMPV2	46	Membership Report group 224.0.0.251
1552	1.83040000	192.168.14.51	173.194.70.99	IPv4	39	IPv6 hop-by-hop option (0)
Frame 1552: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface 0						
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)						
Internet Protocol version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 173.194.70.99 (173.194.70.99)						
Data (5 bytes)						
Data: 48656c6c6f [Length: 5]						
0000	00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00C..*..E.			
0010	00 19 00 01 00 00 40 00 b7 e3 c0 a8 0e 33 ad c2@.3..			
0020	46 63 48 65 6c 6c 6f	FcHello				

שימוש ב-Scapy מתוך קובץ סקריפט

כאשר נרצה לכתוב תכנית / סקריפט שמבצע פעולות באופן קבוע ואוטומטי (כלומר - קובץ בעל הסיומת `uk`, ולא עבודה מתוך `the-Interpreter`), ניתן להשתמש ב-Scapy לצרכים אלו באופן דומה לשימוש בחבילות אחרות ב-`python`. לשם כך, יש לבצע `import` בצורה הבאה:

```
from scapy.all import *
```

כעת ניתן להשתמש באובייקטים של Scapy.

לדוגמה, נשתמש בקוד שכתבנו מוקדם יותר בפרק בCDI להציג את כל הדומיינים שעבורם מתבצעת שאליתת DNS:

```
from scapy.all import *
def print_query_name(dns_packet):
    """This function prints the domain name from a DNS query"""
    print dns_packet[DNSQR].qname
def filter_dns(packet):
    """This function filters query DNS packets"""
    return (DNS in packet and packet[DNS].opcode == 0 and packet[DNSQR].qtype == 1)
print 'Starting to sniff!'
sniff(count=10, lfilter=filter_dns, prn=print_query_name)
```

היכולת ליצור ולשלוח חבילות תשמש אותנו רבות בהמשך הספר.

תרגילי 5.4 -Scapy



1. צרו חבילת IP באמצעות Scapy, ודאו שכתובת היעד של תריה השרת של "www.google.com". השתמשו במתודה **show**, וידאו שאכן החבילה שיצרתם מיועדת אל Google.
2. שלחו את החבילה שיצרתם בסעיף הקודם. הסניפו באמצעות Wireshark, ובדקו שאתם מצלחים לראות את החבילה, ושהיא אכן נשלחה אל הכתובת של Google.
3. פתחו את הדףן שלכם, וגילשו אל Facebook. הסניפו באמצעות Scapy את החבילות שנשלחות בין המחשב שלכם לבין Facebook, והדפיסו סיכום שלهن באמצעות המתודה **summary**. שימו לב לסנן רק חבילות שנשלחות ביןיכם לבין השרת של Facebook.

Scapy - סיכום

בפרק זה למדנו **קצת** על Scapy ואיך להשתמש בו. במהלך הפרק רأינו כיצד מתקנים את Scapy, ולאחר מכן למדנו לבנות באמצעותו חבילות, לשולח אותן ולהסניף חבילות המתקבלות בראשת. בתקווה, הצלחתם להבין את דרך העבודה עם הכל' הנהדר הזה, וגם **קצת** לספוג את ההתלהבות מהשימוש בו והכוח הרב שהוא נותן לכם ול老子ך דרך אצבעותיכם (בצירוף המקלדת, מן הסתם).

כפי שנכתב בתחילת הפרק – Scapy הוא כל'. בדומה לקריאה וכתיבה, הוא לא נותן המון כשלעצמו. אך שמחברים אותו עם דברים נוספים (כמו הידע הנרכש שלכם בראשות), בהחלט אפשר להגיע רוחק. Scapy יובילו אותנו לאורך הספר כולו. יעוזר בו בצד' לתרגם נושאים קיימים, וכן געמיך את היכרותנו עימנו.

עמודים להמשך - Scapy

אלו מכם שמעוניינים להעמיק את הידע שלהם ב-Scapy, מוזמנים לבצע את הצעדים הבאים:

קריאה נוספת

ניתן ומומלץ לקרוא עוד רבות על Scapy באתר הפורוייקט: <http://www.secdev.org/projects/scapy/doc> כמו כן, ניתן להיעזר במסמך Scapy Cheat Sheet שנמצא כתובות <http://goo.gl/tUy6Aq>.

תרגיל מתקדם

בתרגיל זה תמשכו שרת DNS באמצעות Scapy. בצעו את התרגיל בשלבים. בכל שלב, בדקו את השרת שלכם לפני המשיכו לשלב הבא. תוכלו להיעזר בכליה nslookup אותו פגשנו בפרק תחילת מסע - איך עובד האינטרנט/DNS, על מנת לבדוק את השרת שלכם. שימו לב שעל מנת לבדוק את תרגיל זה, עליהם להשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת³⁴.

שלב ראשון - שרת עצמאי

- על השרת להאזין לשאילתות DNS נכנסות בפורט 53, ולענות עליהן.
- השרת צריך לתמוך רק בסוגי הרשומות A ו-PTR³⁵.
- על השרת לשומר קובץ TXT שיכיל את מסד הנתונים שלו. בכל רשומה יהיה רשום סוג הרשמה, הערךים וה-TTL.
- כאשר לקוח פונה לשרת בבקשת DNS, אם הוא פונה עבור רשומה שקיימת במסד הנתונים של השרת, על השרת לענות לו תשובה תקינה בהתאם למסד הנתונים.
- אם הלקוח פנה בבקשת רשומה שלא קיימת במסד הנתונים של השרת, על השרת להגיב בתשובה DNS עם השגיאה "no such name" (באמצעות שדה dns-flags ב-Scapy).

³⁴ באופן תאורטי, יכולנו לעשות זאת מעל loopback device – כלומר מעל הכתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של Scapy בשילוח וקבלת מסגרות מעל loopback device ב-Windows, השתמש בשני מחשבים.

³⁵ Scapy יש Bug ידוע עם חבילות PTR. כדי להתגבר עליו, תוכלו לגשת לקובץ "py.layers" בתיקייה "layers", ולהחליף את השורה:

```
elif pkt.type in [2,3,4,5]: # NS, MD, MF, CNAME
elif pkt.type in [2,3,4,5,12]: # NS, MD, MF, CNAME, PTR
```

בשורה הבאה:

שלב שני - שרת חברותי

בשלב הקודם, השרת שלכם פועל לבדו. אם הוא לא הכיר שם דומיין מסוים - הוא לא הצליח לתת שירות טוב ללקוח. כדי如此, שירותי ייחודיים לפועל בצורה חברותית מסוגלים לתת שירות טוב יותר. שפרו את השרת שתכתבם בשלב הקודם:

- במידה שהשרת לא מכיר שם דומיין שעליו הוא נשאל, הוא ישלח את השאלה לשרת DNS אחר.
- במידה שהשרת DNS אחר ידע לענות על השאלה, הוא יחזיר את התשובה התקינה של שרת DNS אל לקוחות.
- במידה שהשרת DNS אחר לא ידע לענות על השאלה, השרת שלכם יחזיר הודעה שגיאה "such name".

שלב שלישי - שרת עם מטמון

הוסף לשרת שלכם יכולות מטמון (Caching).

- במידה שהשרת נשאל על שם שהוא לא הכיר, והשיג את פרטי השם זהה משרת אחר, הוא יוסיף את המידע שהוא גילה אל מסד הנתונים שלו.
- בפעם הבאה שהשרת ישאל על השם זהה, הוא יוכל לענות בעצמו ולא יזדקק לשרת נוסף.

נספח א' - התקנת Scapy

על אף ש-Scapy היא חבילת מעליה, תהליך ההתקנה שלה אינו טריוני-אל.



תוכלו להיעזר בסרטון המדגים כיצד להתקין את Scapy על מערכת הפעלה Windows 7, 64bit. הסרטון

זמן בכתובות: <http://data.cyber.org.il/networks/videos/install-scapy-windows7-64bit.html>



שיםו לב לעקב אחר ההוראות באופן מדויק:

1. אם Python עדין לא מותקן לכם – התקינו אותו. שימו לב שמדובר בגרסה 2.6. תוכלו להוריד אותה

מהכתובת: <http://goo.gl/2LIOYq>

2. אם מותקנת לכם יותר מגרסה אחת של Python – עדיף שתՏוירו את הגירסאות האחרות ותשאירו רק את גרסא 2.6. אחרת, שימו לב שככל ההתקנות הבאות מתיחסות לגרסה 2.6.

3. התקינו את הגרסה האחרונה של Scapy מתוך פרויקט:

<http://www.secdev.org/projects/scapy/files/scapy-latest.zip>

א. הורידו את הגרסה העדכנית ביותר ושמרו אותה אל המחשב.

ב. ייצאו את הקבצים המכוחזים.

ג.פתחו את ה-Command Line, הכנסו לתיקית ההתקנה של scapy (לדוגמא:

cd C:\Downloads\Scapy

ד. והריצו: "python setup.py install", כמoven שבלי המרכאות.

4. התקינו את win32api, מהכתובת: <http://goo.gl/PSNdbf>.

5. וידאו כי Wireshark מותקן אצלכם. הוא מכיל גם את ה-driver הנחוץ לשם הרצת Scapy (driver Winpcap).

6. התקינו את Pyreadline, אותו ניתן למצוא בכתובות: <http://goo.gl/COFL9o>

7. אם ברשותכם מערכת הפעלה מסווג Windows 7 64bit, בצעו את השלבים הבאים:

א. הורידו את הקובץ <http://data.cyber.org.il/networks/c05/libs.zip>

ב. לפתוח אותו והעתיקו את שני הקבצים שבו (pcap.pyd, dnet.pyd) אל התקינה C:\Python26\DLLs

8. אחרת, בצעו את השלבים הבאים:

- א. התקינו את Pypcap מהכתובת: <http://goo.gl/6SKkOR>. שימו לב כי זהה גרסה מיוחדת עבור Windows 7 / Vista. הערה: תחת Scapy, לחצו על כפתור ימני בעט התוכנה ובחירה "Run as administrator".
- ב. התקינו את libdnet, מהכתובת: <http://goo.gl/8RyR95>. הערה: תחת Run as Windows 7 / Vista, לחזו על כפתור ימני בעט התוכנה ובחירה "administrator".

שים לב שמדריך ההתקנה הרשמי נמצא באתר הפרויקט, בכתב:

<http://www.secdev.org/projects/scapy/doc/installation.html>, והוא המדריך המעודכן ביותר שניתן למצוא. בכל מקרה של בעיה, מומלץ לפנות אליו.

פרק 6 - שכבת התעבורה

עד כה התעסקנו באפליקציות. הצלחנו לשלוח מידע מתוכנה שנמצאה במחשב אחד לתוכנה במחשב אחר. אבל איך כל זה קרה? איך עובד הקסם הזה של העברת מידע בין מחשב אחד למחשב אחר? בפרק זה נתחיל להפיג את הקסם, ולהסביר לעומק איך הדברים עובדים.

במהלך הפרק הקרוב נלמד מהם פורטים (ports), נכיר כלים ומושגים חדשים ונלמד על ה프וטוקולים UDP ו-TCP. נכתוב תוכנה להעברת מידע סודי, ונצליח לגלוות אילו שירותים פתוחים במחשב מרוחק. על מנת לעשות זאת, علينا להבין את שכבת התעבורה.

מה תפקידיה של שכבת התעבורה?



שכבת התעבורה אחראית להעביר מידע מתכנית (תהליך) לתוכנית (תהליך) מרוחקת. חלק מכך, יש לה שתי מטרות עיקריות:

- ריבוב מספר אפליקציות עבור אותה הישות - כלומר היכולת לתקשר עם ישות רשות אחת (אל מול אותה כתובת IP בודדת) ולהשתמש בכמה שירותים שונים של הישות, כך שהישות תדע להבדיל איזה זרם מידע שיר לאיזה שירות שהוא מספקת. מטרה זו **קיימת תמיד** בשכבת התעבורה.
- העברת אמינה של מידע. זהה מטרה אופציונלית, ומכאן שהוא לא **קיימת בכל המימושים** של שכבת התעבורה (כלומר, לא בכל הפרוטוקולים של שכבת התעבורה).

ריבוב אפליקציות - פורטים

נאמר ויש לנו חיבור בין שרת ללקוח. עתה, הלקוח שולח בקשה אימייל לשרת מעל חיבור זה:



הדבראגוני בהנחה והשרות מרים שירות של אימייל. עם זאת, יתכן שהלקוח ישלח יותר מבקשה אחת לשרת. למשל, יתכן והשרות מרים גם שירות של שירות אימייל וגם שירות של שירות Web (למשל - HTTP עליי למדנו בפרק שכבת האפליקציה). מה יקרה אם הלוקוח ישלח אל השירות גם בקשה של אימייל, וגם בקשה HTTP?



כעת, על השירות להבין לאיזה שירות שלו נשלחה הבקשה. במקרה זה, תהיה אצל השירות תוכנה שתטפל בבקשת מילוקחות הקשורות באימייל, ותוכנה שתטפל בבקשתות HTTP. על השירות להצליח להפריד ביניהן, כדי להפנות את הבקשה לתוכנה המתאימה:



לשם כך, יש לנו צורך בזיכרון התוכנה. לא מספיק שהלקוח יודע לפנות אל השירות (להזכירם, מזהה השירות באינטרנט הוא כתובת IP), הוא צריך גם לספק מזהה של התוכנה הספציפית אליה הוא פונה. בפרק [תכנות ב-Socket](#), דמיינו זאת לשילוח מכתב דואר בין שתי משפחות הגרות בשכונה של בתים רבים. ציינו כי מזהה הרכיב (במקרה זה - השירות) הינו מזהה הבניין של המשפחה - למשל "רחוב הרצל בתל אביב, בית מספר 1". מזהה התוכנה (במקרה זה - תוכנת האימייל או תוכנת ה-HTTP) הוא מזהה הדירה הספציפית בבניין, למשל "דירה 23".



**מזהה הבניין:
הרצל 1, תל אביב**

בעולם הרשת, מזהה הבניין הוא כתובת IP, ומזהה הדירה נקרא **פורט (Port)**. באמצעות פניה לפורט מסויים בבקשתה, השירות יכול לדעת לאיזו תוכנה אנו פונים. כך לדוגמה, אם נשלח הודעה לפורט מס' 80 (באמצעות פרוטוקול TCP, עליו מדובר בהמשך הפרק), השירות צפוי להבין שאנו פונים לתוכנת ה-HTTP ולא לתוכנה המail, מכיוון שתוכנת ה-HTTP **מzdינה** על פорт 80:



תרגיל 6.1 מודרך - אילו פורטים פתוחים במחשב שלי?



המנוח "פורט פתוח" מתייחס לפורט שתוכנה כלשהי משתמש עליו. למשל, אם יפנו אל הפורט זהה, תהיה תוכנה שמכונה לקבל חיבור. באם יש לנו שרת שמריץ תוכנת HTTP שמאזינה על פорт 80, ואין תוכנות נוספות שמאזינות על פורטים נוספים, אז פорт 80 יקרא "פתוח" בעוד פорт 81 למשל יקרא "סגור".

כעת נלמד כיצד לגלות אילו פורטים פתוחים במחשב שלנו. לשם כך, פיתחו את ה-*Command Line* והריצו את הפקודה **netstat**:

Active Connections			
Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:5354	USER-PC:49160	ESTABLISHED
TCP	127.0.0.1:5354	USER-PC:49161	ESTABLISHED
TCP	127.0.0.1:27015	USER-PC:49200	ESTABLISHED
TCP	127.0.0.1:49160	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49161	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49200	USER-PC:27015	ESTABLISHED
TCP	127.0.0.1:57236	USER-PC:57242	ESTABLISHED
TCP	127.0.0.1:57240	USER-PC:57241	ESTABLISHED
TCP	127.0.0.1:57241	USER-PC:57240	ESTABLISHED
TCP	127.0.0.1:57242	USER-PC:57236	ESTABLISHED
TCP	127.0.0.1:57247	USER-PC:57248	ESTABLISHED
TCP	127.0.0.1:57248	USER-PC:57247	ESTABLISHED
TCP	192.168.14.51:51817	192.168.14.153:microsoft-ds	ESTABLISHED
TCP	192.168.14.51:54209	wb-in-f125:5222	ESTABLISHED
TCP	192.168.14.51:61183	fa-in-f189:https	ESTABLISHED
TCP	192.168.14.51:61437	fa-in-f120:https	ESTABLISHED
TCP	192.168.14.51:61457	bzq-179-154-217:https	ESTABLISHED
TCP	192.168.14.51:61459	bzq-179-17-162:https	ESTABLISHED
TCP	192.168.14.51:61462	173.194.116.181:https	ESTABLISHED
TCP	192.168.14.51:61479	bzq-179-154-251:https	TIME_WAIT

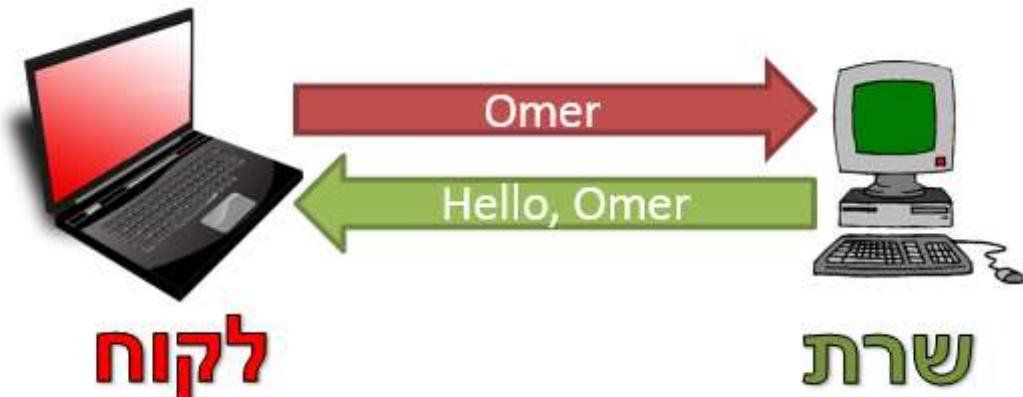
בואו נבחן את הפלט של הפקודה **netstat**:

- בAddon** - אנו רואים את ה프וטוקול שעליו המחשב מבצע האזנה. בשכבה התעבורת ישנו שני פרוטוקולים נפוצים עליהם נלמד בהמשך הפרק, והם TCP ו-UDP.
- בירוק** - הכתובת המקומית עליה המחשב מאזין. הכתובת כתובה בפורמט של "IP:Port" ("IP:Port"). כך לדוגמה בשורה הראשונה, כתובת ה-IP הינה 127.0.0.1, והפורט הינו 5354. התעלמו מכתובות ה-IP בשלב זה, נלמד להכיר אותן בהמשך הספר.
- בכחול** - הכתובת הרחוקה אליו המחשב מחובר. במידה שנחננו לא רק מחכים לחיבור, אלא חיבור כבר קיים, **netstat** יודיע להציג גם את הכתובת המרוחקת של החיבור. כך למשל, החיבור הראשון הינו מפורט 5354 במחשב שלנו, אל פорт 49160 במחשב בשם USER-PC (שהוא למעשה המחשב ממנו רצה הפקודה, מכיוון שבמקרה זה מדובר בתקשורת מקומית על המחשב).
- בכתום** - אנו רואים את מצב החיבור. נלמד על משמעות מידע זה בהמשך הפרק.

נזכיר שרצינו לדעת אילו פורטים פתוחים במחשב שלנו. מכאן שהמیدע שמעניין אותנו נמצא בטור הירוק. כעת ננסה למצוא את החיבור שלנו כשריצא את השרת שכתבנו בפרק תכנות-sockets. הריצו את השרת הראשון

שכתבנו בפרק [תכנות ב-Sockets-תרגיל 2.3 מודרך - השרת הראשון שלו](#), זה שמקבל שם מהלקוח ומחזיר לו

תשובה בהתאם:



לহזיכרכם, השרת בתרגיל האzin על פורט 8820.

הሪיצו את השרת:

```
C:\Windows\system32\cmd.exe - server.py
C:\Users\USER>cd \Cyber
C:\Cyber>server.py
```

אפשרו לשרת להמשיך לרווח. כתע, הריצו שוב את הכלי **netstat**. איןכם צפויים לראות את ההאזנה. דבר זה נובע מכך שבאופן בירית מחדל, **netstat** מציג רק חיבורים קיימים. כמובן, כל עוד אף לקוח לא התחבר לשרת שהרצתם, לא תראו שהמחשב שלכם מאזין על הפורט הרלבנטי. בכך לגרום **-l** ל-**netstat** להציג בכל זאת את החיבור שלנו, נשימוש בדגל **a**³⁶, ככלומר נריץ את הפקודה בצורה הבאה:

netstat -a

כעת אם נביט בפלט, נוכל לראות את ההאזנה שאנו מבצעים:

³⁶ דגל (באנגלית flag) בהקשר זהה הינו פרמטר לפקודה. כך למשל, הפרמטר "a" לפקודה **netstat -a** אשר מציין לפקודה להראות את כל החיבורים.

Active Connections			
Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:22	USER-PC:0	LISTENING
TCP	0.0.0.0:135	USER-PC:0	LISTENING
TCP	0.0.0.0:445	USER-PC:0	LISTENING
TCP	0.0.0.0:5357	USER-PC:0	LISTENING
TCP	0.0.0.0:49152	USER-PC:0	LISTENING
TCP	0.0.0.0:49153	USER-PC:0	LISTENING
TCP	0.0.0.0:49154	USER-PC:0	LISTENING
TCP	0.0.0.0:49155	USER-PC:0	LISTENING
TCP	0.0.0.0:49157	USER-PC:0	LISTENING
TCP	0.0.0.0:49162	USER-PC:0	LISTENING
TCP	0.0.0.0:49165	USER-PC:0	LISTENING
TCP	127.0.0.1:2559	USER-PC:0	LISTENING
TCP	127.0.0.1:5354	USER-PC:0	LISTENING
TCP	127.0.0.1:5354	USER-PC:49160	ESTABLISHED
TCP	127.0.0.1:5354	USER-PC:49161	ESTABLISHED
TCP	127.0.0.1:27015	USER-PC:0	LISTENING
TCP	127.0.0.1:27015	USER-PC:49200	ESTABLISHED
TCP	127.0.0.1:49160	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49161	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49200	USER-PC:27015	ESTABLISHED

למעשה, ניתן לראות את ההאזנה כבר בשורה הראשונה!

משמעותו של State הינו LISTENING. מכך אנו למדים שההמשמעות של LISTENING היא שהמחשב מוכנה ליצור חיבור. שורה שה-State שלו הוא ESTABLISHED, מתארת חיבור רצ'וקי'ם.

בואו נבחן זאת. פיתחו את Python, וכתבו לקוון קטן אשר מתקשר עם השרת אותו יצרתם, כפי שלמדו בפרק תכנות-ב-Sockets. אל תנטקו את החיבור בסופו ועל תסגורו את אובייקט ה-socket, שכן אנו מנסים לשמור על החיבור פתוח.

להלן דוגמה לקוד זהה:

```
import socket

my_socket = socket.socket()
my_socket.connect(('127.0.0.1', 22))

print 'I am connected!'
raw_input()
```

הערה: ההוראה raw_input() תמנע מן הסקריפט לסיים את הריצה כאשר תריצו אותו, שכן היא גורמת לסקריפט לחכות לקלט מהמשתמש.

הሪיצו את הקוד. כתע הריצו שוב את הפקודה **netstat**:

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:22	USER-PC:61493	ESTABLISHED
TCP	127.0.0.1:5354	USER-PC:49160	ESTABLISHED
TCP	127.0.0.1:5354	USER-PC:49161	ESTABLISHED
TCP	127.0.0.1:27015	USER-PC:49200	ESTABLISHED
TCP	127.0.0.1:49160	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49161	USER-PC:5354	ESTABLISHED
TCP	127.0.0.1:49200	USER-PC:27015	ESTABLISHED
TCP	127.0.0.1:57236	USER-PC:57242	ESTABLISHED
TCP	127.0.0.1:57240	USER-PC:57241	ESTABLISHED
TCP	127.0.0.1:57241	USER-PC:57240	ESTABLISHED
TCP	127.0.0.1:57242	USER-PC:57236	ESTABLISHED
TCP	127.0.0.1:57247	USER-PC:57248	ESTABLISHED
TCP	127.0.0.1:57248	USER-PC:57247	ESTABLISHED
TCP	127.0.0.1:61493	USER-PC:ssh	ESTABLISHED
TCP	192.168.14.51:51817	192.168.14.153:microsoft-ds	ESTABLISHED
TCP	192.168.14.51:54209	wb-in-f125:5222	ESTABLISHED
TCP	192.168.14.51:61183	fa-in-f189:https	ESTABLISHED
TCP	192.168.14.51:61457	bzq-179-154-217:https	ESTABLISHED
TCP	192.168.14.51:61459	bzq-179-17-162:https	ESTABLISHED
TCP	192.168.14.51:61462	173.194.116.181:https	ESTABLISHED

הפעם אין צורך בדגל `-a`. מכיוון שהחיבור קיים (במצב ESTABLISHED), הרי ש-**netstat** מציג לנו אותו גם ללא שימוש בדגל זה. בדוגמה לעיל, השורה הרלוונטייה היא השורה הראשונה.



מי מחייב על מספרי הפורטים?

בפועל, פорт הינו מספר בין 0 ל-65,535. על מנת שתוכנה אוחת תוכל להתחבר לתוכנה מרוחקת, עליה לדעת את הפורט שבו התוכנה המרוחקת מازינה.

לשם כך, ישנו **פורטים מוכרים (Ports)** (Well known ports). אלו הם הפורטים מ-0 ועד 1023, והם הוקצו בידי IANA (Internet Assigned Number Authority³⁷). כך למשל ידוע שהפורט 80 משוייך לפרטוקול HTTP. ישנו פורטים נוספים אשר הוקצו בידי IANA ולא נמצאים בטוחה 0-1023. במקרה אחר, מפתחי אפליקציות פשוט צריכים להסכים על הפורט בו הם משתמשים. כך למשל, בשרת הדיבים שכתבנו בפרק [תכנות ב-Sockets-תרג'il](#) - **2.5 - מימוש שרת הדיבים**, החלטנו להאזין על פорт 1729. במקרה זה, כל לקוח שירצה להשתמש בשרת שלנו, צריך לדעת שהוא משתמש במספר הפורט זהה בכדי להצליח לגשת לשרת.

העברה אמינה של מידע

עד כה דיברנו על אוחת המטרות של שכבת התעבורה, והיא ריבוב תקשורת של כמה תוכנות. מטרת נוספת נוספת של שכבת התעבורה הינה סיפוק העברת מידע בצורה אמינה.

³⁷ IANA (דף הבית: <https://www.iana.org>) הוא ארגון האחראי על ניהול והקצאה ייחודית של מספרים באינטרנט.

הרשת בה משתמש שכבת התעבורה כדי להעביר מידע עשויה להיות לא אמינה. למשל, חבילות מידע יכולות "ללאות לאיבוד" בדרך ולא להגיע ליעדן, או אולי להגיע בסדר הלא נכון (ח毕לה מס' 2 הגיע לפני ח毕לה מס' 1). שכבת האפליקציה לא רוצה להתעסק בכך. היא רוצה לבקש משכבות התעבורה להעביר מידע מתוכנה אחת לתוכנה שנייה, ולא לדאוג למקורה שהח毕לה לא הגיע. לשם כך, שכבת התעבורה צריכה לספק העברת אמינה של מידע מצד לצד.

עם זאת, לא תמיד נרצה ששכבת התעבורה תספק העברת אמינה של המידע. לכן, מטרת זה היא אופציונאלית בלבד - ובחלק מהשימושים של פרוטוקולי שכבת התעבורה אין הבטחה שהמידע יגיע ושיגיע בסדר הנכון. בהמשך הפרק נזכיר פרוטוקולים שונים של שכבת התעבורה, וכן נבין מדוע לעיתים נדרש להשתמש בפרוטוקול שmbטיח אמינות, ובמקרים אחרים נעדיף פרוטוקול שלא מבטיח אמינות.

מיקום שכבת התעבורה במודול השכבות

שכבת התעבורה הינה שכבה הרביעית במודול חמש השכבות.



מה השירותים לשכבת התעבורה מספקת לשכבה שמליה?

עבור שכבה החמישית, שכבת האפליקציה, היא אפשררת:

- לשלוח ולקבל מידע מתוכנה (תהליך) מרוחקת.
- במידה שהחיבור אמין - היא מאפשרת ליצור חיבור בין תוכנות שונות, וכן לסגור את החיבור.

מכאן שבעור שכבת האפליקציה, שכבת התעבורה מאפשרת להעביר מידע מהתהליך שלה אל הצד השני של הרשת. דוגמה לכך פגשנו בפרק [התכוות ב-Sockets-תרגול 2.1 מודרך - הלקוח הראשון שלו](#). הזכירו בדוגמה הקוד הבא מפרק זה:

```
import socket

my_socket = socket.socket()
my_socket.connect(('1.2.3.4', 8820))

my_socket.send('Omer')
data = my_socket.recv(1024)
print 'The server sent: ' + data

my_socket.close()
```

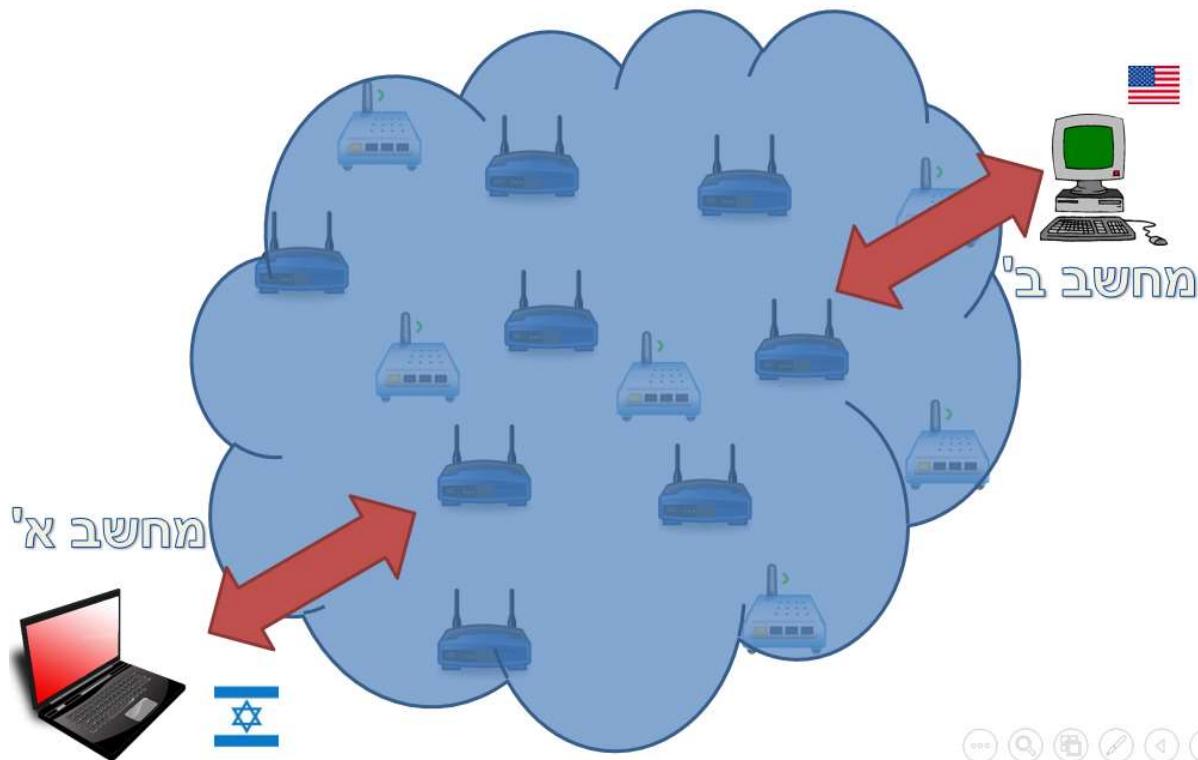
בתווך מתוכנים, שלחנו הודעות מהתוכנה שלנו, לתוכנה שנמצאת על הרשת. במקרה זה, שלחנו את המידע 'Omer' מהתוכנה שלנו, אל תוכנה שנמצאת בשרת המרוחק בכתביות 1.2.3.4 (הכתובת לדוגמה עbor שם

הדומיין networks.cyber.org.il ומאזינה לפורט 8820. דאג לכל שאר התהילה, ולכן שההודעה באמת הגיע מצד לצד. זהו בדיקת השירות שמאפשרת שכבת התעבורה אל שכבת האפליקציה, במקרה זה - באמצעות הממשק של Sockets.



מה השירותים ששכבת התעבורה מקבלת מן השכבה שמתחתייה?

שכבת הרשת, השכבה השלישייה, מספקת לשכבת התעבורה מודול של "ענן", שבו חבילות מידע מגיעות מצד אחד לצד שני. שכבת התעבורה אינה מודעת כלל למבנה הרשת המתוואר, ולמעשה מבחינתה יש פשוט "רשות כלשהי" שמחברת בין מחשב א' למחשב ב'. "תמונה הרשת", מבחינתה, נראה כך:



שימוש לב לשכבת הרשת אינה מודעת לפורטים. לכן, בשכבת הרשת העברת חבילת מידע מתבצעת מישות לישות (לדוגמא - בין מחשב למחשב), ובשכבת התעבורה, היא מתבצעת מתוכנה אחת לתוכנה אחרת (כלומר - מפורט מסויים אל פорт אחר).

פרוטוקולים מבוססי קישור ולא מבוססי קישור

בשכבת התעבורה, פרוטוקולים יכולים להיות מבוססי קישור (Connection Oriented) או לא מבוססי קישור (Connection Less).

פרוטוקולים מבוססי קישור

ניתן להמשיל פרוטוקולים מבוססי קישור למערכת הטלפוניה. כדי לתקשר עם מישוה באמצעות הטלפון, علينا להרים את מכשיר הטלפון, לחיג את המספר שלו, לדבר ואז לנתק את השיחה. לא ניתן פשוט לדבר אל מכשיר הטלפון, ולצפות שאדם בצד השני יקבל את המסר שלנו, אם כלל לא חיגנו אליו. באופן דומה, על מנת לתקשר עם מישוה באמצעות פרוטוקול מוביל קישור, יש ראשית "להקימם" את הקישור, לאחר מכן מכון להשתמש בקשרו שהוקם ולבסוף לנתק את הקישור. מבחינת המשטמש, הוא מתיחס לקישור כמו לשיפורת הטלפון: הוא מzin מידע (במקרה שלנו - רצף של בתים) לקצה אחד, והמשתמש השני יקבל את המידע בצד השני.

דוגמה לפרוטוקול מבוסס קישור היא פרוטוקול TCP (شبקרוב נכיר לעומק), או בשמו המלא - Transmission Control Protocol. כשאנו, בתור מפתחי שכבת האפליקציה, משתמשים ב-TCP כדי להעביר מידע, איננו יכולים פשוט לשלוח חבילה אל תוכנה מרוחקת. ראשית علينا ליצור קישור עם התוכנה המרוחקת, ועתה כל חבילה שנשלח תהיה חלק מאותו קישור.

פרוטוקולים מבוססי קישור מבטחים אמינות בשילוח המידע. כמובן, הם מבטחים שככל המידע שנשלח יגיע אל מקבל, וכן שהוא יגיע בסדר שבו הוא נשלח. עם זאת, לפרוטוקולים מבוססים קישור יש **תקורה (Overhead)** גבוהה יחסית. כאמור, ישנו מידע רב שנשלח בראשת בונוס על המידע שרצינו להעביר. אם נרצה את המידע "שלום לכם" באמצעות פרוטוקול מבוסס קישור, علينا להנify את הקישור לפני שליחת ההודעה, לסיטם את הקישור בסיטם, ולהשתמש במנגנוןים שונים כדי להבטיח שהמסר אכן הגיע אל היעד. פעולות אלו לוקחות זמן ומשאבים, ולכן העברת המידע "שלום לכם" תהיה איטית יותר מאשר שליחת המסר בלבד הרמת הקישור.

תקורה קיימת גם במקומות אחרים בחיים. למשל, על מנת ללמוד שיעור שמתרכש בבית הספר, עליהם לקום מהמייטה, להתלבש, לצאת מהבית, ולהגיע אל בית הספר. במידה שה坦מל מצלם, אתם יכולים להגיע ברגל. אם אתם גרים במרקח מסוים, יתכן עליהם להגיע אל תחנת האוטובוס, להמתין עד שיגיע האוטובוס ולנסוע באמצעותו אל בית הספר. כל זאת הנה תקורה של התהילה - מטרתכם היא אמונה ללמידה בשיעור בבית הספר, אך עליהם לעבור תהילה על מנת לעשות זאת. במקרה זה, ניתן היה למשל להנify את התקורה אם היותם בחורים לישון בבית הספר, ובכך הייתה נמנעת התקורה של תהילה ההגעה. עם זאת, כפי שווידאי מובן להם, לעיתים עדיף לשלם את מחיר התקורה על מנת לקבל את היתרונות שהיא מציעה (שינוי בבית, או העברה אמונה של מידע מעל הרשת).

פרוטוקולים שאינם מבוססי קישור

ניתן להמשיל פרוטוקולים שאינם מבוססי קישור לרשת הדואר. כל מכתב שאנו שולחים באמצעות הדואר כולל את כתובות היעד שלו, וכל מכתב עומד בזכותו עצמו: הוא עובר בראשת הדואר בלבד קשור למכתבים אחרים שנשלחים. ברוב המקרים, אם נשלח שני מכתבים מכתבות אחת לכתבת שנייה, המכתב הראשון שנשלח יהיה זה שיגיע ראשון. עם זאת, אין לכך הבטחה, ולעתים המכתב השני שנשלח יגיע קודם לכן.

דוגמה לפרוטוקול שאינו מבוסס קישור היא פרוטוקול UDP (شبקרוב נכיר לעומק), או בשמו המלא - User Datagram Protocol. כמשמעותו, בטור מפתח שכבת האפליקציה, משתמשים ב-UDP ב כדי לשוחח חבילה, אין הבטחה שהחביבה תגעה ליעדה. כמו כן, אין הבטחה שהחביבות תגעה בסדר הנכוון. אף לכך, אין גם צורך בהרמה וסירה של קישור. באם מתכונת בשכבת האפליקציה רוצה לשוחח חבילה מעל פרוטוקול UDP, הוא פשוט שולח את החביבה.



מתי נעדיף פרוטוקול מבוסס קישור ומתי פרוטוקול שלא מבוסס קישור?

לפרוטוקולים מבוססי קישור, כמו TCP, יתרונות רבים. הם מבטיחים הגעה של המידע לצורה אמינה ובסדר הנכוון. אף לכך, נבחר להשתמש בהם במקרים רבים. לדוגמה, כאשר אנו מורים קובץ מהאינטרנט, היגיון שנעשה זאת מעל TCP: לא נרצה לחלוק מהקובץ היה חסר, שכן אז נוכל לפתח אותו. כמו כן לא נרצה לחלקים מהקובץ הגיעו בסדר לא נכון, וזה הקובץ לא יהיה תקין.

עם זאת, לא תמיד נרצה להשתמש בפרוטוקול מבוסס קישור כגון TCP. כמו שכבר למדנו קודם, ל-TCP יש תקורה גבוהה יחסית: יש צורך בהקמה וסירה של קישור, יש צורך לוודא שהמידע הגיע ליעד והגיע בסדר הנכוון... למעשה, שימוש ב-TCP גורר יותר זמן ומשאבים מאשר שימוש בפרוטוקול שאינו מבוסס קישור כגון UDP. לעיתים, העברת מהירה של המידע תהיה חשובה לנו הרבה יותר מאשר העברת אמינה של המידע.

באו נבחן יחד את המקרים הבאים:

מקרה מבחן: תוכנה להעברת קבצים גדולים

מה דעתכם - האם בתוכנה להעברת קבצים גדולים בין מחשבים נעדיף להשתמש ב-UDP או ב-TCP? התשובה במקרה זה היא TCP. כמו שאמרנו קודם, במקרה של העברת קובץ - נרצה שכל המידע על הקובץ יגיע, ושיגיע בסדר הנכוון. אחרת, יתכן ולא נוכל לפתח את הקובץ בכלל. במקרה זה נעדיף "לשלם" את המחיר של הרמת וסירת קישור, וודיא הגעת המידע וכל ה-Overhead המשתמע משימוש ב-TCP - על מנת שהמידע יגיע באופן אמין.

מקרה מבחן: פרוטוקול DNS

הזכירו בפרוטוקול DNS עליו למדנו בפרק [שכבת האפליקציה](#). מה דעתכם - האם בשימוש ב-S-DNS נעדיף להשתמש ב-UDP או ב-TCP?

התשובה במקרה זה היא UDP³⁸. הסיבה לכך היא ש-DNS הוא פרוטוקול מסווג שאילתא-תשובה. הלוקוח שלוח לשרת שאלת (למשל: "מי זה www.google.com?", שזו חביבה אחת בלבד, ומתקבל עליה התשובה. באם לא הגיעו תשובה, הלוקוח יכול לשלוח את השאלתא שוב. במקרה זה, לא משתלים להרים קישור TCP שלו.



תרגיל 6.2 מודרך - מעל איזה פרוטוקול שכבת התעבורה עובר DNS?

נססה לאמת את ההנחה שלנו - האם באמצעות פרוטוקול DNS עובר מעל UDP? הריצו את Wireshark, ופיתחו הסנהפה. השתמשו במסנן "dns":

Filter: dns

כעת, פיתחו את ה-Command Line והשתמשו בכל **nslookup** כדי לשלוח שאלתא על הדומיין www.google.com:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>nslookup www.google.com
Server: box.privatebox
Address: 192.168.14.1

Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4001:c02::93
          173.194.34.83
          173.194.34.82
          173.194.34.80
          173.194.34.81
          173.194.34.84

C:\Users\USER>
```

מיצאו את השאלה הרלבנטית והסתכו על החביבה:

³⁸ ישנו גם שימוש של DNS מעל TCP, אך השימוש הנרחב הוא מעל פרוטוקול UDP.

```

Frame 12: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 192.168.14.1 (192.168.14.1)
User Datagram Protocol, Src Port: 65522 (65522), Dst Port: domain (53)
  Source port: 65522 (65522)
  Destination port: domain (53)
  Length: 40
  Checksum: 0x9dbe [validation disabled]
Domain Name System (query)
  Response In: 131
  Transaction ID: 0x0004
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    www.google.com: type A, class IN

```

כפי שניתן לראות, הפרטוקול בו מתבצע שימוש הוא פרוטוקול UDP (מוסמן באדום). פורט היעד אליו מתבצעת הפניה הינו פורט 53 (מוסמן בכחול), והוא הפורט המשויך לפרוטוקול DNS.

מקרה מבחן: תוכנה לשיתוף תמונות

מה דעתכם - האם בתוכנה להעברת תמונות רבות בין מחשבים נעדיף להשתמש ב-UDP או ב-TCP?
גם במקרה זה האמיןות חשובה לנו יותר ממהירות, ולכן נשתמש ב-TCP. מכיוון שנרצה שכל התמונות יגיעו באופן תקין ונוכל לצפות בהן, علينا לשלם את המחיר של שימוש בפרטוקול מבוסס קישור.

מקרה מבחן: Skype

מה דעתכם - האם בתוכנה לביצוע שיחות IP over Voice called Skype נעדיף להשתמש ב-UDP או ב-TCP?
במקרה זה בולט מאד הצורך ב מהירות - אנו רוצים שהקהל שלנו יוכל מצד לצד באופן כמו שהיא. גם כאן אין הפסד גדול באמ חילק מהחbillות הללו לאיבוד בדרך. אך במקרה זה נעדיף להשתמש בפרטוקול UDP.

תרגיל 6.3 - מקרה מבחן: שרת HTTP



פרק שכבת האפליקציה למדנו על פרוטוקול HTTP. חשוב בעצמכם - האם נעדיף במקרה של שרת HTTP להשתמש ב-UDP או ב-TCP?
כעת, וודאו את תשובתכם. השתמשו Wireshark ובדפדף בצד לגלוש לשרת HTTP, ומצאו האם פרוטוקול שכבת התעבורה בו משתמש השירות הוא בהתאםprotokol בו חשבתם שהוא ישמש.

שאלת חשיבה: מדוע צריך שכבת תעבורה לא אמינה מעל שכבת רשת לא אמינה?



ננו לחשב על כך: אם שכבת הרשת שלנו אינה אמינה ולא מבטיחה העברת של מידע מצד לצד, מדוע להשתמש בכלל בשכבה תעבורת לא אמינה? מדוע לשימוש בפרוטוקול UDP ולא לשילוח חבילות ישר מעל שכבת הרשת?

לשימוש בפרוטוקול לא אמין של שכבת התעבורת (כדוגמת UDP) מעל שכבת רשת לא אמינה, יש שתי סיבות עיקריות. ראשית, השימוש בפורטים. כפי שהסבירנו קודם בפרק, השימוש בפורטים הוא הכרח בכך לדעת לאיזו תוכנה אנו פונים בשרת המרוחק. UDP מאפשר לנו את השימוש בפורטים.

בנוסף על כן, שימוש של שכבת האפליקציה בשכבת הרשת "ישבו" את מודל השכבות: איןנו רוצים שמתכונת של שכבת האפליקציה יכיר את שכבת הרשת. דבר זה יגרום למפתח של שכבת האפליקציה להעמיק בסוגיות הקשורות לשכבת הרשת, ולכתוב מימוש שונה עבור כל פרוטוקול בשכבה זו. מבחינה מפתח של שכבת האפליקציה, הוא צריך להכיר רק את שכבת התעבורת והשירותים שהואנות ל. בacr, שכבת התעborת "מעילה" את שכבת הרשת משכבת האפליקציה, בין אם היא מספקת אמינות ובין אם לא.

UDP - User Datagram Protocol

עכשו כשהבנו לעומק את מטרותיה של שכבת התעborת, כמו גם את ההבדלים בין פרוטוקולים מבוססי קישור לפרוטוקולים לא מבוססי קישור, הגיע הזמן להכיר את אחד הפרוטוקולים הנפוצים ביותר בשכבה זו - פרוטוקול UDP.

כאמור, פרוטוקול UDP אינם מבוססי קישור. חלק מכך, UDP לא מבטיח הגעה של המידע כלל והגעה בסדר הנכון בפרט. הדבר דומה לשילוח מכתב בדואר רגיל (שאינו רשום): אם ברצוני לשולח מכתב למישחו, עלוי לשים אותו במעטפה ולששל אותו לתיבת. אין לי צורך להגיד לך אל הנமען שהוא צפוי לקבל ממנו את ההודעה, וכן אין הבטחה של רשות הדואר שהמכתב יגיע מהר, או שיגיע בכלל. יתכן והמכתב יאבד בדרך.

תרגיל 6.4 מודרך - התבוננות בפרוטוקול UDP

הריצו את Wireshark. הסניפו עם המסן "kdpn". חכו עד שחבריות UDP תופענה על המסך. לחילופין, תוכלו לשולח שאלתת DNS, שנשלחת מעל UDP כפי שלמדנו קודם. בחרו באחת החבילות. הסתכלו על ה-Header של החבילה:

```

+ Frame 12: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
+ Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
+ Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 192.168.14.1 (192.168.14.1)
- User Datagram Protocol, Src Port: 65522 (65522), Dst Port: domain (53)
  Source port: 65522 (65522)
  Destination port: domain (53)
  Length: 40
  + Checksum: 0x9dbe [validation disabled]
- Domain Name System (query)
  [Response In: 13]
  Transaction ID: 0x0004
  + Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
- Queries
  + www.google.com: type A, class IN

```



מה גודל ה-Header של חבילה UDP?

ב כדי לענות על שאלת זו, נשתמש בעזרתו של Wireshark, שידע לספר עבורנו בתים. נלחץ עם העכבר על שורת ה-UDP (מוסמן באדום):

```

+ Frame 12: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
+ Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
+ Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 192.168.14.1 (192.168.14.1)
- User Datagram Protocol, Src Port: 65522 (65522), Dst Port: domain (53)
  Source port: 65522 (65522)
  Destination port: domain (53)
  Length: 40
  + Checksum: 0x9dbe [validation disabled]
- Domain Name System (query)
  [Response In: 13]
  Transaction ID: 0x0004
  + Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
- Queries
  + www.google.com: type A, class IN

```

0010	00	3c	01	70	00	00	80	11	0b	bc	c0	a8	0e	33	c0	a8	.<.p....3..
0020	0e	01	ff	f2	00	35	00	28	9d	be	00	04	01	00	00	015.(.....
0030	00	00	00	00	00	00	00	77	77	77	06	67	6f	6f	67	6cw	ww.googl
0040	65	03	63	6f	6d	00	00	01	00	01							e.com...	..

User Datagram Protocol (udp, 8 bytes) Packets: 5951 Displayed: 40 Marked: 0 Dropped: 0

קטע Wireshark יסמן לנו את השורה גם היקן שלחצנו, וגם בציגה התחרתונה שמרתא את הבטים שנשלחו (מוסמן **בירוק**). בנוסף, הוא יכתוב לנו למטה את כמות הבטים שסימנו (מוסמן **בכחול**). מכאן שהגודל של Header של חבילה UDP הוא שמונה בתים.

נסו לענות בעצמכם על השאלה הבאה לפני תמשיכו את הקריאה:



אילו שדות יש ב-Header של חבילה UDP? מה התפקיד של כל שדה?

שני השדות הראשונים קלים להבנה:

- **Source Port** (פורט מקור) - הפורט של התוכנה ששלחה את החבילה. במקרה זה, זהו הפורט של התוכנה ששלחה את שאלתת ה-DNS ומחכה לקבל תשובה. שרת DNS צפוי להחזיר את התשובה שלו אל הפורט זהה.
- **Destination Port** (פורט יעד) - הפורט של התוכנה שצפוי לקבל את החבילה. במקרה זה, זהו הפורט של שירות ה-DNS.

השדה הבא הינו שדה האורך (Length).



מה מציין שדה האורך ב-UDP?

האם הוא מציין את אורך המידע של חבילת ה-UDP (במקרה זה, ה-DNS)? האם את אורך ה-Header או את האורך הכולל של ה-Header והמידע?

בדומה לכך בה גילינו את אורך ה-Header של החבילה, נשתמש בספירת הבתים של Wireshark בצד לגלות את גודל שכבת DNS בחבילה זו:

The screenshot shows a Wireshark capture of a DNS query. The tree view highlights the 'Length' field in the User Datagram Protocol (User Datagram Protocol) section. The packet details pane shows the DNS header and the query for 'www.google.com'. The bytes pane displays the raw hex and ASCII data of the packet, with the domain name clearly visible in the ASCII view.

לאחר שנלכד על השורה של ה-**Domain Name System**, היא תסומן בידיו Wireshark (מוסומן באדום בתמונה לעיל).icut, Wireshark יראה לנו גם את גודל השכבה - 32 בתים (מוסומן בירוק).

היות שגילינו קודם לכך שגודלו של ה-Header הוא שווה בתים, ועכשו גילינו שגודלו המידיע במקרה זה הוא 32 בתים, אנו לומדים ששדה האורך ב-Header של UDP מתרחשת גודל ה-Header והמידיע גם יחד.

ב כדי להבין את משמעות השדה הבא, נצטרך לענות על השאלה:



עד כה ציינו שבעיות ברשות יכולות לגרום לחבילה לא להגיע כלל, או לרוץ של חבילות להגיע ברכף הלא נכון. אך בעיות ברשות יכולות גם לגרום לשגיאות בחבילה עצמה - כלומר שהחביבה תגיע עם תוכן שונה מהתוכן שנשלח במקור.

לדוגמה, נביט בפרוטוקול שנועד לשלוח מספרי טלפון נייד מחשב אחד למחשב אחר. בפרוטוקול זה, בכל חבילה, נשלחות 10 ספרות של מספר טלפון אחד. כך למשל, חבילה לדוגמה יכולה להראות כך:



הבעיה היא, שיתכן והחביבה השתנתה בדרך כלל תקלת כלשהי. כך למשל, יתכן והשרת יקבל את החביבה بصورة הhabah:

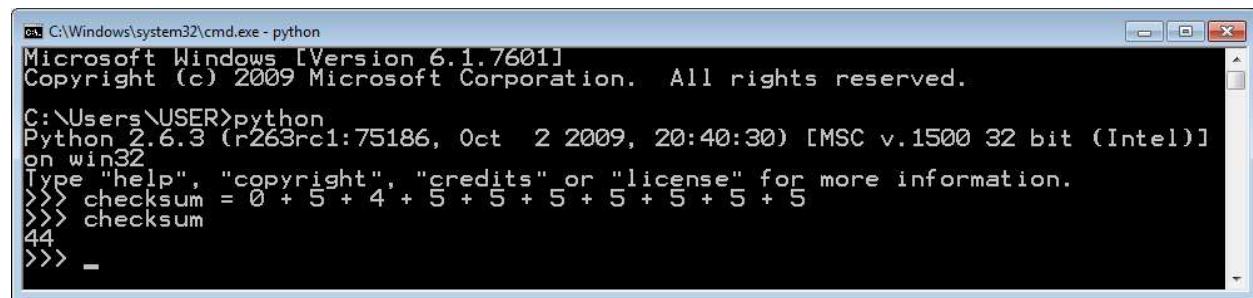


שיםו לב, הספרה הראשונה השתנה, ועכשו היא כבר לא 0 אלא 6. במקורה זה, נרצה שהשרת ידע שאירוע שגיאה, ולא יתייחס לחבילה התקוליה. דרך אחת לעשות זאת, היא להשתמש ב-Checksum. הרעיון הוא כזה: נבצע פעולה כלשהי על המידע שהוא רצים לשלוח, ונשמר את התוצאה. בצד השני (במקורה זה, בצד השרת) החישוב יבוצע שוב, ויושווה לתוצאה שנשלחה. אם התוצאה שונה, הרוי שיש בעיה.

נמשיך עם הדוגמה של מספר הטלפון הנגיד. נאמר ובחרנו בפונקציית `Checksum` הבאה: חיבור כל הספרות של מספר הטלפון. כלומר, עבור מספר הטלפון 054-5555555 שראינו קודם, יבוצע החישוב הבא:

$$0 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5$$

אפשר לעשות זאת באמצעות פיתון ולראות את התוצאה:



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>python
Python 2.6.3 (r263rc1:75186, Oct  2 2009, 20:40:30) [MSC v.1500 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> checksum = 0 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5
>>> checksum
44
>>> -
```

כעת, השולח ישלח לא רק את המידע שהוא רוצה לשלוח (כלומר את מספר הטלפון), אלא גם את התוצאה של ה-Checksum. בדוגמה זו, תשלח החבילה הבאה:



עכשו, במידה שתקרה אותה השגיאה שהתרחשה קודם לכן, השרת יקבל את ההודעה הבאה:



כעת השירות ינסה לבצע את החישוב של ה-Checksum על המידע עצמו:

$$6 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5$$

שוב, נוכל להשתמש בפייטון:

```
C:\Windows\system32\cmd.exe - python
C:\Users\USER>python
Python 2.6.3 (r263rc1:75186, Oct  2 2009, 20:40:30) [MSC v.1500 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> checksum = 6 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5
>>> checksum
50
>>> _
```

התוצאה יוצאה 50, אך ה-Checksum שהלוח שלח היה 44.³⁹ אי לכך, יש שגיאה בחבילה - והיא צריכה להיזרק.

במציאות שדה נוסף בן 2 ספרות, הצלחנו לוודא שהמידע ששלחנו ב-10 הספרות הקודומות הגיע בצורה תקינה. עם זאת, הפונקציה שלנו אינה מושלמת, מן הסתם. אם, בדוגמה הקודמת, השירות היה מקבל את המספר 0635555555, התוצאה של ה-Checksum הייתה עדין 44, וזה לא המספר אותו הלוח התכוון לשלח. בספר זה לא נסביר את הפונקציה שבה משתמשים כדי לחשב את ה-Checksum ב프וטוקול UDP, אך חשוב שבני את המשמעות של השדה הזה ושהוא נדרש למציאת שגיאות.

באם אתם מעוניינים לראות דוגמה נוספת ל-Checksum, אתם מוזמנים לקרוא על סירת ביקורת במספר הזאות בישראל, בכתבות: <http://goo.gl/CvYtgt>. לכל אזרח בישראל יש מספר זהות בעל תשע ספרות. למעשה, שמונה הספרות השמאליות הן מספר הזאות עצמו, והספרה הימנית ביותר היא סירת הביקורת - תפקידה לוודא שאין שגיאה בכתיבה של שמונה הספרות שלפניהם.

³⁹ אלא אם כן, הייתה שגיאה בשדה ה-Checksum עצמו. גם במקרה זה, החבילה צפוייה להיזרק.

אגב, אין חובה להשתמש בChecksum בפרוטוקול UDP. אם הלקוח לא מעוניין להשתמש בChecksum, ניתן לשלוח 0 בשדה של הChecksum.

נסכם את מה שלמדנו על שדות ה-Header של UDP:

- (פורט מקור) - הפורט של התוכנה ששלחה את החבילה.
- (פורט יעד) - הפורט של התוכנה שצפוייה לקבל את החבילה.
- (אורך) - אורך החבילה (כולל Header ומידע).
- - חישוב כדי לוודא שהחביבה הגיעה באופן תקין.

UDP של Socket

עכשו שלמדנו על פרוטוקול UDP, הגיע הזמן להשתמש בקוד ששולח הודעות UDP.



תרגיל 6.5 מודרך - ליקוח UDP ראשון

עת נכתוב את ליקוח UDP הראשון. הלקוח יהיה דומה מאוד ללקוח הראשון שכתבנו בפרק [תכנות ב-Sockets](#). לצורך התרגיל, ישנו שרת שאיתו נרצה לתקשר. השרת נמצא באינטרנט, והוא בעל שם הדומיין networks.cyber.org.il. לצורך ההסביר, נניח כי כתובת ה-IP של השרת, אותה עליים למצוא באמצעות `nslookup` או כל אחר לבחירתכם. על השרת זהה, יש תוכנה שמאזינה בפורט 8821. אנו נתחבר אל השרת זהה, ונסלח לו את השם שלנו (לדוגמא: "Omer"). בהמשך, נכתוב את השרת שישתמש במידע זהה.



Omer

לקוח

נתחל מכתב ללקוח פשוט באמצעות Python. ההתחלה זהה לחלוון לנו שכתבנו בפרק [תכנות ב-Sockets](#). הדבר הראשון שעשינו לעשות לשם כך הוא ליבא את המודול של [תרגיל 2.1 מודרר - הלקוח הראשון שלו](#):

```
import socket
```

כעת, עליינו ליצור אובייקט מסווג **socket**. נקרא לאובייקט זה בשם `:my_socket`:

```
my_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

כאן אנו נתקלים בהבדל הראשון בין הלקוח שכתבנו בפרק [תכנות ב-Sockets-תרגיל 2.1 מודרר - הלקוח הראשון שלו](#) לבין הלקוח אותו כתובים עכשווי. להזכירם, כאשר כתבנו את הלקוח הקודם, השתמשנו בשורה הבאה:

```
my_socket = socket.socket()
```

מכיוון שלא סיפקנו פרמטרים ל-(`socket`), פיתון הניח לנו להשתמש בפרמטרים ברירת המחדל, שהם:

```
my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

נעלם כרגע מהפרמטר הראשון (`socket.AF_INET`) ונטרכז בפרמטר השני, שיכל לקבל בין השאר את הערכים הבאים:

- `SOCK_STREAM` - הכוונה היא לשימוש בחיבור מבוסס קישור. בפועל, השימוש הוא ב프וטוקול TCP.
- `SOCK_DGRAM` - הכוונה היא לשימוש בחיבור שאינו מבוסס קישור. בפועל, השימוש הוא בפרוטוקול UDP.

מcean שהלkoח שכתבנו בפרק [תכנות ב-Sockets](#) השתמש, מבל' שצינו זאת באופן מראש, בפרוטוקול TCP. כעת, מכיוון שאנו מציינים את הparameter `socket.SOCK_DGRAM`, הוא ישתמש בפרוטוקול UDP:

```
my_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

lezיכרכם, כאשר כתבנו את הלkoח הקודם שלנו, השתמשנו בשלב זה בmethod `connect`. מודה זה יוצרה קישור בין התוכנה שלנו (הסקרייפט של הלkoח) לבין התוכנה המרוחקת (הסקרייפט של צד השרת על המחשב המרוחק). מכיוון שאנו כותבים באמצעות UDP, אין צורך בmethod זה, ונוכל ישירות לשלוח את המידע שלנו באמצעות method `sendto`:

```
my_socket.sendto('Omer', ('1.2.3.4', 8821))
```

כפי שניתן לראות, method `sendto` קיבלת את המידע שברצוננו לשולח ('Omer') וכן את ה-tuple שמתאר את התוכנה המרוחקת ומכיל כתובת IP ומספר פורט. בשורה זו שלחנו את המחרוזת 'Omer' אל התוכנה שמАЗינה לפורט 8821 בשרת בעל הכתובת "1.2.3.4".

על מנת לקבל מידע, علينا להשתמש בmethod `recvfrom`. שימו לב, שמכיוון שלא נוצר קישור ביןינו לבין השרת המרוחק, ניתן גם שנקלט מידע מישות אחרת. לכן, `recvfrom` גם מאפשר לנו לדעת ממי קיבלנו את המידע שקיבלנו:

```
(data, remote_address) = my_socket.recvfrom(1024)
```

ונכל כמובן להדפיס את המידע שקיבלנו:

```
print 'The server sent: ' + data
```

ולבסוף, "נסגור" את אובייקט ה-`socket` שיצרנו בכך לחסוך במשאבים:

```
my_socket.close()
```

להלן כל הקוד שכתבנו:

```
import socket

my_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

my_socket.sendto('Omer', ('1.2.3.4', 8821))

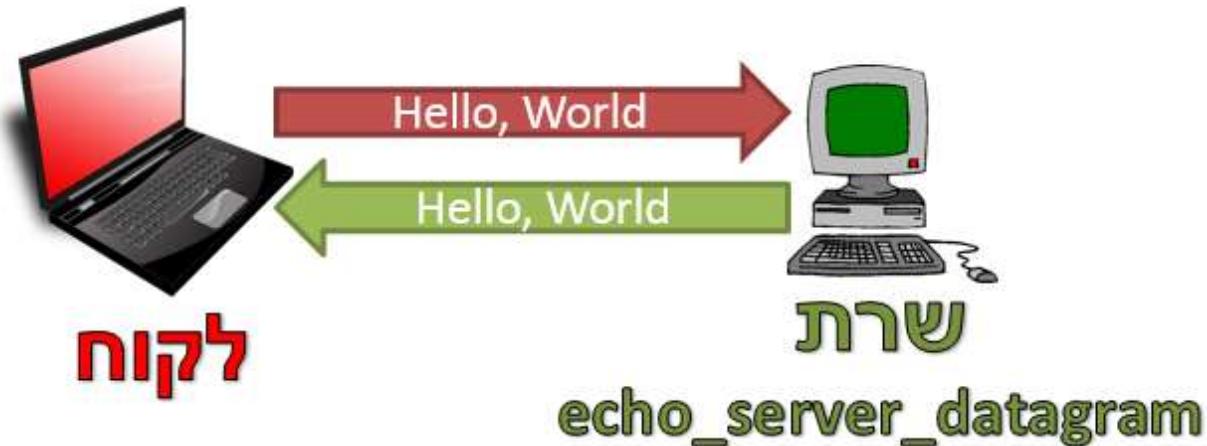
(data, remote_address) = my_socket.recvfrom(1024)

print 'The server sent: ' + data

my_socket.close()
```

תרגיל 6.6 - ל��וח לשרת הדימ

כעת כתבו ל��וח לשרת הדימ, בדומה ל��וח שתכתבם בפרק [תכנות ב-Sockets 2.2 - ל��וח לשרת הדימ](#). בתרגיל זה השרת כבר מושך עבורהכם. להזכירם, שרת הדימ משכפל כל מידע שתשלחו לו, ושולח אותו אליכם בחזרה, כמו הdad. כך למשל, אם תכתבו אל השרת את המידע: "Hello, World" (שםו לב - הכוונה היא למחוזת), הוא יענה: "Hello, World"



הורידו את השרת מהכתובת: http://data.cyber.org.il/networks/c06/echo_server_datagram.pyc. שמרו את הקובץ למיקום הבא:
C:\echo_server_datagram.pyc

על מנת להריץ את השרת, הכנסו אל ה-Command Line, והריצו את שורת הפקודה:
python C:\echo_server_datagram.pyc
השרת מازין על הפורט 1729.

תרגיל 6.7 - השוואת זמנים בשרת הדימ

כעת, נסדרג את הלוקה. עליכם לחשב כמה זמן לוקח מזמן שליחתם את הודעה אל השרת, ועד שהתקבלה תשובה (רמז: השתמשו במודול **time** של Python). הדפיסו למסך את הזמן זהה.

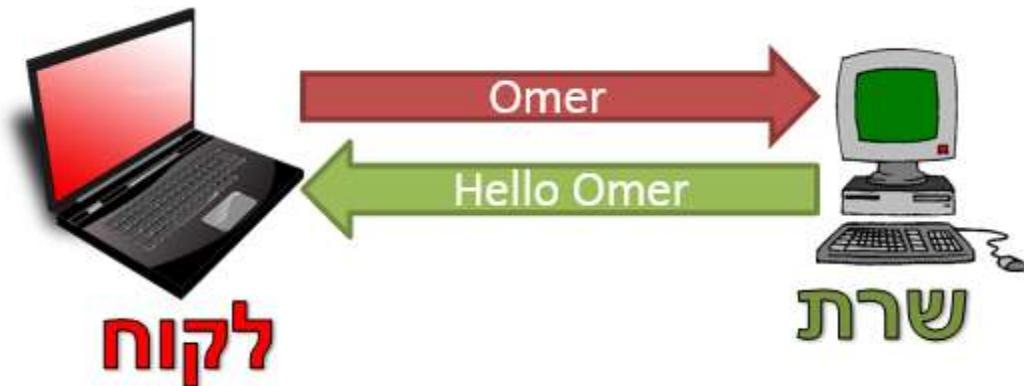
לאחר מכן, השתמשו בקוד שתכתבם בפרק [תכנות ב-Sockets - ל��וח לשרת הדימ](#), בו השתמשנו, כאמור, ב-TCP. הוסיףו גם ללקוח זה את היכולת למדוד זמן מהרגע שבו נשלחה הודעה אל השרת, לבין התשובה.

כעת, הריצו את הלקוחות ובדקו את הזמן. האם יש הפרש בין הזמן שלקח לתשובה להגיא בימוש ה-UDP לבין הזמן שלקח לתשובה להגיא בשימוש ה-TCP?

שיעור ללב: עליכם להריץ את הלקוחות אל מול שרת שנמצא במחשב מרוחק, ולא אל מול שרת שנמצא במחשב שלכם.

תרגיל 6.8 מודרך - שרת UDP ראשון

邏輯 מוקדם יותר, יצרנו ל��וח ששולח לשרת את שמו, לדוגמה: "Omer". כעת, נגרום לשרת לקבל את השם שה לקוחות שולח, ולענות לו בהתאם. לדוגמה, השרת יענה במקרה זה: "Hello, Omer"



גם ב-UDP, הדרך לכתיבת שרת דומה מאד לכתיבה של לקוח. גם הפעם, הדבר הראשון שעשינו לעשות הוא **לייבא את המודול של socket לפיתון:**

```
import socket
```

כעת, עליינו ליצור אובייקט מסווג **socket**. שוב, עליינו להגיד שמדובר בחיבור UDP. נקרא לאובייקט זה בשם **:server_socket**

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

בשלב הבא, עליינו לבצע קישור של אובייקט ה-**socket** שיצרנו לכתובת מקומית. לשם כך נשתמש בMETHOD **bind**. המתוודה זהה למקורה של שימוש ב-TCP. נשתמש בה, לדוגמה, כך:

```
server_socket.bind(('0.0.0.0', 8821))
```

בצורה זו יצרנו קישור בין כל מי שמנסה להתחבר אל הרכיב שלנו לפורט מספר 8821 - אל האובייקט **.server_socket**

הפעם, בניגוד לשרת ה-TCP שמיימנו בעבר, אין צורך להשתמש במתודה `listen`, וגם לא במתודה `accept` למשהו, אנו מוכנים לקבל מידע:

```
(client_name, client_address) = server_socket.recvfrom(1024)
```

מכיוון שלא הקמנו קישור, המתודה `recvfrom` מחזירה לנו לא רק את המידע שהלך שלח (אותו שמרנו אל המשתנה `client_name`), אלא גם את הכתובת של הלוקה (`client_address` המשווה במשתנה `address`). כתובות זו תשמש אותנו כנדרשה לשЛОח מידע חזרה אל הלוקה:

```
server_socket.sendto('Hello ' + client_name, client_address)
```

השימוש זהה למשה לקבלת ושליחת מידע בצד הלוקה, ומשתמש במתודות `recvfrom`-ו-`sendto` אשר פגשנו קודם לכן. שימושו לב שבנייגוד לתקשורת TCP, לא נוצר לנו אובייקט `socket` חדש עבור כל לוקה, שכן לא הרמננו קישור עם הלוקה.

cut נוכל לסגור את אובייקט ה-`socket`:

```
server_socket.close()
```

להלן כל הקוד של השרת שיצרנו:

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_socket.bind(('0.0.0.0', 8821))

(client_name, client_address) = my_socket.recvfrom(1024)

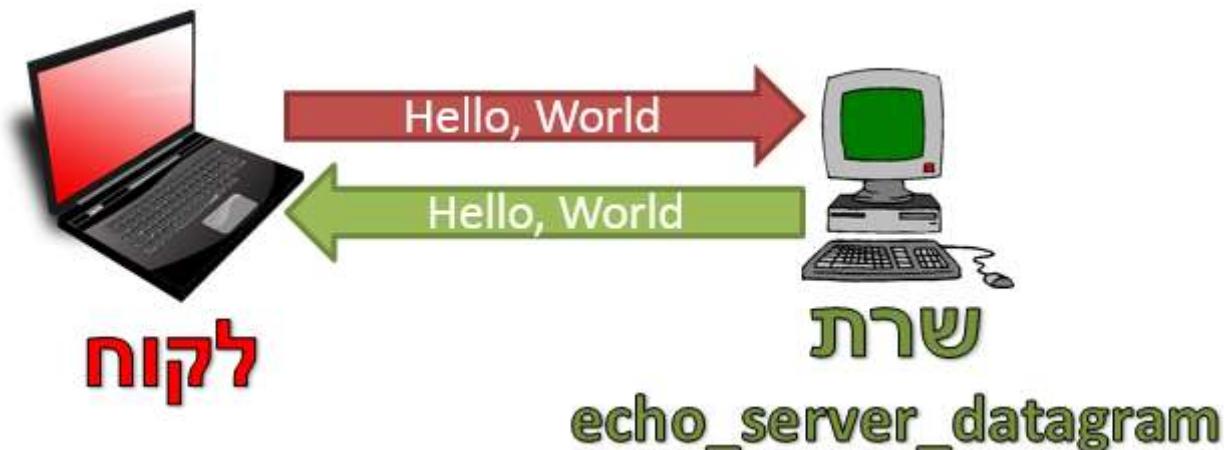
server_socket.sendto('Hello ' + client_name, client_address)

server_socket.close()
```

תרגיל 6.9 - מימוש שרת הדימ

בתרגיל קודם, כתבתם לוקה שהתחבר לשרת מוקן. כעת, באמצעות המידע שצברנו במהלך כתיבת השרת שביצענו קודם לכן, תממשו בעצמכם את השרת בו השתמשתם בתרגיל הקודם.

זכור כי השרת משכפל כל מידע שתשלחו לו, ושולח אותו אליכם בחזרה, כמו היד. כך למשל, אם תכתבו אל השרת את המידע: "Hello, World", הוא יענה: "Hello, World"



כעת אתם מצוידים בכל המידע הדרוש לכם על מנת לפתור את התרגיל. בהצלחה!

Scapy ב-UDP

כעת נלמד כיצד לשלוח חבילות UDP באמצעות Scapy.



תרגיל 6.10 מודרך - שליחת שאלות DNS באמצעות Scapy

בתרגיל זה נשלח בעצמו שאלת DNS באמצעות Scapy. ראשית, פתחו את Scapy. כעת, נתחילה מבנות חבילה של DNS. נסתכל על מבנה החבילה:

```
>>> DNS().show()
```

```
C:\> C:\Windows\system32\cmd.exe - scapy
>>> DNS().show()
###[ DNS ]###
id= 0
qr= 0
opcode= QUERY
aa= 0
tc= 0
rd= 0
ra= 0
z= 0
rcode= ok
qdcount= 0
ancount= 0
nscount= 0
arcount= 0
qd= None
an= None
ns= None
ar= None
>>>
```

על מנת לבנות חבילת שאליתא ב-DNS, علينا לציין כמה שאילתות אנו שולחים. שדה זה נקרא 'qdcount'. לכן, ניצר את חבילת ה-DNS כאשר בשדה זה ישנו הערך 1, המציין לנו שולחים שאילתא אחת:

```
>>> dns_packet = DNS(qdcount = 1)
>>> dns_packet.show()
```

```
cat C:\Windows\system32\cmd.exe - scapy
>>> dns_packet = DNS(qdcount=1)
>>> dns_packet.show()
###[ DNS ]###
id= 0
qr= 0
opcode= QUERY
aa= 0
tc= 0
rd= 0
ra= 0
z= 0
rcode= ok
qdcount= 1
ancount= 0
nscount= 0
arcount= 0
qd= None
an= None
ns= None
ar= None
>>>
```

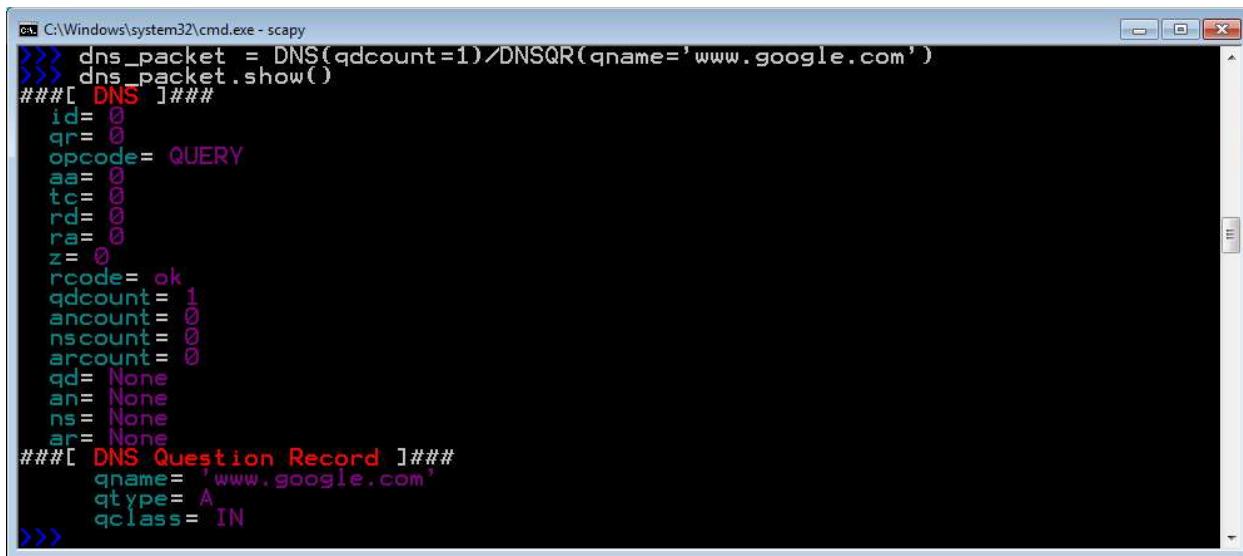
כעת علينا לבנות את השאליתא. ראשית נסתכל על הדרך שבה Scapy מציג השאילתא:

```
>>> DNSQR().show()
```

```
cat C:\Windows\system32\cmd.exe - scapy
>>> DNSQR().show()
###[ DNS Question Record ]###
qname=
qtype=A
qclass=IN
>>>
```

כפי שנitin לראות, Scapy מניח בעצמו שהשאליתא היא מסוג A, כלומר מיפוי של שם דומיין לכתובת IP. מכאן שעلينו לשנות רק את שם הדומיין, שהוא בשדה `qname`:

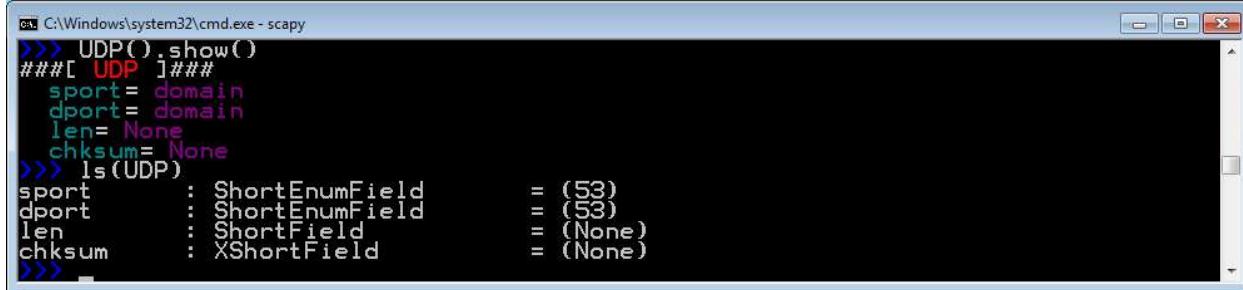
```
>>> dns_packet = DNS(qdcount=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
```



```
C:\Windows\system32\cmd.exe - scapy
>>> dns_packet = DNS(qdcount=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
###[ DNS ]###
 id= 0
 qr= 0
 opcode= QUERY
 aa= 0
 tc= 0
 rd= 0
 ra= 0
 z= 0
 rcode= ok
 qdcount= 1
 ancount= 0
 nscount= 0
 arcount= 0
 qd= None
 an= None
 ns= None
 ar= None
###[ DNS Question Record ]###
 qname= 'www.google.com'
 qtype= A
 qclass= IN
>>>
```

עת ברשותנו יש חיבורו שモרכבת משכבה DNS בלבד. על מנת לשולח אותה, נצטרך להרכיב גם את השכבות התתונות. נתחיל מלהרכיב את שכבת ה-UDP. על מנת לעשות זאת, נבחר להשתמש ב-53 כפורט יעד (מכיוון שהוא הפורט המשויך ל-DNS), ובחר בפורט מקור כרצונו, לדוגמה: 24601. ראשית, נסתכל על הדרך בה Scapy קורא לשדות השונים של UDP. נוכל לעשות זאת באמצעות המטודה **show** על חיבור UDP כלשהו, או באמצעות הפקודה **ls**:

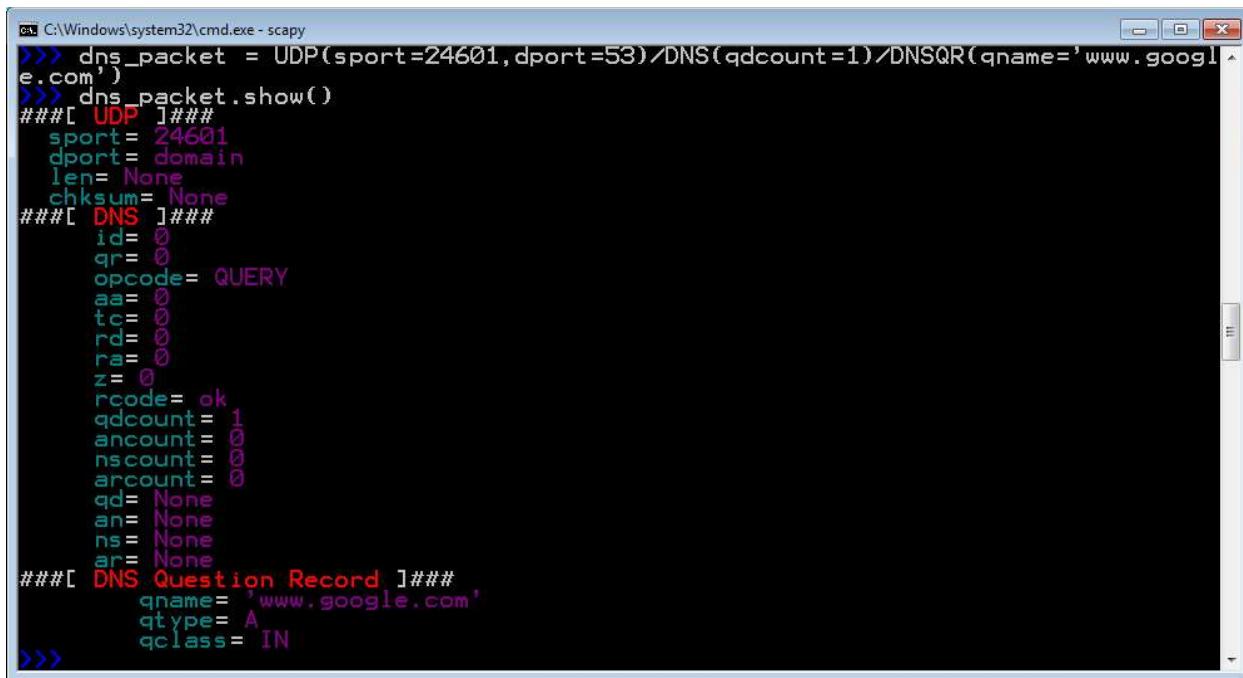
```
>>> UDP.show()
>>> ls(UDP)
```



```
C:\Windows\system32\cmd.exe - scapy
>>> UDP().show()
###[ UDP ]###
 sport= domain
 dport= domain
 len= None
 checksum= None
>>> ls(UDP)
sport : ShortEnumField      = (53)
dport : ShortEnumField     = (53)
len   : ShortField          = (None)
checksum : XShortField     = (None)
>>> _
```

עכשו ניתן את החיבור:

```
>>> dns_packet = UDP(sport=24601,
dport=53)/DNS(qdcount=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
```



```

C:\Windows\system32\cmd.exe - scapy
>>> dns_packet = UDP(sport=24601,dport=53)/DNS(qdcount=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
###[ UDP ]###
sport= 24601
dport= domain
len= None
checksum= None
###[ DNS ]###
id= 0
qr= 0
opcode= QUERY
aa= 0
tc= 0
rd= 0
ra= 0
z= 0
rcode= ok
qdcount= 1
ancount= 0
nscount= 0
arcount= 0
qd= None
an= None
ns= None
ar= None
###[ DNS Question Record ]###
qname= 'www.google.com'
qtype= A
qclass= IN
>>>

```

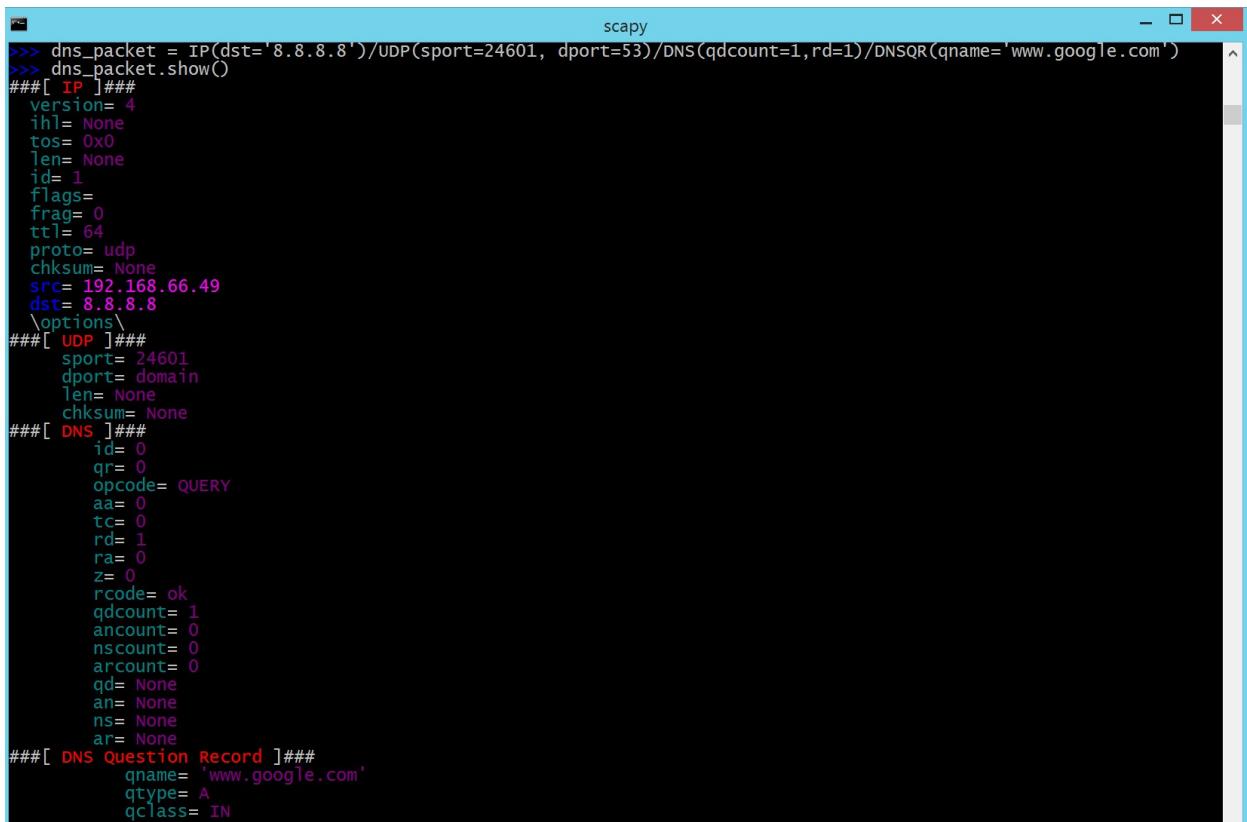
שים לב - לא הצמנו אף ערך בשדות האורך (שנקרא על ידי Scapy בשם **len**) וה-Checksum (שנקרא על ידי Scapy בשם **checksum**). אל דאגה, ערכים אלו יתملאו באופן אוטומטי כאשר החבילה תישלח!

כעת עליינו להחליט לאיזו כתובת IP לשלוח את החבילה. לצורך התרגיל, נשלח לכתובת "8.8.8.8", שמשמשת שרת DNS באינטראנט. נבנה את החבילה המלאה:

```

>>> dns_packet = IP(dst='8.8.8.8')/UDP(sport=24601,
dport=53)/DNS(qdcount=1,rd=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()

```



```

scapy
>>> dns_packet = IP(dst='8.8.8.8')/UDP(sport=24601, dport=53)/DNS(qdcount=1,rd=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
###[ IP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= udp
chksum= None
src= 192.168.66.49
dst= 8.8.8.8
\options\
###[ UDP ]###
sport= 24601
dport= domain
len= None
checksum= None
###[ DNS ]###
id= 0
qr= 0
opcode= QUERY
aa= 0
tc= 0
rd= 1
ra= 0
z= 0
rcode= ok
qdcount= 1
ancount= 0
nscount= 0
arcount= 0
qd= None
an= None
ns= None
ar= None
###[ DNS Question Record ]###
qname= 'www.google.com'
qtype= A
qclass= IN

```

בטרם נשלח את החבילה, פיתחו את Wireshark והריצו הסניפה עם המסן dns. כעת, שילחו את החבילה:

```
>>> send(dns_packet)
```

אתם צפויים לראות את חבילת השאלה, כמו גם התשובה שהגיעה מהשרת:

Filter: dns						Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info			
9	1.39747000	192.168.14.51	8.8.8.8	DNS	74	Standard query 0x0000 A www.google.com			
10	1.46074700	8.8.8.8	192.168.14.51	DNS	330	Standard query response 0x0000 A 212.179.180.121 A 212.179.180.123			

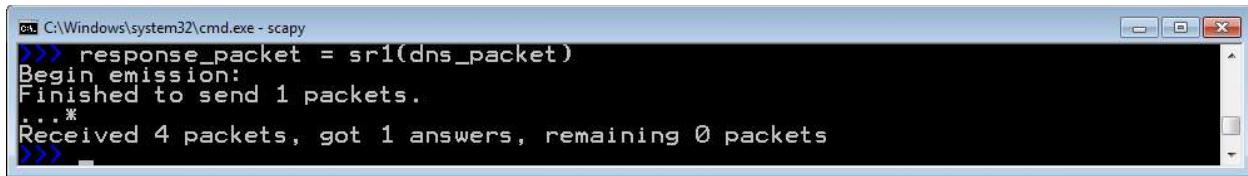
תרגיל 6.11 מודרך - קבלת תשובה לשאלת DNS באמצעות Scapy



از הצלחנו לשלוח שאלה של DNS לשרת המרוחק, וגם ראיינו Wireshark שההודעה נשלחה כמו שצריך ואף התקבלה תשובה. אך עכשו נרצה להצליח לקבל את התשובה באמצעות Scapy. ישנו מספר דרכים לעשות זאת, ובשילוב זה נלמד דרך שהיא שימוש בפונקציה **sr** המשמשת לשילוח חבילה אחת וקבלת תשובה עליה.

השתמשו באותה חבילת השאלה שיצרנו קודם לכן, ושילחו אותה. אך הפעם, במקום להשתמש ב-**send**, השתמשו בפונקציה **sr1**, ושמרו את ערך החזרה שלה. פונקציה זו תשלח את החבילה, וזה תסניף את הרשות (כמו הפקודה **sniff**), ותשמר את התשובה לחבילה שנשלחה. עשו זאת כך:

```
>>> response_packet = sr1(dns_packet)
```

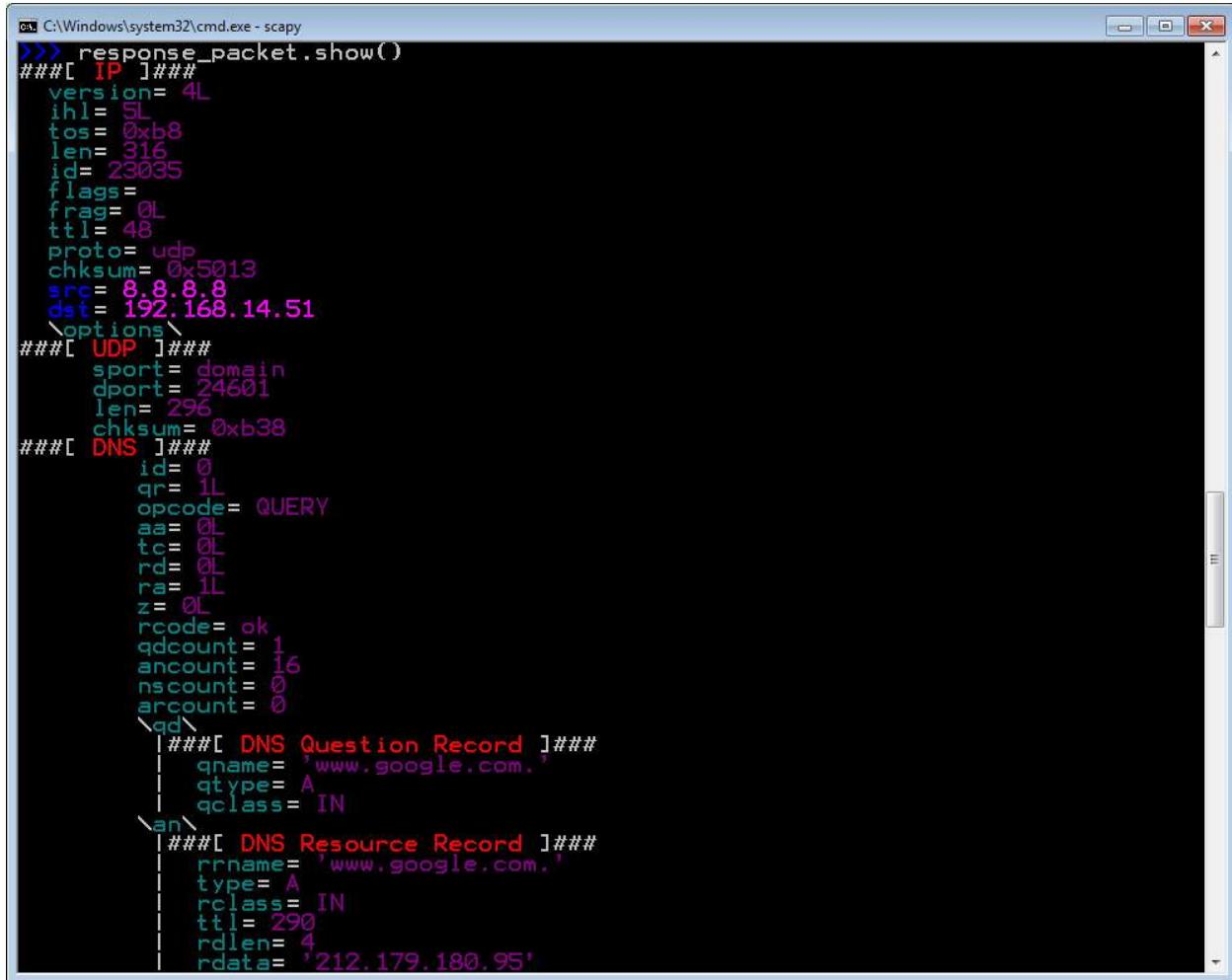


```
C:\Windows\system32\cmd.exe - scapy
>>> response_packet = sr1(dns_packet)
Begin emission:
Finished to send 1 packets.
...
Received 4 packets, got 1 answers, remaining 0 packets
>>>
```

שימוש לב לשורות התחתיונות. Scapy הציג, תוך כדי ריצה, שלוש נקודות (*). כל נקודה צו היא פקטה ש-Scapy הסניף שלא הייתה קשורה לפקטה שלחנו (כלומר לא תשובה לשאלת ה-DNS שלחנו קודם לכן). הכווצית היא פקטה שן קשורה (כלומר פקטת התשובה לשאלתא שלחנו). לסיום מסכם ואומר זאת במילים – "קיבلت 4 פקודות, 1 מהן הייתה פקחת תשובה. יש 0 פקודות שעדיין מחכות לתשובה". Scapy מציין שיש 0 פקודות שמחכות לתשובה מכיוון שהפעם שלחנו חבילת שאלה אחת בלבד. יש פונקציות אחרות המאפשרות לשלוח יותר משאלת שאלה אחת בכל פעם.

כעת, נוכל להסתכל על התשובה של שרת ה-DNS:

```
>>> response_packet.show()
```



```
C:\Windows\system32\cmd.exe - scapy
>>> response_packet.show()
###[ IP ]###
version= 4L
ihl= 5L
tos= 0xb8
len= 316
id= 23035
flags=
frag= 0L
ttl= 48
proto= udp
chksum= 0x5013
src= 8.8.8.8
dst= 192.168.14.51
\options\
###[ UDP ]###
sport= domain
dport= 24601
len= 296
chksum= 0xb38
###[ DNS ]###
id= 0
qr= 1L
opcode= QUERY
aa= 0L
tc= 0L
rd= 0L
ra= 1L
z= 0L
rcode= ok
qdcount= 1
ancount= 16
nscount= 0
arcount= 0
\qd\
###[ DNS Question Record ]###
qname= 'www.google.com.'
qtype= A
qclass= IN
\an\
###[ DNS Resource Record ]###
rrname= 'www.google.com.'
type= A
rclass= IN
ttl= 290
rdlen= 4
rdata= '212.179.180.95'
```

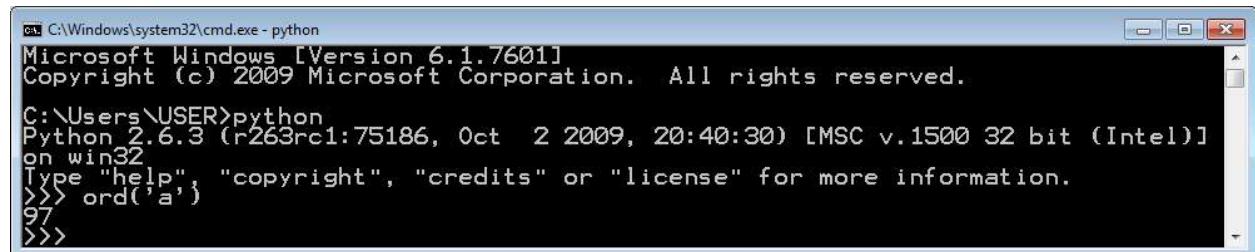
תרגיל 6.12 - תשאל שרת DNS באמצעות Scapy

עד כה ייצרנו ביחד שאלת DNS, שלחנו אותה אל השרת וקיבלנו את התשובה. כעת, כתבו סקריפט אשר מקבל מהמשתמש את הדומיין שלו והוא רוצה לשאול, ומדפיס את כתובת-h-IP הרלוונטי. לדוגמה, אם המשתמש יזין את הכתובת "www.google.com", על הסקריפט להדפיס את כתובת-h-IP הרלוונטי (למשל - 212.179.180.95⁴⁰). במידה שמוזרת יותר מתשובה אחת, הדפיסו רק את כתובת-h-IP הראשונה.

תרגיל 6.13 - תקשורת סודית מעל מספרי פורט

בתרגיל זה עלייכם לסייע לשני תלמידים, יואב ומאור, לתקשר בצורה סודית מעל הרשת. מטרת התלמידים היא להצליח להעביר מסרים אחד לשני, מבל' שאף אדם יוכל לקרוא אותם, גם אם הוא יכול להסניף את התעבורה ביניהם.

התלמידים החליטו על הפיתרון הבא: על מנת להעביר אותות ביניהם, הם ישלחו הודעה ריקה למספר פורט שמסמל אותה, כמשמעותו לפי קידוד ASCII (לקראת נוספת <http://en.wikipedia.org/wiki/ASCII>). לדוגמה, נאמר שיואב רוצה לשולח למאור את האות 'a'. לשם כך, עליוראשית להבין מה הערך ה-ASCII שלה. בכך, לעומת זאת, הוא יכול להשתמש בפונקציה `ord` של פירטו:



```

C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>python
Python 2.6.3 (r263rc1:75186, Oct  2 2009, 20:40:30) [MSC v.1500 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ord('a')
97
>>>

```

כעת כשียวאב גיליה שערך ה-ASCII של האות 'a' הוא 97, הוא ישלח הודעה UDP ריקה לפורט 97 של מאור. במידה שיואב ירצה להעביר למאור את ההודעה "Hello", עליו יהיה לשולח הודעה ריקה לפורט 72 (הערך של

⁴⁰ במהלך העבודה על התרגיל, יתכן שתתקלו בשגיאה שנובעת מ-Bug בחרבילה Scapy. במקרה זה, אתם צפויים לראות את כל מידע DNS בתווך שכבת Raw (בדומה לאופן שבו Scapy מציג חבילות HTTP) ולא בתווך שכבת DNS נפרדת. על מנת לפתור את הבעיה, אנא בצעו את השלבים הבאים:

(1) היכנסו לתיקייה שבה הותקן Python, ובו לתיקייה `.Lib\site-packages\scapy`.

לדוגמה: `C:\Python26\Lib\site-packages\scapy`.

(2) פתחו לעריכה את הקובץ `py.pyc`.

(3) בשורה ה-66, החליפו את "return inet_ntoa(addr)" ב- "return socket.inet_ntoa(addr)".

(4) מחקו את `py.pyc`.

הטו 'H'), לאחר מכן לשלוח הודעה לפורט 101 (הערך של התו 'e'), שתי הודעות ריקות לפורט 108 (הערך של התו 'I') ולבסוף הודעה ריקה לפורט 111 (הערך של התו 'o').

בתרגיל זה עלייכם למש את הסקורייפים בהם ישתמשו יואב ומאור בכך להעביר מסרים זה לזה:

- כתבו סקריפט בשם `secret_message_client.py`. הסקריפט יבקש מהמשתמש להקליד הודעה, ולאחר מכן ישלח אותה אל השרת באופן סודי, כפי שתואר לעללה. את כתובות ה-`IP` של השרת אתם יכולים לכלול בקוד שלכם בכלם באופן קבוע ולא לבקש אותה מהמשתמש. השתמשו ב-`Scapy` בכך לשלוח את החבילות.
- כתבו סקריפט בשם `secret_message_server.py`. הסקריפט ידפיס למסך מידע שהוא הבין כתוצאה שליחחה של הסקריפט `secret_message_client.py`. השתמשו ב-`Scapy` בכך להסניף ולקבל את החבילות.

שיםו לב שעל מנת לבדוק את תרגיל זה, עלייכם להשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת⁴¹.

בונוס: כפי שלמדתם, פרוטוקול UDP אינו מבוסס קישור, ולכן יתכן שחלק מהמידע ששלחו מלקוח לשרת לא יגיע, או לחלופין יגיע בסדר הלא נכון. חשבו כיצד ניתן להתגבר על בעיות אלו, וממשו פיתרון אמין יותר.

TCP - Transmission Control Protocol

TCP הינו פרוטוקול שכבת התעבורה הנפוץ ביותר באינטרנט לחיבורים מבוססי קישור. כאמור, בתור מפתחי שכבת האפליקציה, משתמשים ב-TCP בכך להעביר מידע, איננו יכולים פשטוט לשולח חבילה אל תוכנה מרוחקת. ראשית علينا ליצור קישור עם התוכנה המרוחקת, ועתה כל חבילה שנשלחה תהיה חלק מאותו קישור. דבר זה דומה לשיחת טלפון: על מנת לדבר עם אדם אחר, אני יכול פשוט להגיד את הודעה שלי (למשל: "נפגש היום בשעה חמיש ליד בית הספר"). עלי' ראשית לחיג את המספר שלו, לשם צילח חיבור, ולהחות עד שירם את הטלפון ובכך יוקם בינינו קישור.

TCP תוכנן וועצב לרוץ מעל שכבת רשת שאינה אמינה. למשל, ההנחה הבסיסית היא שהשכבה הרשת חבילות יכולות ללקת לאיבוד או להגיא שלא בסדר הנכון. בתור פרוטוקול מבוסס קישור, TCP מבטיח לשכנת האפליקציה שהמידע יגיע אל היעד בסדר הנכון.

⁴¹ באופן תאורטי, ניתן לעשות זאת מעל `loopback device` – ככלומר מעל הכתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של `Scapy` בשליחת וקבלת מסגרות מעל `loopback device`-bs-`Windows`, נשתמש בשני מחשבים.



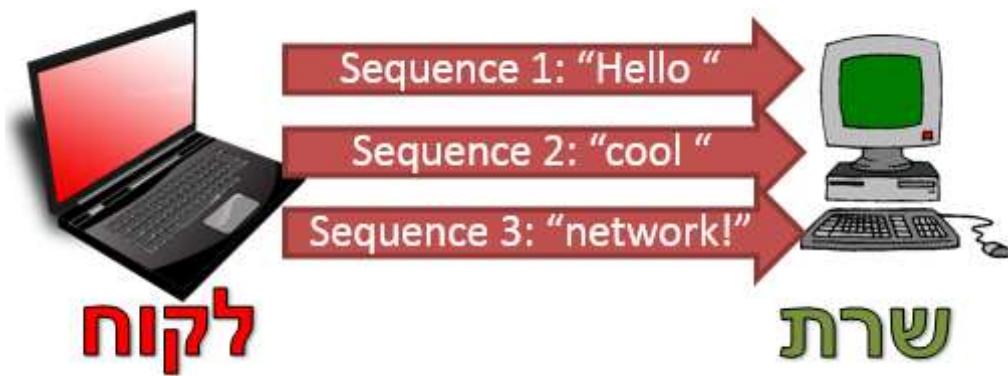
כיצד ניתן לוודא שהמידע מגיע אל היעד? כיצד ניתן לוודא שהוא מגיע בסדר הנכון?

על מנת לעשות זאת, TCP מנצל את העבודה שהוא פרוטוקול מבוסס קישור. מכיוון שכל החבילות (שנקראות בשכבת התעבורה בשם **סגמנטים**⁴²) הן חלק מקשר, אנו יכולים לבצע דברים רבים.

ראשית, אנו יכולים לתת מספר סידורי לחבילות שלנו. נאמר שבשכבת האפליקציה רצינו לשלוח את המידע "Hello cool network!". בשכבת התעבורה, נאמר שהמידע חולק לחבילות בצורה הבאה:

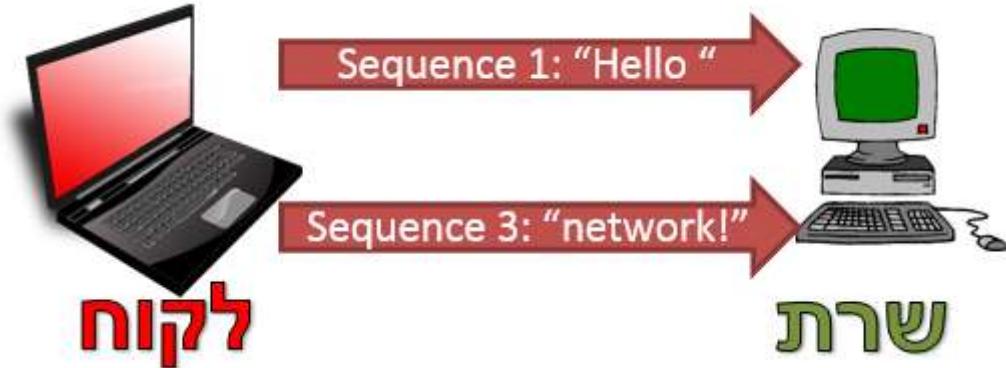
- חבילה מס' אחד - "Hello"
- חבילה מס' שני - "cool"
- חבילה מס' שלישי - "network!"

כעת נוכל לשלוח את החבילות כשלצדן יש מספר סידורי (Sequence Number):



כעת, נסתכל על צד השרת. נזכיר כי בשרת יתכן וחבילות מסוימות "נופלות" ולא מגיעות לייעדן. כך למשל, יתכן שחביבה מס' שני נפלה" בדרך, והשרת רואה מהלך רק שתי חבילות:

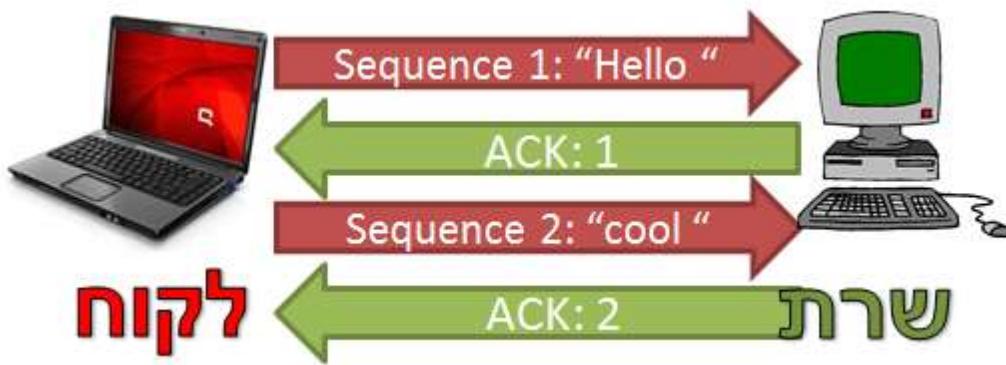
⁴² גושי מידע בשכבת התעבורה נקראים "סגמנטים". עם זאת, כל סגמנט הוא למעשה גם חבילה של שכבה השלישית (שמכילה בתוכה את השכבה הרביעית, בהתאם למודל השכבות). על כן, ניתן לומר שכל סגמנט הוא גם פקטה (מונח זה שיר לשכבת הרשת, שכבה השלישית) וניתן לקרוא לו כך.



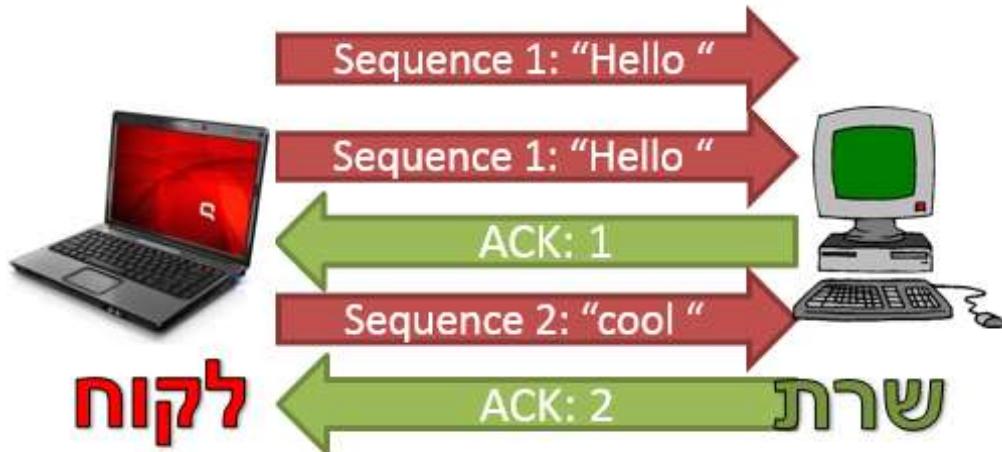
כעת, השרת יכול להבין ש.ssורה לו חבילה מסוימת שתיים! הוא יכול לעשות זאת מכיוון שהוא יודע שהחיבור הנוכחי בינו לבין הלוקוט, הוא קיבל את חבילה מסוימת וחבילה מסוימת שלוש, וכך הוא אמרור היה גם לקבל את חבילה מסוימת שתיים.

ניתן להשתמש במספרי החבילות כך לוודא שחבילה אכן הגיעה ליעדה. כך למשל, ניתן להחליט שעל כל חבילה שהגיעה, השרת ישולח אישור ללקוח. חבילה צזו נקראת בדרך כלל **ACK** (קיצור של **Acknowledgement**) ומשמעותה - "קיבלתי את החבילה שלך".

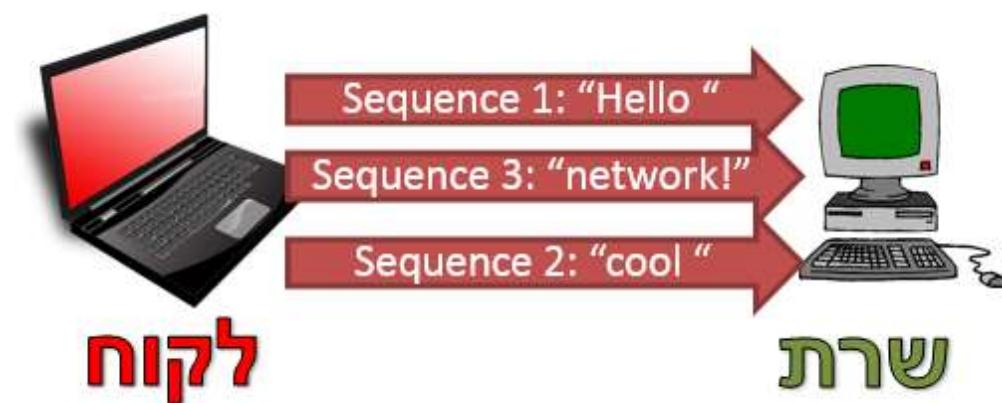
הלקוח יזכה לקבל ACK על כל חבילה אותה הוא שולח לשרת:



בצד הלקוח, אם לא התקבלה חבילת ACK מהשרת לאחר זמן מסוים, נראה שהחבילה שהוא שלח "נפלה בדרך". במקרה זה, החבילה תשלוח שוב:



כך הצלחנו להבטיח שהחבילות שלחנו באמצעות הגיעו ליעד!
השימוש במספר סידורי לחבילה מאפשר לנו להתמודד עם בעיות נוספות. בגלל שהרשות לא אמינה, יתכן
שהחבילות הגיעו לשרת בסדר לא נכון:



במקרה זה, חבילה מספר שלוש הגיעה לפני חבילה מספר שניים. עם זאת, מכיוון שהשרת רואה את המספר הסידורי של כל חבילה, הוא יכול לסדר אותן מחדש בסדר הנכון. שכבת האפליקציה לא תדע בכלל שהחבילות הגיעו במקור בסדר שונה מאשר הלקוקו התכוון.

השימוש ב-ACKים ובמספרים סידוריים מבטיח לנו אמינות: המידע שלחנו יגיע, וגם יתקבל בסדר הנכון. לשם כך הינו צריכים להרים קישור, ולשלוח את החבילות חלק מה קישור. בסיום הקישור, נרצה לסגור אותו.

איך TCP משתמש במספר סידורי?

למדנו את חשיבותם של מספרים סידוריים ו-ACKים. עצת נסביר את השימוש שלהם בפrotocol TCP.

פרוטוקול TCP לא נותן מספר סידורי לכל חבילה, אלא לכל בית (byte). כזכור, אנו מעבירים מצד לצד רצף של בתים. לכל אחד מהבתים ברצף יש מספר סידורי מסוון. בכל חבילה שנשלוח, יהיה המספר הסידורי שמצין את הבית הנוכחי בחבילה. כך למשל בדוגמה הבאה:



הערה: "Seq" משומש כקיצור ל-"Sequence Number".

הטו "H" הוא הבית בעל המספר הסידורי 100 בתקשרות בין הלקוח לשרת. "e" הוא בעל המספר 101, ה-"l" הראשונה היא מסטר 102, ה-"l" השנייה היא מסטר 103, "o" הוא מסטר 104 והרוווח שנמצא לאחריו הוא הבית בעל המספר הסידורי 105. מכיוון שהבית האחרון שנשלח היה הבית בעל המספר הסידורי 105, הבית הבא יהיה בעל המספר 106. לכן, המשך התקשרות יראה כך:



החבילה השנייה התחליה עם המספר הסידורי 106. המשמעות של כך היא שהבית הראשון שבhä, כלומר התו "c", הוא בעל המספר הסידורי 106. ה-"o" שלו הוא בעל המספר 107, וכך הלאה.

שימוש לבשתקשרות TCP היא למעשה שני Stream'ים של מידע: רצף בתים לכל צד. התקשרות שבין הלקוח לשרת מחייב רצף בתים בפני עצמה, וה-Sequence Number בכל מקטע מתיחס לרצף בין הלקוח לשרת בלבד, ולא לרצף שנשלח מהשרת אל הלקוח.

תרגיל 6.14 מודרך - צפייה ב-Squence Numbers של TCP

פתחו את Wireshark והריצו הסנפה. השתמשו במסנן התצוגה "http", כפי שLEARנו בפרק **שכבות האפליקציה**. פיתחו דףדף, וגילשו אל הכתובת: <http://www.ynet.co.il>. עצרו את ההסנפה, וביחרו באחת מהחbillות ה-HTTP. לחצו על החבילה באמצעות המקש הימני של העכבר, וביחרו באפשרות "Follow TCP Stream":

The screenshot shows the Wireshark interface with a list of network packets. A context menu is open over the 453 packet (HTTP GET /Ext/App/...). The menu items include: Mark Packet (toggle), Ignore Packet (toggle), Set Time Reference (toggle), Time Shift..., Edit or Add Packet Comment..., Manually Resolve Address, Apply as Filter, Prepare a Filter, Conversation Filter, Colorize Conversation, SCTP, Follow TCP Stream (which is highlighted with a red box), Follow UDP Stream, and Follow SSL Stream.

כעת Wireshark מציג בפנינו את התקשרות ביןינו לבין הרשת של Ynet, והוא מציג את קישור ה-TCP שבחרנו בלבד. עד סוף הפרק, נבין כיצד Wireshark יודיע לעשויות זאת.

ביחרו באחת החbillות שהשרת שלח אליכם, עדיף חבילה שאחריה נשלחה מיד עוד חבילה מהשרת:

The screenshot shows a detailed view of a selected TCP segment (Frame 226) from the previous list. The packet details pane shows the following information:

- Source:** 81.218.31.137
- Destination:** 192.168.14.51
- Protocol:** TCP
- Sequence Number:** 2958 (relative sequence number)
- Next Sequence Number:** 4318 (relative sequence number)
- Acknowledgment Number:** 1804 (relative ack number)
- Header Length:** 20 bytes
- Flags:** 0x010 (ACK)
- Window Size Value:** 9103
- Calculated Window Size:** 18206
- Window Size Scaling Factor:** 2
- CHECKSUM:** 0x453f [validation disabled]
- [SEQ/ACK analysis]:** [reassembled PDU in frame 221]
- TCP Segment Data:** (1360 bytes)

בדוחן ניתן לראות את ה-Sequence Number הנוכחי, קלומר מהו המספר של הבית הראשון בסegment זה, והוא **בדוגמה** שלמו. **בכחול** ניתן לראות את גודל המידע של הסegment הנוכחי - **1,360** בתים.

באמצעות שני נתונים אלו, המספר הסידורי של הבית הנוכחי, וכמות הבטים שנשלחים בסוגמנט הנוכחי - נוכל לחשב את המספר הסידורי של הסוגמנט הבא! נעשה זאת יחס:

$$\textcolor{red}{2,958} + \textcolor{blue}{1,360} = \textcolor{green}{4,318}$$

גם Wireshark מציין לנו שזה יהיה המספר הסידורי הבא (תחת הסעיף [Next sequence number]). נמשיך לבדוק זאת בעצמנו. נבחר את החבילת הבאה ונראה מה המספר הסידורי שלה:

Wireshark screenshot showing a sequence of TCP segments. The selected segment (Frame 227) has its sequence number highlighted in green. The details pane shows the sequence number as 4318.

```

Frame 227: 1414 bytes on wire (11312 bits), 1414 bytes captured (11312 bits) on interface 0
Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
Internet Protocol Version 4, Src: 81.218.31.137 (81.218.31.137), Dst: 192.168.14.51 (192.168.14.51)
Transmission Control Protocol, Src Port: http (80), Dst Port: 54768 (54768), Seq: 4318, Ack: 1804, Len: 1360
Source port: http (80)
Destination port: 54768 (54768)
[stream index: 8]
Sequence number: 4318 (relative sequence number)
[Next sequence number: 5678 (relative sequence number)]
Acknowledgment number: 1804 (relative ack number)
Header length: 20 bytes
Flags: 0x10 (ACK)
Window size value: 9103
[calculated window size: 18206]
>window size scaling factor: 2]
Checksum: 0x303f [validation disabled]
[SEQ/ACK analysis]
[Reassembled PDU in frame: 331]
TCP segment data (1360 bytes)

```

בירוק אנו רואים שהמספר הסידורי הוא אכן **4,318**, כמו שהישבנו קודם לכן.

נסו לעשות את החישוב הזה גם על חבילות נוספות.



איך TCP משתמש ב-Acknowledgement Numbers?

היות שהמספרים הסידוריים של TCP מתייחסים לבטים (bytes) ברכף המידע, כך גם מספרי ACK. מספר ה-ACK ב-TCP מציין את המספר הסידורי של הבית הבא שצפוי להתקבל. כך למשל, בדוגמה הקודמת שלנו:



ציינו שהבית הבא שאמור להשלוח מהלקוח יהיה בעל המספר הסידורי 106. אי' לך, ה-ACK אמור להכיל את הערך 106:



בצורה זו קל מאד לבצע מעקב אחרי התקשרות. מכיוון שה-ACK מכיל את המספר הסידורי הבא, הרי שזה יהיה המספר הסידורי שישלח בחבילת המידע הבאה. כך בדוגמה זו, רצף החבילות יראה כדלקמן:



בנוסף, כאשר נשלח ACK ב-TCP, הכוונה היא שכל המידע שהגיע עד לבית שמצוין ב-ACK הגיע נכון. כך לדוגמה, במקרה לעיל השרת יכול היה לא לשЛОח ACK עבור הabiliaה שכילה את המידע "Hello", אלא רק לאחר קבלת הabiliaה שכילה את המידע "cool". במקרה זה, ערך ה-ACK צריך להיות המספר הסידורי הבא -

והוא יהיה 110 (שכן הוא כולל את ערך הבית הראשון בחבילת השניה, שהוא 106, ובנוסף גודל החבילות - שהוא 4 בתים):



במקרה זה, הלקוח מבין שתי החבילות, הן זאת שמכילה את המידע "Hello", והן זאת שמכילה את המידע "cool", הגיעו כמו שצריך. זאת מכיוון שכשהשרת שלח ACK עם הערך 110, הוא למעשה אמר: " קיבלתי את כל הבתים עד הבית ה-110 בהצלחה".

לאחר שליחת החבילות אלו, הלקוח מזכה זמן מסוים לקבל ה-ACK. אם ה-ACK לא הגיע עד לסיום הזמן זהה, הוא שולח אותו מחדש.

תרגיל 6.15 מודרך - צפייה של TCP Acknowledgement Numbers

נסתכל שוב בחבילת האחורונה שליטה הסתכלנו באמצעות Wireshark:

Index	Time	Source	Destination	Protocol	Information
226	3.90828400	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
227	3.90907300	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
228	3.90910900	192.168.14.51	81.218.31.137	TCP	54 54768 > http [ACK] Seq=1804 Ack=5678 win=66
229	3.90987200	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
230	3.91065000	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
231	3.91069600	192.168.14.51	81.218.31.137	TCP	54 54768 > http [ACK] Seq=1804 Ack=8398 win=66
232	3.91146300	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
233	3.91225800	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
234	3.91226000	81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]

Frame 227: 1414 bytes on wire (11312 bits), 1414 bytes captured (11312 bits) on interface 0
 Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
 Internet Protocol Version 4, Src: 81.218.31.137 (81.218.31.137), Dst: 192.168.14.51 (192.168.14.51)
 Transmission Control Protocol, Src Port: http (80), Dst Port: 54768 (54768), Seq: 4318, Ack: 1804, Len: 1360
 Source port: http (80)
 Destination port: 54768 (54768)
 Stream index: 81
 Sequence number: 4318 (relative sequence number)
 Next sequence number: 5678 (relative sequence number)
 Acknowledgment number: 1804 (relative ack number)
 Header length: 20 bytes
 Flags: 0x010 (ACK)
 Window size value: 9103
 Calculated window size: 18206
 Window size scaling factor: 2
 Checksum: 0x303f [validation disabled]
 SEQ/ACK analysis
 Reassembled PDU in frame: 3311
 TCP segment data (1360 bytes)

נחשב את המספר הסידורי של החבילות הבאה, כפי שלמדנו לעשות קודם לכן:

$$4,318 + 1,360 = 5,678$$

מכאן שהמזהה הסידורי של הבית הבא אמור להיות 5,678. כפי שלמדנו זה עתה, זה צפוי להיות גם הערך של חבילת ה-ACK. בואו נבחן זאת בחבילת ACK הרלבנטית:

227 3.90907300 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
228 3.90910900 192.168.14.51	81.218.31.137	TCP	54 54768 > http [ACK] Seq=1804 Ack=5678 Win=66640 Len=0
229 3.90987200 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
230 3.91065000 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
231 3.91069600 192.168.14.51	81.218.31.137	TCP	54 54768 > http [ACK] Seq=1804 Ack=8398 Win=66640 Len=0
232 3.91146300 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
233 3.91225800 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]
234 3.91226000 81.218.31.137	192.168.14.51	TCP	1414 [TCP segment of a reassembled PDU]

```

Frame 228: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 81.218.31.137 (81.218.31.137)
Transmission Control Protocol, Src Port: 54768 (54768), Dst Port: http (80), seq: 1804, Ack: 5678, Len: 0
    Source port: 54768 (54768)
    Destination port: http (80)
    [Stream index: 8]
    Sequence number: 1804 (relative sequence number)
    Acknowledgment number: 5678 (relative ack number)
    Header length: 20 bytes
    Flags: 0x010 (ACK)
        window size value: 16660
        [calculated window size: 66640]
        [window size scaling factor: 4]
    Checksum: 0x4059 [validation disabled]
    [SEQ/ACK analysis]

```

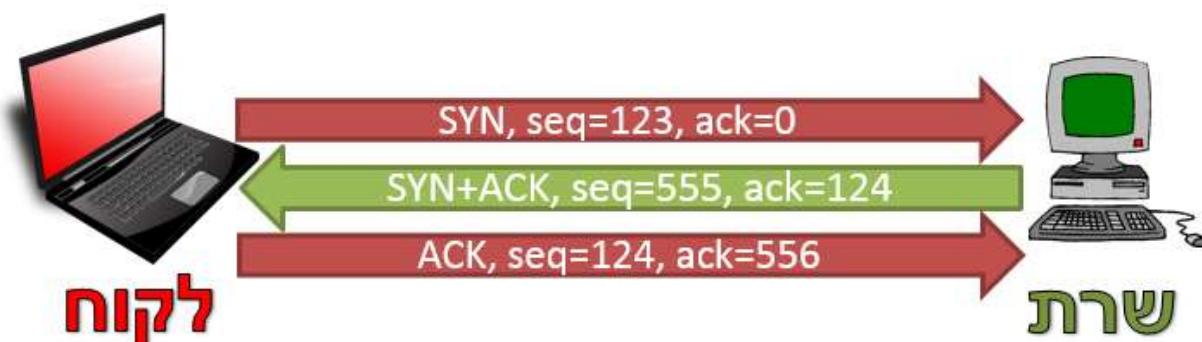
כמו שניתן לראות, אכן התקבל הערך הצפוי.

נסו עתה לחשב בעצמכם את ערכי ה-ACK שצפויים להתקבל עבור חבילות נוספות שנקלטו בהסכמה שלכם, ומצאו אותם. שימו לב כי לא בהכרח נשלחת חבילת ACK עבור כל חבילת מידע.



הקמת קישור ב-TCP

כשדיברנו על פרוטוקולים מבוססי קישור, חזרנו על כך שיש צורך להקם את הקישור בין הצדדים לפני שלב העברת המידע ביניהם. באמצעות הקמת הקישור, אנו מודיעים לצד השני שאנחנו מולו בתקשורת וublisherו יהיה מוכן לכך. בנוסף, לעיתים יש לאמת פרמטרים בין שני הצדדים בצד אחד שה קישור יעבד בצורה יעילה יותר. באופן כללי, הקמת קישור ב-TCP נקראת **Three Way Handshake** (לחיצת יד משולשת), ונראית כך:



כפי שניתן לראות, במהלך הרכבת הקישור נשלחות שלוש חבילות. ישנו שימוש בשדות Sequence Number וה-Acknowledgement Number של כל חבילה כדי להציג להרים את הקישור.icut, נבין את התפקיד של כל חבילה ואת האופן בו מוחשבים הערכים בשדות האלו.

חביבה ראשונה - SYN

בשלב הראשון, הלוקו שולח לשרת חבילה שמטרתה להתחיל את הקמת הקישור. באופן זה, הלוקו מצין: "אני רוצה להקים קישור מולך". בכל חבילת TCP יש כמה דגלים שניתן לצין, חלק מה-Header⁴³. בחברה זו, הדגל SYN דלוק. משמעות הדגל SYN היא תחילת תקשורת. Sequence Number של חבילה זו הינו ה-Initial Sequence Number ההתחילתי של הלוקו עבור הקישור זהה עם השרת, ונקרא בשם Sequence Number (Number) ISN.



איך נבחר ה-Initial Sequence Number?

ניתן היה להסכים שה-ISN, אותו מספר התחילתי עבור הקישור, יהיה תמיד ערך קבוע - כגון 0. דבר זה יכול להקל מאוד על הבנת התקשורות. למשל, הבית עם המספר הסידורי 0 יהיה תמיד הבית הראשון בתקשורת, הבית עם המספר הסידורי 1 יהיה הבית השני בתקשורת וכך הלאה.

עם זאת, ה-ISN נבחר באופן רנדומלי. הסיבה העיקרית לכך היא למניעת התנגשויות של חיבורים. נדמיין לעצמנו מצב שבו כל החיבורים היו מתחילה עם המזהה 0. נאמר שהлокו שלח לשרת חבילה עם המספר הסידורי 100. במידה שהקישור בין הлокו לשרת נפל (למשל, מכיוון שהייתה שגיאה אצל הлокו או אצל השרת), יקום חיבור חדש אף הוא עם המזהה 0. אז עשויה להגיע חבילה מהחיבור הקודם ליעדה, והשרת יחשוף אותה לחביבה מהחיבור החדש. אי לכך, על מנת למנוע מקרים כאלה, נבחר בכל פעם מספר באופן רנדומלי.

בדוגמה שלנו, המספר הסידורי שנבחר הינו 123. דגל ACK של החביבה הראשונה כבוי, שהרי לא ניתן ACK על אף חבילה קודמת.

⁴³ במודון זה, דגל הינו בית שמצוין אפשרות מסוימת. הסבר על כל הדגלים קיימם ב[נספח א' TCP Header](#).

בשלב זה, התקשרות נראה כך:

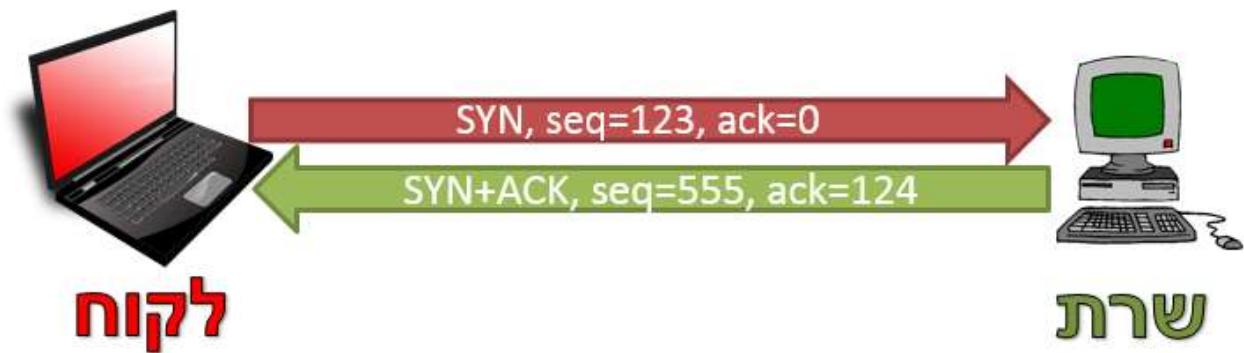


חכילה שנייה - SYN + ACK

בשלב השני, בהנחה שהשרת הסכים להקים את הקישור, הוא עונה בחכילה בה דלוקים שני הדגלים: SYN ו-ACK. הדגל SYN דלוק מכיוון שהוא שזוע חכילה שמודיעה על הקמה של קישור. הדגל ACK דלוק מכיוון שהשרת מודיע על רקוח שהוא קיבל את החכילה הקודמת שהוא שלח, שהוא חכילת ה-SYN.

ה-Sequence של החכילה של השרת יהיה ה-*ISN* של התקשרות בינו לבין הלקוח. כלומר, יתאר את המספר הסידורי ההתחלתי של הבטים שנשלחו מהשרת אל הלקוח. נציג שוב שתקשרות TCP היא למעשה שני Stream של מידע: רצף בתים לכל צד. המספר הסידורי של התקשרות של הלקוח (שמהחיל ב-123) מצין את המספר הסידורי של הבטים בין הלקוח לשרת, והמספר הסידורי שהשרת ישלח בשלב זה יתאר את המספר ההתחלתי של הבטים בינו לבין הלקוח. גם השרת ייגריל את ה-*ISN* באופן רנדומלי, מהסתיבות שתוארו קודם לכן. בדוגמה שלנו, המספר שנבחר הוא .555

בנוסף, על השרת לציין את מספר ה-ACK כדי להודיע ללקוח שהוא קיבל את החכילה שלו. כפי שהסבירנו קודם, ה-ACK מציין את המספר הסידורי של הבית הבא שצפוי להגיע. במקרה של חכילת SYN, החכילה נספרת בגודל של בית אחד (על אף שלא נשלח שום מידע). כלומר,ערך ה-ACK יהיה המספר הסידורי של החכילה שהלקוח שלח (בדוגמה שלנו, 123) ועוד 1 עבור ה-ACK. מכאן שערך ה-ACK יהיה 124. כך נראה התקשרות בשלב זה:



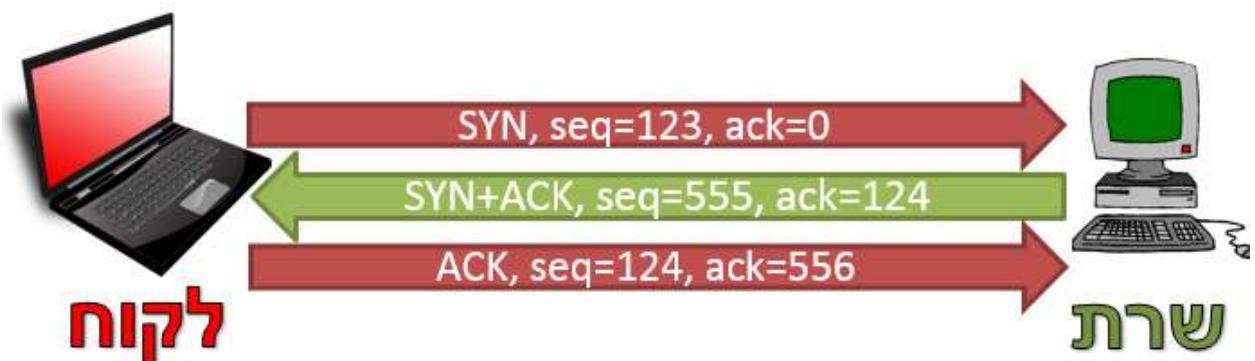
חבייה שלישית - ACK

על מנת שהקשר יוקם בהצלחה, על השרת לדעת שהחבייה הקודמת שהוא שלח, חביתה ה-SYN+ACK הגיעה אל הליקוּחַ בהצלחה. אם החבייה אכן הצלחה להגיע, גם הליקוּחַ וגם השרת יודעים שהקשרו קם, הסכימו להתחילה אותו, וכן מסונכרנים על המספרים הסידוריים הראשונים (כלומר ה-*N*-SN) אחד של השני.

על מנת לעשות זאת, הליקוּחַ שלוח חבייה כshedagl ACK דלוק, ומספר ה-ACK מצין את הבית הבא שהוא מ暢פה לקבל מהשרת. הבית הבא מחושב על ידי שימוש במספר הסידורי שהשרת שלח (במקרה שלנו - 555) ועוד 1 עבור הבית של SYN. מכאן שבדוגמה שלנו, הערך יהיה 556.

שים לב שהdagel SYN כבוי, שכן זו כבר לא החבייה הראשונה שנשלחת מלהיקוּחַ לשרת בקשר הנוכחי.

כמובן מלהיקוּחַ צריך גם לכלול את המזהה הסידורי של הבית שהוא שלח, כמו בכל חבייה של TCP. הערך זהה הינו הערך שהוא ב-ACK של החבייה שהתקבל מהשרת, לחושב באמצעות לקיחת המספר הסידורי הראשוני (123) והוספת 1 עבור ה-*N*-SYN. מכאן שהמספר הסידורי הוא 124:



בשלב זה הוקם קשר קייזר בין הליקוּחַ לשרת, ועכשו ניתן לשלוח מעליוּ חביות מיד!

תרגיל 6.16 מודרך - צפייה ב-Three Way Handshake של TCP



פתחו את Wireshark והריצו הסנפה. גילשו שוב אל האתר `http://www.bewan.com`, בזמן שטמון התצוגה שלכם הוא "http://www.bewan.com". השתמשו שוב ב-`Follow TCP Stream` כדי לראות קישור TCP יחיד. כתע נתמקד יחד בחבילה הראשונה של הקישור:

```

Frame 211: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 81.218.31.137 (81.218.31.137)
Transmission Control Protocol, Src Port: 54768 (54768), Dst Port: http (80), Seq: 0, Len: 0
    Source port: 54768 (54768)
    Destination port: http (80)
    [Stream index: 8]
    Sequence number: 0 (relative sequence number)
    Header length: 32 bytes
    Flags: 0x002 (SYN)
        Window size value: 8192
        [calculated window size: 8192]
    Checksum: 0x4065 [validation disabled]
    Options: (12 bytes), Maximum segment size, No-Operation (NOP), window scale, No-Operation (NOP), No-operation (NOP), SACK permitted
0000  00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00  ....c.. *..E.
0010  00 34 21 bf 40 00 80 06 00 00 c0 a8 0e 33 51 da  .4!@.6. ..Q.....
0020  1f 89 d5 f0 00 50 fa ba de bb 80 12  3.P..... .
0030  20 00 40 65 00 00 02 04 03 b4 01 03 03 02 01 01  ..@. .....
0040  04 02

```

כפי שניתן לראות בכותרת, הדגל הדלוק בחבילה הוא דגל SYN, שמצוין הקמת חיבור. ה-Sequence Number קבוע באדום, ומתקבל את הערך 0. עם זאת, Wireshark מצוין לנו כי זהו ערך ייחודי. ככלומר, Wireshark מזהה עבורנו שמדובר ב-ISN של השיחה ולכן נותן לו את הערך 0. על מנת לראות את הערך האמיתי, נוכל ללחוץ על השדה זהה, וכעת Wireshark יציג לנו גם בשדה המידע של החבילה את הערך האמיתי (מוסמן בירוק). בדוגמה שלנו, ה-ISN הוא 0xfabadeba בסיס הקסיה-דצימלי.

בנוסף, ניתן לראות בכחול את שדה ה-Options. אלו הן אפשרותויות שיופיעו על כל המשך הקישור, ויש לציין אותן כבר בשלב הקמת הקישור. לא עמוק עליון בשלב זה.

כעת נסתכל בחבילה הבאה:

```

Frame 212: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
Internet Protocol Version 4, Src: 81.218.31.137 (81.218.31.137), Dst: 192.168.14.51 (192.168.14.51)
Transmission Control Protocol, Src Port: http (80), Dst Port: 54768 (54768), Seq: 0, Ack: 1, Len: 0
    Source port: http (80)
    Destination port: 54768 (54768)
    [Stream index: 8]
    Sequence number: 0 (relative sequence number)
    Acknowledgment number: 1 (relative ack number)
    Header length: 32 bytes
    Flags: 0x012 (SYN, ACK)
        Window size value: 14600
        [calculated window size: 14600]
    Checksum: 0x5aff [validation disabled]
    Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, N
        [SEQ/ACK analysis]
0000  d4 be d9 d6 0c 2a 00 0c c3 a5 16 63 08 00 45 00  ....*.. .c..E.
0010  00 34 00 00 40 00 36 06 01 86 51 da 1f 89 c0 a8  .4!@.6. ..Q.....
0020  0e 33 00 50 d5 f0 cc e1 1e 8c fa ba de bb 80 12  .3.P..... .
0030  39 08 5a ff 00 00 02 04 03 50 01 01 04 02 01 03  9.Z..... .P.....
0040  03 01

```

כפי שניתן לראות בכתום, הדגלים הדולקים בחבילה הם אלו של SYN ו-ACK. באדום, ניתן לראות כי גם הפעם Wireshark מציין כי ערך ה-ISN הוא 0, באופן ייחודי. אם נלחץ על שדה זה, Wireshark יציג לנו את הערך האמתי (בירוק), שבדוגמא שלנו הוא 0xcce11e8c בבסיס הקסיה-דצימלי. בכחול, אנו יכולים לראות את ערך ה-ACK. מכיוון ש-Wireshark מציג ערכים ייחודיים, הוא מציין לנו כי הערך הוא 1 (ערך הגadol ב-1 מה-ISN, שצוין כ-0). אם נלחץ על שדה זה, נראה שהוא ערך האמתי הינו 0xfabadebb בbasis הקסיה-דצימלי, ערך שאכן גדול ב-1 מהערך האמתי שנשלח.

לסיום נסתכל בחבילת ה-ACK שמסיימת את הרמת הקישור:

```

+ Frame 214: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on i
+ Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63
+ Internet Protocol version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 81.
+ Transmission Control Protocol, Src Port: 54768 (54768), Dst Port: http (8)
  Source port: 54768 (54768)
  Destination port: http (80)
  [Stream index: 8]
  Sequence number: 1      (relative sequence number)
  Acknowledgment number: 1    (relative ack number)
  Header length: 20 bytes
+ Flags: 0x010 (ACK)
  window size value: 16660
  [calculated window size: 66640]
  [window size scaling factor: 4]
+ Checksum: 0x4059 [validation disabled]
+ [SEQ/ACK analysis]

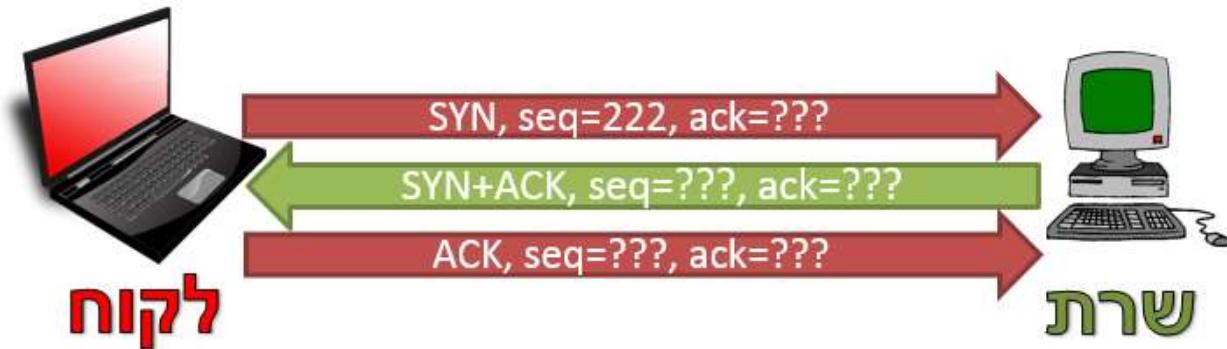
0000  00 0c c3 a5 16 63 d4 be   d9 d6 0c 2a 08 00 45 00  . ....c..  ....*..E.
0010  00 28 21 c0 40 00 80 06 00 00 c0 a8 0e 33 51 da  .(!..@.... 3Q.
0020  1f 89 d5 f0 00 50 fa ba de bb cc e1 1e 8d 50 10  ....P.....P.
0030  41 14 40 59 00 00 A.@@Y..
```

כפי שניתן לראות בכתום, הדגל הדולק הוא אכן דגל ה-ACK, ואף לא דגל אחר. באדום ניתן לראות את המספר הסידורי של הבית הנוכחי. כפי שכבר ציינו, Wireshark מציג אותו באופן ייחודי כ-1, ולהזיהה עליו תראה לנו בירוק כי ערך האמתי הינו 0xfabadebb בbasis הקסיה-דצימלי. בכחול ניתן לראות את ערך ה-ACK, שהוא באופן ייחודי 1, ולהזיהה עליו תגלה לנו כי הערך הוא d8c11e8d בbasis הקסיה-דצימלי, צפוי.



תרגיל 6.17 - חישובי Three Way Handshake

בشرطוט שלפניכם מוצגת לחיצת יד משולשת, אך חלק מהערכים בה חסרים ובקומם מופיעים שלושה סימני שאלה:



השלימו את סימני השאלה האלו בערכים משלכם. אם ערך מסוים צריך להיות רנדומלי, בחרו ערך כלשהו.



תרגיל 6.18 מודרך - מימוש Three Way Handshake

בתרגיל זה תממשו Three Way Handshake באמצעות Scapy. נתחילה מכר שנפתה הסופה ב-k .Wireshark כעת, נעה את Scapy וניצור חבית TCP

```
>>> syn_segment = TCP()
>>> syn_segment.show()
```

```
C:\> C:\Windows\system32\cmd.exe - scapy
C:\> cd \Python26\Scripts
C:\> scapy
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: No IPv6 support in kernel
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python Crypto lib. Won't be able to decrypt WEP.
INFO: Can't import python Crypto lib. Disabled certificate manipulation tools
Welcome to Scapy (2.2.0-dev)
>>> syn_segment = TCP()
>>> syn_segment.show()
###[ TCP ]###
sport= ftp_data
dport= http
seq= 0
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
checksum= None
urgptr= 0
options= ()
>>>
```

למעשה, ניתן לראות ש-Scapy יוצר עבורהנו באופן ברירת מחדל חבילה מסוג SYN, על ידי כך שרשום: flags = S. ערך ה-Sequence Number הינו 0. פורט היעד הוא פורט 80, ו-Scapy מציג אותו עבורהנו כ-HTTP. עם זאת, נרצה ליצר בעצמנו את החבילה ולשלוט בפרמטרים האלו. וכך, ניצור בעצמנו סגמנט TCP עם הדגל SYN, כשפורט היעד הוא פורט 80, ונitin בערך ה-Sequence Number 123 את המספר 123:

```
>>> syn_segment = TCP(dport=80, seq=123, flags='S')
>>> syn_segment.show()
```

```
C:\Windows\system32\cmd.exe - scapy
>>> syn_segment = TCP(dport=80, seq=123, flags='S')
>>> syn_segment.show()
###[ TCP ]###
sport= ftp_data
dport= http
seq= 123
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
checksum= None
urgptr= 0
options= ()
```

כעת נרצה "להעמיס" את שכבת ה-TCP שיצרנו מעל שכבה של IP, בכך שנוכל לשלוח אותה. ננסה לשלוח את החבילה אל google, וכן ניצור אותה בצורה הבאה:

```
>>> my_packet = IP(dst='www.google.com')/syn_segment
>>> my_packet.show()
```

```
C:\Windows\system32\cmd.exe - scapy
>>> my_packet = IP(dst='www.google.com')/syn_segment
>>> my_packet.show()
###[ IP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= tcp
checksum= None
src= 192.168.14.51
dst= Net('www.google.com')
\options\
###[ TCP ]###
sport= ftp_data
dport= http
seq= 123
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
checksum= None
urgptr= 0
options= ()
```

ראשית, ננסה פשוט לשלוח את החבילה בזמן שהוא מסניף:

```
>>> send(my_packet)
```

נמצא את החבילה בהסנהפה, ונראה גם את החבילות שאחריה:

Filter: tcp.stream eq 1						Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info			
5	1.24617400	192.168.14.51	173.194.113.147	TCP	54	ftp-data > http [SYN]	Seq=0	Win=8192	Len=0
6	1.26118000	173.194.113.147	192.168.14.51	TCP	60	http > ftp-data [SYN, ACK]	Seq=0	Ack=1	Win=4080 Len=0 MSS=1360
25	4.25926000	173.194.113.147	192.168.14.51	TCP	60	http > ftp-data [SYN, ACK]	Seq=0	Ack=1	Win=4080 Len=0 MSS=1360
415	10.2604080	173.194.113.147	192.168.14.51	TCP	60	http > ftp-data [SYN, ACK]	Seq=0	Ack=1	Win=4080 Len=0 MSS=1360

כפי שניתן לראות, חבילת ה-SYN שלנו הגיעו אל Google. השרת של Google, ניסה להשלים את לחיצת היד המשולשת ושלח אלינו חבילת תשובה - SYN + ACK. עם זאת, לא הגבינו על חבילה זו. השרת של Google, שהנich שאלוי החבילה לא הגיעו אלינו, ניסה לשלוח אותה אלינו פעמיים נוספים. לאחר מכן, הוא התייאש. החיבור לא קם.

כעת נשלח שוב את החבילה באמצעות Scapy, אך הפעם משתמש בפקודה **sr1** שפגשנו קודם לכן, בצד ימין של Google לשמר את חבילת התשובה של Google:

```
>>> response_packet = sr1(my_packet)
>>> response_packet.show()
```

```
C:\Windows\system32\cmd.exe - scapy
>>> response_packet = sr1(my_packet)
Begin emission:
Finished to send 1 packets.

...
Received 5 packets, got 1 answers, remaining 0 packets
>>> response_packet.show()
###[ IP ]###
version= 4L
ihl= 5L
tos= 0x0
len= 44
id= 30529
flags= DF
frag= 0L
ttl= 250
proto= tcp
chksum= 0x1b59
src= 173.194.113.147
dst= 192.168.14.51
'options\
###[ TCP ]###
sport= http
dport= ftp_data
seq= 3447352025L
ack= 124
dataofs= 6L
reserved= 0L
flags= SA
window= 4080
checksum= 0x6125
urgptr= 0
options= [(('MSS', 1360)]
###[ Padding ]###
load= '\x00\x00'
>>>
```

ניתן לראות שערך ה-ACK הינו אקן 124, כמו שציפינו והסבירנו קודם לכן בפרק זה. בנוסף, ערך ה-Sequence-ACK הוא ערך רנדומלי שהוגרל בשרת של Google. בשדה הדגלים, הערך הוא "SA", כלומר SYN (שמיוצג על ידי "S") ו-ACK (שמיוצג על ידי "A").

כעת, כשבידכם חבילת ה-ACK+SYN, המשיכו **בעצמכם**. השתמשו בחבילת ההזו בצד ימין ליצור את חבילת ה-ACK שתשלים את החיבור, ושלחו אותה אל Google בזמן. אם תצליחו, תראו בהසנהה ש-Google לא שולח לכם עוד שתי חבילות ACK + SYN מכיוון שהחיבור הוקם בצורה מוצלחת.



תרגיל 6.19 - איזה שירותי פתוחים?

באמצעות ביצוע Three Way Handshake, אנו יכולים לגלוות אילו שירותי פתוחים אצל מחשב מרוחק. כיצד? קודם לכן, כאשר שלחנו חבילת SYN אל פורט 80 של Google, קיבלנו בתשובה חבילת SYN+ACK. מכך למדנו שפורט 80 "פתוח" אצל Google, ככלומר יש אצל תוכנה שמאזינה על פורט 80. מכיוון שהוא יודע שעל פורט 80 מאזינה בדרך כלל תוכנה שנותנת שירות HTTP, גילינו שכרגע השירות ה-HTTP "פתוח" אצל Google ונitinן לגשת אליו.

מה יקרה אם נשלח חבילת SYN לפורט "סגור", ככלומר לפורט שאף תוכנה לא מאזינה עליו?
בואו ננסה זאת. נשלח חבילת SYN לשרת של Google, אך לפורט 1024:

Filter: tcp.stream eq 9						Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info			
84	24.6544140	192.168.14.51	173.194.112.242	TCP	54	ftp-data > 24601 [SYN] Seq=0 Win=8192 Len=0			

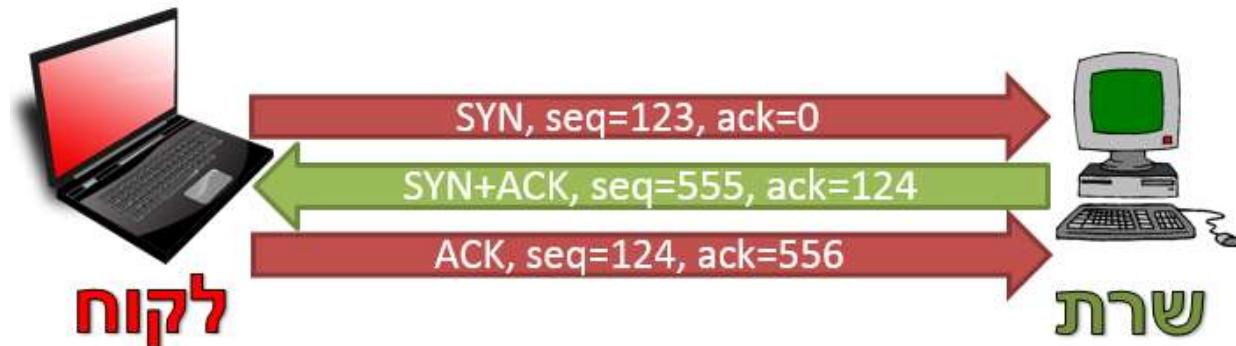
השרת של Google כלל לא נתן תשובה של ACK+SYN! במקרים מסוימים, שירותי מרוחקים יענו חבילת כshedgal-h-RST דולק, ומשמעותו שהשרות לא מוכן להרים את הקישור.

השתמשו בהתנהגות זאת בצד לכתוב סקריפט אשר מקבל מהמשתמש כתובת IP, ומძפיס למסך איזה פורטים פתוחים במחשב המרוחק, בטווח ה포רטים 1024-20. מכיוון שהסקריפט עתיד לשלוח תעבורת רבה, **אל תבדקו אותו על שירותי אינטרנט**, אלא רק על מחשבים נוספים בביביטם או בכייתכם.

העברת מידע על גבי קישור שנוצר

הבנו כיצד מרים מרים קישור ב-TCP. לאחר ביצוע ה-Three Way Handshake, קיימן קישור בין שני הצדדים. כעת, כל סגמנט שמגיע משוייר בידי TCP לקישור מסוים. לבסוף, TCP יודע לסדר את החבילות שהוא מקבל לפי סדר השילחה שלhn ולקבל מחדש חבילות שהגיעו בצורה משובשת, או לא הגיעו כלל, זאת על ידי שימוש ב-Acknowledgement Numbers ו-Sequence Numbers.

הчисוב של שדות ה-Acknowledgement Number- Sequence Number לאורך הקישור, ממשיך באותו אופן שבו הוא בוצע במהלך ה-Three Way Handshake. כך לדוגמה, נאמר שהלכה והשרות הרימו קישור בצורה הבאה:



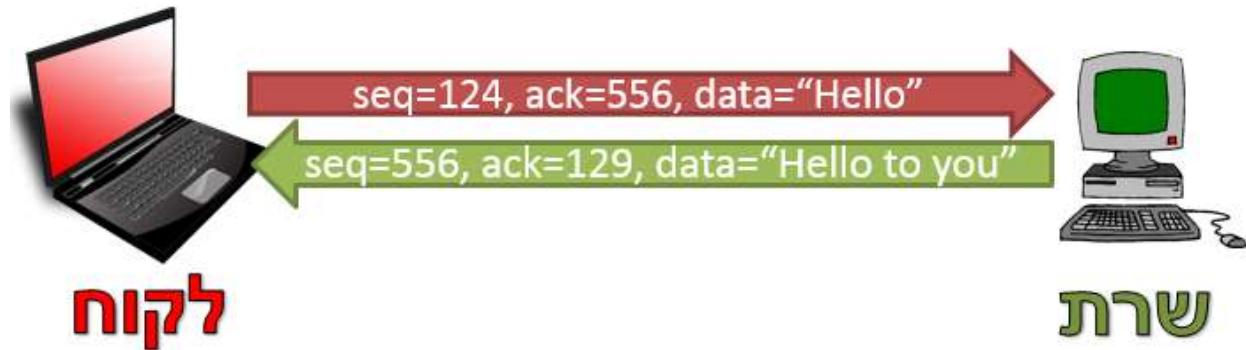
כעת, הלוקוט רוצה לשלוח הודעה נוספת, ובה המידע: "Hello". כיצד יראו המזהים? ובכן, שדה ה-Sequence Number יהיה 124, שכן זה הבית הבא שנשלח. בהודעה האחורונה שמוצגת בצייר לעיל, הלוקוט שלח ACK, אך הוא לא שלח כל מידע. שדה ה-ACK ישאר בעל הערך 556, שכן לא נשלח מידע חדש מהשרות:



כעת נאמר שהשרות רוצה להשיב ב-ACK, אך גם לכלול את ההודעה: "Hello to you". מה יהיו הערכים בהודעה?

שדה ה-Sequence Number יכול את הערך 556, מכיוון שהוא הבית הבא שהשרות צפוי לשלוח. שמו לב שנייתן לדעת זאת מכיוון שהוא ערך ה-ACK בחבילה האחורונה של הלקוב.

בשביל לחשב את הערך של שדה ה-ACK, علينا לקחת את ה-Sequence Number האחרון של הלקוב (והוא 124), ולהחבר אליו את אורך ההודעה שהוא שלח (האורך הוא 5, מכיוון שהוא האורך של המחרוזת "Hello"). אי' לכך, ערך ה-ACK יהיה 129:



כעת הלקוב רוצה לשלוח לשרת ACK על החבילה שהתקבל, אך להוסיף מסר - ".?How are you". נסו לחשב בעצמכם: מה יהיה ערך ה-ACK? מה יהיה שדה Sequence Number?

ראשית נחשב את שדה ה-Sequence Number של ההודעה. הערך יהיה זהה לערך ה-ACK של החבילה הקודמת: כולם 129.

כעת נחשב את שדה ה-ACK. שדה ה-ACK מחושב על פי שימוש ב-Sequence Number של השירות (שהיה 556), ועוד האורך של המידע שהוא שלח. האורך של המחרוזת "Hello to you" הוא 12 בתים, ולכן הערך יהיה $(12 + 556) 568$:





תרגיל 6.20 - חישובי ערכים בשיחת TCP

בشرطוט שלפניכם מוצג המשך השיחה לעיל, אך חלק מהערכים בה חסרים ובמקרה מופיעים שלושה סימני שאלה:



השלימו את סימני השאלה האלו בערכים המתאימים.

תפקידים ושיפורים נוספים של TCP

דיברנו על TCP לא מעט, ועם זאת - נגענו רק בחלק קטן מהתפקידים של TCP ודריכי המימוש שלו.

TCP אחראי גם לכך שהמידע יעבור מצד לצד בצורה יעילה עד כמה שניתן. לשם כך, למשל, TCP לא תמיד מחייב ל-ACK על מנת לשלוח את החבילה הבאה - אלא שולח מספר חבילות יחד. עם זאת, TCP לא מעוניין לשולח יותר מידע ממה שהמחשב שמקבל יהיה מוכן לאכسن. אי כך, על הצדדים להסכים על גודל מסוים של כל מקטע⁴⁴. כמו כן, TCP מנסה למנוע היוצרות של עומס על הרשות כולה. באם TCP מזיהה שיש עומס על הרשות, הוא פועל בכך לשלוח פחות מידע ולאפשר לרשת "להתאושש".

ל-TCP יש גם שיפורים נוספים לאורק הזמן וקיימים בחלוקת מהIMPLEMENTATIONS שלו. לדוגמה, בחלוקת מהמקרים, לא צריך לחכות לכך שלא יגיע ACK בצדדי לשלוח חבילה מחדש. לעיתים, צד אחד של החיבור יכול להבין שהסירה לו חבילה שלא הגיעה, ולהודיע על כך לשלוח באמצעות חבילות מיוחדות בשם NAK.

על תפקידים אלו ועוד לא נהנית במסגרת ספר זה, אך אתם מוזמנים להרחב אופקים ולקראן עליהם באינטרנט, ובפרק [עדים להמשך](#) של פרק זה.

⁴⁴ גודל המידע שהמחשב מקבל מוקן לקבל נקרא גודל החלון. הוא דינامي, ונשלח בכל מקטע TCP בשדה Window Size. תהליך ניהול החלון הוא מורכב, ותוכלו להרחב את הידע שלכם בסעיף [עדים להמשך](#) של פרק זה.

תרגיל 6.21 – תקשורת סודית מעל TCP (אתגר)

קודם לכן, בתרגיל "תקשרות סודית מעל מספרי פורט", סייעتم לשני תלמידים, יואב ומאור, לתקשר בצורה סודית מעל הרשת. להזכירכם, מטרת התלמידים הייתה להצליח להעביר מסרים מהאחד לשני, מבלי ש愧 אדם יוכל לקרוא אותם, גם אם הוא יכול להסניף את התעבורה ביניהם. בתרגיל זה תסייעו לשתי תלמידות, זהה ושיר, להשיג מטרה דומה, אך בדרך אחרת.

את הפתרון הקודם מימוש באמצעות UDP, הפעם תשלחו את המסר הסודי מעל TCP.

תרגיל מכין- מימוש Three Way Handshake ע"י Scapy

בשלב הראשון תממשו Three Way Handshake באמצעות Scapy. מימוש הלוקוח יהיה כמו בתרגיל מודרך 6.18, אך בנוסף תממשו גם את צד השרת. מימוש Three Way Handshake גם הלוקוח:

- ישלח פקודות מסוג SYN לשרת, לפורט אקראי כלשהו שיבחר ע"י הלוקוח (חשוב להמשך התרגיל). גם ה-source port צריך להיות אקראי כפי שמקובל בתקשרות TCP.
- יצפה לקבלת SYN-ACK מהשרת. אם לא מתקבלת פקטה זו, ישלח SYN בולולאה כל זמן מוגדר.
- עם קבלת SYN-ACK, הלוקוח ישלח את הודעת ACK שחותמת את התהליך. יש להקפיד על ערכי seq ו-ack נכונים, בהתאם לערכים שנשלחו בהודעות קודמות.

השרות:

- יזין לפקודות מסוג SYN. שימוש לב שהפקודות עשוויות להשליח לכל פורט (בניגוד לתקשרות TCP רגילה בה השירות מבצע listen לפורט ספציפי) יכול השירות לא יכול לעזור במספר הפורט כדי לסנן את הפקודות. חישבו איך בכל זאת ניתן לסנן פקודות SYN.
- ענה עם פקחת SYN-ACK. יש להקפיד על ערכי seq ו-ack נכונים.

בסיום התהליך עליכם להיות מסוגלים למצוא את הפקודות שלוחתם הן בשרת והן בלוקוח. לדוגמה:

No.	Time	Source	Destination	Protocol	Length	Info
14	4.96560800	10.0.0.4	10.0.0.8	TCP	54	2222-80 [SYN] Seq=0 Win=8192 Len=0
17	5.78265300	10.0.0.8	10.0.0.4	TCP	60	80-2222 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
18	5.96839100	10.0.0.4	10.0.0.8	TCP	54	2222-80 [ACK] Seq=1 Ack=1 Win=8192 Len=0
No.	Time	Source	Destination	Protocol	Length	Info
561	13.0079590	10.0.0.4	10.0.0.8	TCP	60	2222-80 [SYN] Seq=0 Win=8192 Len=0
597	13.8201310	10.0.0.8	10.0.0.4	TCP	54	80-2222 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
607	14.0082450	10.0.0.4	10.0.0.8	TCP	60	2222-80 [ACK] Seq=1 Ack=1 Win=8192 Len=0

יש לציין שימוש ה-Three Way Handshake באמצעות Scapy לא יוצר socket אמיתי של TCP אלא משתמש רק לטובת תרגול, במילויים אחרים, לא נוצר קשר אמיתי ברמת שכבת התעבורה וההודעות הבאות שיועברו בין השירות ללוקוח לא יקבלו ACK (אלא אם כן אתם תממשו זאת בעצמכם...)

מימוש תקשורת סודית

כעת נעברו לחלק בו אנחנו ממשיכם תקשורת סודית. ב כדי שהעברת המידע תבוצע כך שלא ניתן יהיה להבין אותה, תשלחו את הפקטות בסדר לא נכון, ובכל פעם לפורט אחר. הערך של ה-Sequence Number תקין ביחס לקישור שהוקם, אלא ביחס למסר הכללי אותו התלמיד מנסה להעביר.

נסביר באמצעות דוגמה: במידה שישר מעוניינט לשלוח לזרה את המסר: "TCP connections are awesome" היא תוכל, למשל, לעשות זאת כך:

- להרים קישור (באמצעות Three Way Handshake) לפורט 24601.
- לשולח את המסר "c". ערך ה-Sequence Number יהיה 0. לסגור את הקישור.
- להרים קישור (באמצעות Three Way Handshake) לפורט 1337.
- לשולח את המסר "awesome". ערך ה-Sequence Number יהיה 20. לסגור את הקישור.
- להרים קישור (באמצעות Three Way Handshake) לפורט 555.
- לשולח את המסר "connections are". ערך ה-Sequence Number יהיה 5. לסגור את הקישור.

שים לב: התקשרות איתה תיארנו לעיל אינה תקשורת TCP תקינה. לדוגמה, כאשר נשלח המסר "c", ערך ה-Sequence Number אמור להיות מחושב בהתאם לערך ה-ISBN שהוגرسل בתחילת הקישור, ולא להיות 0. הוא מקבל את הערך 0, שכן אלו הבטים הראשונים במסר שישר מנסה לשלוח לזרה. מעבר לכך, אין חשיבות למספר הפורט.

עבור המסתכל מן הצד, התקשרות נראה כmo תקשורת TCP שאינה תקינה ואיינה הגיונית. עם זאת, מי ש מכיר את דרך הפעולה של התקשרות הסודית, יכול להבין את המסר.

חלק א' - לקוח ששלוח מסר סודי

כתבו סקריפט בשם `tcp_secret_message_sender.py`. על הסקריפט לבצע את התהליך הבא:

- לבקש מהמשתמש להקליד הודעה.
- לבקש מהמשתמש מספר שימושו לכמה חלקים לחלק את ההודעה בהעברת המסר הסוד.
- לשולח את ההודעה באופן סודי, כפי שתואר לעל, ובהתאם למספר החלקים שביקש המשתמש.

dagshim

- עליכם להשתמש ב-Scapy כדי להסניף, לקבל ולשלוח את החבילות.
- את כתובות ה-IP של השירות אתכם יכולים כולל בקוד שלכם באופן קבוע ולא לבקש אותה מהמשתמש.
- את הפורט עליכם לקבוע באופן אקראי (בכל מקרה השירות לא יוכל להניח שידוע לו מספר הפורט).
- במידה שמספר החלקים שהמשתמש שלח גדול יותר מסופר התווים שבהודעה, הציגו לו הודעה שגיאה.
- על מנת לבדוק את תרגיל זה, עליכם להשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת⁴⁵.

חלק ב' - שירות שקורא מסר סודי

כתבו סקריפט בשם `tcp_secret_message_receiver.py`. על הסקריפט לבצע את התהליך הבא:

- לחכות לפניות TCP (סגןוט עם הדגל SYN).
- להרים קישור (באמצעות Three Way Handshake) עם מי שפתח את חיבור ה-TCP.
- לקבל מעל חיבור ה-TCP את המידע של השולח, ולהבין מה המיקום שלו במסר הכלול.
- לבסוף, לחבר את המידע בסדר הנכון ולהדפיס את המסר הסודי שהתקבל.

dagshim

- עליכם להשתמש ב-Scapy כדי להסניף, לקבל ולשלוח את החבילות.
- בצד ימין את המסרים שהתקבלו מלהלכו, על השירות להבין متى מסר אחד נגמר, ומסר אחר מתחילה. חשבו על דרך לעשות זאת. תוכלו לשנות את קוד הלוקה לשם כך.
- בדקו את השירות באמצעות הלוקה שתכתבם בחלק א'. וודאו כי אתם מצליחים לקרוא נכון את המסרים הנשלחים אליכם. בדקו הן מסרים שונים והן מספר שונה של חלקים אליו הם המסר יחולק.

חלק ג' – תקשורת אמינה

הוסיפו אמינות לתקשרות בין השירות והלקוח שלכם. במקרים אחרים, חישבו על מצב בו פקטה לא הגיעו וטפלו בו. אתם מוזמנים למשתמש פתרון כרצונכם, אפשר ללמוד על מנגנון ה-ACKים של TCP ולשאוב ממנו רעיונות. עם זאת עקב המורכבות שלו עדיף למשתמש מנגנון פשוט יותר.

חלק ד' - לcko ושרת משולבים

כתבו סקריפט בשם `tcp_secret_messenger.py`. הסקריפט ישלב את הלוקה והשירות שתכתבם בחלקים הקודמים של התרגיל, ויאפשר לשני לקוחות לתקשר זה עם זה כמו בצד, באמצעות העברת מסרים סודיים.

⁴⁵ באופן תאורטי, יכולנו לעשות זאת מעל `loopback device` – כלומר מעל כתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של Scapy בשילחה וקבלת מסגרות מעל `loopback device` ב-Windows, נשתמש בשני מחשבים.

שכבה התעבורה - סיכום

בפרק זה סקרנו את השכבה הרביעית במודל חמש השכבות, היא שכבה התעבורה. התחלנו מלימוד על מטרות השכבה, וACHINE מכך הסבירנו את המושג **פורט**. הבנו את מיקום השכבה במודל חמש השכבות, הכרנו את הכלים **netstat** ו**afp** תירגלו את השימוש בו. לאחר מכן הכרנו את המושגים **חיבור מבוסס קישור וחיבור שאינו מבוסס קישור**. למדנו על ההבדלים ביניהם, ומתי נעדיף להשתמש בכל אחד.

ماוחר יותר העמכונו בפרוטוקול UDP. באמצעות Wireshark למדנו להכיר את השודות השונות של הפרוטוקול. לאחר מכן למדנו איך לכתוב לקווי ושרות משתמשים בפרוטוקול UDP באמצעות Sockets. כמו כן, למדנו לשЛОוח ולקבל פקודות UDP באמצעות Scapy. כתבנו מספר שירותים, שלחנו שאלתת DNS וכן העברנו מידע סודי באמצעות שימוש במספרי פורטים.

לאחר מכן למדנו על פרוטוקול TCP. הבנו כיצד TCP משתמש ב-Sequence Numbers וב-ACKים ב כדי לשמר על אמינות. למדנו כיצד מרימים קישור ב-TCP באמצעות Three Way Handshake. צפינו ב-Wireshark כיצד נראות חבילות של TCP, בעת הקמת קישור ובכלל. בהמשך מישנו בעצמנו Three Way Handshake באמצעות Scapy, והשתמשנו ביכולת זו ב כדי לאגור איזה שירותים פתוחים במחשב מרוחק. אז העמכונו בדרך בה נראה תקשורת לאחר שחיבור קם, והבנו כיצד מחושבים ערכיהם של שדות שונים בחבילות TCP. בסוף, הזכרנו בקצרה פקידים של TCP עליהם לא העמכונו בפרק זה.

במהלך הפרק, הכרנו לעומק מספר בעיות אפשריות בראשת, ודריכים להתמודד עימן. להלן טבלה המסכםת את אתגרים ומנגנוני התמודדות אלו:

האם קיימים ב- TCP?	האם קיימים ב- UDP?	מנגנון התמודדות	בעיה
כן	לא	Acknowledgement שליחת סignalization על מידע שנשלח	חbillah לא מגיעה ליעדה
כן	לא	כלילת מס' סידורי עבור כל מידע שנשלח	חbillah מגיעה ליעדה, אך בסדר לא נכון מתוך רצף חbillות
כן	כן	שימוש ב- <u>Checksum</u>	חbillah מגיעה ליעדה, אך באופן לא תקין

בפרק הבא, נמשיך ללמידה על מודל השכבות ונלמד להכיר את שכבת הרשות. נדבר על האתגרים הניצבים לפני שכבה זו, וכייז הוא מתקשרות לשכבה התעבורה עליה למדנו עתה.

שכבה התעבורה - צעדים להמשך

כפי שהבנתם מחלק [תפקידים ושיפורים נוספים של TCP בפרק זה](#), ישנו עוד דברים רבים ללמוד אודות TCP בפרט, ושכבה התעבורה בכלל.

אלו מכם שמעוניינים להעמק את הידע שלהם בשכבה התעבורה, מוזמנים לבצע את הצעדים הבאים:

קריאה נוספת

בספר המצוין Computer Networks (מהדורה חמישית) מאת David J. - Andrew S. Tanenbaum ו-Wetherall, הפרק השישי מתיחס במלואו לשכבה התעבורה. באופן ספציפי, מומלץ לקרוא את החלקים:

- 6.5.4 - מספק מידע על ה-TCP Header, יכול להשלים את המידע השני [בנספח א' של פרק זה](#).
- 6.5.6 - מתאר על תהליך סיום תקשורת TCP, אשר לא סקרו במהלך פרק זה.
- 6.5.8 - ניהול החלון של TCP.
- 6.5.10 - ניהול עומסים של TCP.

בספר Computer Networking: A Top-Down Approach (מהדורה ששית) מאת James F. Kurose, הפרק השלישי מוקדש כולו לשכבה התעבורה. באופן ספציפי, מומלץ לקרוא את החלקים:

- 3.3.2 - למתעניינים בדרך בה מחושב ה-Checksum של UDP.
- 3.4 - שיטות ועקרונות בימוש פרוטוקול אמין.
- 3.6 - עקרונות ושיטות לניהול עומסים.
- 3.7 - ניהול עומסים של TCP.

כמו כן, ניתן להרחיב את אופקיכם בפרק על TCP ו-UDP מתוך [The TCP/IP Guide](#), אותו ניתן למצוא בכתבota:

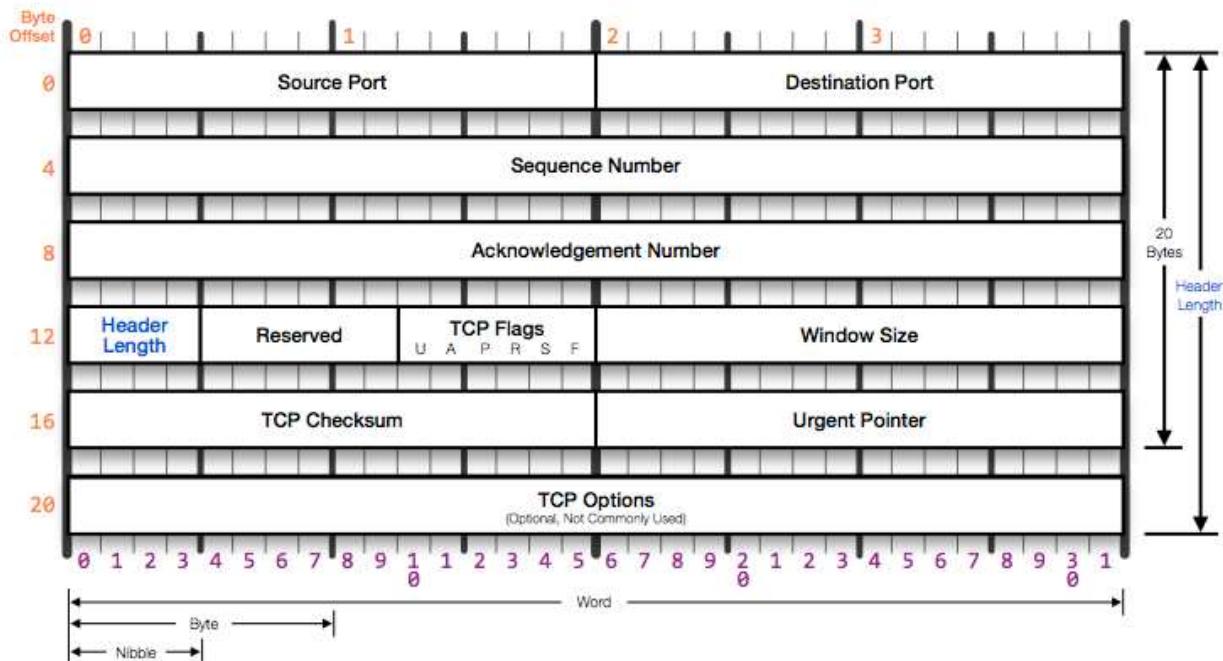
<http://goo.gl/GC69q4>

נספח א' - TCP Header

נספח זה נועד כדי לתאר את כל השדות של ה-Header של TCP. לא חוני להבין את כל השדות עד הסוף.

מידע נוסף ניתן למצוא בכתובת: <http://goo.gl/CyVbvX>

TCP Header

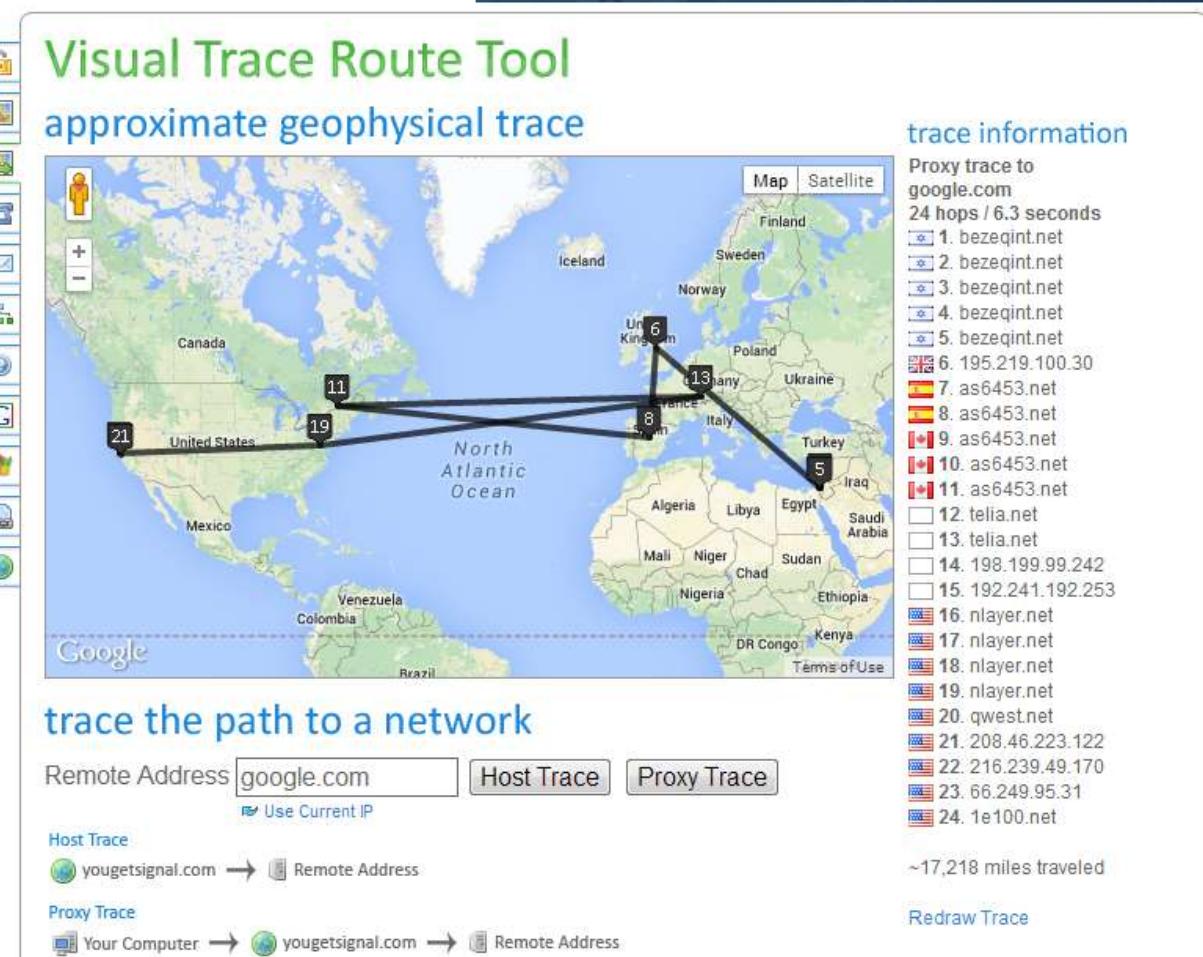


- **Source Port** - פורט המקור.
- **Destination Port** - פורט היעד.
- **Sequence Number** - מספר סידורי עוקב, המזהה את המקטע בתוך זרם המידע הכלול. כל Sequence Number מתאר בית אחד ברצף המידע.
- **Acknowledgment Number** - מתאר את הבית הבא שצפוי להתקבל.
- **Header Length** - מתאר את אורכו של החביליה, היוט שהוא עשוי לשנתנות מחייביה לחביליה, מכיוון שישנם שדות של TCP Options אותם ניתן בהמשך. הערך של השדה מתאר את הגודל ביחידות מידיה של 32 ביטים. כך למשל, עבור חבילה בגודל 20 בתים, הערך של השדה יהיה 5 (מכיוון שהוא כפוף ב-160 ביטים, שהם 5 פעמיים 32 ביטים). הערך 5 (שمتאר 20 בתים) הוא הערך הנפוץ ביותר עבור שדה זה.
- **Reserved** - ביטים ששמורים עבור שימוש עתידי.

- Flags - הדגלים
 - C - דגל CWR - קיצור של Congestion Window Reduced - מתייחס להקטנת Congestion Window וקשרו לטיפול בעומסים. איןנו מרחיבים על נושא זה בספר זה.
 - E - דגל ECN-ECHO - קיצור של Explicit Congestion Notification - מודיע לשלוח שביקשת Congestion Experienced התקבלה. חלק משיפורים חדשים יותר של TCP. איןנו מרחיבים על נושא זה בספר זה.
 - U - דגל URG - קיצור של Urgent. הסוגמנט מכיל מידע דחוף, והשדה Urgent Pointer מצביע על מידע דחוף זה.
 - A - דגל ACK - קיצור של Acknowledgement - אישור על קבלת מקטע קודם. מאשר את המספר הסידורי שモופיע בשדה Acknowledgement Number.
 - P - דגל PSH - מבקש ממערכת הפעלה להעביר את המידע ללא דיחוי. איןנו מרחיבים על נושא זה בספר זה.
 - S - דגל SYN - בקשה להקמת קישור.
 - F - דגל FIN - בקשה לסיירת קישור קיים.
- Window Size - גודל חלון השיליחה של TCP. מצין כמה מידע השולח מוכן לקבל ברגע זה.
- Checksum - מזودא תקינות המידע בתוך הסוגמנט. ה-Checksum מתבצע גם על המידע עצמו, ולא רק על ה-Header.
- Urgent Pointer - מצביע למיקום המידע הדחוף בסוגמנט, שהדגל URG מורה על קיומו.
- Options - אפשרויות נוספות.
- דוגמה לאופציה: ניתן לכלול מקטע המקסימלי המותר לשיליחה.

פרק 7 - שכבת הרשת

פרק תחילת מסע - איך עובד האינטרנט/ ענן האינטרנט, נחשפטם לכל Visual Traceroute, שמאפשר לנו, למשל, לראות את הדרכ שבה עבר מידע בין המחשב שלנו לבין השירות של Google:



כיצד Visual Traceroute יודע לעשות זאת? איך הוא מבין מה הדרכ שבה עבר המידע? עד סוף פרק זה תוכלן לענות על שאלה זו, וכן לכתוב בעצמכם כל שימצא את הדרכ בה עבר המידע בין המחשב שלכם לבין שירות מרוחק.

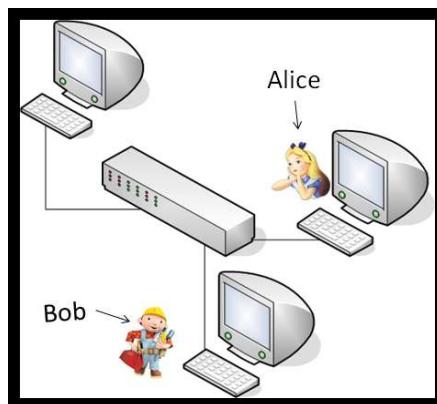
לשם כך, עליינו לענות על שאלות רבות ומרתקות - מהי תפקידה של שכבת הרשת? מהו פרוטוקול IP? מה היא כתובת IP? מה זה נתב? על כל שאלות אלו (ועוד) תוכלן לענות בעצמכם בתום פרק זה.

נתחיל מהשאלה הראונה:

? מה תפקידה של שכבת הרשות?

ראשית علينا להבין מה מהותה של השכבה השלישייה במודל השכבות, הלא היא שכבת הרשות. בפרק הקודם למדנו על שכבת התעבורה, והבנו שהיא מאפשרת לנו לשולח הודעה ממוחשב אחד למוחשב אחר. כמו שמדובר, שכבת התעבורה עשויה לדאוג לכך שלא יהיה כישלון בהעברת המידע בין הצדדים. אם כך, מדוע צריכים את השכבה השלישייה? **חשבו על כך ונסו לענות על השאלה זו בעצמכם.**

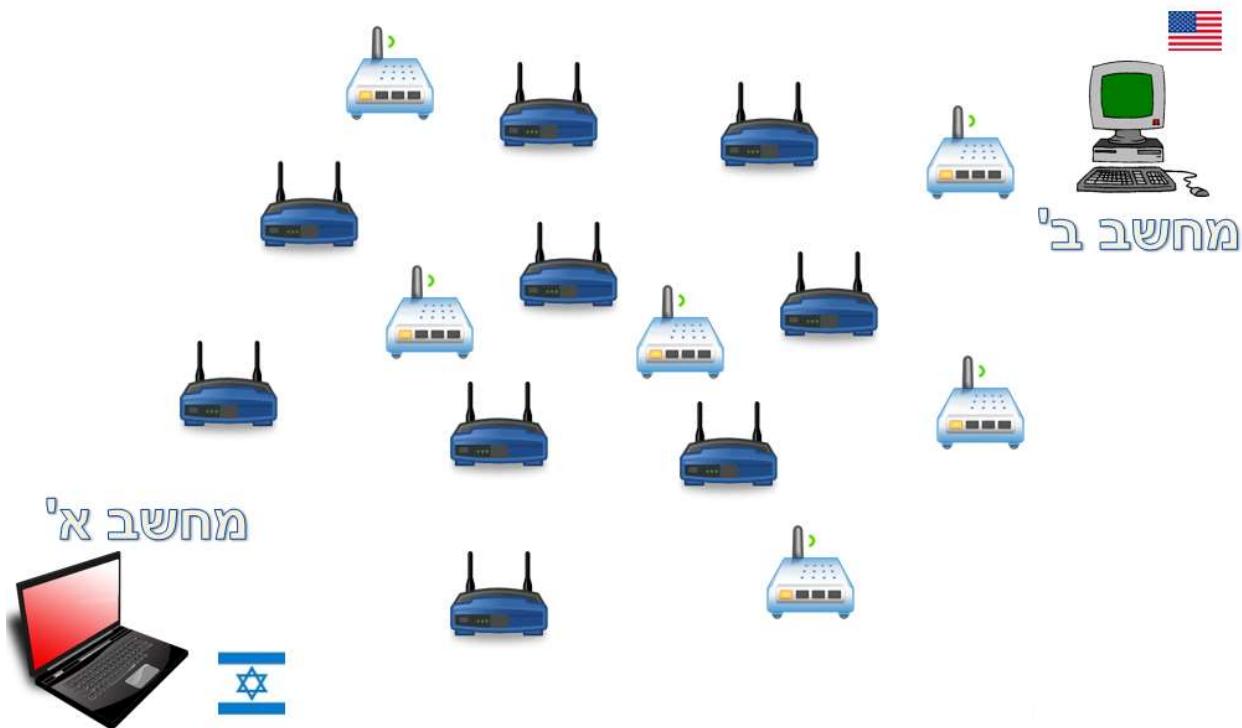
כדי לענות על השאלה הזאת, נתחיל מלחשוב על רשותות ישותות, שהיו קיימות לפני כ-30 שנים. עצמו את העניינים, דמיינו עולם ללא טלפונים ניידים ולא רשת האינטרנט שאתם מכירים כיום. רשת סטנדרטית בתקופה הזאת הייתה יכולה לכלול כמה מחשבים בודדים, שהוברו באמצעות כבלים וקופסאות קטנה.



בפועל זהה, בו רשותות כוללות מספר מצומצם של מחשבים בלבד, קליחסית לגרום למיידע לעبور מצד לצד (בדוגמא שבצ'יר - מהמחשב של Bob אל המחשב של Alice). אך רשותות אלו אינן דומות כלל וכלל לרשותות שאנו מכירים היום! המחשב שלכם, שנמצא כאן בישראל, כמו גם הסמארטפון שלכם, מצויים באותה רשת כמו שרת Google שנמצא בארצות הברית, והוא רשת האינטרנט. על מנת להעביר מידע בין מחשבים⁴⁶ ברשת האינטרנט, علينا לעبور בדרך בהרבה רכיבים שונים.

⁴⁶ המושג "מחשבים" משומש כאן מטעני נוחות. למעשה, הדבר נכון עבור כל ישות רשת - כמו נתבים או שרתים. שמו לב שכאשר אנו מתיחסים ל"ישות" ברשת, אנו מתיחסים לכרטיס רשת מסוים. כך למשל, מחשב עם שני כרטיסי רשת מייצג למעשה שתי "ישויות רשת" שונות.

הביתו בשרטוט הרשות הבא:



בشرطוט זה ניתן לראות את מחשב א', שנמצא בישראל, ומחשב ב', שנמצא בארצות הברית. עתה, נניח והמשתמש במחשב א' רוצה לשלוח הודעה אל מחשב ב'.

שכבה התעבורת, עליה למדנו בפרק הקודם, מניהה כי אפשר להעביר חבילת מידע בוודدت ממחשב א' למחשב ב'. שכבת הרשות היא האחראית לתהליך זה.



מטרת שכבת הרשות היא להעביר חבילות מידע מיישות*⁴⁷* אחת אל ישות אחרת.

נסתכל שוב בשרטוט לעיל. ישות אחת (מחשב א') מעוניינת לשלוח מידע לשות שנייה (מחשב ב'). המידע הזה מחולק בתורו לחבילות מידע. כאשר מתיחסים ל"חבילות מידע" בשכבת הרשות, אנו קוראים להן בשם **Packets** **ובעברית - חבילות (או "פקטות")**. להיות שרוב המידע שעובר באינטרנט מועבר באמצעות השכבה השלישית, אנו מכנים לרוב כל מידע כזה בשם "פקטה", כפי שראיתם לאורך הספר. בדוגמה לעיל, שכבת הרשות אחראית להעביר את חבילות המידע בין מחשב א' לבין מחשב ב'.

⁴⁷ למשל ממוחשב אחד לאחר או ממוחשב לשרת.

שימוש לב של נקודת קצה בפני עצמה מכירה את מבנה הרשת הכלול. כלומר, מחשב א' (המקור) לא יודע איזה רכיבים נמצאים בין למחשב ב' (היעד). הוא "מבקש" משכבה הרשת לשלוח את החבילה עבורה, ובאחריות שכבת הרשת להבין את מבנה הרשת.



אילו אתגרים עומדים בפני שכבת הרשת?

שימוש לב שבפני שכבת הרשת עומדת ממשימה לא קללה בכלל. חבילה שmagiuha ממחשב א' לממחשב ב' עוברת בדרך קשה ומפותלת. בין השאר, שכבת הרשת עשויה להתמודד עם:

1. חומרות שונות - יתכן שבדרך בין מחשב א' לממחسب ב' יהיו רכיבים שונים לחלווטין, כאשר אחד מהם הוא שרף עצום ואחד מהם הוא קופסא קטנה.
2. תקנים שונים - יתכן שהתקשרות בין מחשב א' למעבור בלויין בחלל, לאחר מכן באמצעות כבל רשת סטנדרטי⁴⁸, לאחר מכן באמצעות WiFi ובחזרה.

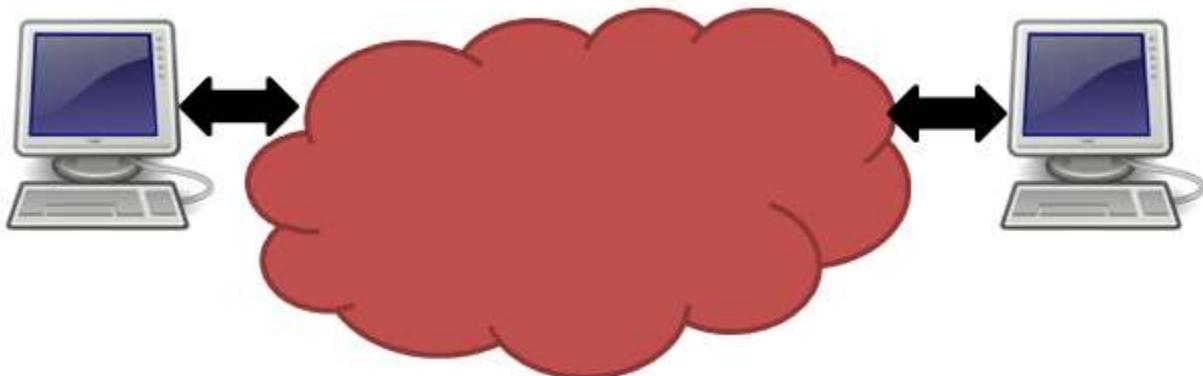
מייקום שכבת הרשת במודל השכבות

שכבת הרשת היא השכבה השלישית במודל חמש השכבות.



מה השירותים ששכבת הרשת מספקת לשכבה שמعلיה?

שכבת הרשת מספקת לשכבת התעבורה, שכבה הרכיבית במודל השכבות, מודל של "ענן", שבו חברות מידע מגיעות מצד אחד לצד שני. שכבת התעבורה אינה מודעת כלל למבנה הרשת המתוואר, ולמעשה מבחינתה יש פשוט "רשות כלשה" שמחברת בין מחשב א' לממחسب ב'. "תמונה הרשת", מבחינתה, נראה כך:



⁴⁸ הכוונה היא לכבל Ethernet, עליו נלמד בפרק שכבת הקו.

בצורה זו, שכבת הרשות מצלילה "להעלים" את הרשות מבחינה שכבת התעבורה מבקשת "שלחי לי חビלה מכאן לכאן" (למשל מחשב א' למחשב ב'), ו scavbet הרשות דואגת לכל התהילה ממש ואילך.



מה השירותים ש scavbet הרשות מקבלת מהשכבה שמתוחתיה?

שכבת הקו מספקת לשכבת הרשות ממשק להעברת מידע בין שתי יישויות הקשורות זו לזה באופן ישיר. באופן זה, scavbet הרשות לא צריכה לדאוג לסוגיות הקשורות לחיבור בין שתי תחנות. את scavbet הרשות לא מעוניין אם היישויות הקשורות בכבול, בלויין, או באמצעות יוני דואר. היא רק אחראית להבין מה המסלול האופטימלי. כמו ש-Waze רק אומרת לרכיב באיזו דרך לעבור, ולא מסבירה לנוג שהוא צריך לתרדיק, ללחות על הגז או לאותה. בזה טיפול הנagger, או במקרה שלנו - scavbet הקו. על כל זאת, נלמד לעומק בפרק הבא.

מסלולים ברשות



תרגיל 7.1 מודרך - מי נמצא בדרך שלי?

עכשו נסו בעצמכם - כיצד תגלו מה הדרכך בה חביתה מידע עוברת מן המחשב שלכם אל Facebook מוביל להשתמש באתר חיצוני?

לשם כך נוכל להיעזר בכל'i אשר פגשנו קודם לכן בספר, בשם **traceroute**.
פתחו את שורת הפקודה (CMD). הנכם כבר יודעים כיצד לעשות זאת.
כעת, הקישו את הפקודה:

tracert -d www.facebook.com

אתם צפויים לראות פלט הדומה לכך:



```

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>tracert -d www.facebook.com

Tracing route to star.c10r.facebook.com [31.13.72.65]
over a maximum of 30 hops:

 1 <1 ms    <1 ms    <1 ms  192.168.14.1
 2 1112 ms   981 ms   1088 ms  212.179.37.1
 3 1085 ms   660 ms   1029 ms  81.218.103.210
 4 1117 ms   983 ms   135 ms   212.179.75.198
 5 1059 ms   1057 ms   1183 ms  212.179.124.193
 6 1037 ms   1002 ms   74 ms   212.179.124.122
 7 934 ms    1000 ms   1075 ms  195.66.225.69
 8 1112 ms   1107 ms   1119 ms  74.119.77.53
 9 868 ms    1213 ms   1354 ms  31.13.72.65

Trace complete.

C:\Users\USER>

```

כל שורה כאן מייצגת תחנה נוספת בדרך בין המחשב שלכם לבין Facebook. מכאן שהחבריה שנשלחה אל www.facebook.com הגיעו ראשית אל 192.168.14.1, אחר מכן אל 212.179.37.1 וכן הלאה. **שים לב:** הפלט במחשביכם יהיה שונה מהפלט של הפקודה שבדוגמה. בהמשך נקבעו מינהליים אלו.

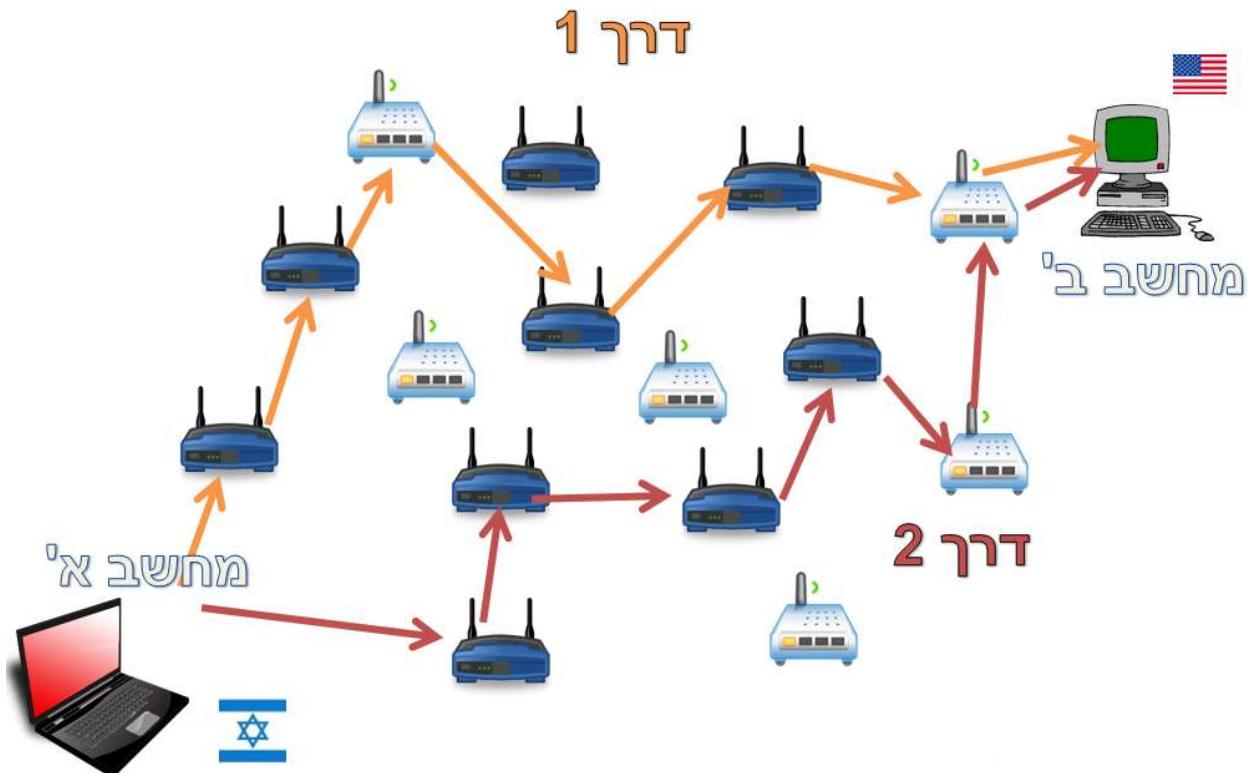
cut, נסו בעצמכם להשלים את הטבלה הבאה לפי הפלט של הפקודה **tracert** במחשביכם:

תחנות בדרך בין המחשב שלכם אל Facebook

כתובת	מספר תחנה
	1
	2

מתרגיל זה אנו לומדים על מטרה נוספת חשובה של שכבות הרשת: **ניתוב (Routing)**. הכוונה היא החלטה על הדרך שבה יש להגיע מנקודה א' לנקודה ב'. הדבר דומה למציאת דרך נסעה ברכב, למשל כמו שעשו האפליקציה Waze. על האפליקציה להבין מה הדרך שבה על הרכב (או במקרה שלנו - חבילת המידע) לעבור כדי להגיע מהמקורה אל היעד.

נביט שוב בשרטוט הרשת הקודם שלנו:



כאן מוצגות שתי דרכי שונות בהן יכולה לעבור חבילת מידע ממחשב א' למחשב ב': דרך 1, המסומנת בכתום, ודרך 2, המסומנת באדום. אם נחזור לעולם המושגים של Waze, החיצים מסומנים למעשה כבישים בהם הרכב יכול לעבור. כזכור שניין לבחור בהרבה דרכים אחרות ואין מנעה לכך. על שכבות הרשת להחליט באיזה דרך להעביר כל חבילה מידע שהגיעה אליה, והוא יכול לבחור בכל דרך שתרצה.

מה צריכה שכבת הרשת לדעת בצדיה להחליט כיצד לנתב?



נסו לחשב על כך בעצמכם לפני תמשיכו בקריאה.

ראשית, על שכבות הרשת להכיר את מבנה הרשת. אם שכבות הרשת תדוע על כל הרכיבים שנמצאים בցיר, היא תוכל להחליט על דרך מלאה אותה יש לעבור בצדיה להגיע ממחשב א' למחשב ב'. קל להקליל זאת למציאת דרך

נסעה ברכב: על מנת ש-Waze תוכל לדעת מה הדרך הטובה ביותר להגעה מTEL אביב לירושלים, עליה להכיר את כל הכבישים במדינה.

שנית, על שכבת הרשות לדעת האם קיים בכלל חיבור בין המקור ליעד. אם כל הכבישים בין תל אביב לירושלים חסומים כרגע, עדיף ש-Waze יגיד לנו להישאר בבית. כך גם צריכה שכבת הרשות לעשות: באם לא קיים כרגע אף חיבור בין מחשב ב', עליה להודיעו למחשב א' שהוא אינה מסוגלת להעביר את חבילת המידע שלו.

שלישית, שכבת הרשות צריכה להבין מהי הדרך הכי מהירה.שוב, בדומה ל-Waze, המטרה של השכבה היא לאפשר לחבילת המידע להגיע בדרך המהירה ביותר.

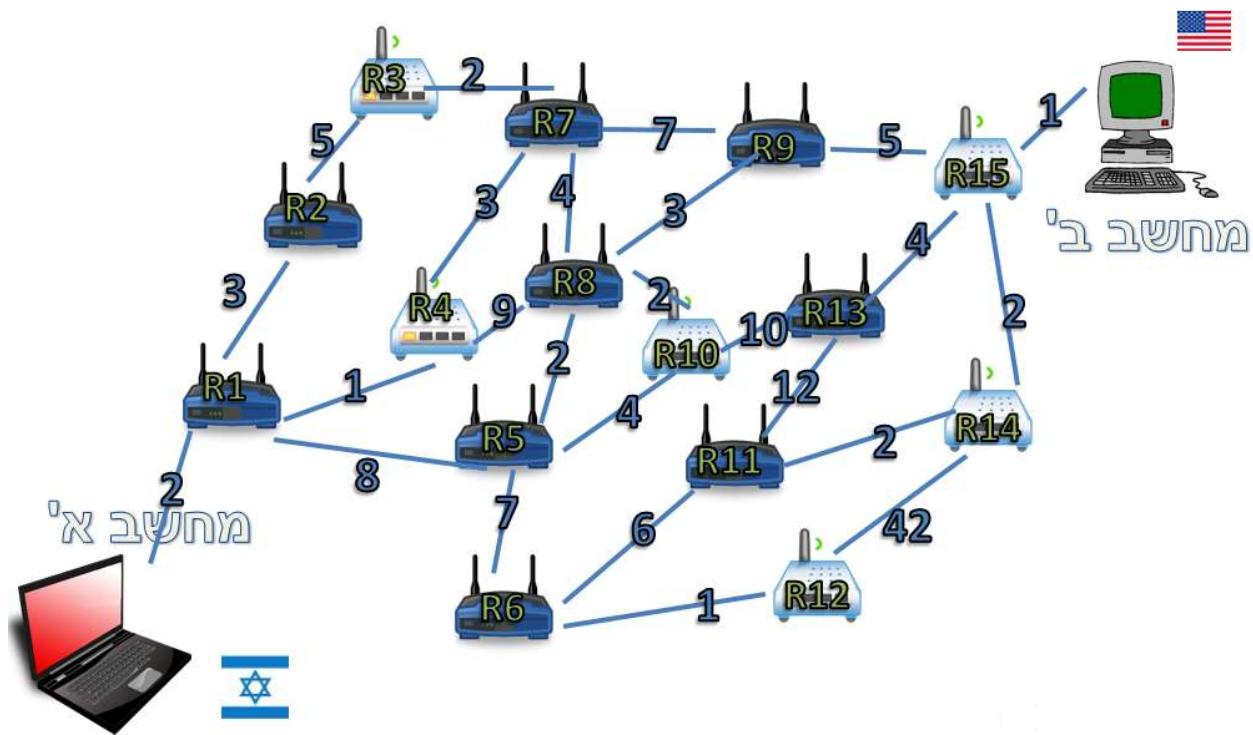
רביעית, באחריות שכבת הרשות להבין אילו דרכי אסورة. כפי שתיכון וכביש מסוים חסום כרגע, או שאינו אפשר לעبور בו בגלל סכנת מפולת, כך גם בעולם הרשות - ישנו נתיבים אשר לא ניתן לעبور בהם.



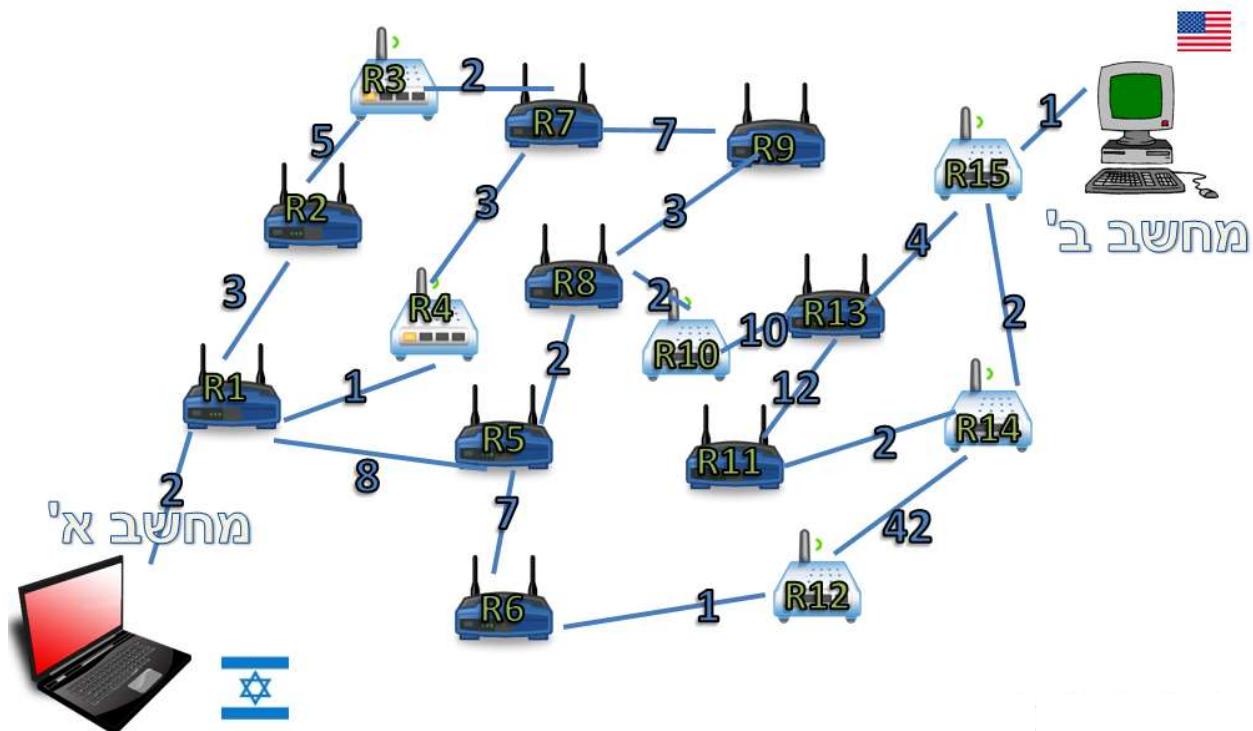
עשה זאת בעצמך - תכנן מסלול ניתוב

לפניכם שובشرطוט הרשות הקודם. הפעם, לכל רכיב בדרך יש אותן ומספר שמותיהם אותו (לדוגמא: R1), ולכל קישור בין רכיב אחר יש "עלות". "עלות" זו יכולה להיקבע על פי גורמים שונים - למשל מרחק פיזי, איכות הkop, סוג הקישור (קווי, אלחוטי) ועוד. כך למשל, שליחת חבילה ממחשב א' אל R1 "עליה" 2, שליחת חבילה מ-R1 אל R2 "עליה" 3, וכן שליחת חבילה ממחשב א' אל R2 דרך R1 "עליה" 5.

מצאו את הדרך ה"זולה" ביותר להגעה ממחשב א' אל מחשב ב'. לאיזו עלות הגיעם? באיזה רכיבים עברתם בדרך?



תקלות שונות גרמו לכך שחלק מהקישורים בין הרכיבים כבר לא פעילים. הסתכלו בתמונה הרשות החדשה, ומצאו שוב את הדרך ה"זולה" ביותר להגעה מחשב א' למחשב ב':



האם הדרך השתנה?

כך הבנו את אחת הסיבות שכבת הרשות משנה את החלטת הניתוב עבור כל פקטה (חבילה). בכל פעם גם מצב הרשות משתנה, ולכןו נהוג בהתאם. בהקבלה לנסיעה ברכבת, זיכרו כי Waze עשויה לבחור עבורו דרך שונה להגעה מביתנו לבית הספר - שכן עכשווי יש עומסי תנועה בדרך מסוימת, ודרך אחרת חסומה בשל עבודות בכביש.

פרוטוקול IP

הבנו מודיעו צריך את שכבת הרשות, לפחות חלק מהתקידים שלה. הבנו שבאחריותה להעביר חבילות מידע מישות אחת ברשות לישות אחרת, וכן הבנו שהיא אחראית על ניתוב החבילות. עכשווי הגיע הזמן לראות קצת איך הדברים קורים במצבות. לשם כך, נזכיר את IP (Internet Protocol), הפרטוקול של האינטרנט.

כתובות IP



מה כתובת ה-IP שלי?

ראשית נגלה מה כתובת ה-IP שלנו. לשם כך, הכנסו לCMD (Command Line), והקישו את הפקודה .ipconfig

הפלט שלכם יהיה דומה לפלט הבא:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix  . : privatebox
  Link-Local IPv6 Address . . . . . : fe80::419b:9669:cfb6:705%11
  IPv4 Address . . . . . : 192.168.14.51
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.14.1
```

כתובת ה-IP של המחשב ממנו הוריצה הפקודה מסומנת במלבן אדום, והוא הכתובת 192.168.14.51. מהי כתובת ה-IP שלכם?



מה זו כתובות IP?

דיברנו במהלך הספר לא מעט על כתובות IP. עצת אנו יכולים לציין כי כתובות IP הן כתובות של שכבת הרשת - שכבה זו משתמשת בהן כדי לדעת מה הכתובת של היישויות השונות ברשת⁴⁹. חשוב להבין שכתובת IP היא כתובת **לוגית** בלבד, בניגוד לכתובת פיזית. כלומר, זו לא כתובת ש"צרכובה" על כרטיס הרשת, אלא עשויה להשתנות עם הזמן ולהתחלף.

כתובת IP, כפי שכבר רأינו, מיוצגת באמצעות ארבעה בתים (bytes)⁵⁰, ולכן תיראה צירוף של ארבעה מספרים, שכל אחד נع בין הערבים 0 ו-255. להלן דוגמא לכתובת IP:

192.168.2.5



חשוב: האם הכתובת הבאה הינה כתובת IP תקינה?

192.168.275.2

התשובה היא - לא. להיות ש-275 הינו מספר גדול מדי (גדול יותר מ-255, ולכן אין נכנס בגודל של בית אחד), הוא לא יכול להיות חלק מכתובת ה-IP.



כמה כתובות IP אפשריות קיימות?

מכיוון שכתובת IP מיוצגת על ידי ארבעה בתים (bytes), היא למעשה מיוצגת על ידי 32 ביטים (bits), שכל אחד מהם יכול להיות 0 או 1. אי כך, ישן 2^{32} אפשרויות לכתובות IP שונות.

כתובת IP מחולקת לשני חלקים:

- **מזהה רשת (Network ID)** - לאיזו רשת שייכת כתובת ה-IP הזו? לדוגמה: האם היא חלק מהרשת של בית הספר?
- **מזהה ישות (Host ID)** - לאיזה כרטיס הרשת שייכת הכתובת הזו, בתוך הרשת⁵¹? למשל - האם היא שייכת לתלמיד משה או לתלמיד אהרון?

⁴⁹ יכולות להיות כתובות אחרות, במידה שלא משתמשים ב-IP אלא בפרוטוקול אחר. לצורך הנוחות, נניח לאורך הפרק שהשימוש הוא תמיד בפרוטוקול IP בטור ה프וטוקול של שכבת הרשת.

⁵⁰ הדבר נכון כמובן רק עבור כתובות IP בגירסה 4 (IPv4), ולא עבור גירסאות אחרות (למשל IPv6). מטעמי נוחות, במהלך הספר, נתיחס לפרוטוקול IPv4 IP בשם "IP".

ניקח לדוגמא את הכתובת הבאה: **200.100.0.1** המשויכת למחשב מסוים. נקרא למחשב זה "מחשב'ה". נאמר והגדרנו את שני הבטים הראשונים (המסומנים באדום) בתור **מזהה הרשת**, ושני הבטים הבאים (המסומנים בכחול) בתור **מזהה הישות**. מכאן שכל כתובת שתתחל ב-**200.100** תתאר ישות שנמצאת באותה הרשת, וכל כתובת אחרת - לא.

עבור כל אחת מכתובות ה-IP הבאות, כתבו האם היא נמצאת באותה הרשת של מחשב'ה, או שما



ברשת נפרדת:

- 1. 200.100.0.2
- 2. 200.100.2.0
- 3. 100.200.0.5
- 4. 200.100.200.100
- 5. 1.2.3.4
- 6. 200.200.100.100
- 7. 1.0.200.100



פתרון מודרך - בדקו את עצםכם

לאחר שתכתבם את תשובותיכם בסעיף הקודם, בואו נביט ביחד בכתובות ונסמן את מזהה הרשת:

- 1. מזהה הרשת זהה למזהה של מחשב'ה, ולכן מדובר בכתובת שנמצאת באותה הרשת.
- 2. מזהה הרשת זהה למזהה של מחשב'ה, ולכן מדובר בכתובת שנמצאת באותה הרשת.
- 3. מזהה הרשת שונה, ולכן מדובר בכתובת שלא נמצא לה חילופין (בדוגמא זו משנה אם החליפו את הסדר או עשו כל דבר אחר. כל עוד מזהה הרשת לא זהה לחילופין).
- 4. מזהה הרשת זהה למזהה של מחשב'ה, ולכן מדובר בכתובת שנמצאת באותה הרשת.
- 5. מזהה הרשת שונה, ולכן מדובר בכתובת שלא נמצא לה חילופין!
- 6. מזהה הרשת שונה, ולכן מדובר בכתובת שלא נמצא לה חילופין!
- 7. מזהה הרשת שונה, ולכן מדובר בכתובת שלא נמצא לה חילופין!

כמו שראינו, בהינתן שגם יודעים מהו מזהה הרשת, אנו יכולים לדעת אילו ישותות (מחשבים, נתבים, שרתים ועוד) נמצאות באותה הרשת. לכל ישות יהיה **מזהה ישות** שונה. כך למשל, במקרה של המחשב מחשב'ה, כתובתו

⁵¹ להזכירם, ישות יכולה להיות מחשב, נתב, שרת או רכיב אחר. באופן ספציפי, מזהה הישות מתיחס לכרטיס רשת מסוים באותו מחשב, נתב, שרת או רכיב.

היתה כזכור: 1.00.0.200, ומזהה הרשת שלו היה **200.100**. מכאן שמדובר בהשota שלו ברשת הינו: **0.1**. נאמר שיש ברשת של מחשביל'ה ישות נוספת, למשל הדפסת של מחשביל'ה, הנקראת צפוי מדפסתלה. מזהה ההשota של מדפסתלה צריך להיות שונה מזו של מחשביל'ה, והוא יוכל להיות למשל: **0.2**. כך תהיה כתובתה המלאה: **200.100.0.2**.

- הכתובת של מחשביל'ה היא: **200.100.0.1**
- הכתובת של מדפסתלה היא: **200.100.0.2**

היות שלמחשביל'ה ומדפסתלה יש את אותו מזהה הרשת (**200.100**), אנחנו יודעים שהם נמצאים באותה הרשת. עם זאת, מכיוון שלכל אחד מהם יש מזהה ישות שונה, אנו יכולים לפנות אליהם בנפרד ולדעת האם המידע שהוא שולחים מיועד למחשביל'ה או למדפסתלה.

בדוגמא לעיל ראיינו מזהה רשת בגודל של שני בתים. שימוש לב Ci מזהה רשת יכולם להיות בגודל משתנה. לדוגמה, יכולים גם להגיד את כתובתו של מחשביל'ה כך:

200.100.0.1

כלומר, הבית הראשון (**200**) מייצג את מזהה הרשת, ושלושת הבתים האחרים (**100.0.1**) את מזהה ההשota. במקרה ומזהה הרשת הוגדר כך, הרי שכל כתובות IP שמתחליה בערך 200 מייצגות כתובות באותה הרשת. כך למשל, הכתובת הבאה: **200.50.2.3**, נמצאת בכתובת של מחשביל'ה. זאת בגיןוד להגדרה הקודמת של כתובתו של מחשביל'ה, שבה כתובות הינה צריכה להתחילה ברכף הבתים 100.200 כדי להיות חלק מן הרשת.

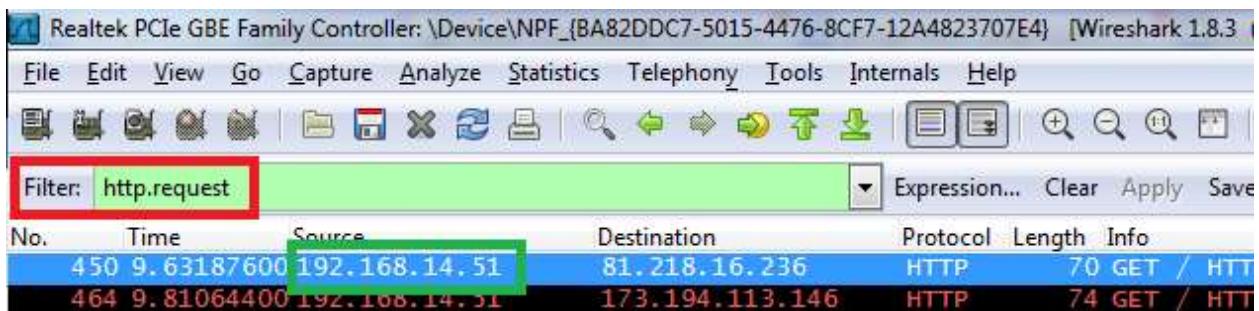


תרגיל 7.2 מודרך - מציאת כתובות ה-IP באמצעות הסנפה

מוקדם יותר בפרק, מצאתם את כתובות ה-IP שלכם באמצעות הפקודה `ipconfig`. על מנת לוודא שהכתובת שמצאתם היא הכתובת הנכונה, נשלח בקשה לאתר חיצוני (למשל ל-Google), ונסניף אותה באמצעות Wireshark. כאן נוכל כבר להסתכל לראשונה על חבילת IP.

בואו נעשו זאת ייחדי. פתחו את Wireshark והתחילו להסניף. השתמשו במסנן התצוגה (display filter) הבא: `http.request`

לאחר מכן, פתחו את הדפדפן האbove עליהם, וגלשו אל הכתובת `www.google.com`. כתעט, הפסיקו את ההסניפה. החלון שלכם אמור להראות בערך כך:



בריבוע **האDOI**, מוצג **display filter** בו השתמשTEM בCDI לסקן בקשות HTTP, כפי שלמדנו בפרק [שכבות האפליקציה/תרגיל 4.1 מודרך - התנסות מעשית בתקשורתHTTP](#).

כבר עתה, ניתן לראות בפקטה את כתובת המקור המסווגת בריבוע **הירוק**. שימו לב כי אכן מדובר בכתובת אותה מצאתם קודם לכן, באמצעות **ipconfig**.

כעת נסתכל גם בפקטה עצמה. הסתכלו בשדות השונים ב-Wireshark, ופתחו את שכבת ה-IP:

```
Internet Protocol Version 4, Src: 192.168.14.
Version: 4
Header length: 20 bytes
+ Differentiated Services Field: 0x00 (DSCP 0)
  Total Length: 56
  Identification: 0x2432 (9266)
+ Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 128
  Protocol: TCP (6)
+ Header checksum: 0x0000 [incorrect, should
  Source: 192.168.14.51 (192.168.14.51)
  Destination: 81.218.16.236 (81.218.16.236)
```

לא נסתכל על כל השדות עכשו, אך נשים לב שבשדה **Source** (כתובת המקור) אכן מצוינת כתובת ה-IP שראינו קודם.

כתובת היעד (Destination) של החבילה היא כמובן כתובת ה-IP של www.google.com. כך אנו רואים שבאמת חבילת ה-HTTP נשלחה מהמחשב שלנו (המקור) אל [Google](http://www.google.com) (היעד).

 מה חסר לנו?

از גילינו את כתובת ה-IP שלנו. עם זאת, משחו עדין חסר. בהינתן החומר שלמדנו בין היתר, נסו לחושב איזה פרט מידע חסר לנו לפני שתמשיכו לקרוא את השורה הבאה.

ובכן, כת ברשוטנו כתובת ה-IP המלאה שלנו. עם זאת, כפי שלמדנו, הכתובת מחולקת למזהה רשות ומזהה ישות (במקרה זה - מזהה המחשב שלנו בתוך הרשת). כיצד נדע מהם המזהאים?

לדוגמא, כיצד נדע האם המחשב בעל כתובת IP 192.168.0.5 נמצא בתחום הרשות? כמובן - עליינו לדעת, מתוך כתובת ה-IP שلونו, מהו מזחה הרשות ומהו מזחה הישות. מכאן שעליינו להבין אילו מהabitיטים מגדירים את מזחה הרשות, ואילו מהabitיטים מגדירים את מזחה הישות.



מהו מזחה הרשות שלי? מהו מזחה הישות?

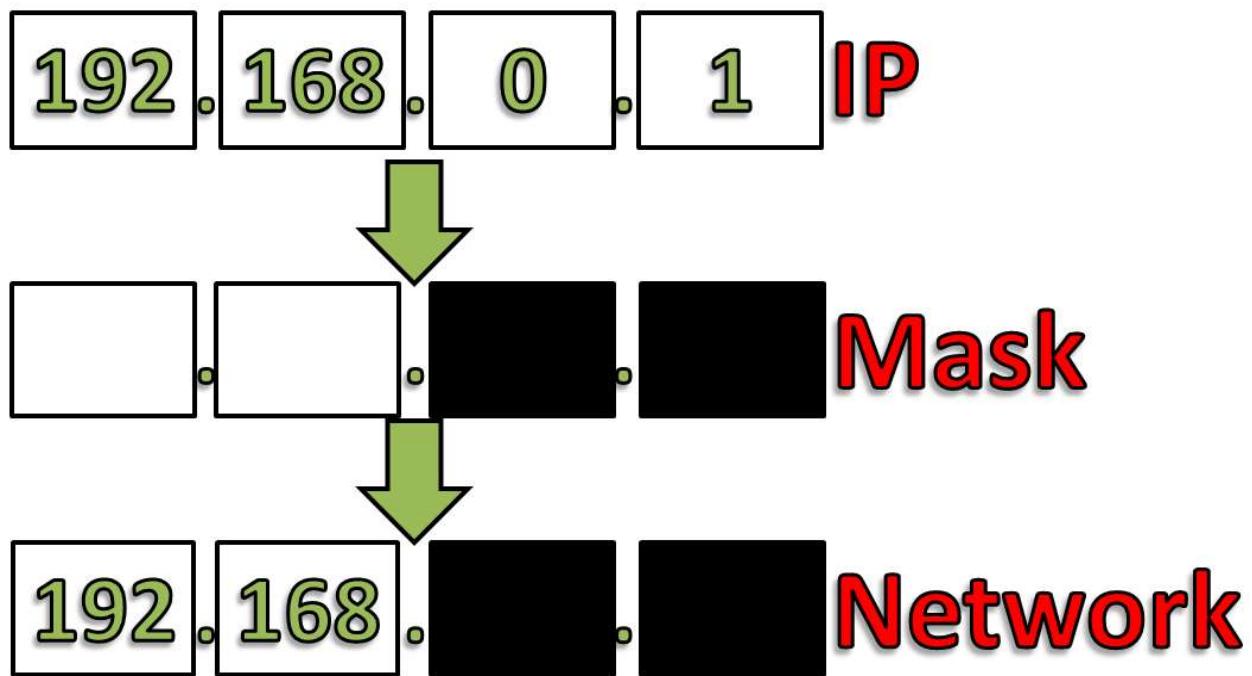
בכדי לענות על שאלה זו, עליינו ללמידה נוספת מונח חדש בשם **Subnet Mask** (טסית רשות). עבור כל כתובת IP, עליינו לדעת מהי ה-Subnet Mask שלה, על מנת לדעת מהו מזחה הרשות. ה-Subnet Mask מגדיר כמה ביטים (bits) מתוך כתובת ה-IP מייצגים את מזחה הרשות.

נשתמש בדוגמה. הביטו בכתובת הבאה: 192.168.0.1. נאמר שטסית הרשות שלה מוגדרת כ-16 ביטים (או שני בתים⁵²). מכאן ש-192.168.0.1 הינו מזחה הרשות, ו-0.1 הינו מזחה הישות. את מקורה זה ניתן להציג בדרכים שונות: 192.168.0.1/16 - הוספה "/"16" בסוף הכתובת מצינית שה-Subnet Mask מכיל 16 ביטים, כלומר שהוא מזחה הרשות היררכוני.

דרך נוספת היא לציין שהכתובת היא 192.168.0.1 וה-Subnet Mask הינו 255.255.0.0 (16 הביטים הראשונים דולקים, ולכן הם מזחה הרשות. 16 הביטים הבאים כבויים, ולכן הם מזחה הישות).

⁵² אם איןכם מרגשים עדין בטוחים במונחים "ביטים" ו-"בתים" אל תדאגו, הביטחון נרכש עם הזמן. עם זאת, קראו לאט וידאו כי אתם מבינים את הכוונה בדוגמאות שניתנות לפניכם.

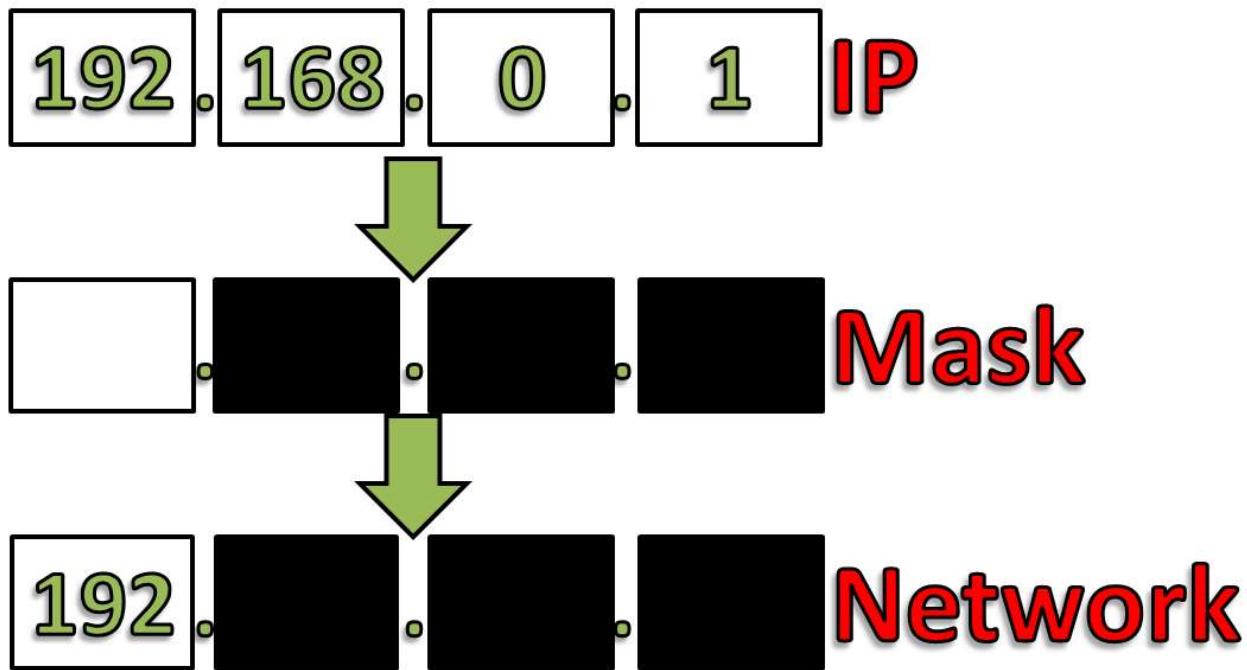
בדוגמה זו, המסכה "נראית" כהה:



כלומר, המסכה גורמת לנו להבין שרק 16 הביטים (שני הביטים הראשונים הם הרלבנטיים עבור מזזה הרשות, ובכך "מעליהם" את 16 הביטים (שני הביטים) הנוטרים.

בואו נראה דוגמאות נוספות:

עבור הכתובת 192.168.0.1/8, המסכה נראה כך:



כמובן, המסכה גורמת לנו להבין שרק 8 הביטים הראשונים (כלומר הבית הראשון) הם הרלוונטיים עבור מזהה הרשת, ובכך "מעלימה" את 24 הביטים (שלושת הבטים) הנדרשים.

תשובות על השאלות הבאות עבור כתובת ומסכה אלו (192.168.0.1/8):

האם הכתובת 192.168.0.2 נמצאת באותה הרשת? התשובה היא כן - הרי יש לה את אותו מזהה הרשת (192).

האם הכתובת 192.5.0.2 נמצאת באותה הרשת? התשובה היא כן - הרי יש לה את אותו מזהה הרשת (192).

שים לב לדבר זה בכך כיון שמדובר רק ב-8 ביטים, ככלומר את 192, ולא מתחשב למעשה בבית השני, המכיל את הערך 168.

האם הכתובת 100.200.0.1 נמצאת באותה הרשת? התשובה היא לא - כיון שמדובר רשת אחרת (לא 192).

את אותה הכתובת עם מסכת הרשת ניתן היה גם להציג כך: הכתובת 192.168.0.1, המסכה: 0.255.0.0.0.



שים לב: עליה לנו כאן נקודה מעניינת. מחשב בעל כתובת ה-IP הבאה: 192.160.0.1, הינו חלק מאותה הרשת של המחשב 192.168.0.2/8, אך לא חלק מאותה הרשת של המחשב 192.168.0.2/16. מכאן שעל מנת לדעת אילו ישויות נמצאות באותה הרשת, علينا להבין גם את ה-Subnet Mask, ולא רק את כתובת ה-IP.

הערה: מזהה הרשת מוגדר באמצעות מספר ביטים (bytes) ולא בתים (bits), ולכן מסכת רשת יכולה להיות מוגדרת לא רק כמספר שmagdir בתים שלמים (8, 16, 24), אלא גם באמצעות מספר ביטים בווד (למשל 9 או 23). לא נתעכט על נקודה זו בספר זה.

נחזיר לשאלת השאגנו קודם:



מהו מזהה הרשת שלי? מהו מזהה הישות?

נסו לענות על כך בעצמכם.

נסתכל קצת בדוגמה שהאגנו קודם, באמצעות הפקודה **ipconfig**:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix  . : privatebox
  Link-local IPv6 Address . . . . . : fe80::419b:ac69:cfb6:705%11
    IPv4 Address . . . . . : 192.168.14.51
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.14.1
```

אנו רואים שמסכת הרשת שלנו היא 255.255.255.0, כלומר מזהה הרשת מוגדר באמצעות 24 ביטים (או 3 בתים).

נחזיר לשאלת נוספת שהאגנו מוקדם יותר: האם מחשב בעל כתובת 192.168.0.5 נמצא באותו הרשת?

התשובה היא - לא. זאת מכיוון שמדובר מזהה הרשת שלנו מוגדר באמצעות 24 ביטים, והוא למעשה כתוב ב-192.168.14.0. הכתובת שהאגנו אינה מכילה את מזהה הרשת זה, ולכן המחשב בעל כתובת זו אינו נמצא באותו הרשת כמו המחשב שעלי הרצינו את הפקודה **ipconfig**.



מצאו את כתובת ה-IP שלכם ואת ה-Subnet Mask שלכם. כתבו כתובת IP אחת שנמצאת אתם באותה הרשת, וככתובת IP אחת שלא נמצא אתם באותה הרשת.



כעת הסניפו את הרשת שלכם ממש דקוט. הסתכלו בקובץ ההסנהה, ומצאו את כתובות ה-IP השונות שבו. אילו כתובות נמצאות ברשת שלכם? אילו כתובות לא?

כתובות IP מיוחדות

"שנן כמה כתובות IP ש모וגדרות כ"כתובות מיוחדות", ושווא להכיר אותן. כאמור, נכיר רק חלק מהן:

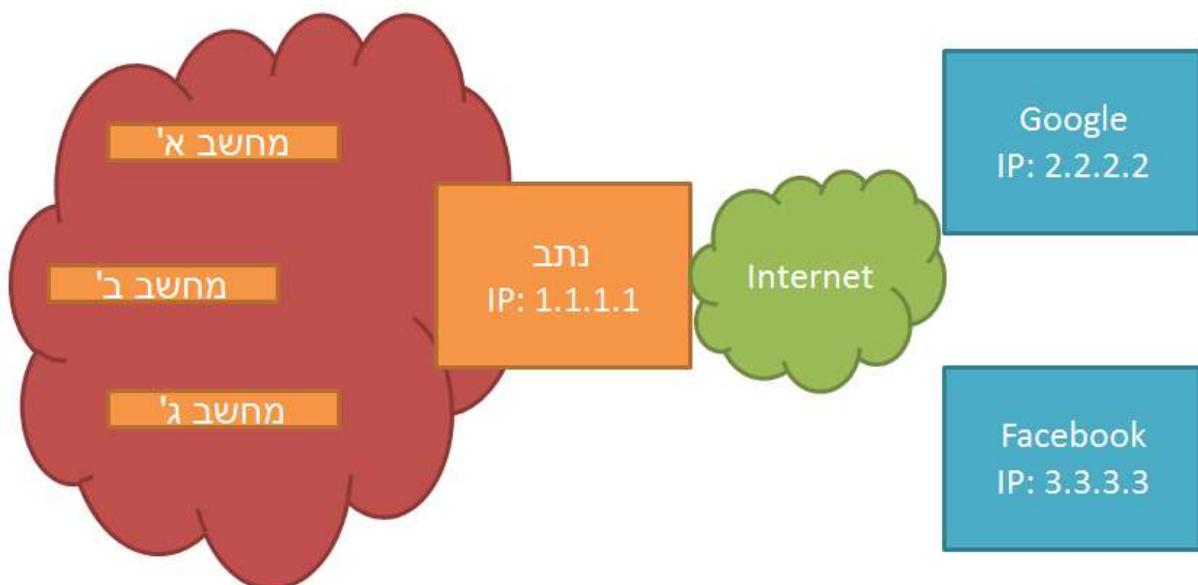
- 255.255.255.255 - כתובות זו היא כתובות Broadcast. הכוונה היא שחבילה שנשלחת לכתובות זו מיועדת לכל הishiות ברשת. לדוגמה: ברשות בה יש ארבעה מחשבים: של דני, דינה, אורית ואורי, אם דני שולח חבילה לכתובות היעד "255.255.255.255", היא תגיע לדינה, אורית ואורי - כלומר לכל שאר המחשבים ברשות.
- 127.0.0.0/8 - כתובות "Loopback", הנקראות גם "Local Host". כתובות אלו מציניות למשזה שהחביבה לא צריכה לעזוב את כרטיס הרשת, אלא "להישאר במחשב" (בפועל - נשלחת לכרטיס הרשת הווירטואלי של מערכת הפעלה). הכתובת המוכרת ביותר הנמצאת בטוווח זה היא הכתובת 127.0.0.1, אך מכיוון שמדובר ברשת הינו בגודל 8 ביטים (או בית אחד), לכתובות 127.5.6.7 (לדוגמה) יש את אותה המשמעות.

כאמור, ישן כתובות מיוחדות נוספות עליהן לא נרchie'ב בשלב זה.

כתובות פרטיות ו-NAT

החל מסוף שנות ה-80' - נוצרה בעולם בעיה אמיתית ומוחשית - נגמרו כתובות ה-IP. מסיבות שונות, נוצר מצב שבו למרות IPv4 מספק כמעט 4.3 מיליארד⁵³ כתובות שונות, התבצעה הקזאה לא יילה שלhn ולא נותרו כתובות IP שנית היה להקצתו לרכיבי רשת חדשים שהזדקקו להן. בתחילת שנות ה-90', כשה האינטרנט זכה לגיליה מהירה מאוד, המהסור בכתובות ה-IP החל לפגוע בספקיות האינטרנט שפשות לא יכול להקצת כתובות IP לлокחות שלhn.⁵⁴

.(Network Address Translation) NAT, קוראים NAT נוצר אפוא צריך למצוא פתרון מהיר לבעה. לפtron זהה, נביט בתמונה הרשות הבאה: על מנת להסביר את הרעיון הכללי, נביט בתמונה הרשות הבאה:



לפנינו נמצאת "הרשות האדומה", ובها שלושה מחשבים. הרשות מחוברת, באמצעות הנטב בעל כתובת ה-IP של 1.1.1.1, אל האינטרנט. בנוסף, ישנו השירותים של Google ו-Facebook אשר ירצו מחשבים ברשות לגשת. עד אשר החל השימוש ב-NAT, היה צריך לספק כתובת IP ייחודית לכל אחת מהישויות. כמו שמחשב א', מחשב ב' ומחשב ג', יזכו כל אחד לכתובת IP אמיתית ייחודית בעולם. דבר זה חיובי מהרבה בחינות, אך במצבות בה אין כבר כתובות IP לתת - הדבר לא נכון. אי לכך, נוצר הרעיון של NAT. לפי רעיון זה, כל הישויות בתוך הרשות - מחשב א', מחשב ב' ומחשב ג', יקבלו **כתובות פרטיות** - כלומר כתובות שאין אותן בתוך הרשות בלבד, ולא בעולם החיצוני. כתובות אלו אינן ניתנות לניתוב - כמובן, נתב באינטרנט שראה חבילת שמיעדת לכתובת שכזו עתיד "לזרוק" אותה.

⁵³ מיליאון שבארבעה בתים (bytes) יש 32 ביטים (bits), שקל אחד מהם יכול להיות 0 או 1. אי לכך, מספר האפשרויות הינו

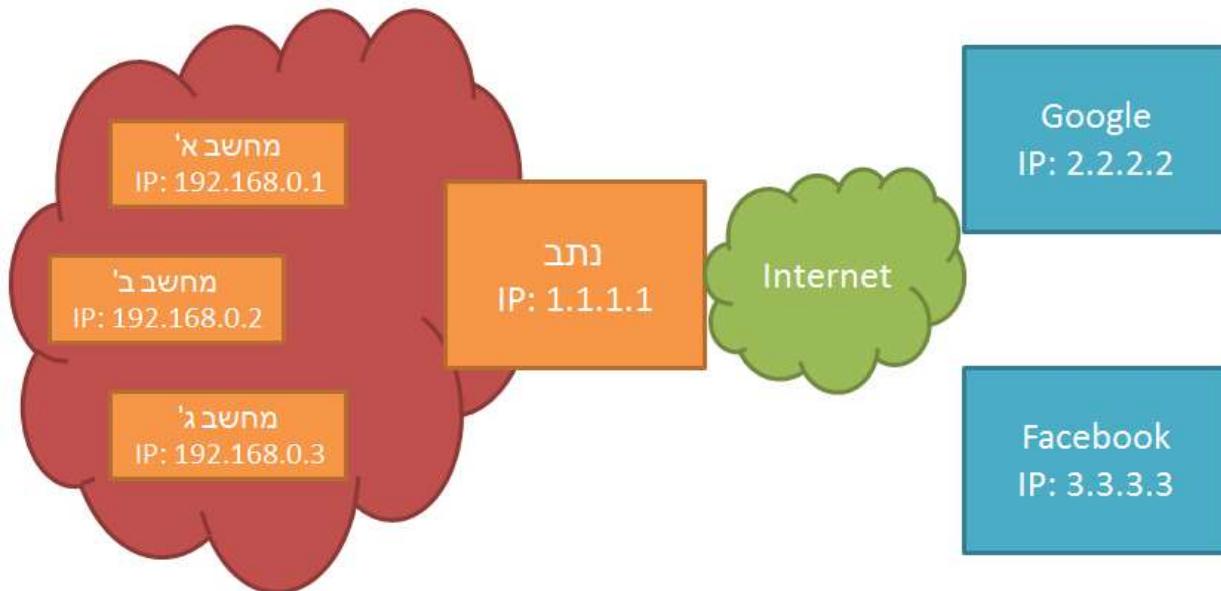
⁵⁴ 2³²

⁵⁴ לקריאה נוספת על תופעה זו - קראו בעמוד: http://en.wikipedia.org/wiki/IPv4_address_exhaustion

לשם כך, הוגדרו שלושה טוווחים של כתובות פרטיות:

- 10.0.0.0/8 - בטוח זה ישן 16,777,216 כתובות.
- 172.16.0.0/12 - בטוח זה ישן 1,048,576 כתובות.
- 192.168.0.0/16 - בטוח זה ישן 65,536 כתובות.

לצורך הדוגמא, הרשת שלנו עכשו תיראה כך:



בצורה זו, הרשת האדומה "ביזזה" רק כתובת IP אחת - זו של הנטב שלב, ולא ארבע כתובות כמו שהוא היה צריך לפני השימוש ב-NAT.

עם זאת, כיצד תצליח הרשת לעבוד? כיצד יצליח עכשו לפנות מחשב א', שהינו בעל כתובות פרטיות שאסור לנטב, אל Google? חמור מכך - כיצד Google יצליח להחזיר תשובה אל כתובת IP פרטית שאסור לנטב, וכן שיכת להרבה רכיבים שונים ברחבי העולם?

באופן כללי, התהליך יעבוד כך:

בשלב הראשון, מחשב א' ישלח הודעה ממנה (כתובת המקור: 192.168.0.1) אל Google (כתובת היעד: 2.2.2.2). את החביליה הוא ישלח אל הנטב.

בשלב השני, הנטב ⁵⁵ יקבל את החביליה, ויחליף בה את כתובת המקור לכתובת שלו. כלומר, החביליה עכשו תשלח מכתובת המקור 1.1.1.1, אל כתובת היעד 2.2.2.2.

⁵⁵ מימוש NAT לא חייב להתבצע אצל הנטב של הרשת. עם זאת, בפועל, ברוב המקרים הנטב הוא אכן זה שמשמש את NAT.

בשלב השלישי, הרשות של Google קיבל את החביליה. שימו לב: הרשות של Google כלל לא מודע לכתובת ה-IP של מחשב א', או לעובדה ישינה ישות רשות מאחוריו האינטרנט. הוא מודע אך ורק לשיטת הרשות בעלת הכתובת 1.1.1.1, וזהו האינטרנט של הרשות. בעת, Google יעבד את הבקשה של מחשב א', וישיב עליה - אל האינטרנט. כאמור, מבחןינו של Google, הוא קיבל את הבקשה באופן ישיר מן האינטרנט.

בשלב הרביעי, האינטרנט קיבל את התשובה מהרשות של Google. החביליה תכיל את כתובת המקור של הרשות של Google (כלומר 2.2.2.2), וכתובת היעד של האינטרנט (1.1.1.1). בעת, האינטרנט יבין שמדובר בחביליה המיועדת למשה אל מחשב א'. אי כך, הוא יבצע **החלפה של כתובת היעד**. ככלומר, הוא ישנה את כתובת היעד מ-1.1.1.1 ל-192.168.0.1, ועביר את החביליה למחשב א'.

בשלב החמישי, מחשב א' קיבל את הودעת התשובה מ-Google. מבחינת מחשב א', החביליה הגיעה מכתובת ה-IP של Google (שהיא 2.2.2.2), היישר אל הכתובת שלו (192.168.0.1), ולכן מבחןינו מדובר בתהיליך "רגיל" לכל דבר.

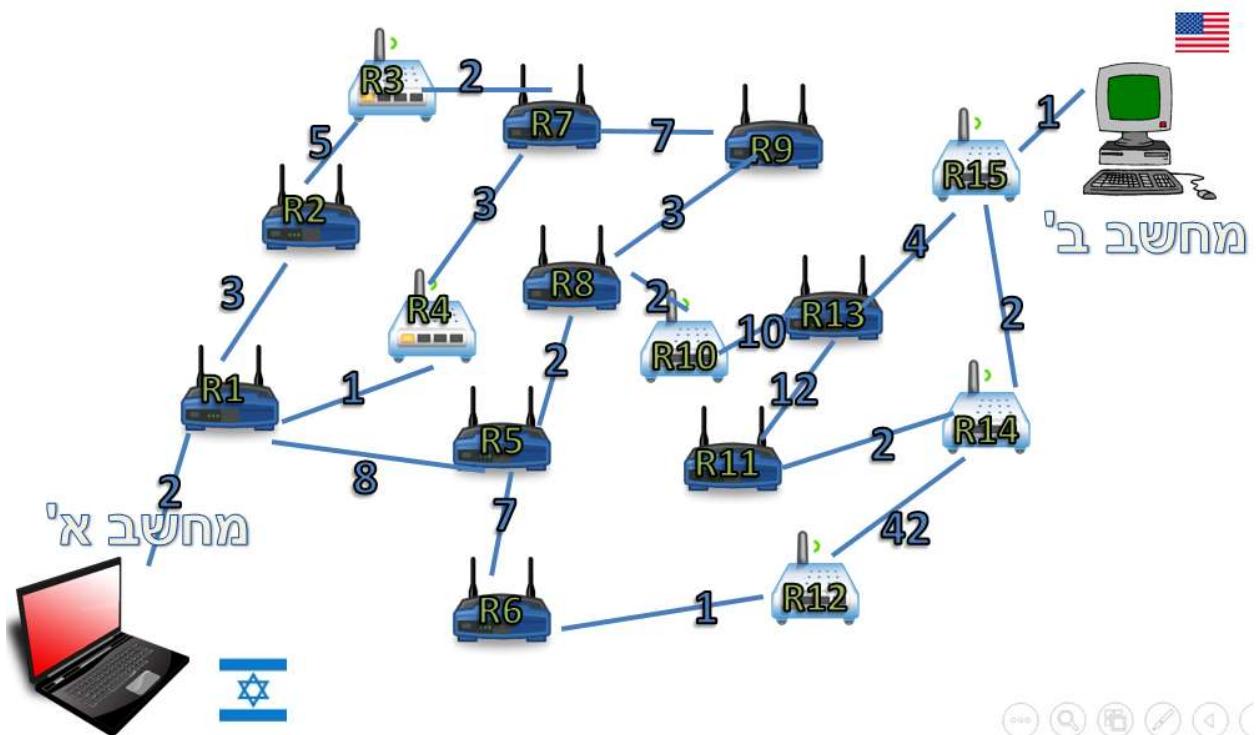
בצורה זו מצליחות ישיות רשות מתוך הרשות האדומה שלנו לתקשר עם רכיבים מחוץ לאינטרנט, על אף שאין להם כתובת IP ייחודית. עם זאת, ישנה סוגיה לא פתרה - כיצד, בשלב הרביעי שהציגו, יידע האינטרנט שהחביליה שהגיעה מ-Google מיועדת למשה אל מחשב א' ולא למחשב ב'? כיצד הוא ידע לעשות זאת מידת שמחשב א' ומחשב ב' פנו שניהם, באותו הזמן בדיק, אל Google? על מימושים שונים של NAT לא נתעכט בספר זה, אך אתם מוזמנים להרחיב על כך בעמוד: http://en.wikipedia.org/wiki/Network_address_translation

ניטוב

עכשו כשאנו יודעים כיצד בנוייה כתובת IP, הגיע הזמן להתרכז בתהיליך הניטוב. הסברנו קודם שימושות הניטוב היא ההחלטה על הדרך שבה חביליה תעבור בין שתי נקודות. אם משתמש שוב בהקלה לעולם הרכיבים, הריצף הניטוב הוא ההחלטה על הדרך בה הרכיב צריך לנוטע בכדי להגיע מנקודת המוצא אל היעד. הרכיבים שמבצעים את רוב מלאכת הניטוב בעולם רשתות המחשבים, נקראים **נתבים**.

נתב (Router)

הנתב (Router) הוא רכיב רשת בשלב שלישי. מטרתו היא לקשר בין מחשבים ורשתות ברמת ה-IP. נזכר בתרשימי הרשות שהציגו קודם:



כל רכיב בדרך כאן הינו למעשה נתב, ומכאן גם נובע הייצוג שלו (R1 הוא למעשה קיצור עבור "Router 1"). על הנתב לקבל כל חבילה, ולהחליט איך לנצל אותה הלאה בדרך הטובה ביותר.

ציוו שנתב הוא רכיב של שכבה הרשת. מה המשמעות של קר? לכל נתב יש כתובת IP משלה⁵⁶. מעבר לכך, הנתב "מבין" את שכבת הרשת - הוא יודע מה היא כתובת IP, מכיר את מבנה חבילת ה-IP, קורא את ה-Header (תחילית) של החבילה ומתקבל החלטות בהתאם.

למעשה, כאשר ביצענו traceroute קודם לכן וקיבliśmy את הדרך שאויה עברה חבילה מהמחשב שלנו ואל www.facebook.com, קיבלנו את **רשימת הנתבים** אצלם עברה החבילה בדרך. בקרוב נבין כיצד ניתן ליצור רשימה זו, ככלומר - איך traceroute פועל.

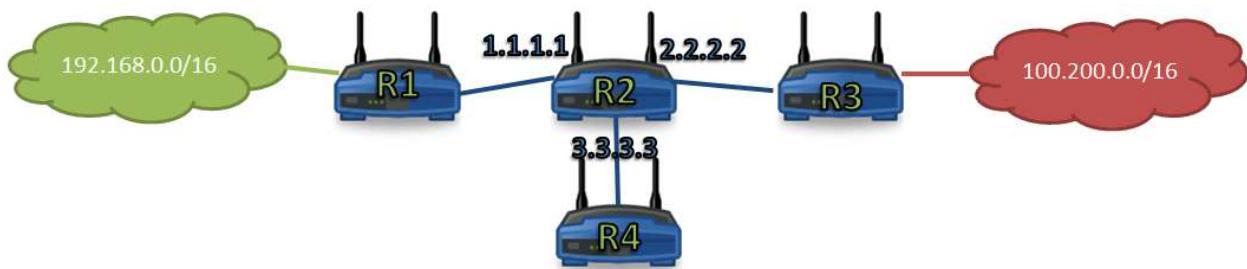
האם נתב מחליט לאן לנצל?

נסתכל בתמונה לעיל, ונניח כי נתב R1 קיבל את החבילה של מחשב א', שנשלחה אל מחשב ב'. כיצד יודע הנתב R1 אם להעביר את החבילה אל הנתב R2, אל R4 או אל R5?

⁵⁶ ישנו גם נתבים בפרוטוקולים שאינם פרוטוקולי IP. עם זאת, מטעם הנוחות, נניח במהלך הספר שכל הנתבים הם נתונים IP.

על מנת להחליט כיצד לנתח את החבילה, לכל נתב יש **טבלת ניתוב (Routing Table)** משלו. טבלה זו מתחארת لأن יש להעביר כל חבילה שmagiuha אל הנתב. ברוב המקרים, הטבלה היא דינמית - כלומר, היא יכולה להשתנות בהתאם למצב הרשת. היזכרו בתרגיל שביצענו קודם לכן בפרק זה, בו מצאנו את הדרך ה"זולה" ביותר להגעה מנקודה אחת ברשת לנקודה אחרת. עקב מספר תקלות - "מצב הרשת" השתנה, שכן חלק מהמחיבורים לא היו תקינים עוד. בעקבות כך, הנתבים בדרכם צריכים לשנות את דעתם על הרשות ולהחליט על דרך חדשה לננתב.

נביט לדוגמה בתמונה הרשות הבאה (הערה: מדובר בתמונה רשות חלקית בלבד):



בתמונה הרשות הזו ישנו ארבעה נתבים. הנתב R1 מחובר באופן ישיר לרשות בעל המזהה 192.168.0.0/16. הנתב R3 מחובר באופן ישיר לרשות בעל המזהה 100.200.0.0/16. הנתב R2 מחובר באופן ישיר לנתבים R1 ו-R3, וגם לנתב נוסף בשם R4.

שים לב שננתב R2 נמצא למעשה בשלוש רשתות שונות: הרשת שלו ושל R3, הרשת שלו ושל R1, והרשת שלו ושל R4. עברו כל אחת מהרשתות האלה, ל-R2 יש כתובות IP אחרות.

- כתובות ה-IP של R2 ברשות שלו ושל R1 הינה: 1.1.1.1
- כתובות ה-IP של R2 ברשות שלו ושל R3 הינה: 2.2.2.2
- כתובות ה-IP של R2 ברשות שלו ושל R4 הינה: 3.3.3.3

כל אחד מהנתבים עשוי להיות מחובר גם לנתבים נוספים, אך נועלם מכך לצורך ההසבר.

אל הנתב R2 מגיעה חבילה, כשכתובת היעד שלה היא: 100.200.5.8. כיצד ידע הנתב R2 אל מי להעביר את החבילה? אם ל-R1, ל-R3 או ל-R4?

על מנת לענות על שאלת זו, נסתכל בטבלת הניתוב של נתב R2:

מספר שורה	יעד (Destination)	מסכת רשת (Mask)	ממשק (Interface)
1	0.0.0.0	0.0.0.0	3.3.3.3
2	192.168.0.0	255.255.0.0	1.1.1.1
3	100.200.0.0	255.255.0.0	2.2.2.2

על הנטב להסתכל בטבלת הניתוב, ולראות לאיזה **הרשומות הניתוב** (שורות בטבלה) שלו היא מתאימה. את החיפוש שלו מבצע הנטב מלמטה למעלה.

נבחן יחד את הפעולה של הנטב. ראשית, הוא מסתכל ברשומה התחתונה ביותר - רשומה מספר שלוש, המייצגת את הרשת 100.200.0.0 עם המסכה 255.255.0.0. כתה הוא שואל את עצמו:
"אם הכתובת 100.200.5.8 שייכת לרשת זו?"

התשובה היא, כפי שלמדנו, כן. הרי מזאה הרשת הינו 100.200, והכתובת 100.200.5.8 אכן תואמת את מזאה זה.

אי לכך, החבילה מתאימה לחוק המצוין בשורה 3, ولكن הנטב **יעביר (forward)** את החבילה אל הממשק⁵⁷ R3, כולם אל R3. בטורו, R3 יעביר את החבילה הלאה, עד שזו תגיע אל הישות בעלת הכתובת 100.200.5.8.

נראה דוגמה נוספת. כתה הגיעו אל הנטב חביבה עם כתובת היעד 192.168.6.6. הנטב יבצע את הפעולות הבאות:

בטור התחלת, הוא ינסה לבדוק האם הכתובת מתאימה לרשות הניתוב الأخيرة שיש לו, כולם לרשותה מספר שלוש. לשם כך הוא שואל:
"אם הכתובת 192.168.5.8 שייכת לרשת זו?"

התשובה היא לא, זאת מכיוון שמזאה הרשת הינו 100.200, והכתובת 192.168.5.8 אינה עונה על מזאה זה. כתה, הנטב ימשיך לרשותה הבאה, רשומה מספר שניים, אשר מתארת את הרשת 192.168.0.0 עם המסכה 255.255.0.0. הנטב שוב שואל:
"אם הכתובת 192.168.5.8 שייכת לרשת זו?"

התשובה היא כן, שכן מזאה הרשת הינו 192.168, והכתובת 192.168.5.8 אכן עונה על מזאה זה. אי לכך, החבילה מתאימה לחוק זה, והנטב יעביר את החבילה על הממשק 1.1.1.1, כולם אל R1. בטורו, R1 יעביר את החבילה הלאה, עד שזו תגיע אל הישות בעלת הכתובת 192.168.5.8.

נראה דוגמה שלישיית. כתה הגיעו אל הנטב חביבה עם כתובת היעד 5.5.5.5. הנטב ינסה לבדוק האם הכתובת מתאימה לרשות הניתוב الأخيرة שיש לו, כולם לרשותה מספר שלוש, המתארת את הרשת 100.200.0.0/16. בשלב זה אנו כבר מבינים שהכתובת לא נמצאת ברשף המתוארת ברשימה זו, ולכן הנטב

⁵⁷ המילה "ממשק" מתארת כרטיס רשת. לנטב יש מספר כרטיסי רשת, וכל אחד מהם מתואר באמצעות כתובת IP אחרת. בדוגמה זו, המשמעות של "ממשק 2.2.2.2", היא כרטיס הרשת בעל הכתובת 2.2.2.2, כולם הכרטיס המחבר את הנטב R2 אל הנטב R3.

יעבור אל הרשומה הבאה, המתארת את הרשת 192.168.0.0/16. גם כאן, הכתובת 5.5.5.5 אינה חלק מהרשות, ולכן הנטב יעבור אל הרשומה الأخيرة בטבלה.
כעת, הנטב שואל את עצמו - "אם הכתובת 5.5.5.5 היא חלק מהרשות 0/0?"?

התשובה לשאלה זו היא - כן. למעשה, החוק שמתאר את הרשת 0.0.0.0 עם המסכה 0.0.0.0 הוא חוק גנרי, וכל כתובת IP תואמת אליו. היהות שמסכת הרשת היא בגודל 0 ביטים, המשמעות היא שככל כתובות IP שהיא - תהיה חלק מן הרשת זו.

אי לכך, הנטב יעביר את החבילה אל הממשק 3.3.3.3 - כלומר אל R4, שבתורו ימשיך את הטיפול בחבילה. מכאן אנו למדים למעשה שככל חבילה אשר הנטב לא מעביר אל הנטים R1 או R3, הוא צפוי להעביר אל R4.



מהי טבלת הניתוב שלי?

גם למחשבים, בדומה לנטים, יש טבלתנית. על מנת לראות את טבלת הניתוב שלכם, היכנסוikut ל-**:route print**, והקישו את הפקודה Command Line

```
C:\Users\USER>route print
=====
IPv4 Route Table
=====
Active Routes:
Network Destination      Netmask     Gateway       Interface   Metric
          0.0.0.0        0.0.0.0   192.168.14.1  192.168.14.51    20
         127.0.0.0        255.0.0.0   On-link      127.0.0.1    306
         127.0.0.1        255.255.255.255  On-link      127.0.0.1    306
  127.255.255.255        255.255.255.255  On-link      127.0.0.1    306
         192.168.14.0        255.255.255.0   On-link      192.168.14.51   276
         192.168.14.51        255.255.255.255  On-link      192.168.14.51   276
         192.168.14.255        255.255.255.255  On-link      192.168.14.51   276
         224.0.0.0          240.0.0.0   On-link      127.0.0.1    306
         224.0.0.0          240.0.0.0   On-link      192.168.14.51   276
  255.255.255.255        255.255.255.255  On-link      127.0.0.1    306
  255.255.255.255        255.255.255.255  On-link      192.168.14.51   276
```

לעת עתה, נתעלם מהעמודות "Network Destination", "Netmask", "Metric"- ו-"Gateway", ונתעמק בעמודות "Interface" ו-"Interface".

ראוי לציין שניין לראות שני ממשקים (Interfaces) למחשב זה:

- הממשק בעל הכתובת "192.168.14.51". זוכרים שמצאנו את כתובות זו בתחילת הפרק? זהו כרטיס הרשת שלנו.
- הממשק בעל הכתובת "127.0.0.1". זוכרים שדיברנו על הכתובת זו קודם קודם תחת סעיף כתובות מיוחדת? חבילות שנשלחות לממשק זה לא באמת יגיעו לכרטיס הרשת, אלא "ישארו במחשב".

שימוש לב לרשותה הראשונה בטבלה: אותה רשותה שעלייה תאמת כל כתובת IP, זו שמתארת את הרשת :0.0.0.0/0

Network	Destination	Netmask	Gateway	Interface	Metric
	0.0.0.0	0.0.0.0	192.168.14.1	192.168.14.51	20

רשומה זו מתרמת את ה-**Default Gateway** של המחשב - כלומר מי הנטב המשויך אל המחשב. כל חבילה שלא התאימה על חוק ספציפי בטבלת הנטוב, תשלח אל ה-Default Gateway. מכאן שכל חבילה שתשלח אל הממשק "192.168.14.1", תשלח למעשה אל הנטב 192.168.14.51.



תרגיל 7.3 – ניתוב על פי טבלת ניתוב

על פי טבלת הנטוב שלעיל, ענו על השאלות הבאות:

1. لأن תשלח חבילה עם כתובת היעד "?255.255.255.255"?
2. لأن תשלח חבילה עם כתובת היעד "?192.168.14.51"
3. لأن תשלח חבילה עם כתובת היעד "?127.0.0.1"
4. لأن תשלח חבילה עם כתובת היעד "?127.5.5.6"
5. لأن תשלח חבילה עם כתובת היעד "?1.2.3.4"

ICMP

כעת נלמד על פרוטוקול נוסף - ICMP, ששמו המלא הוא: Internet Control Message Protocol. פרוטוקול זה נועד לעזור לטכנאים ולט (אנשים המעוניינים להבין לעומק את דרך הפעולה של רשתות מחשבים) למצוא תקלות ברשת ולהבין את מצב הרשת.

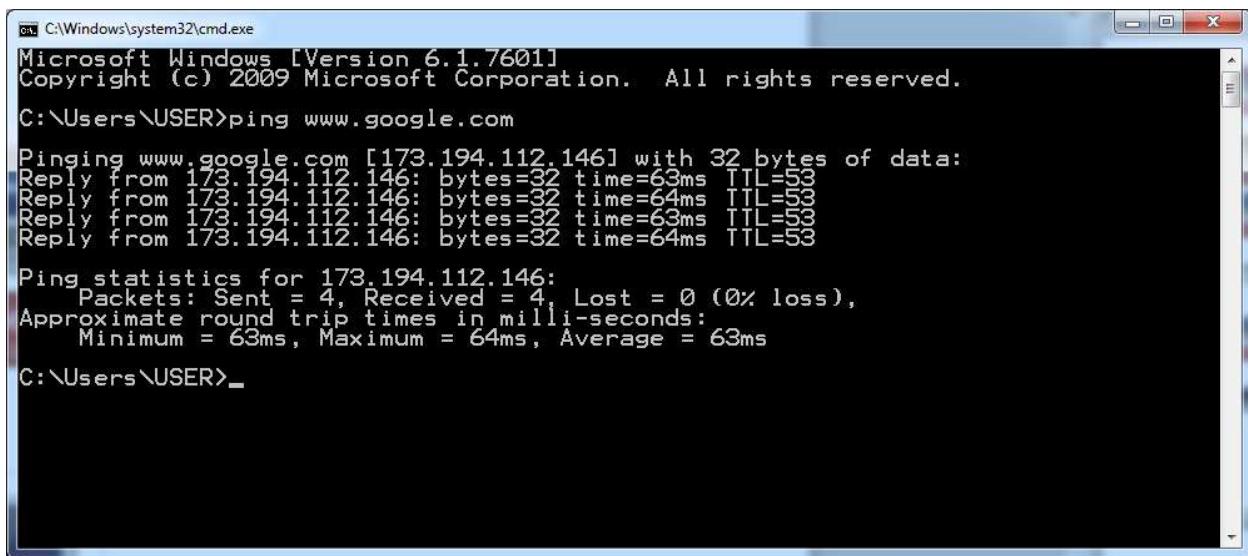
הכלי המוכר ביותר המשמש בפרוטוקול ICMP הוא הכל*ping*, אשר פגשנו מספר פעמים לאורך הספר. כלי זה נועד בכדי להבין האם ישות מסוימת "למעלה" - כלומר דילוקת, עובדת ומגיבת לשילוח הודעות אליה. לחופין, הוא יכול לוודא שיש תקשורת מהמחשב שלנו אנו מרכיבים את הפקודה אל הרשת אליה הוא מחובר.



תרגיל 7.4 מודרך - בדיקת קישורים ל-Google

נסו זאת בעצמכם - בצעו ping ל-"www.google.com", וידאו שאתם מקבלים תשובה. פעולה זו יכולה לעזור לנו לוודא אם המחשב שלנו מחובר לאינטרנט - מכיוון שלא סביר ש-Google "נפל", אנו יכולים לשלווה אליו ping ולקווות לתשובה. אם לא קיבלנו תשובה, נראה שיש בעיה כלשהי אצלנו.

המסמך שלכם אמר להראות פחות או יותר כך:



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>ping www.google.com

Pinging www.google.com [173.194.112.146] with 32 bytes of data:
Reply from 173.194.112.146: bytes=32 time=63ms TTL=53
Reply from 173.194.112.146: bytes=32 time=64ms TTL=53
Reply from 173.194.112.146: bytes=32 time=63ms TTL=53
Reply from 173.194.112.146: bytes=32 time=64ms TTL=53

Ping statistics for 173.194.112.146:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 63ms, Maximum = 64ms, Average = 63ms

C:\Users\USER>_

```

כפי שניתן לראות, **ping** מציג את כתובת ה-IP של "www.google.com", שולח באופן בירית מחדל ארבע הודעות ומספר לנו כמה מהן הגיעו לעדן ונענו, ובאיזה מהירות.



איך Ping עובד?

מה, לא הסנפתם כשהרצתם קודם לכן את **ping**? ובכן, אתם אכן עדין חדשים בעולם הרשות. פתחו עתה את Wireshark, הסניפו והריצזו שוב את פקודת **the-ping** שהרצתם קודם. אל תשחו להשתמש ב-filter בכך לسان את הפקחות הללו רלבנטיות:

No.	Time	Source	Destination	Protocol	Length	Info
19	4.90664200	192.168.14.51	173.194.113.148	ICMP	74	Echo (ping) request id=0x0001, seq=131/33536, ttl=128
20	4.98088400	173.194.113.148	192.168.14.51	ICMP	74	Echo (ping) reply id=0x0001, seq=131/33536, ttl=51
21	5.90732300	192.168.14.51	173.194.113.148	ICMP	74	Echo (ping) request id=0x0001, seq=132/33792, ttl=128
22	5.98162000	173.194.113.148	192.168.14.51	ICMP	74	Echo (ping) reply id=0x0001, seq=132/33792, ttl=51
24	6.90834300	192.168.14.51	173.194.113.148	ICMP	74	Echo (ping) request id=0x0001, seq=133/34048, ttl=128
25	6.98264600	173.194.113.148	192.168.14.51	ICMP	74	Echo (ping) reply id=0x0001, seq=133/34048, ttl=51
29	7.91034000	192.168.14.51	173.194.113.148	ICMP	74	Echo (ping) request id=0x0001, seq=134/34304, ttl=128
30	7.98515900	173.194.113.148	192.168.14.51	ICMP	74	Echo (ping) reply id=0x0001, seq=134/34304, ttl=51

cut, נסתכל באחת החבילות שנשלחו. שימו לב לבחור בחבילת בקשה (request):

```

Frame 18: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 173.194.113.146 (173.194.113.146)
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x4cd1 [correct]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 138 (0x008a)
Sequence number (LE): 35328 (0x8a00)
[Response In: 19]
Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]

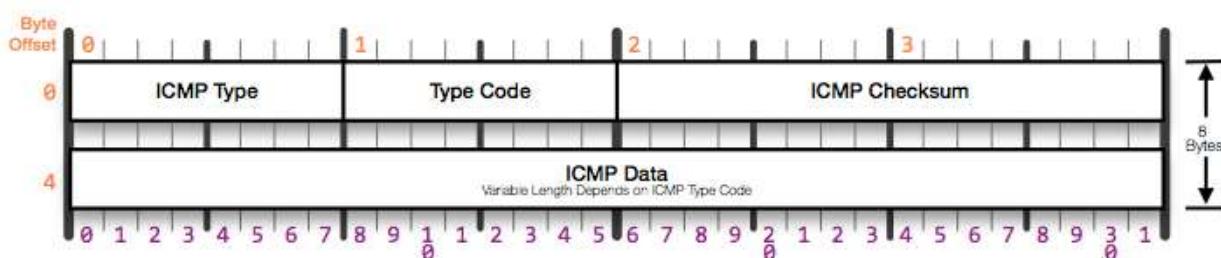
0000 00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00 . . . . . . . .
0010 00 3c 01 b4 00 00 80 01 4a dd c0 a8 0e 33 ad c2 . < . . . . . 3 . .
0020 71 92 08 00 4c d1 00 01 00 8a 61 62 63 64 65 66 q . . L . . . abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcde fg hi

```

נשים לב למודל השכבות:

- בשכבה השנייה, ניתן לראות את השימוש ב프וטוקול Ethernet⁵⁸. דבר זה מלמד כי כרטיס הרשות ממנו שלחה החבילה הוא כרטיס מסוג Ethernet⁵⁹.
- בשכבה השלישית, י箇נו שימוש בפרוטוקול IP. כתובות המקור היא הכתובת של המחשב שליח את ה- ping, וכתובות היעד היא הכתובת של Google.
- לאחר מכן, בהתאם לשכבה השלישית, י箇נו פרוטוקול ICMP.

cut נבחן את מבנה ה-Header של חבילת ICMP:



ה-Header של חבילת ICMP הוא לעולם בגודל קבוע של 8 בתים:

- בית 0⁶⁰ - Type: כולל את סוג החבילה. ישנו סוגים שונים של חבילות ICMP. בקרוב נראה דוגמה.
- בית 1 - Code: זה למעשה תת-סוג של Type-Code שהוגדר בביית הקודם. שוב, בקרוב נראה דוגמה.
- בתים 2-3 - Checksum: ערך שמחושב על שדות ה-Header והמידע של ICMP. מועד כדי לוודא שאין שגיאות בחבילה.⁶¹.

⁵⁸ על השכבה השנייה בכלל, ופרוטוקול Ethernet בפרט, נרחב בפרק הבא.

⁵⁹ יתכן שבמחשב שלכם תראו שכבה שנייה אחרת, בהתאם לכרטיס הרשות שלכם. למשל, יתכן ותראו כי מדובר בשכבה שנייה של WiFi, אם אתם מחוברים באמצעות כרטיס רשת WiFi.

⁶⁰ שימו לב לכך שהבית הראשון הוא תמיד בית באינדקס 0. הבית השני הוא בית באינדקס 1, וכך הלאה.

⁶¹ להזכירם, למדנו על [Checksum](#) בפרק [checksum התعبורה/ מה זה?](#)

- בתים 7-4 - כל השאר: ערך זהה יהיה שונה בהתאם לסוג החבילה, שהוגדר ביד' הบทים Type ו-Code.

נבחן לחבילה שלחנו ל-Google ובו נקבע בمزחים:

```

Frame 18: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 173.254.14.191
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4cd1 [correct]
        Identifier (BE): 1 (0x0001)
        Identifier (LE): 256 (0x0100)
        Sequence number (BE): 138 (0x008a)
        Sequence number (LE): 35328 (0x8a00)
    Response In: 191
    Data (32 bytes)
        Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
        Length: 32
0000  00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00 . ....C... *..E.
0010  00 3c 01 b4 00 00 80 01 4a dd c0 a8 0e 33 ad c2 .<..... J....3...
0020  71 92 08 00 4c d1 00 01 00 8a 61 62 63 64 65 66 q...L... abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69 wahcddefg hi

```

- **באדיט** - אנו רואים את הבית הראשון, הלא הוא Type, סוג החבילה. מדובר בסוג 8, כמו ש- Wireshark מועל בטובו להגיד לנו - מדובר ב-Echo Request. זאת ועוד, Wireshark מגדיל לעשوت ואף מתאר בסוגרים כי זו חבילה שקשורה ל-ping. השימוש ב-ping כה נפוץ, עד שabitot מסוג 8 (שהוגדרו במקור כ-"Echo Request") נקראות לעיתים "Ping Request".
- **בחול** - אנו רואים את הבית השני, ה-Code. עבור Type מסווג 8 (חbillot Type), שדה ה-Code תמיד מכיל את הערך 0.
- **בירוק** - הบทים השלישי והרביעי, המציגים את ה-Checksut. זהו אותו חישוב שמתבצע במחשב ששולח את ההודעה, כמו גם במחשב שמקבל את ההודעה. אם התוצאה היא אותה התוצאה - אין שגיאה בחבילה.
- **בכתום** - אלו מזחים ייחודיים להודעת Echo. הלוקוט, אשר שולח את בקשת ה-Ping, יכול להשתמש במזחים אלו כדי לזהות את חbillot הבקשתו שלו כשהוא שולח מספרabitot. כמובן, הוא יכול לציין כאן את הערך "1", ובוחbillot הבקשתו הבאה את הערך "2". כשהשרת יענה, הוא יgive לכל החבילה עם המספר שלו. מבולבלים? אל דאגה, המשיכו לעקוב אחר הדוגמה.
- **בסגול** - ישים ה-"מידע" של חbillot Echo Request. בשימוש ב-Windows, כפי שתתאפשרם יוכלו לראות, נשלח פשוט כל האנגלי עד האות 'w', ואז שוב מהאות 'a' ועד 'z'.



שאלת מחשבה: אילו המרכיבים זהים בין החבילה שבדוגמה לחבילה שלחتم במחשב שלכם? מדוע דוקא הערכים האלו נותרו קבועים, בעוד אחרים השתנו?

מחשב היעד (במקרה זהה, השירות של Google) קיבל את החבילה ששלחנו.icut, הוא מסתכל עליה, ומבחן שמדובר בחבילת ICMP. לאחר מכן הוא מסתכל בשדה Type, ומבחן שמדובר בבקשת Echo Request (או Ping). אי לכך, הוא מחליט שעליו לענות. כעת נבחן את החבילה הבאה, הלווא היא חבילת התשובה:

```

Frame 19: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on in
Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a
Internet Protocol Version 4, Src: 173.194.113.146 (173.194.113.146), Dst:
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x54d1 [correct]
        Identifier (BE): 1 (0x0001)
        Identifier (LE): 256 (0x0100)
        Sequence number (BE): 138 (0x008a)
        Sequence number (LE): 35328 (0x8a00)
    Response To: 181
    Response Time: 74.398 ms
Data (32 bytes)
    Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
0000 d4 be d9 d6 0c 2a 00 0c c3 a5 16 63 08 00 45 b8 .....*.. . . . . E.
0010 00 3c 5c d3 00 00 33 01 3c 06 ad c2 71 92 c0 a8 .<\....3. <...q...
0020 0e 33 00 00 54 d1 00 01 00 8a 61 62 63 64 65 66 .3..T.... abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuvwxyz
0040 77 61 62 63 64 65 66 67 68 69 wabcdefg hi

```

מבנה השכבות נותר, מן הסתם, זהה ולכך לא נתעכט עליו. נעבר על השדות הרלוונטיים ל-ICMP:

- **בAddon** - אנו רואים את הבית הראשון, הלווא הוא Type, סוג החבילה. מדובר בסוג 0, אשר מציין בפניהם שמצין Echo Reply. גם הפעם, Wireshark לא עוצר כאן ומוסיף בסוגרים כי זו חבילת שקשורה ל-Ping.
- **בכחול** - אנו רואים את הבית השני, Code. גם עבור חבילות מסווג Echo Reply, שדה Type תמיד מכיל את הערך 0.
- **בירוק** - הבטים השלישי והרביעי, המציגים אתChecksum.
- **בכתום** - כאן Google העתיק את המזהים שנשלחו בחבילת הבקשה. הסתכלו בחבילה הקודמת, ושימו לב כי אכן מדובר באותם ערכים בדיקון! לאחר מכן, הסתכלו בבקשת Ping הבאה, ככלומר בחבילת השאלה. תראו שהמזהים האלו שונים. מצאו את חבילת התשובה שלה, ככלומר חבילת תשובה שבה מזהים אלו זהים למזהים שבבקשה.
- **בסגול** - ישנו ה"מידע" של חבילת Echo Reply. השירות (במקרה שלנו, Google) חייב להעתיק את אותו המידע שנייתן בחבילת הבקשה. כפי שניתן לראות, זהו אכן אותו המידע.



שיםו לב: לקחנו פקודה מוכרת (**ping**), והשתמשנו ב-Wireshark על מנת להבין איך היא באמת עובדת. ניתן לראות כאן שימוש נוסף בכל Wireshark, שמאפשר לנו להבין איך הדברים עובדים מאחורי הקלעים!



תרגיל 7.5 מודרך - Ping - עשה זאת בעצמך

כתבו, באמצעות Scapy, סкриיפט ששולח חבילה Echo Request אל "www.google.com", ומקבל את התשובה. הסניף את התובורה תוך כדי⁶². לאחר שניתנסתם לבצע את התרגיל לבדכם, עשה זאת ייחודי.

ראשית, טענו את Scapy. כעת, נתחל מלבנות את פקעת השאלה, שלב אחר שלב. בטור התחלה, ניצור פקטה עם שכבת IP, כשמיולה יש ICMP. עשה זאת כך:

```
>>> request_packet = IP()/ICMP()
```

כעת, נביט בחבילה שנוצרה:

```

>>>
>>> request_packet = IP()/ICMP()
>>> request_packet.show()
###[ IP ]###[
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= icmp
chksum= None
src= 192.168.14.51
dst= 127.0.0.1
`options`###[ ICMP ]###[
    type= echo-request
    code= 0
    checksum= None
    id= 0x0
    seq= 0x0
>>> -

```

ניתן לראות ש-Scapy יצר עבורנו חבילה עם ערכים שהוא מוחש שישיעו לנו. כך למשל, כתובות המקור בחבילה ה-IP מכילה את כתובות ה-IP שלנו, במקרה זה - "192.168.14.51". עם זאת, כתובות היעד אינה הכתובות של "www.google.com". נסו לשנות זאת, ובדקו שהחbillah אכן השתנתה. ניתן לעשות זאת כך:

⁶² כפי ששמתתם לב לאורך הספר, הסניפה היא פעולה אשר אנו מבצעים באופן שוטף וועזרת לנו במספר תחומים - תוך כדי כתיבת קוד, במהלך הריצת כלי קיימ (כגון Ping), ובמקרים נוספים. זכרו להשתמש בכל עוצמתי זה!



```

C:\Windows\system32\cmd.exe - scapy
>>> request_packet[IP].dst="www.google.com"
>>> request_packet.show()
###[ IP ]###
  id= 0x0
  seq= 0x0
>>> request_packet[IP].dst="www.google.com"
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= icmp
  checksum= None
  src= 192.168.14.51
  dst= Net('www.google.com')
  \options\
###[ ICMP ]###
  type= echo-request
  code= 0
  checksum= None
  id= 0x0
  seq= 0x0
>>>

```

שים לב - לא נתנו ל-Scapy כתובת IP אמיתית, אלא את שם ה-DNS של "www.google.com". מעבר לכך, Scapy גם שמר אצלו את הכתובת בתור ('www.google.com', Net('www.google.com')) בדרך זו, Scapy מקל علينا ומבצע את תרגום כתובת ה-DNS לכתובת IP בעברינו.

להלן פירוט הפעלה, יכלו כמובן להשתמש בכתובת IP, ולכתוב לדוגמא:

>>> request_packet[IP].dst="173.194.113.178"

כעת יש לנו חבילת IP, שכתובת המקור שלה היא המחשב שלנו, וככתובת היעד שלה היא "www.google.com" החביבה כוללת גם Header של ICMP. אם נבחן את השדות, נראה כי Type מכיל "echo-request". מכאן ש-Scapy מוחש שם אמו מבקשים ליצור חבילה ICMP, אנו ככל הנראה רוצים חבילה מסווג Echo Request שהוא כפי שלמדנו שאלת Ping.

שדה ה-Code מכיל את הערך 0, כפי שחייבת Echo Request לצורך להראות. לאחר מכן, שדה ה-CRC (שנקרא בפי Scapy בשם "checksum") הינו ריק (מתיפוס None). דבר זה קורה מכיוון שהChecksum מחושב בזמן שליחת הפקטה, ולא בזמן יצירה שלה. שאר השדות מכילים את הערך 0x0, אך לא באמת מעניינים אותנו כרגע.

על מנת ליצור את חבילה זו בשורה אחת ולזוזה שאכן נוצרת חבילה מסווג Echo, יכלו להשתמש בשורה הבאה:

>>> request_packet = IP(dst="www.google.com")/ICMP(type="echo-request")

נוידא זאת:

```

C:\Windows\system32\cmd.exe - scapy
>>> request_packet = IP(dst="www.google.com")/ICMP(type="echo-request")
>>> request_packet.show()
###[ IP ]###[ ICMP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= icmp
chksum= None
src= 192.168.14.51
dst= Net('www.google.com')
\options\
###[ ICMP ]###[ echo-request ]
type= echo-request
code= 0
checksum= None
id= 0x0
seq= 0x0
>>>

```

כמה נוח ויפה להשתמש ב-**Scapy**! שימו לב שהשורה הזו משתמשת על כך ש-**Scapy** משתמש בכתובות ה-IP שלנו בתור כתובות מקור, וב-0 בתור שדה **Code** של ICMP. היות שאין צורך לשנות דברים נוספים בחבילה, ניתן לשלוח אותה:

```
>>> send(request_packet)
```

הסתכלו ב-**Wireshark**. אתם צפויים לראות שתי חבילות:

Filter: icmp							Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info				
9703	1122.13583	192.168.14.51	173.194.113.18	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64				
9704	1122.19841	173.194.113.18	192.168.14.51	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=55				

החבילה הראשונה היא חבילת השאלה שלנו. החבילת השנייה היא חבילת התשובה של Google, שכן ענה לנו! אם נבחן את שדות הבקשה שלנו, נראה שהם אינם אמורים שיכרנו (מלבד ל-**Checksum** שהוא אחר מכן):

```

Frame 9703: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on
Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 17
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf7ff [correct]
    Identifier (BE): 0 (0x0000)
    Identifier (LE): 0 (0x0000)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)

```

0000 00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00c..*..E.
0010 00 1c 00 01 00 00 40 01 8d 30 c0 a8 0e 33 ad c2@. .0...3..
0020 71 12 08 00 f7 ff 00 00 00 00 q..... ..

cut הסתכלו בחבילת התשובה. ציינו קודם לכן שהשרת (במקרה זה "www.google.com") צריך להשתמש באותם מזהים שנשלחו בבקשתו. במקרה שלו, התשובה צריכה לכלול את הערך 0 בכל השדות: Identifier (BE) ,Sequence Number(LE) ,Sequence Number (BE) ,Identifier (LE)

הצלחנו לראות את בקשת התשובה! אך עשים זאת רק ב-Wireshark, Scapy, ו- Scapy נותר אידיש למדוי נוכח התשובה של "www.google.com"

```

C:\Windows\system32\cmd.exe - scapy
>>> send(request_packet)
Sent 1 packets.
>>>

```

cut, נשתמש בפונקציה שמאפשרת גם לקבל תשובה: **sr1**. את הפונקציה החזינו לראשונה בפרק [שכבות התקשורת/תרגיל 6.11 מודרך - קבלת תשובה לשאלות DNS באמצעותעדי](#).

נשתמש בפונקציה זו לצורך הבהה:

```
>>> response_packet = sr1(request_packet)
```

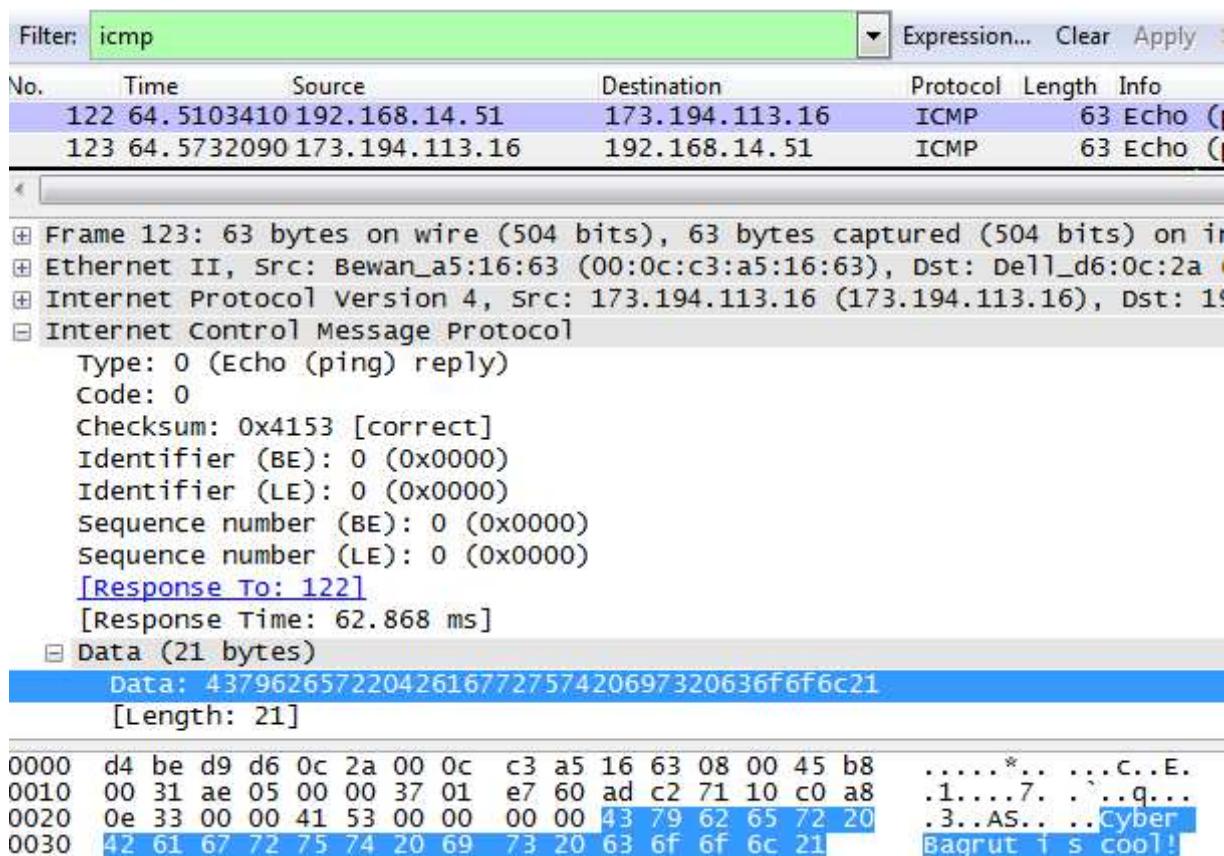
cut Scapy ישלח את החבילת, ויככה לתשובה. כאשר התשובה תחזור, היא תשמור במשתנה **response_packet**. בדקו זאת על ידי סוטכלות בתשובה:

עתה הגיעו העת לבחון הנחות שהוא לנו קודם לכך. טענו שהשרות יספק כל מידע שנשלח לו בחבילת התשובה. נסו לשלוח אל השירות בchein תבילה Echo Request, עם המידע "Cyber Bagrut is cool!". אם תצליחו, אתם צפויים לקבל את המידע הזה מהשירות.

אפשר לעשות זאת בצורה הבאה:

```
>>> request_packet = IP(dst="www.google.com")/ICMP(type="echo-request")/"Cyber Bagrut is cool!"  
>>> response_packet = sr1(request_packet)
```

ווחכל ב-www.google.com. ונראה ש-www.google.com אכו היגיר לנו. ועכשיו על המידע שאותנו לו:



נוכל לוודא זאת גם באמצעות Scapy

```
C:\Windows\system32\cmd.exe - scapy
>>> request_packet = IP(dst="www.google.com")/ICMP(type="echo-request")/"Cyber Bagrut is cool!"
>>> response_packet = sr1(request_packet)
Begin emission:
Finished to send 1 packets.
...
Received 4 packets, got 1 answers, remaining 0 packets
>>> response_packet[Raw]
<Raw load='Cyber Bagrut is cool!'>
>>>
```

תרגיל 7.6 - תרגילי Ping

- השתמשו ב-Scapy ובידע שלכם כדי למשת סקרייפטים אשר יבצעו משימות שונות:
1. שלחו הודעת Ping ל-"www.facebook.com". השתמשו ב-Identifier מסוים, וודאו שהשרת החזיר לכם תשובה בעלת אותו Identifier.
 2. שלחו שתי הודעות Ping ל-"www.facebook.com", ולכל אחת תננו Identifier אחר. שימו לב שאתם מקבלים את התשובות ומבינים איפה תשובה שייכת לאיזו שאלה.
 3. כתבו סקרייפט אשר ניתן להריץ משורת הפקודה. על הסקרייפט לקבל כפרמטר כתובת של שרת, לשЛОח אליו 4 חבילות Echo Request, ולכתוב למסך כמה תשובות הגיעו בהצלחה.

לדוגמה, שימוש בסקריפט יראה כך:

my_ping.py 2.3.4.5

תשובה לדוגמא תהיה:

Sending 4 packets to 2.3.4.5.

Received 3 reply packets.

רמז: קראו את התיעוד של הפונקציה `send`, והבינו אילו פרמטרים נוספים היא יכולה לקבל.

4. שמרו את הסקריפט הקודם, אך שיקבל מספר שונה של פקודות לשילוח (לאו דואק 4). כמו כן, וידאו שהסקריפט שולח את כל הפקודות, ורק אז מחרכה לתשובה.

לדוגמה, שימוש בסקריפט יראה כך:

my_ping 2.3.4.5 3

תשובה לדוגמא תהיה:

Sending 3 packets to 2.3.4.5

Received 2 reply packets.

רמז: קראו על הפונקציה `send`.

איך Traceroute עובד?

ראינו בתחילת הפרק את הכללי Traceroute אשר מאפשר לנו להבין את הדרך שחביבה עוברת בין שתי נקודות קצה. כעת, נלמד להבין איך Traceroute עובד ונממש את הפונקציונליות של הכללי - בעצמנו.

ברור שבשלב זה הנכם מתרגשים לקריאת כתיבת כלי כה מגניב בעצמכם. אתם יכולים להירגע ולשtotות כו"ם מים בטרם תמשיכו.

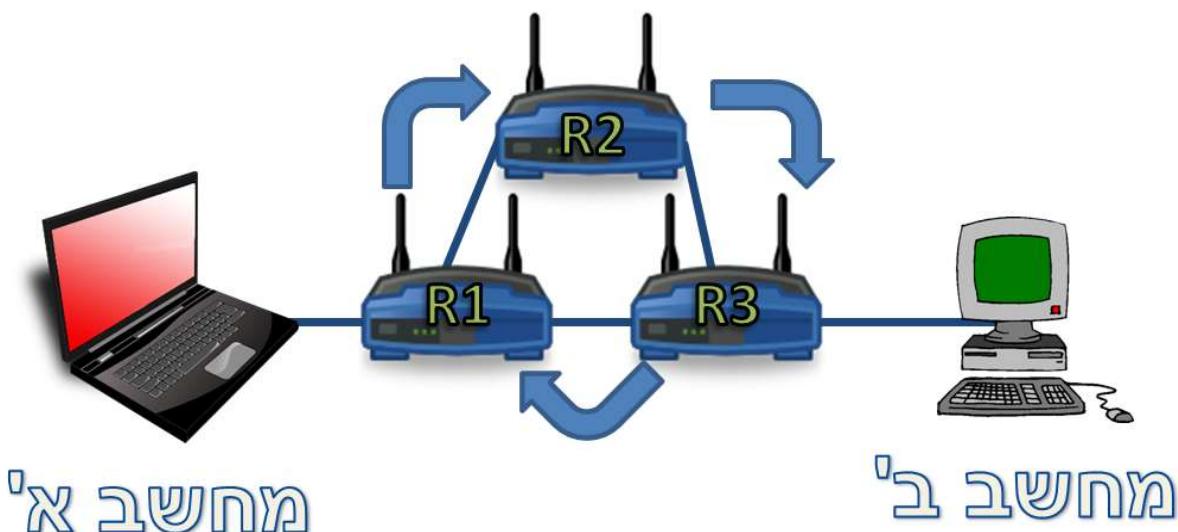
ראשית, נלמד על אחד השדות ב-Header של IP, שנקרא **Time To Live**, או בקיצור - **TTL**. השדה הוא באורך בית אחד, כלומר שהערך שנייתן בו יכול להיות בין 0 ל-255. על מנת להבין את השדה, נתחיל מלהסביר את הצורך בו.

הסתכלו בתמונה הרשות הבאה:



נאמר שמחשב א' שלוח חבילה למחשב ב'. החבילה תגיע ראשית אל R1. כעת, הנטב R1 יסתכל בטבלת הניתוב שלו, ויחליט שהדרך הטובה ביותר להעביר את החבילה למחשב ב' היא דרך הנטב R2. لكن החבילה תעבור אל R2. בתווך, הנטב R2 יבחן את החבילה, ויגע למסקנה שהדרך הטובה ביותר להעביר אותה למחשב ב' היא דרך R3, וכך יעביר אותה אליו. כעת, החבילה תגיע אל R3, שיבחן אותה ויגע למסקנה שהדרך הטובה ביותר להעביר את החבילה למחשב ב', היא דרך R1. הנטב R3, יעביר את החבילה לנטב R1, שכעת ייגע למסקנה שהדרך הטובה ביותר להעביר את החבילה למחשב ב' היא דרך הנטב R2... .

כפי שואדי הבחנתם, החבילה "לכודה" הגיע בין הנטבים R2, R1 ו-R3, ותמשך להיות מועברת בתוך ה"lolaha" שנוצרה:



כעת חשבו - איזה נזק נגרם כתוצאה לכך שהחbillה הזו נשארת "תקועה" ולא הגיע אל יעדה?

כמובן, מחשב ב' לא יקבל את הchipila שיעודה לו. אך מעבר לכך, הchipila זו יוצרת עומס על הרשות. שלושת הנתבים - R1, R2 ו-R3, ממשיכים לעבד אותה בכל פעם מחדש. הם מקדישים לה זמן ומשאבים שהיו יכולים להיות מוקדשים לחבילות אחרות. כך גם הקישוריהם בהם הchipila מועברת (הקישור בין R1 ל-R2, R2 ל-R3 והקישור בין R3 ל-R1), עומסים יותר כיון שהchipila זו ממשיכה לעبور בהם. קיומם של חבילות "לכודות" causal ברטת משפיע באופן עליון על הביצועים של הרשות ופוגע בהם!

אי לכך, علينا למנוע מקרים כאלה. שיטה אחת לעשות זאת היא להשתמש במנגנון ה-TTL, שקובע למעשה כמה פעמים הchipila יכולה להיות מועברת הלאה. נשתמש בדוגמה. נאמר שערק ה-TTL הראשון של הchipila אותה: TTL: שלח מחשב א', הוא 4. את ערך TTL זה קבע מחשב א'. נבחן את מסלולו של הchipila, וכן את ערך השדה TTL: • הchipila נשלה מחשב א' ומגיעה אל הנתב R1. הנתב בוחן את ערך ה-TTL, והוא שהוא 4. אי לכך, הוא מוריד ממנו 1, ועביר אותו הלאה, אל נתב R2. כמובן, הוא משנה את השדה $4 - 1 = 3$, שכן הוא כבר טיפול בחיפילה.

- הchipila מגיעה אל הנתב R2. הוא בוחן את ערך ה-TTL, והוא שהוא 3. הוא מוריד ממנו 1, ועביר אותו אל הנתב R3. כמובן, הוא משנה את השדה $3 - 1 = 2$, ועביר אותו אל הנתב R3.
- כעת הנתב R3 קיבל את הchipila. הוא בוחן את ערך ה-TTL, והוא שהוא 2. במידה שהוא עבר אותו אל מחשב ב' - תהליך השילוח הסטטי, הchipila הגיעה ליעדה והכל בסדר. אך מכיוון שלא כך המצב, הנתב מוריד את ערך ה-TTL, ועביר את הchipila לנtab R1, כשער ה- $TTL = 1$.
- כעת הנתב R1 מסתכל על הchipila. הוא בוחן את ערך ה-TTL, והוא שהוא שווה $1 - 1 = 0$. הוא מחסיר 1 מערך זה, ומגיע למצב שבו $0 = TTL$. אי לכך, הנתב מבין שאסור לו להעביר את הchipila הלאה - והוא משמש את הchipila!

כך למעשה מנגנון ה-TTL מונע מהchipila להישאר "תקועה" לנצח. עם זאת, במקרה זה, מן הראוי להודיע למחשב א' שהchipila שלו לא הגיעו ליעדה. לשם כך, מוצרה הודעה ICMP בשם Time-to-live exceeded. כאשר הנתב R1 מבין שעליו להשמש את הchipila היה שערק ה-TTL שלה נמוך מדי, הוא ישלח אל מחשב א' הודעה מסוג Time-to-live exceeded, כדי לידע אותו על כך.

שימוש לב לשימוש שנווכל לעשות בשדה זה: נניח כי אין בעיות ניתוב וחיפילה יכולה להגיע מחשב ב', כשהיא עובר בדרך הנתב R1, לאחר מכן בנtab R2, משם היא מועברת לנtab R3 שלבסוף מעביר אותה אל מחשב ב'. אם נשלח מחשב א' למחשב ב' חבית IP עם ערך 1 = TTL, הרי שהיא תגיע לנtab R1 שיגיד לנו שהוא לא יכול להעביר את הchipila. אם נשלח מחשב א' למחשב ב' חבית IP עם ערך 2 = TTL, היא צפוייה להגיע לנtab R2 שיגיד לנו שהוא לא יכול להעביר את הchipila הלאה. כך נוכל לגלוות את כל הרכיבים בדרך מחשב א' למחשב ב'!

תרגיל 7.7 מודרך - Traceroute - עשה זאת בעצמך



בשלב זה נעשה שימוש בשדה TTL של חבילה IP, כדי לגלו את הרכיבים שנמצאים בין המחשב שלנו לבין "www.google.com". נסן לערוך זאת בעצמכם, ולמצוא את הרכיב הקרוב ביותר אליום.

כעת נעשה זאת ייחד. ראשית, פיתחו את Wireshark והריצו הסנפה. השתמשו ב-Scapy, וצרו חבילה IP כאשר בשדה TTL ישנו הערך 1. על מנת להימנע מטעות של הכל' tracert, החביבה תכיל הודעה ICMP Echo Request (כלומר הודעה Ping). עם זאת, אין בכך הכרח. בנו את החביבה ושלחו אותה אל :"www.google.com"

```
C:\Windows\system32\cmd.exe - scapy
>>> traceroute_packet = IP ttl=1, dst="www.google.com")/ICMP()
>>> send(traceroute_packet)
Sent 1 packets.
```

הסתכלו בהסנפה. אתם צפויים לראות שתי חבילות - הראשונה, החביבה שלוחתם. השנייה, החביבה שהתקבלה מהרכיב הראשון ביןיכם לבין :"www.google.com

Filter: icmp							Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info				
186	42.7310060	192.168.14.51	173.194.112.49	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1				
187	42.7318360	192.168.14.1	192.168.14.51	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)				

נביט בחבילה שאנו שלחנו. שימו לב לשדה ה-**Time to live** המסומן בתכלת:

```
+ Frame 186: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
+ Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
+ Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 173.194.112.49 (173.194.112.49)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 28
    Identification: 0x0001 (1)
  Flags: 0x00
    Fragment offset: 0
+ Time to live: 1
  Protocol: ICMP (1)
+ Header checksum: 0xcd11 [correct]
  Source: 192.168.14.51 (192.168.14.51)
  Destination: 173.194.112.49 (173.194.112.49)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
+ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xf7ff [correct]
  Identifier (BE): 0 (0x0000)
  Identifier (LE): 0 (0x0000)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
```

כעת הסתכלו בחבילת התשובה:

```

Frame 187: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
Internet Protocol Version 4, Src: 192.168.14.1 (192.168.14.1), Dst: 192.168.14.51 (192.168.14.51)
Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xf4ff [correct]
Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 173.194.112.49 (173.194.112.49)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 28
    Identification: 0x0001 (1)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 1
        Protocol: ICMP (1)
    Header checksum: 0xcd11 [correct]
        Source: 192.168.14.51 (192.168.14.51)
        Destination: 173.194.112.49 (173.194.112.49)
        [Source GeoIP: Unknown]
        [Destination GeoIP: Unknown]
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf7ff
    Identifier (BE): 0 (0x0000)

0000 d4 be d9 d6 0c 2a 00 0c c3 a5 16 63 08 00 45 c0 . .... *... . . . . .
0010 00 38 a8 50 00 00 40 01 34 30 c0 a8 0e 01 c0 a8 .8.P..@. 40. .....
0020 0e 33 0b 00 f4 ff 00 00 00 00 45 00 00 1c 00 01 .3. .... . . . . .
0030 00 00 01 01 cd 11 c0 a8 0e 33 ad c2 70 31 08 00 ..... .3..pi...
0040 f7 ff 00 00 00 00 00 ..... .


```

נבחן בדgesים הבאים:

- באדוק** - מסומנת כתובות השולח של החבילה. זהו למשה הנטב שקיבל את חבילת-h-
.Time-to-live exceeded
- שאנו שלחנו, הבני שהוא לא יכול להעביר אותה הלאה, ושלח את חבילת-h-
במילים אחרות, זיהי הכתובת של הנטב הראשון ביןינו לבין "www.google.com"
- בכחול** - אלו השדות המאפיינים חבילה מסווג ICMP, כאשר שדה ה-h-
Code קיביל את הערך 0. ניתן לראות שתחת שדה ה-Type יש את הערך 11, ותחת השדה
יש את הערך 0.
- בכתום** - זהה למשה העתקה של החבילה המקורי, כלומר החבילה שאנו שלחנו אל
Header ה-192.168.14.1 העתיק אותה ושלח לנו אותה לאחר ה-ICMP.

ובכן, הצלחנו למצוא את הכתובת של הנטב הראשון! עם זאת, לא עשינו זאת באופן תכונתי. נסו לשפר את הקוד
שלכם כך שימצא את הכתובת של הנטב הראשון באופן תכונתי.
ניתן לעשות זאת כך:

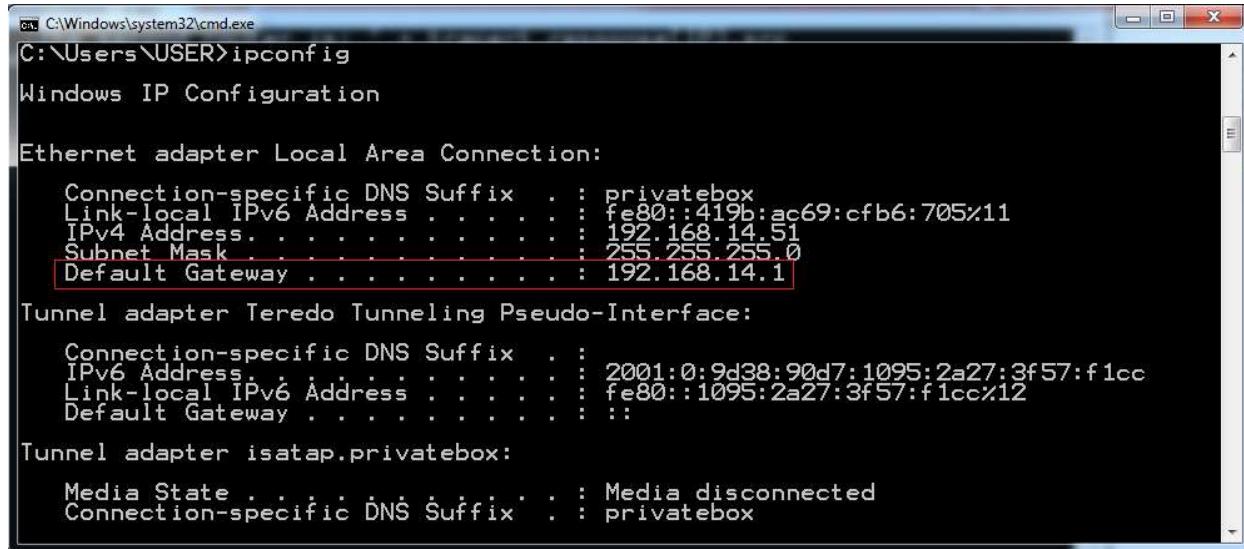
```

C:\Windows\system32\cmd.exe - scapy
>>> tracert_packet = IP(dst="www.google.com")/ICMP()
>>> tracert_response = sr1(tracert_packet)/ICMP()
Begin emission:
Finished to send 1 packets.

....*
Received 5 packets, got 1 answers, remaining 0 packets
>>> print 'The first router is: ' + tracert_response[IP].src
The first router is: 192.168.14.1

```

הכתובת שמצאנו היא כמובן של הנטב המחבר אל המחשב שלנו, והוא מהוות את ה-**Default Gateway** שלנו.
בכדי לוודא זאת, נוכל להריץ את פקודה **ipconfig** אוטה הכרנו קודם לכך:



```
C:\Windows\system32\cmd.exe
C:\Users\USER>ipconfig

Windows IP Configuration

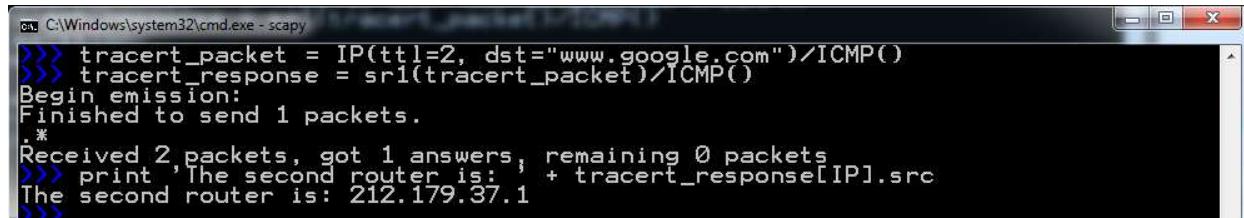
Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix . : privatebox
  Link-local IPv6 Address . . . . . fe80::419b:ac69:cfb6%11
  IPv4 Address . . . . . 192.168.14.51
  Subnet Mask . . . . . 255.255.255.0
  Default Gateway . . . . . 192.168.14.1

Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . :
  IPv6 Address . . . . . 2001:0:9d38:90d7:1095:2a27:3f57:f1cc
  Link-local IPv6 Address . . . . . fe80::1095:2a27:3f57:f1cc%12
  Default Gateway . . . . . ::

Tunnel adapter isatap.privatebox:
  Media State . . . . . Media disconnected
  Connection-specific DNS Suffix . : privatebox
```

נסו בעצמכם למצוא את הכתובת של הנטב השני באמצעות השיטה.

עשה זאת ייחד:



```
C:\Windows\system32\cmd.exe - scapy
>>> tracert_packet = IP(ttl=2, dst="www.google.com")/ICMP()
>>> tracert_response = sr1(tracert_packet)/ICMP()
Begin emission:
Finished to send 1 packets.
.
.
Received 2 packets, got 1 answers, remaining 0 packets
>>> print(The second router is: + tracert_response[IP].src
The second router is: 212.179.37.1
>>>
```

כעת נקבע גם בהסנפה:

No.	Time	Source	Destination	Protocol	Length	Info
902	11.9179240	192.168.14.51	173.194.113.176	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=1
903	11.9188030	192.168.14.1	192.168.14.51	ICMP	70	Time-to-live exceeded (Time to Live exceeded in transit)
1123	14.8578990	192.168.14.51	173.194.113.176	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=2
1124	14.8725700	212.179.37.1	192.168.14.51	ICMP	70	Time-to-live exceeded (Time to Live exceeded in transit)

הסנפה כוללת ארבע פקודות:

1. פקחת ה-**Echo request** שלחנו אל "www.google.com", כאשר שדה ה-TTL מכיל את הערך 1.
2. תשובה **Time-to-live exceeded** מהנטב הראשון בדרך מהמחשב שלנו אל "www.google.com".
3. פקחת ה-**Echo request** שלחנו אל "www.google.com", כאשר שדה ה-TTL מכיל את הערך 2.
4. תשובה **Time-to-live exceeded** מהנטב השני בדרך מהמחשב שלנו אל "www.google.com".

מcean שرك הסתכלות בהסנפה יכולה להראות לנו את רשימת הנטבים ביןינו לבין "www.google.com". עט,
נשתמש בכל tracert כדי למצוא את שני הנטבים הראשונים בדרך ביןינו לבין "www.google.com", ונבדוק כי
התשובה זהה⁶³:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>tracert -h 2 www.google.com
Tracing route to www.google.com [173.194.113.178]
over a maximum of 2 hops:
 1 <1 ms    <1 ms    <1 ms  box.privatebox [192.168.14.1]
 2 14 ms    14 ms    14 ms  bzq-179-37-1.static.bezeqint.net [212.179.37.1]

Trace complete.

C:\Users\USER>
```

התשובה אכן כוללת את הנטבים שמצאו בעצמן. מעבר לכך, היא כוללת מידע נוסף לא מದמנו. אם נביט בהסנפה, נראה שהחבילות tracert שולח דומות מאוד לחבילות שלחו בעצמן.

תרגיל 7.8 - עכשו תורכם - תרגילי traceroute



- כתבו סקריפט אשר מדפיס את כל התחנות ביןינו לבין "www.google.com". על הסקריפט להדפיס את כתובות ה-IP של התחנה בלבד.

dagshim:

- כasher הגעתם לכתובת ה-IP של "www.google.com", הפסיקו את פעולת הסקריפט.
- אל תעלו את ערך שדה ה-TTL באופן ידני כמו שעשינו עד כה. השתמשו בולאה.

- שפו את הסקריפט שלכם. מודיעו את הזמן שלוקח לכל נתב להגיב להודעה שלוחתם לו (כלומר הזמן שלוקח מהרגע שלוחתם את חבילת השאלה, ועד אשר קיבלתם הודעה מהנתב), בדומה לכלי tracert הדפסו גם את מידע זה למסך.

- שפו את הסקריפט כך שהוא משתמש יעיר בשורת הפקודה את הכתובת אליה הוא רוצה לבצע traceroute. שימוש לדוגמא בסקריפט יראה כך:
my_traceroute.py www.google.com

⁶³ כפי שכבר למדנו, שכבת הרשת יכולה לבחור לשנות את הנитוב עבור כל חבילה וחביבה, ולכן תיכון והתשובה לא תהיה זהה. עם זאת, בדרך כלל הנטבים הראשונים בדרך כן יישארו זהים, מסיבות שלא נפרט עליין כרגע.

הערה: על מנת לבצע traceroute באמצעות Scapy, כמו גם מנת לבצע הרבה דברים אחרים, ישן דרכים נוספת, אלגנטיות יותר מאשר שown בספר. עם זאת, מטרת הספר היא ללמד ולהבין את הדרך שבה הדברים עובדים, ולא ללמד שיטות מגניבות להשתמש ב-Scapy. אתם מוזמנים להרחיב את הידע שלכם ב-Scapy על ידי קריית ה-Tutorial שלו, שנמצא כתובות: <http://www.secdev.org/projects/scapy/doc/usage.html>

DHCP

למדנו בunitים על שכבת הרשת בכלל, ועל פרוטוקולי IP ו-ICMP בפרט. מונה שחרר הרבה מהפרק הוא " כתובות IP" - אותה כתובות לוגית בשכבה הרשת המשמשת כל ישות ברשת כדי להזדהות. אך שאלת גודלה עדין נותרה ללא תשובה:

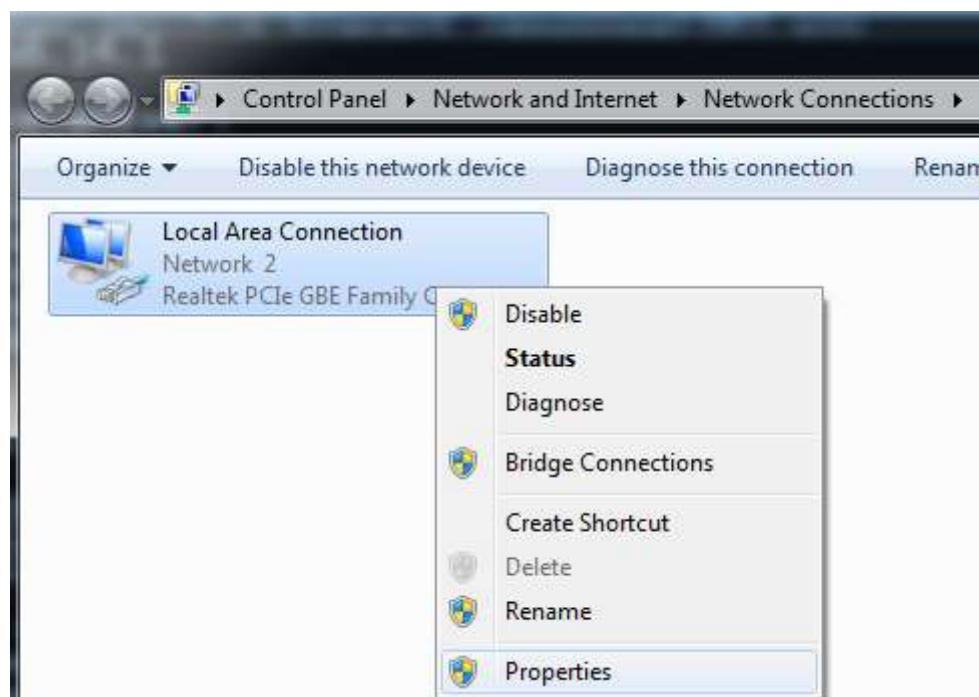


איך רכיב מקבל כתובות IP?

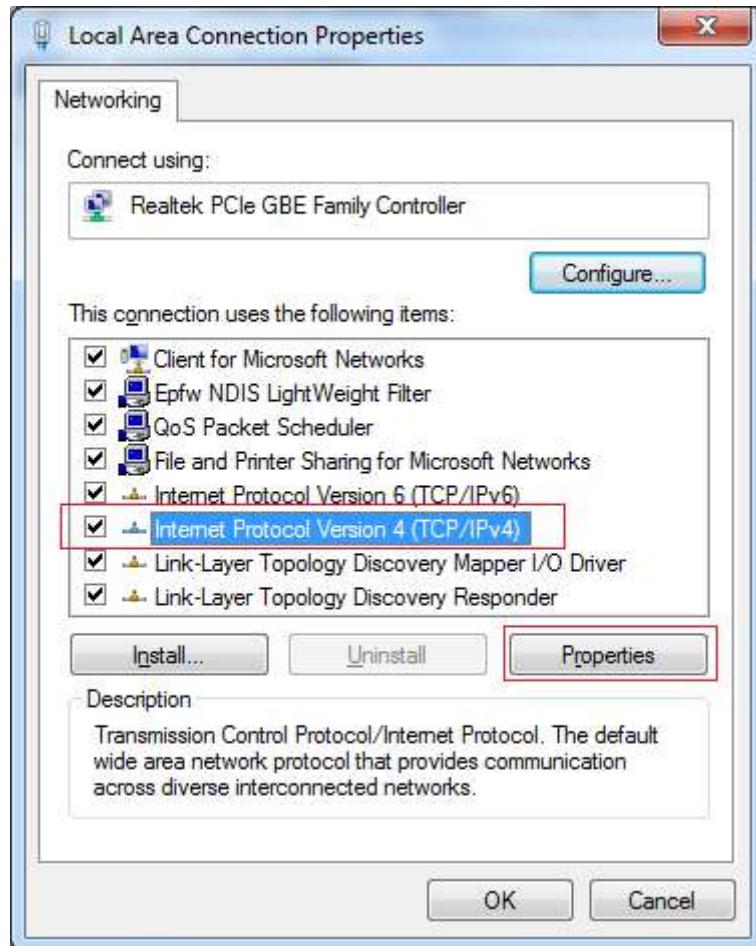
לצורך הדוגמא, כיצד המחשב שלו יודע מה כתובות ה-IP שלו?

ישן מספר דרכים לקבל כתובות IP, ולאណון בכלל. הפשטה שבן נקראת **הקצת סטטיטית של כתובות IP**. על מנת לבצע כתובות IP בצדה סטטיטית, היכנסו ללוח הבקרה (Control Panel) ובחרו ניהול קישורי רשת (Network Connections). תוכלו לעשות זאת גם על ידי הקשה על WinKey+R, והקשת **ncpa.cpl**.

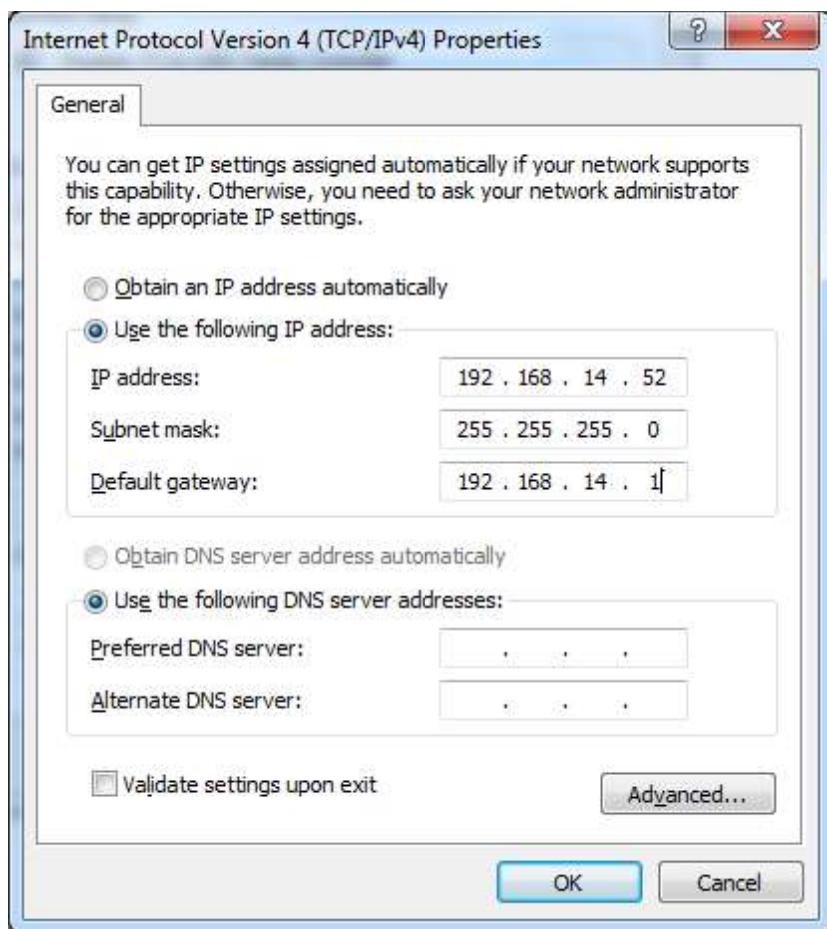
לאחר מכן, לחזו על החיבור שלכם באמצעות הממשק הימני של העכבר, ובחרו **במאפיינים** או **Properties**:



מתקן החולון שנפתח, בחרו ב-**Internet Protocol Version 4 (TCP/IPv4)**, ולהצט שוב על **מאפיינים** (או **:Properties**)



כעת תוכלו לבחור באפשרות **השתמש בכתובת IP הבאה** (או **Use the following IP address**), ולאחר מכן:
תוכלו לבחור כתובת IP, Subnet Mask, Default Gateway וכן **כרזונכם**. לדוגמה:



על מנת שאפשרות זו תעבור,عليكم לבחור בכתובת שלא קיימת כבר ברשת. כמו כן, שימו לב להשתמש באופןן הגדרות Default Gateway ו- Subnet Mask שהיו לכם קודם⁶⁴.

דרך נוספת לקלוט כתובת IP היא הדרך הדינמית, שמתבצעת בדרך כלל באמצעות הпрוטוקול **DHCP** (Dynamic Host Configuration Protocol).

נאמר לנו מחברים מחשב חדש לרשת. המחשב אינו ידוע את כתובת ה-IP שלו, ולכן עליו לברר אותה. הדבר הראשון שהוא יעשה לשם כך, הוא שליחת הודעה בשם **DHCP Discover** בבקשת זו, המחשב למעשה פונה לעולם ומבקש: "שלום, אין לי כתובת IP. האם תוכלו לעזור לי? אם יש כאן שרת DHCP שיכל להציג לי כתובת IP?" בקשה זו נשלחת כموבן broadcast, כלומר לכל הישויות ברשת. המטרה היא ששרת DHCP שיראה את הבקשה יוכל לענות אליה, בעוד ישויות אחרות יתעלמו מבקשת זו.

⁶⁴ ניתן להגדיר כתובת IP בצורה זו רק עבור כתובת פרטית המשמשת רק בתוך הרשת שלכם, ולא עבור כתובות חיצונית. כמובן, אם מדובר בכתובת שERICA להיות מוקצת על ידי ספק האינטרנט - תהיליך שינוי הכתובת מסווג בהרבה. לפרטים נוספים - קראו את [הנספח על כתובות פרטיות ו-NAT](#).

כעת, נאמר וישנם שני שירותים DHCP ברשות. שניים יכולים לענות למחשב המבקש כתובת IP עם הצעה. להודעה זו קוראים **DHCP Offer**.

במציאות, השירות כותב למחשב איזו כתובת IP הוא יכול לקבל, וכן פרטיים נוספים - כגון משך הזמן אותו הוא מבטיח להציג את כתובת ה-IP זו למחשב המבקש ולאף ישות אחרת ברשות. הודעה זו נשלחת אף היא לכטובת Broadcast - שכן אין עדין כתובת IP למחשב שאמור לקבל את הבקשה.

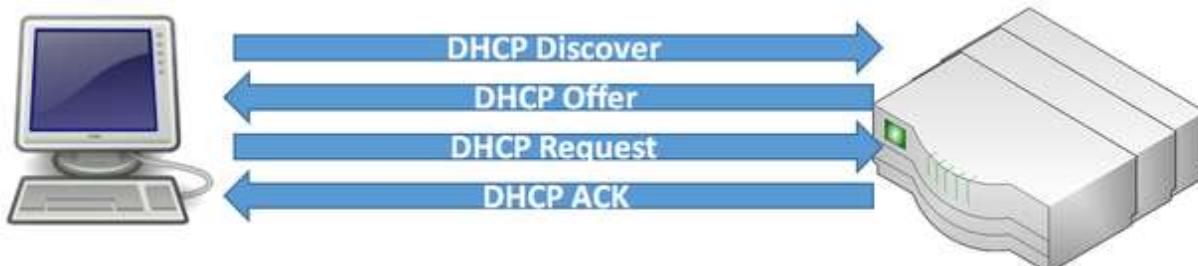
המחשב קיבל שתי הצעות שונות לכטובות IP. עליו לבחור באחת מהן, ואז לשלו בקשה לקבל באממת את הכתובת הזאת. הודעה זו נקראת **DHCP Request**.

בחבילת הבקשה, הלוקו מציין מי השירות DHCP ממנו הוא רוצה לקבל את הכתובת, כמו גם את הפרטים אשר השירות הציע לו. אם הודעה זו נשלחת ב-Broadcast, וזאת על מנת ששאר השירותים ידעו שהמחשב בחר בשרות הספציפי הזה, ולא ישמרו עבורי את הכתובת. לדוגמה, אם המחשב קיבל הצעה משרת A והצעה נוספת משרת B, והוא בחר בהצעה של שרת A, כאשר הוא שולח את הודעת ה-**DHCP Request** שלו לכולם, גם השירות B יראה אותה ויבין שמחשב א' לא בחר בו.

לבסוף, השירות צריך להחזיר למחשב אישור סופי שהוא אכן הופך להיות השירות DHCP שלו. להודעה זו קוראים בשם **DHCP ACK**.

בהודעה זו, השירות חוזר על הפרטים שהוא נותן לлокו. החל מעכשיו, הלוקו יכול להשתמש בכטובת ה-IP שניתנה לו באמצעות השירות DHCP.

לסיכום, התהליך נראה כך:

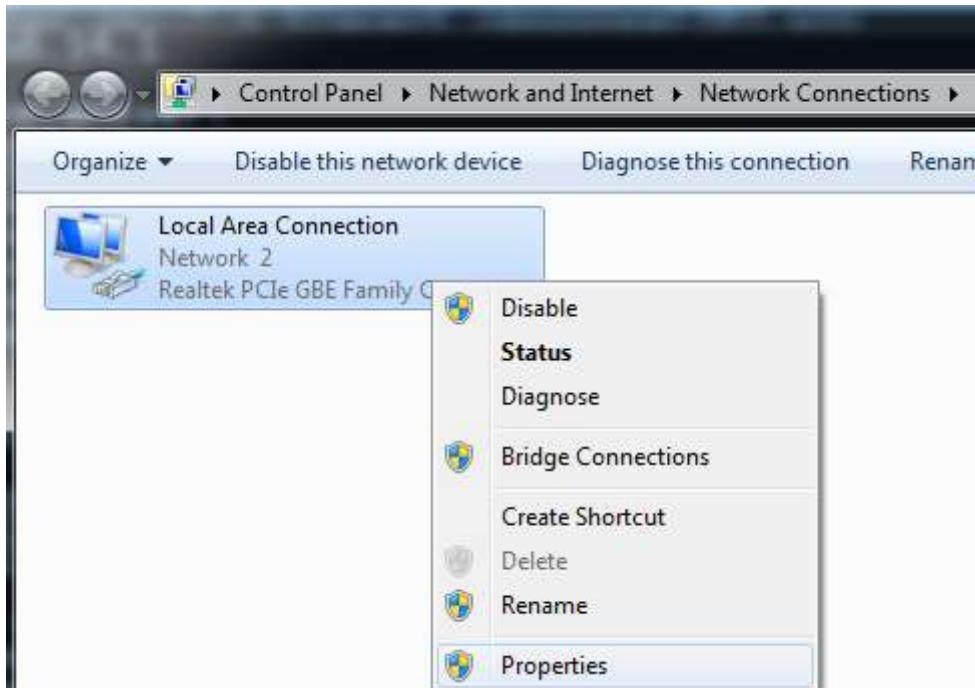


לשימוש DHCP יתרונות רבים. בין השאר, הוא מאפשר לחסוך בכטובות IP בכך שהוא מקצה כתובות לлокוחות רק כשם זקנים להן, ולא כל הזמן. בנוסף, באמצעות DHCP ניתן לתת פרטיה קונפיגורציה נוספים לлокוחות מלבד לכטובות IP (למשל - מי השירות DNS ישמש את הלוקו). על כן לא נרchia במספר זה.

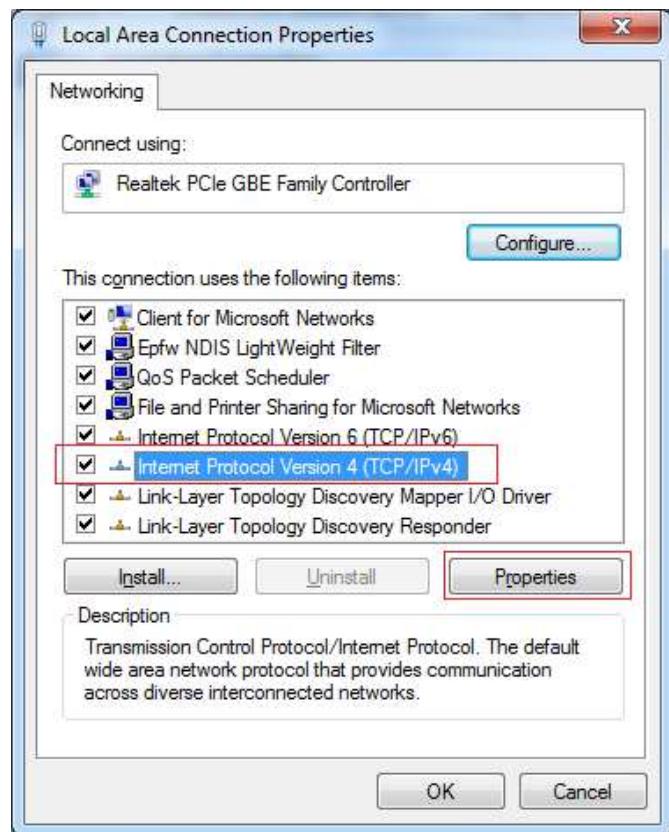
תרגיל 7.9 מודרך - קבלת IP באמצעות DHCP

כעת נקבל בעצמנו כתובת IP משרת DHCP שלנו. ראשית, עלינו לוודא שכרטיס הרשות שלנו מקבל כתובת IP בצורה דינמית ולא סטטית. על מנת לעשות זאת, היכנסו ללוח הבקרה (Control Panel) ובחרו בניהול קישורי רשת (Network Connections). תוכלו לעשות זאת גם על ידי הקשה על **R**, **WinKey+R**, והקשת **ncpa.cpl**.

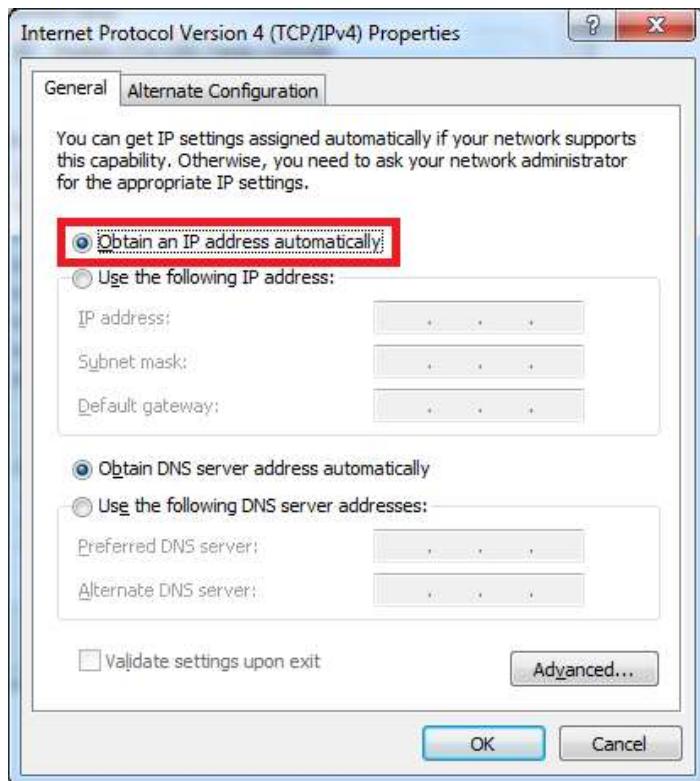
לאחר מכן, לחזו על החיבור שלכם באמצעות המקש הימני של העכבר, ובחרו במאפיינים (Properties).



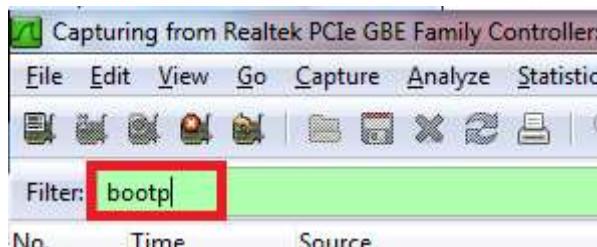
מתוך החלון שנפתח, בחרו ב-**Internet Protocol Version 4 (TCP/IPv4)**, ולחצו שוב עיל **מאפיינים** (Properties).



כעת, וודאו כי האפשרות המסומנת הינה האפשרות



כעת הריצו את Wireshark, והשתמשו במסנן התצוגה (display filter) הבא: `.bootp`



cut the connection to the Command Line, run ipconfig /release, and then run ipconfig /renew.

```
cmd C:\Windows\system32\cmd.exe
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\USER>ipconfig /release

Windows IP Configuration

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix . .
  Link-local IPv6 Address . . . . . fe80::419b:ac69:cfb6:705%11
  Default Gateway . . . . . ::

Tunnel adapter Teredo Tunneling Pseudo-Interface:

  Connection-specific DNS Suffix . .
  Link-local IPv6 Address . . . . . fe80::ffff:ffff:ffff%12
  Default Gateway . . . . . ::

Tunnel adapter isatap.privatebox:

  Media State . . . . . Media disconnected
  Connection-specific DNS Suffix' . .

C:\Users\USER>
```

כעת אין למחשבים כתובת IP. אם נסתכל בהסנהה, נגלה שלמעשה המחשב שלח הודעה לשרת ה-DHCP שמצין כי הוא מבקש להתנקק ממנו:



לא נתמך על הודעה זו. עצה, נסו לקבל כתובת IP חדשה. לשם כך, הריצו את הפקודה:

⁶⁵ DHCP הינו למעשה הרחבה של פרוטוקול יישן יותר בשם BOOTP, Wireshark, ו-BOOTP לא מכיר את מסן התצוגה "dhcp". על ההבדלים בין DHCP ל-BOOTP ישנו נטען בספר זה, אך אתם מוזמנים לבחש על כך באינטראקט.

```
C:\Windows\system32\cmd.exe
C:\Users\USER>ipconfig /renew
Windows IP Configuration

Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix  . : privatebox
  Link-local IPv6 Address . . . . . : fe80::419b:ac69:cfb6:705%11
  IPv4 Address . . . . . : 192.168.14.51
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.14.1

Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix  . :
  IPv6 Address . . . . . : 2001:0:9d38:90d7:3827:2a6e:3f57:f1cc
  Link-local IPv6 Address . . . . . : fe80::3827:2a6e:3f57:f1cc%12
  Default Gateway . . . . . :

Tunnel adapter isatap.privatebox:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . . :
```

cutet הביטו בהסנה ומצאו את החבילות הרלוונטיות:

No.	Time	Source	Destination	Protocol	Length	Info
59	7.13522600	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xc8bcef81
70	10.11095001	192.168.14.1	192.168.14.51	DHCP	346	DHCP Offer - Transaction ID 0xc8bcef81
71	10.11141500	0.0.0.0	255.255.255.255	DHCP	352	DHCP Request - Transaction ID 0xc8bcef81
72	10.12026301	192.168.14.1	192.168.14.51	DHCP	346	DHCP ACK - Transaction ID 0xc8bcef81

נעבור בקצרה על החבילות. הסתכלו על החבילה הראשונה, חבילת ה-DHCP. להזכירם, זהו חבילה שהמחשב שולח כדי למצוא שרת DHCP ולבקש מהם לתת לו כתובת IP.

Frame 59: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0

Ethernet II, Src: Dell_d6:0c:2a (44:b7:d9:d6:0c:2a) [REDACTED] Dst: Broadcast (FF:FF:FF:FF:FF:FF)

Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0) [REDACTED] Dst: 255.255.255.255 (255.255.255.255)

User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)

Bootstrap Protocol

Message type: Boot Request (1)

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0xc8bcef81

Seconds elapsed: 3

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)

Client hardware address padding: 000000000000000000000000

Server host name not given

Boot file name not given

Magic cookie: DHCP

Option: (53) DHCP Message Type

Option: (61) client identifier

Option: (50) Requested IP Address

Length: 4

Requested IP Address: 192.168.14.51 (192.168.14.51)

Option: (12) Host Name

Option: (60) Vendor class identifier

Option: (55) Parameter Request List

Option: (255) End

נתחל מודל השכבות:

- בשכבה השנייה, ניתן לראות את השימוש בפרוטוקול ⁶⁶Ethernet⁶⁶. דבר זה מלמד כי כרטיס הרשות ממנו נשלחה החבילה הוא כרטיס מסווג Ethernet.
- בשכבה השלישית, יש שימוש בפרוטוקול IP. שימו לב שכותבת המקור המסומנת באדום היא הכתובת: 0.0.0.0, שמצוינת כי החבילה נשלחה מ"המחשב שלו". הייתה שלמה לנו עדין אין כתובת IP, הוא משתמש בכתובת זו. כתובת היעד המסומנת בכחול היא הכתובת: 255.255.255.255, המצוינה כי החבילה נשלחת ב-Broadcast, כלומר לכל הישויות ברשת. זאת הייתה שהמחשב מנסה להגיע לכל שרתים DHCP, והוא אינו יודע את הכתובות שלהם.
- בשכבה הרביעית, יש שימוש בפרוטוקול UDP, עליו למדנו בפרק הקודם. ישן מספר סיבות להעדפת פרוטוקול זה על פני TCP במקורה של DHCP. ראשית, פרוטוקול DHCP הוא פרוטוקול מסווג "בקשה-תשובה", כלומר: אם בקשה אחת "תלך לאיבוד", ניתן פשוט לשלוח בקשה נוספת. עם זאת, הסיבה המהותית יותר, היא שאנו מבקשים לשЛОח הודעות ב-Broadcast, דבר אשר לא אפשרי בפרוטוקול TCP, כפי שלמדנו בפרק שכבת התעבורה.
- בשכבה החמישית, יש שימוש בפרוטוקול DHCP, אשר Wireshark מזהה עבורה כפרוטוקול BOOTP או בשמו המלא - (Bootstrap Protocol).

לא נתעכט על כלל השדות של פרוטוקול DHCP, אך נשים לב לנקודה מעניינת: אחד השדות אשר הלקוח שלו, מסומן בירוק, הוא השדה **Requested IP Address**. הייתה שלא מדובר במחשב חדש ברשת, המחשב זוכר את כתובת ה-IP שהיאיה לו קודם לכן, וمبקש לקבל אותה שוב.

⁶⁶ כמו שציינו קודם לנו - על השכבה השנייה בכלל, ופרוטוקול Ethernet בפרט, נרחיב בפרק הבא.

החבילה הבאה הינה חבילת ההצעה של השירות: **DHCP Offer**

```

# Frame 70: 346 bytes on wire (2768 bits), 346 bytes captured (2768 bits) on interface 0
# Ethernet II, Src: Bewan_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
# Internet Protocol Version 4, Src: 192.168.14.1 (192.168.14.1), Dst: 192.168.14.51 (192.168.14.51)
# User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
└ Boot Protocol
    Message type: Boot Reply (2)
    Hardware type: Ethernet
    Hardware address Length: 6
    Hops: 0
    Transaction ID: 0xc8bcef81
    ┌ Seconds elapsed: 3
    ┌ Bootp flags: 0x0000 (Unicast)
    ┌ Client IP address: 0.0.0.0 (0.0.0.0)
    └ Your (client) IP address: 192.168.14.51 (192.168.14.51)
    Next server IP address: 192.168.14.1 (192.168.14.1)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client MAC address: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
    Client hardware address padding: 000000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    ┌ Option: (53) DHCP Message Type
    ┌ Option: (54) DHCP Server Identifier
    ┌ Option: (51) IP Address Lease Time
    ┌   Length: 4
    ┌   IP Address Lease Time: (86400s) 1 day
    ┌ Option: (58) Renewal Time Value
    ┌ Option: (59) Rebinding Time Value
    ┌ Option: (28) Broadcast Address
    ┌ Option: (15) Domain Name
    ┌ Option: (6) Domain Name Server
    ┌ Option: (3) Router
    ┌ Option: (1) Subnet Mask
    ┌   Length: 4
    ┌   Subnet Mask: 255.255.255.0 (255.255.255.0)
    ┌ Option: (255) End

```

מודל השכבות זהה ولكن לא נתעכט עליו. עם זאת, שימו לב לכתובות ה-IP בשימוש. כתובות המקור של החבילות, המסומנת ב**אדום**, היא כתובות ה-IP של שירות DHCP שהולחן את ההצעה. **בכחול**, נמצאת כתובות ה-IP אשר הלקוח ביקש. דבר זה אפשרי רק כאשר הלקוח מציין במפורש את הכתובת שאויה הוא רוצה, כמו שראינו בשלב הקודם. במקרים אחרים, כתובות זו תהיה "0.0.0.0".

גם בשכבת DHCP, נמצאת **בכחול** כתובות ה-IP אשר המחשב ביקש. כאן היא נמצאת תחת השדה **Your (client) IP address**, כלומר - זו הכתובת שהשרת מציע ללקוח.

כפי שניתן לראות, השירות שולח פרטים נוספים רבים. חשוב בשלב זה לשים לב לשדה **Lease Time**, אשר מסומןocabub **ירוק**. המשמעות שלו היא הזמן שבו מובטח ללקוח שכותבת ה-IP המוצעת מוקצה עבורו ולא עבר אף אחד אחר. כלומר, לאחר מעבר הזמן הזה (בדוגמה שלנו - יום אחד), אם הלקוח לא ביצע חידוש של ה-**Lease Time**, אסור לו להמשיך ולהשתמש בכתובת ה-IP הזאת, שכן יתכן והוא מוקצה לשימוש אחר.

בכתום ניתן לראות ששרת DHCP מספק גם את ה-**Subnet Mask**. דבר זה הגיוני בהתאם למה שלמדנו במהלך הפרק - על המחשב לדעת לא רק מה כתובת ה-IP שלו, אלא גם מה מזגה הרשות שלו, ולשם כך עלי להכיר את ה-Subnet Mask שלו. כתוב לנו מבינים שהמחשב יודע אותה באמצעות הודעה DHCP.

לאחר מכן נשלחת הודעה DHCP Request, המצינית בפניו שרת DHCP זהה (וגם בפניהם נוספים), אם יש כאלה), שהלקוח בחר בו ומעוניין להשתמש בהצעה שלו. היה ששהודה זו חוזרת באופן כמעט מלא על הودעות קודמות, לא נתעכבר עליה.

לבסוף, נשלחת הודעה DHCP ACK, המצינית שהשרת קיבל את הבקשה של הלוקוח, וכי הוא רשאי להשתמש בכתובת ה-IP שהוקצתה עבורו. היה ששהודה זו חוזרת באופן כמעט מלא על הודעות קודמות, לא נתעכבר עליה.

בתום תהליך זה, המחשב שלנו ידע מה כתובת ה-IP שלו ומה ה-Subnet Mask הרלוונטי. כמו כן הוא גילה פרטים חשובים נוספים על הרשות, כגון ה-Default Gateway שלו, שרת ה-DNS בו עליו להשתמש ועוד. בעצם, לאחר שהושלם תהליך DHCP, הוא יכול להשתמש בכרטיס הרשות שלו ולצאת לתקשורת עם העולם.

שכבה הרשת - סיכום

במהלך פרק זה למדנו להכיר את שכבה הרשת. התחלנו מלהבין לעומק את תפקידה במודל השכבות, וכייזד היא משתלבת בשכבות עליון למדנו עד כה. למדנו על האתגרים בפניהם עומדת שכבה הרשת, ועל דרכי בהן היא מתמודדת איתם.

הכרנו את **פרוטוקול IP**, הпрוטокול של האינטרנט, והתעמקנו ב**כתובות IP** והמבנה שלהן. לאחר מכן הכרנו את מונח **הניטוב**, כמו גם את הרכיב **נתב**. במהלך הפרק השתמשנו בכלים שונים כגון **ping**, **tracert**, **ipconfig** ו-**route**.

הכרנו גם פרוטוקולים נוספים, כגון **פרוטוקול ICMP**. באמצעות למידת פרוטוקול זה הבנו כיצד ממומשות הפקודות Ping ו-Traceroute ולמדנו למשוך אותן בעצמנו. לבסוף, למדנו על דרכי לקבל כתובות IP, והרחבנו בעיקר על **פרוטוקול DHCP**.

בפרק הבא, נרד שכבה נוספת במודל השכבות ונלמד להכיר את שכבת הקו. נדבר על האתגרים הניצבים בפניהם שכבה זו, וכייזד היא מתחשרת לשכבה הרשת עליה למדנו עתה.

שכבה הרשת - צעדים להמשך

על אף שלמדנו רבות על שכבה הרשת, נותרו נושאים רבים בהם לא נגענו. לא הסבכנו כמעט בכלל כיצד מtbodyות החלטות הניתוב - כלומר איך הנתבים בונים את טבלאות הניתוב שלהם. לא הרחכנו על שיטות להתמודד עם עומסים ברשת, ולא נגענו בברחת איכות (Quality Of Service). לא סיפרנו על אבטחה ב-IP (באמצעות IPSec), ולא דיברנו על אפשרויות נוספות של השכבה.

אלו מכם שמעוניינים להעמק את הידע שלהם בשכבה הרשת, מוזמנים לבצע את הצעדים הבאים:

קריאה נוספת

בספר המצוין Computer Networks (מהדורה חמישית) מאת David J. - Andrew S. Tanenbaum, הפרק החמישי מתיחס במלואו לשכבה הרשת. באופן ספציפי, מומלץ לקרוא את החלקים:

- 5.2.1-5.2.6 - ניתובים ואלגוריתמי ניתוב.
- 5.3 - בקרת עומסים.
- 5.4.1-5.4.2 - בקרת איכות.
- 5.6.6-5.7.7 - על אלגוריתמי הניתוב בשימוש באינטרנט.

בספר המצוין Computer Networking: A Top-Down Approach (מהדורה ששית) מאת James F. Kurose, הפרק הרביעי מוקדש כולו לשכבה הרשת. באופן ספציפי, מומלץ לקרוא את החלקים:

- 4.3 - התהיליך שקורה בתגובה.
- 4.4 - ניתובים ואלגוריתמי ניתוב.
- 4.6 - על אלגוריתמי הניתוב בשימוש באינטרנט.
- 4.7 - ניתוב Multicast ו-Broadcast.

כמו כן, ניתן להרחב את אופקיכם בפרק על IP מתוך The TCP/IP Guide, אותו ניתן למצוא בכתבתו: <http://goo.gl/igqBmk>.

תרגיל 7.10 - פרוגמנטציה של IP (אתגר)

בתרגיל זה תמשכו תקשורת מעל פרוגמנטציה של IP (על פרוגמנטציה תוכלו לקרוא [בנוסף לפרק זה](#)). השתמשו במודול **socket** של פיתון כדי לכתוב שרת פשוט המאזין על פורט 55555 לחבילות UDP נוכנות. על הסкриיפט להציג למסך כל חבילה מידע שהוא מקבל.

עת כתבו סкриיפט המקבל מהמשתמש מסר שעליו לשלוח (כלומר, מחרוזת - למשל: "hello world", ומספר פרוגמנטים. עלייכם לשלוח את המסר שהליך שלח אל שרת ה-UDP שכתבתם קודם לכן, ולחلك אותו באמצעות

פרגמנטציה של IP למספר הrogramnetics שהליך ביקש. כך למשל, אם הליך ביקש לשלוח את המסר " hello world " בשלשה חלקים, תוכלו לשלוח אותו כך:

- חלק ראשון - "hel"
- חלק שני - "lo worl"
- חלק שלישי - "d"

כמובן שתוכלו לשנות את מספר התווים שנשלחים בכל חלק.

וודאו כי השירות מצליח להציג נכון הודעה שלחכם לו. כמו כן, הסניף באמצעות Wireshark וודאו שגם שולחים מספר נכון של פרגמנטים. שימו לב שעיל מנת לבדוק את תרגיל זה, עלייכם להשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת⁶⁷.

dagshim

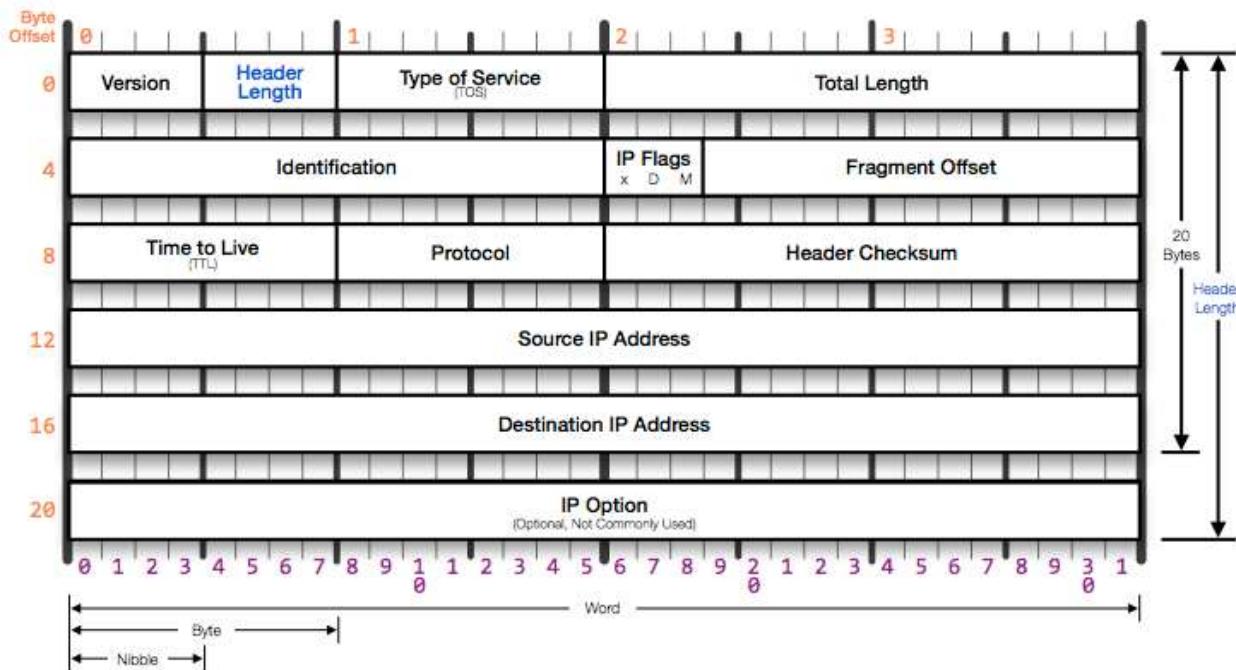
- יש לבנות ולשלוח את החבילות באמצעות Scapy.
- ה-Header של UDP צריך להופיע רק פעם אחת, אין צורך לחזור עליו בכל פרוגמנט מחדש.

⁶⁷ באופן תאורי, יכולנו לעשות זאת מעל设备 loopback – כזכור מעל כתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של Scapy בשילחה וקבלת מסגרות מעל loopback device ב-Windows, נשתמש בשני מחשבים.

נספח א' - IP Header

נספח זה מועד כדי לתאר את כל השדות של ה-Header של IPv4. לא חיוני להבין את כל השדות עד הסוף. מידע נוסף ניתן למצוא בכתובות: http://en.wikipedia.org/wiki/IPv4_header#Header

IP Header

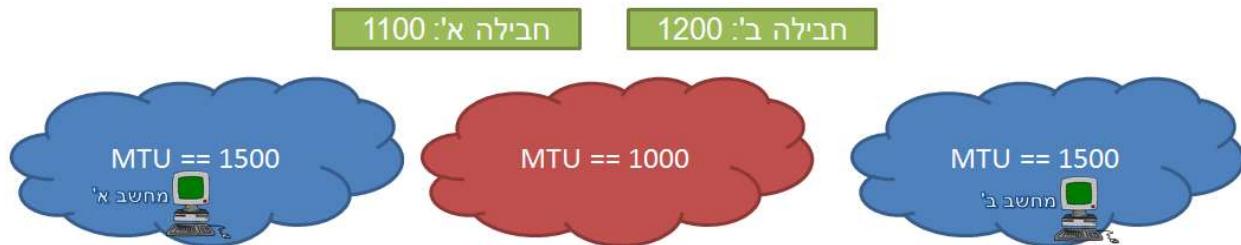


- שדה זה מתאר את גירסת ה-IP. לדוגמה, עבור פקטה של IPv4, השדה יכיל את הערך 4. •
עבור פקטה של IPv6, השדה יכיל את הערך 6.
- מתאר את אורך ה-Header של החבילה, להיות שהוא עשוי לשינוי בהתאם לחבילה •
לחבילה, מכיוון שישנם שדות של IP Options אוטם נקבע בהמשך. הערך של השדה מתאר את הגודל ביחידות מידת של 32 ביטים. כך למשל, עבור חבילה בגודל 20 בתים, הערך כאן יהיה 5 (מכיוון שמדובר ב-160 ביטים, שהם 5 פעמיים 32 ביטים). הערך 5 (שנתואר 20 בתים) הוא הערך הנפוץ ביותר עבור שדה זה. שדה זה דומה מאוד לשדה ה-Header Length של פרוטוקול TCP, עליו למדנו בפרק שכבת התעבורה.
- לשדה זה יהיו משמעותות שונות לאורך השנהים והוא גם הוגדר מחדש. במקור, השדה אפשר לציין את העדויות של חבילה מסוימת על פני חבילות אחרות - בכך לבקש מהנתבים להעביר חבילות בעלות עדויות גבוהה לפני חבילות בעלות עדויות נמוכה יותר. בפועל, לא היה שימוש נרחב בשדה זה. •

Header - שדה זה מצין את הגודל, בbytes, של כל החבילות בשכבה ה-IP (כולל ה-Header) • והמידע).

- שימוש לב - הגודל המינימלי של חבילה IP הוא 20 bytes, שכן זהו הגודל המינימלי של התחלית. הגודל המקסימלי הוא 65,535 bytes - שכן השדה Total Length הוא באורך של 16 bytes.

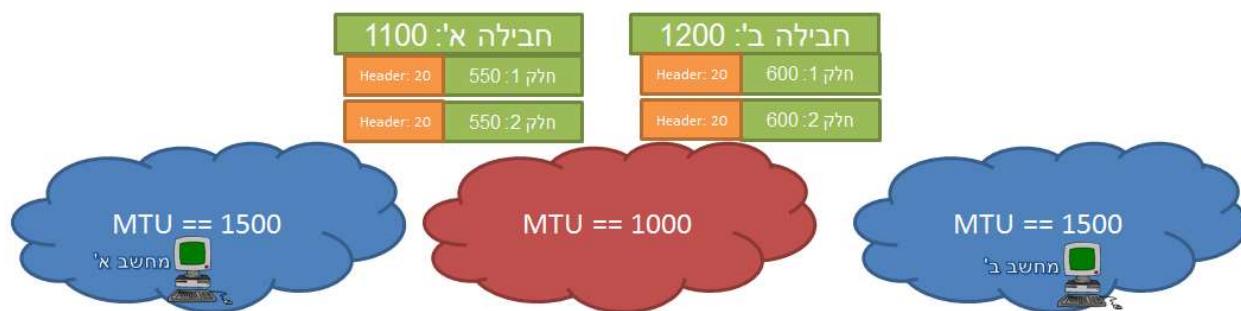
בטרם נמשיך לשדות הבאים, علينا להסביר שני מונחים חדשים: **פרגמנטציה (Fragmentation)** ו-**MTU**. רשתות שונות יכולות לטפל בגודל שונה של חבילות. כך למשל, יתכן ורשת אחת משתמשת ב프וטוקול Ethernet של השכבה השנייה - שכבת הנקו, וכך יכולה להעביר חבילות עד גודל של 1,500 bytes, ולא יותר מכך. אי-כך, נאמר שה-**MTU (Maximum Transmission Unit)** של רשת זו הינו 1,500 bytes. נביט בתמונה הרשות הבאה:



אם מחשב א' נמצא ברשות שה-MTU שלו הוא 1,500 bytes - כלומר, ברשות ניתן להעביר חבילות עד גודל של 1,500 bytes ולא יותר מכך. מחשב א' רוצה להעביר למחשב ב' שתי חבילות: האחת בגודל 1,100 bytes, והשנייה בגודל 1,200 bytes.

מחשב ב' נמצא אף הוא ברשות שה-MTU שלו הוא 1,500 bytes. עם זאת, על מנת להגיע למחשב ב', החבילות צריכות לעבור ברשות שה-MTU שלה הוא 1,000 bytes בלבד (הרשות האדומה). מכיוון שלא חבילה א', ולא חבילה ב' יכולות לעבור בה. אם חבילה שכוונה להגיע לרשות האדומה, היא צפופה להיזרק.

על מנת לפתור את סוגיה זו, ניתן לבצע **פרגמנטציה** של חבילות: נחלק אותן לשני חלקים (fragments), שכל אחד מהם בגודל של 1,000 bytes או פחות. נביט בתמונה הבאה:



אם חבילה א' התחלקה לשני חלקים, שכל אחד מהם בגודל של 550 bytes של מידע, ועוד 20 bytes של Header.

חבילה ב' התחלקה אף היא לשני חלקים, שכל אחד מהם בגודל של 600 בתים של מידע, ועוד 20 בתים של .Header

כעת יש ארבעה חלקים חבילות, ש愧 אחד מהם לא עבר את גודל ה-UTM המותר בראשת האדומה. לכן, מחשב א⁶⁸ יכול לשלוח את כלן מבלי להיתקל בבעיית ה-UTM. כאשר החלקים יגיעו למחשב ב' בראשת הכהולה, יהיה לעליו להבין איזה חלקים היו שייכים לאיזה חבילה, לחבר אותם מחדש ולקבל את החבילות שמחשב א' ניסה לשלוח אליו במקור.

ניתן לקרוא מידע נוסף על פרוגמנטציה בכתובות: http://en.wikipedia.org/wiki/IP_fragmentation

כעת נמשיך להסביר על השדות השונים של IP Header:

- Identification - שדה זה משומש במקרה של פרוגמנטציה. בדוגמה לעיל, מחשב ב' צריך לדעת להבדיל בין חלק 1 של חבילה א' לבין חלק 1 של חבילה ב', כדי לדעת להרכיב נכון את החלקים. אי לכך, גם החלק 1 וגם חלק 2 של חבילה א' יהיה אותו המזהה בשדה ה-Identification (לדוגמא - המזהה 100), בעוד חלק 1 וחלק 2 של חבילה ב' יהיה מזהה אחר (לדוגמא - המזהה 200).
- Flags - דגלים שונים לשימוש. מוגדרים שלושה דגלים:
 - Reserved - בית זה נשמר תמיד על הערך 0.
 - (DF) Don't Fragment - אסור לבצע פרוגמנטציה לחבילה שנשלחת עם הדגל הזה דולק⁶⁹.
 - במקרה של עיל, במידה שחבילה א' נשלחה עם הדגל DF דולק, אף נתב בדרך אסור לפצל אותה ולבצע פרוגמנטציה. אם כך, החבילה לא תוכל להשליח למחשב ב', ותשלח על כך הודעה שגיאיה.
 - (MF) More Fragments - דגל זה משומש במקרה של פרוגמנטציה. במידה שנשלחת חבילה מפוצלת, בכל fragment השני ה-fragment האחרון בחבילה הביט הזה יהיה דולק. בדוגמה לעיל, בחלק 1 של חבילה א' הדגל יהיה דולק (מכיוון שיש גם את חלק 2). בחלק 2 של חבילה א' הדגל יהיה כבוי (כיון שהוא חלק האחרון של החבילה). באופן דומה, בחלק 1 של חבילה ב' הדגל יהיה דולק, ובחלק 2 של חבילה ב' הדגל יהיה כבוי.
- Time To Live (TTL) - נדרש כדי למנוע חבילות להסתובב לנצח ברחבי הרשת. הרחכנו על שדה זה תחת הסביר על פרוטוקול ICMP בפרק זה.
- Protocol - שדה זה מציין מהו הפרוטוקול שנמצא מעל שכבת ה-IP. לדוגמא, הערך "6" מציין שהשכבה שמעל לשכבה ה-IP היא שכבת TCP. הערך "17" מציין שמדובר בשכבה UDP.
- Header Checksum - נדרש לוודא את תקינות ה-Header של החבילה (שים לב שוואידוא התקינות מתבצע על ה-Header בלבד, ולא על המידע של החבילה). כאשר חבילה מגיעה אל ישוט כלשה' בראשת,

⁶⁸ בפועל, בחלק גדול מהמקרים, יהיה זה אחד הנתבים בדרך שיבצע את הפרוגמנטציה ולא מחשב הקצה.

⁶⁹ המשמעות של "דגל דולק" היא שערך הבית הוא 1. "דגל כבוי" משמעו שערך הבית הוא 0.

היא מחשבת את ערך ה-Checksum של ה-Header ומשווה אותו לערך שמצוּ בשדה ה-Checksum. במידה שהערכים לא זהים, יש לזרוק את החבילה. להזכירם, למדנו על [Checksum בפרק שכבת התעבורה/ מה זהChecksum ?](#)

- כתובות המקור של החבילה, כמו כתובות ה-IP של שולח החבילה.
- כתובות היעד של החבילה, כמו כתובות ה-IP של היעד הסופי.
- שדה זה מאפשר ל-Header להיות גמיש ולכלול בתוכו אפשרות נוספת. השימוש בו נדר לימדי.

נספח ב' - IPv6

כפי שלמדנו בפרק זה כאשר הסבכנו את נושא ה-NAT, בסוף שנות ה-80' נוצרה בעיה אמיתית ומוחשית - נגמרו כתובות ה-IPv4. הדריך התשתיית להתרמודד עם בעיה זו, היא ליצור גירסה חדשה של פרוטוקול IPv6, וכך שתתאפשר בהרבה יותר כתובות מאשר 2^{32} כתובות של IPv4. בנוסף, הייתה שפרוטוקול IPv6 היה בשימוש כבר זמן מה, הוסקו מסקנות לגבי השימושים שלו בראשת האינטרנט, וניתן להפיך מהן ל开玩笑 וליצור גירסה טובה יותר של הפרוטוקול. לשם כך, עלתה בשנת 1995 ההצעה הראשונה לגירסה 6 של פרוטוקול IPv6, הידועה בשם IPv6.

בנספח זה נתאר רק חלק מהມידע הרלוונטי ל-IPv6, ונתמקד בשינויים העיקריים בין IPv4.

כתובות IPv6

הבדל הראשון הוא, כמובן, בכתבאות. כתובות של IPv6 יהיו, כאמור, באורך של ארבעה בתים (bytes), מהם 32 ביטים (bits), ומכך 2^{32} האפשרויות השונות לכתובות של IPv4. ב-IPv6 החולט שכל כתובות תהיה באורך של 16 בתים (bytes), כלומר 128 ביטים (bits). אי לכך, ישן 2^{128} אפשרויות לכתובות IPv6. זהו מספר עצום של כתובות שלא אמרו להיגמר גם כאשר לכל מקרה, מצטמם ומידח תהיה כתובת IPv6 מסוימת.

כתובות IPv6 מחולקות לשוגים: ישן כתובות **Unicast** השונות מכתובות **Multicast**. בנספח זה נתאר כתובות Unicast בלבד. עבור כתובות אלו, 64 הביטים (bits) העליונים מצינים את **מזהה הרשות**, בעוד 64 הביטים התחתוניים מצינים את **מזהה הيسות**.

כתבאות מוצגות באמצעות שמוֹנה "קבוצות" של ארבע ספרות הקסה-דצימליות, כאשר כל "קבוצה" מייצגת למעשה שני בתים (bytes). ה"קבוצות" מופרדות באמצעות הוו נקודות (:). להלן דוגמה של כתובת IPv6:

2340:0000:0000:0000:0000:0000:0001

מכיוון שכתובות כאלה היא ארוכה מאד, ישנים חוקים המאפשרים להציג את הכתובת בצורה קצרה יותר, בייחודה כאשר יש שימוש באפסים. למשל, בכל קבוצה, ניתן להשמיט את האפסים הראשונים. כך למשל, הקבוצה 0010, יכולה להיות מוצגת כ-10 בלבד. באמצעות חוקים אלו, ניתן להציג את הכתובת עילית גם כך:

2340:0:0:A:0:0:0:1

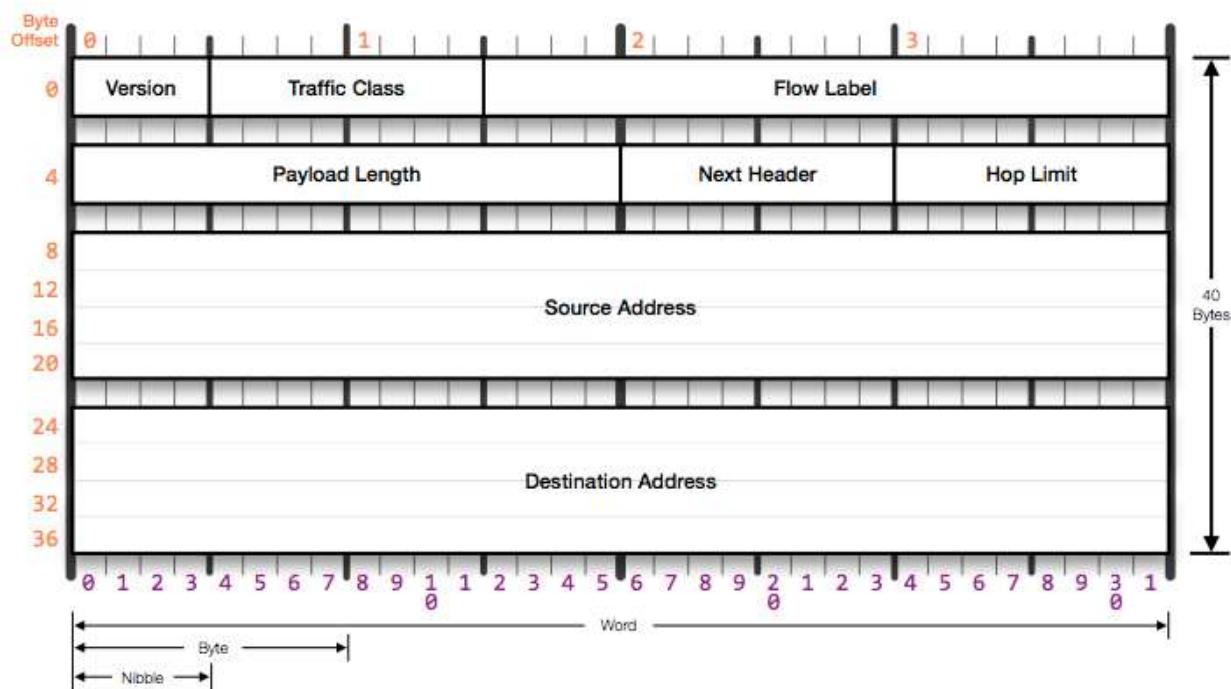
זאת ועוד, קבוצות שכוללות רק את הספרה אפס, שמופיעות ברצף זו אחר זו, יכולות להיות מוצגות באמצעות שני תווי נקודות (:). עם זאת, ניתן להשתמש בשני תווי נקודותיים רק פעם אחת בכתבאות. כך למשל, את הכתובת לעיל ניתן להציג גם בצורה הבאה:

2340::A:0:0:0:1

IPv6 Header

כך נראה ה-Header של חבילה IPv6:

IPv6 Header



השודות הם:

- שדה זה מתאר את גרסת פרוטוקול IP. במקרה של IPv4, ערך שדה זה יהיה 4. במקרה זה, מכיוון שמדובר ב-IPv6, הערך יהיה 6.
- שדה זה דומה לשדה Type Of Service (ToS) ב-IPv4. השדה מאפשר לציין את העדיפויות של חבילה מסוימת על פני חבילות אחרות - כדי לבקש מהנתבים להעביר חבילות בעלות עדיפויות גבוהה לפני חבילות בעלות עדיפויות נמוכה יותר.
- שדה זה נדרש כדי לאפיין חבילות השתייכות לאותו "flow". הכונה היא לחבילות מאותו זרם מידע. על אף-Sh-IP הינו פרוטוקול שלא מבוסס קישור, יש כאן דרך לשער חבילה בודדת לקישור מלא. כך למשל, ניתן לתת את אותו ערך בשדה Flow Label לכל החבילות הקשורות לשיחת VoIP מסוימת, או לתקשורת בין דפדן לבין אתר. במידה שהחbillה לא מקושרת לאף "flow", בשדה זה יהיה הערך 0. הרעיון הוא שכל הנתבים בדרך יטפלו בכל החבילות הקשורות לאותו ה-"flow" באותה דרך. וכך, כל החבילות הקשורות לאותו קישור, ינותבו באותו האופן.
- אורך ה-Payload - Payload Length. האורך כאן מדבר רק על ה-Payload, ולא כולל את ה-Header כמו ב-IPv4. הסיבה לכך היא שאורך ה-Header של IPv6 הוא קבוע.

- Next Header - מTARGET איזה Header מגיע אחריו ה-IPv6 Header. כך למשל, ערך של 6 מזזה זה-Header.
- הבא הוא של TCP, בעוד הערך 17 מזזה זה-Header הבא הינו Header של UDP.
- TTL - זהה במשמעות לשדה ה-TTL ב-IPv4. ההבדל הוא רק בשם. העבודה היא שדה ה-TTL Hop Limit.
- לא היה קשור בזמן, אלא למספר הקפיצות (hops) שחביבה יכולה לעבור. לכן, Hop Limit מהו שמתאים יותר מאשר Time To Live.
- Source Address - כתובת המקור של החביבה, כמובן - כתובת ה-IPv6 של שולח החביבה.
- Destination Address - כתובת היעד של החביבה, כמובן - כתובת ה-IPv6 של היעד הסופי של החביבה.

הבדלמשמעותי אחד בין ה-Header של IPv4 לבין ה-Header של IPv6 הינו קבוע ועומד תמיד על אבעים בתים (bytes). זאת בניגוד ל-IPv4 Header, שכללזיכרון שדה של Options שעשוי להשפיע על האורך שלו.

מעבר לכך, שימושו לב שאין יותר שדה checksum כמו שהוא ב-IPv4. הסיבה לכך היא ששנים של ניסיון הוכיחו שבדרך כלל חבילת IPv4 רצתה מעל שכבה שנייה שכוללת checksum (כגון Ethernet), ומעלה נמצאת שכבה רבייעית שגם כוללת checksum (כגון TCP או UDP, עליהם נלמד בפרק הבא). אי לכך, ה-IPv6 מחושב בכל כרטיס רשת בדרך, והן על ידי מכשירי הקצה. מכאן שאין צורך לחשב את ה-checksum גם בכל נתב ונתב. פועלות חישוב checksum הינה יקרה ומעמיסה על הנתב. אי לכך, נתבים העובדים עם IPv6 יכולים להשיק את זמני בנייטוב של פקודות, ולא בחישוב של checksum.

ל-IPv6 יתרונות רבים נוספים על IPv4, ביניהם תמיימה נוחה בכתובות Multicast, אפשרות לישויות לחתת לעצמן כתובות IP מלבני צורך ב-DHCP (תהליך הנקרא SLAAC) ועוד. על אלו לא נרחב בנספח זה, אך אתם מוזמנים להרחב אופקיכם.

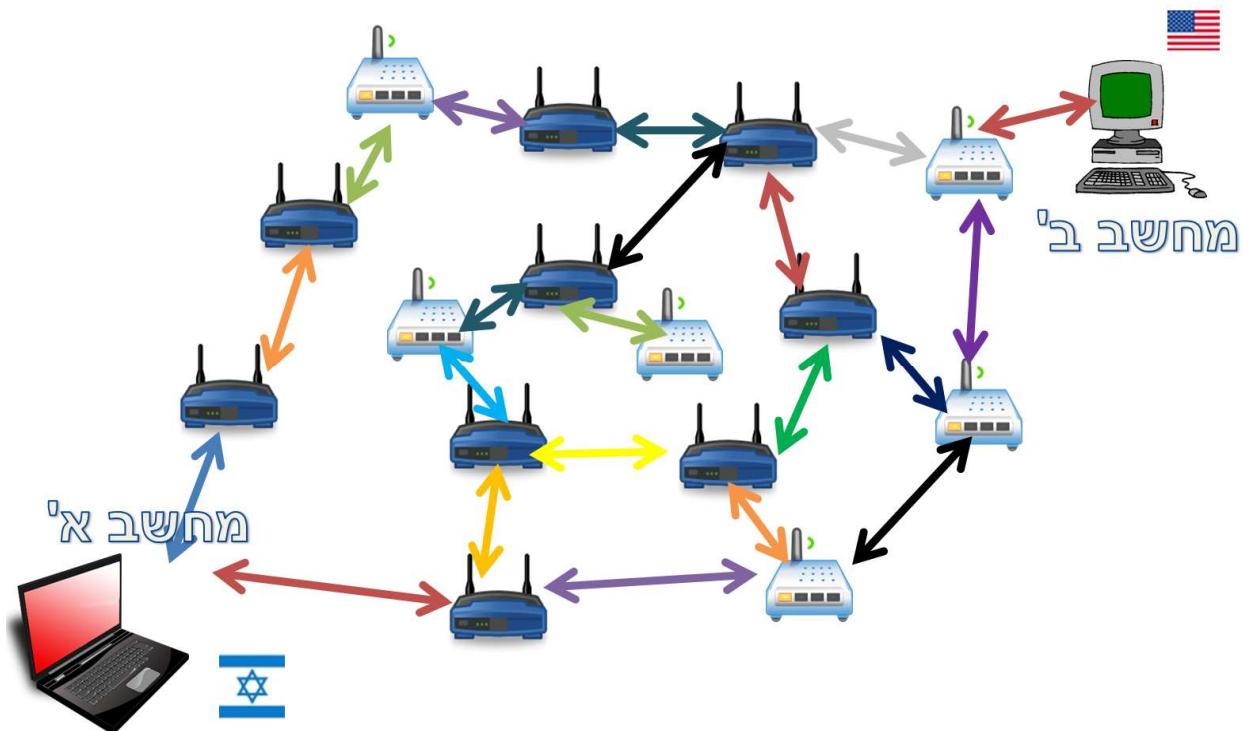
פרק 8 - שכבת הקן

בפרק הקודם למדנו על שכבת הרשת, וצעת אנו מבינים שהabilות מידע שעוברות בין שתי נקודות קצהה עוברות בדרך כלל בין מספר רכיבים בדרך (למשל נתבים). לכל אורך הפרק הקודם, הנחנו שנייה להעביר חבילת בין ישות אחת לשות אחרת כשאלו צמודות זו לזו. עם זאת, פועלה זו אינה כה פשוטה. במהלך הפרק הקרוב נבין את מטרתה של שכבת הקן, נלמד על פרוטוקול Ethernet, כמו גם פרוטוקול ARP, ובנין את האטגרים עימם מתמודדת השכבה.



מה תפקידיה של שכבת הקן?

_nb: בתרמונה המלאה יותר אוננו לאור הספר:



בפרק הקודם ראיינו ששכבת הרשת אחראית להעביר חבילת בין מחשב א' למחשב ב'. כמו כן, הבנו שהיא אחראית על המסלול שבה החביבה מעבור. השכבה השנייה אחראית על התקשרות בין כל שתי יישויות הקשורות זו לזו באופן ישיר. באIOR לעיל, כל חץ צבעוני מייצג תקשורת בשכבה השנייה בין שתי יישויות - כל עוד הן מחוברות זו לזו באופן ישיר, הטיפול של העברת הودעה ביניהן ישיר לשכבה זו. בשכבת הקן אין הבנה של הדרך המלאה שהחביבה עוברת מהמקור אל היעד כמו בשכבת הרשת, אלא רק בין יישויות סמוכות - ככלומר כל חץ צבעוני באIOR לעיל בלבד.



מטרת שכבת הkn היא להעביר מידע בין שתי ישותים מחוברות זו לזו באמצעות ישר

המשמעות של חיבור ישר היא שמיידע יכול לעבור בין הישויות מבלתי עבור בישות אחרת בדרך. חיבור ישר יכול להיות לצורךות שונות. תcan ומדובר בחיבור קווי - משתמשים בקבל פיזי כדי לחבר ישות אחת לאחרת. למשל, נאמר שמחשב א' מחובר לאחד הנתבים הקרובים אליו בכבול. חיבור אחר יכול להיות חיבור אלחוטי - לדוגמה, תcan והנתב של מחשב א' מחובר לנット הבא באמצעות WiFi. תcan אףלו והחיבור יהיה באמצעות יוני דואר - למשל, תcan שהתקשרות בין מחשב ב' לבין הנתב הקרוב אליו מתבצעת באמצעות יוני דואר. מקרים אלו שונים אחד מהשני מאוד, והשכבה השנייה צריכה לדאוג לכך שabitutes המידע יצליחו לעבור מישות לשות בקרה אמינה.

השכבה צריכה להתמודד עם תקלות שיכולות להיות בהkn, עליה נפרט בהמשך.



שכבת הkn מספקת לשכבת הרשות מושך להעברת מידע בין שתי ישותים מחוברות זו לזו באמצעות ישר

באופן זה, שכבת הרשות לא צריכה לדאוג לסוגיות הקשורות לחיבור בין שתי תחנות. את שכבת הרשות לא מענין אם הישויות מחוברות בכבול, בלויין, ב-WiFi, או באמצעות יוני דואר. היא רק אחראית להבין מה המסלול האופטימלי. כמו Sh-Waze רק אומרת לרכב באיזו דרך עבור, ולא מסבירה לנו הגשה הוא צריך לרדוף, ללחוץ על הגז, לאוותת ולעוצר ברמזור או להולך רגל. בזה יטפל הנגה, או במקרה שלנו - שכבת הkn.



אייפה ממומשת שכבת הkn?

השימוש של שכבת הkn "נמצא" בכל ישות ברשות - וספציפית, בכרטיס הרשות של הישות. כך למשל כרטיס Ethernet ימשח את פרוטוקול Ethernet, וכרטיס WiFi ימשח את פרוטוקול WiFi. כרטיס רשות שונים מתקשרים זה עם זה. עם זאת, כרטיס רשות Ethernet לא יכול לתקשר ישירות עם כרטיס רשות של WiFi.

Ethernet פרוטוקול

בפרק זה נתמקד בפרוטוקול Ethernet, בו משתמשים כרטיסי רשת מסוג Ethernet. כשאנו מדברים על כרטיסי Ethernet, הכוונה היא לכרטיס רשת המתחבר באופן קוו, עם כבל שנראה כך⁷⁰:



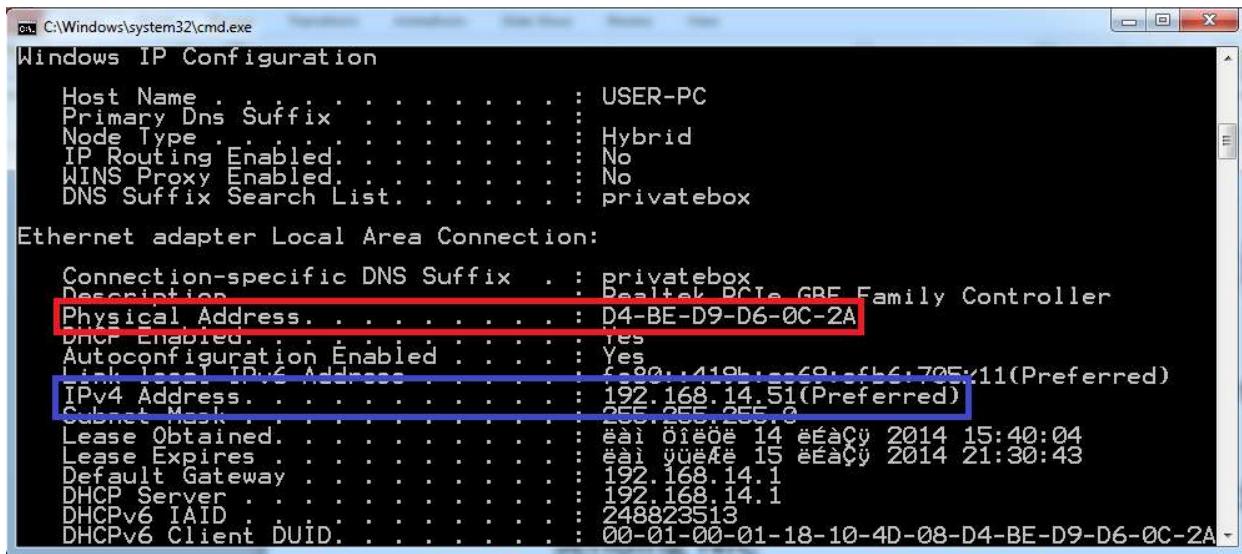
מהי כתובת שלי בשכבה הקו?

לצורך הסבר זה נניח שכרטיס הרשת שלכם הוא מסוג Ethernet. אם הוא לא, אנא עבדו על מחשב שיש לו כרטיס רשת צזה.

על מנת לתקשר זה עם זה, כרטיסי הרשת צריכים שיהיו להם מזהים - או כתובות, בהם הם יכולים להשתמש. דבר זה נכון מכיוון שבחלק מהמקרים בשכבה הקו, שתי הישויות שמנוטות לתקשורת מוחוברות באופן ישיר לא רק אחת לשניה, אלא גם לישויות נוספות. לדוגמה, חישבו על רשת WiFi - כלל הישויות המוחוברות לרשת יכולות לתקשר זו עם זו באופן ישיר, ככלם בלי לעבור באף תחנה אחרת בדרך. באם ישות מסוימת רוצה לפנות אל ישות אחרת, היא צריכה לפנות אל כתובתה שלה. כתובות בשכבה השנייה נקראות **כתובות MAC (באנגלית - MAC Addresses)**.

היכנסו שוב ל-*Command Line*, וחקישו את הפקודה הבאה: **ipconfig /all**. שימוש לב שהשתמשנו בפקטור **all**, שכן אחרת הפקודה ipconfig לא מציגה את כתובת השכבה השנייה.

⁷⁰ [פרק השכבה הפיזית / הרשת המשדרית](#), למד יותר על כבל זה.



ניתן לראות שהמידע מוצג עבור הכרטיס עם כתובת ה-IP שמצאנו בפרק הקודם, 192.168.14.51, כפי שמסומן **בכחול**. מעת לעלה יותר, יש שדה בשם **Physical Address**, זהה לכתובת ה-MAC, המסומנת **באדום**. שימושו לבשcomo שהפקודה מציגה לנו, אכן מדברת **בכתובת פיזית** - כתובת זו "צרכובה" על כרטיס הרשות עצמו, ולא אמורה להשתנות⁷¹. כמו כן, היא אמורה להיות ייחודית - כמובן, לא אמור להיות עוד כרטיס רשות בעולם בעל אותה כתובת בדיק. בהמשך הפרק נסביר כיצד מורכבת כתובת MAC.

מהי כתובת ה-MAC שלכם? מיצאו אותה עצם.



תרגיל 8.1 מודרך - הסניף כתובות ה-MAC ברשות המקומית



בתרגיל זה נבצע הסניפה, ובמהלכה נדפיס את כל כתובות ה-MAC של היישויות שפנו אל כתובת ה-MAC שלנו. לשם כך, נשתמש בכלי Scapy. נפתח את Scapy, ונבצע הסניפה פשוטה של שתי חבילות מידע. בשלב זה נכיר מונח נוסף בשם **מסגרת (Frame)**. בעוד חבילת מידע בשכבה הרשות נקראת חבילה או פקטה (Packet) בשכבה הקרו רצף המידע שעבר נקרא מסגרת. לאחר מכן, נסתכל על אחת המסגרות שהסניפה:

```
>>> frames = sniff(count=2)
>>> frame = frames[0]
>>> frame
```

⁷¹ חלק מהמיימושים של השכבה השנייה יש אפשרות להשתמש בכתובת שאינה צרכובה על הכרטיס.

```

C:\Windows\system32\cmd.exe - scapy
>>> frames = sniff(count=2)
>>> frame = frames[0]
>>> frame
<Ether dst='00:0c:c3:a5:16:63' src='d4:be:d9:d6:0c:2a' type='0x800' |>>>
  ihl=5L tos=0x0 len=41 id=703 flags=DF frag=0L ttl=128 proto=tcp checksum=0x0 src='192.168.14.51' dst='173.194.78.125' options=[] |>>> TCP sport=53442 dport=5222 seq=2228948902L ack=3568354024L dataofs=5L reserved=0L flags=A window=16325 checksum=0xcb36 urgptr=0 options=[]
>>> -

```

כפי שניתן לראות, בשכבה ה-Ethernet נמצאות ראשית כתובת היעד של המsgרת (destination address) שמופיעיה בשם "dst"), וכתובת המקור של המsgרת (source address) בשם "src".

כפי שראנו בפרקים הקודמים, ניתן להשתמש בפורמטר **ifilter** של הפונקציה **sniff** של Scapy כדי לסקן מסגרות (או חבילות) המתאימות לתנאי שלנו. על מנת לסנן מסגרות שפוננות לכתובת ה-MAC של הכרטיס שלנו בלבד, علينا לכתוב פונקציה שתחזיר True אם שדה כתובת היעד של המsgרת תואם את כתובת ה-MAC שלנו. לשם כך, נגדיר קודם את כתובת ה-MAC אותה מצאנו קודם لكن באמצעות **ipconfig**, בצורה הבאה:

```
>>> MY_MAC = 'd4:be:d9:d6:0c:2a'
```

שימוש לב Ci הכתובות שלכם תהיה שונה מהכתובות שמופיעיה בדוגמה. כמו כן, על אף ש-**ipconfig** הציג את הכתובות כאשר כל בית (byte) מופרד באמצעות התו מקף ('-'), אנו משתמשים בפורמט של Scapy בו כל בית מופרד באמצעות התו נקודות ('.'). בנוסף, אנו כותבים באותיות קטנות ('a') כפי שעושה Scapy, ולא באותיות גדולות ('A') כפי שעושה **ipconfig**.

עתה נוכל לכתוב את הפונקציה שלנו:

```
>>> def filter_mac(frame):
    return (Ether in frame) and (frame[Ether].dst == MY_MAC)
```

בשלב הראשון, יידאו שמדובר במסגרת Ethernet. לאחר מכן, הפונקציה מחזירה True במידה שכתובת היעד של המsgרת היא הכתובת של כרטיס הרשות שלנו. אם לא, היא מחזירה False. לשם הבהרה, ניתן להזכיר גם את הפונקציה בצורה הבאה:

```
>>> def filter_mac(frame):
...     if Ether not in frame:
...         return False
...     if frame[Ether].dst == MY_MAC:
...         return True
...     else:
...         return False
```

```

C:\Windows\system32\cmd.exe - scapy
>>> MY_MAC = 'd4:be:d9:d6:0c:2a'
>>> def filter_mac(frame):
...     return (Ether in frame) and (frame[Ether].dst == MY_MAC)
>>>

```

כעת נוכל להסניף רק מסגרות שייעדו אל כתובת ה-MAC שלנו באמצעות פונקציית הфиילטר. נעשה זאת כך:

```
>>> frames = sniff(count=10, lfilter=filter_mac)
```

נוודא זאת על ידי התבוננות בכתובת היעד של שתי מסגרות:

```
>>> frames[0][Ether].dst
```

'd4:be:d9:d6:0c:2a'

```
>>> frames[4][Ether].dst
```

'd4:be:d9:d6:0c:2a'

```

C:\Windows\system32\cmd.exe - scapy
>>> MY_MAC = 'd4:be:d9:d6:0c:2a'
>>> def filter_mac(frame):
...     return (Ether in frame) and (frame[Ether].dst == MY_MAC)
>>> frames = sniff(count=10, lfilter=filter_mac)
>>> frames[0][Ether].dst
'd4:be:d9:d6:0c:2a'
>>> frames[4][Ether].dst
'd4:be:d9:d6:0c:2a'
>>> -

```

כעת בראצנו להציג את כתובות ה-MAC של היעדים והפוניות אליהם. לשם כך - נשתמש בפרמטר **prn** של הפונקציה **sniff**, אשר גם אותו הכרנו בפרקם קודמים בספר. נגדיר את הפונקציה שתתפל בכל מסגרת, וכך שתוודף למסך כתובות המקור של המסגרת:

```
>>> def print_source_address(frame):
```

```
    print frame[Ether].src
```

כעת נבצע את ההסנפה:

```

C:\Windows\system32\cmd.exe - scapy
>>> def print_source_address(frame):
...     print frame[Ether].src
>>> frames = sniff(count=10, lfilter=filter_mac, prn=print_source_address)
00:0c:c3:a5:16:63
>>> -

```

כפי שראיתם, בדוגמה זו כל הכתובות היו שייכות לאוותה היעשת (עליה למד בהמשך). במצבו הנוכחי, הסקריפט שכתבנו לא כל כך מועיל.

תרגיל 8.2 - מציאת כתובות MAC שפונגו אל כרטיס הרשת שלך



שפרו את הסקריפט שכתבנו עד כה. גירמו לכך שהסקריפט "יזכור" האם הוא הדףיס כתובת מסויימת למסך, ואם כן - לא ידפיס אותה שוב. אפשרו לסקריפט לזרז במשר חמש דקודות (רמז) - קראו על פרמטרים נוספים לפונקציה `(sniff)`.

על מנת לגרום לסקריפט להדפיס כתובות נוספות, נסו לפחות אליהם מישיות נוספות ברשות שלכם, למשל מחשב אחר שקיים ברשות. תוכלו להיעזר לשם כך בפקודה `ping` עליה למדנו בפרק שכבת הרשת/ איך Ping עובד?.



איך בנויה כתובת Ethernet?

עד כה בפרק הספקנו לראות מספר כתובות MAC של Ethernet. כפי שווידאי הבחנתם, כתובות Ethernet בנויות מישיה בתים (bytes). ניתן לייצג את הכתובות בדרךים שונות. ניתן, כפי שעשו הפקודה `ipconfig`, לייצג את הכתובות באמצעות הפרדה עם התו מקף ('-') בין הבתים השונים, לדוגמה:

D4-BE-D9-D6-0C-2A

ניתן גם, כפי שעשו Scapy, לייצג את הכתובת באמצעות הפרדה עם התו נקודותיים ('.') בין הבתים, למשל:
D4:BE:D9:D6:0C:2A

ניתן גם לבצע הפרדה רק בין צמדים של בתים, למשל:

D4BE:D9D6:0C2A

ישנן דרכים נוספות לייצג את הכתובות, אך במקרה חסוב להבין שמדובר ברצף של שישה בתים.

עם זאת, לא לכל הבתים יש את אותה המשמעות. באופן כללי, כתובת Ethernet מחלוקת לשני חלקים:

- **מזהה יצור (Vendor ID)** - מזהה מי החברה שייצרה את כרטיס הרשת.
- **מזהה ישות (Host ID)** - מזהה של כרטיס הרשת הספציפי.

שלושת הבתים העליונים (הראשונים) שייכים למזהה היצור, בעוד שלושת הבתים התחתונים שייכים למזהה הישות.

כך למשל, בכתובת שהציגנו קודם לכן:

D4:BE:D9:D6:0C:2A

שלושת הבתים העליונים (המסומנים ב**אדום**) הם מזהה היצור, ושלושת הבתים התחתונים (המסומנים ב**כחול**) שייכים לכרטיס הרשת הספציפי.
כך, אם נסתכל בכתובת הבאה:

D4:BE:D9:11:22:33

ונכל לדעת שני כרטיסי הרשת יוצרים באותו ידי אותו יצור, שכן מזהה היצור שלהם (המסומן ב**אדום**) זהה.

מזהה היצרנים ידועים, ولكن ניתן לדעת בקלות לאיזה יצran שיכת כתובות מסוימת. לדוגמה, נכנס לאתר http://www.coffer.com/mac_find ונצין לתוךו את אחת משתי כתובות MAC לעיל, שמתחלות בmazeה היצran D4:BE:D9. האתר יספר לנו שמדובר בכתבota השיכת ליצרנית Dell. השימוש לבשילצראניות שונות עשוי להיות יותר mazeה יצran אחד. כך למשל, גם mazeה היצran E0:DB:55 שייר ל-Dell, גם mazeה DB:BA:A4, וגם רבים נוספים.



תרגיל 8.3 - מציאת היצרנית של כרטיס הרשות שלו

באמצעות האתר שהוצג לעיל, כמו גם פקודה **ipconfig**, מצאו מי היצרנית של כרטיס הרשות שלכם.



תרגיל 8.4 - מציאת יצרניות של כרטיסי רשות מתוך הסנפה

הורידו את קובץ ההסנפה **Layer2_1.pcap** מהכתובת:
http://data.cyber.org.il/networks/c07/Layer2_1.pcap. היעזרו בהסנפה כדי למצוא את שתי כתובות ה-MAC שנמצאות בה, ולאחר מכן גלשו לאתר שהוצג לעיל ומצאו את היצרנים של כתובות MAC המופיעות בקובץ. הוציאו אותם לטבלה הבאה:

יצran	כתובת MAC

Broadcast

הכתובת FF:FF:FF:FF:FF:FF הינה כתובת Ethernet מיוחדת. כתובת זו היא כתובת Broadcast - כלומר כל הישויות ברשת. שליחת מסגרת עם כתובת היעד FF:FF:FF:FF:FF:FF משמעotta שליחת המסגרת לכל הישויות שנמצאות איתנו ברשת⁷².

⁷² כתובת Broadcast שיכת למעשה לקבוצה כל הישויות ברשת. על כתובות Ethernet של קבוצות נלמד [בנספח א' של פרק זה](#).

תרגיל 8.5 - כתובות בהסנפה



הורידו את קובץ ההסנפה Layer2_Broadcast.pcap מהכתובת:

ענו על השאלות הבאות: http://data.cyber.org.il/networks/c07/Layer2_Broadcast.pcap

1. כמה מסגרות נשלחו אל כתובות Broadcast בرمת-h-Ethernet? מה המספר הסידורי של מסגרות אלו בהסנפה?
2. איזה מסן תצוגה (display filter) יש לחת ל-Wireshark כך לסנן רק את המסגרות שנשלחו לכתובות Broadcast בرمת-h-Ethernet?

תרגיל 8.6 - כתובות Ethernet



בתרגיל זה תכתבו בפייתון סקריפט אשר מבקש מהמשתמש כתובות MAC ומדפיס עליה מידע.

1. קבלו מהמשתמש כתובות MAC. על הכתובת להיות בפורמט AA:BB:CC:DD:EE:FF (הפרדה של כל בית באמצעות התוו נקודות). האותיות יכולות להכתב כאותיות קטנות ('a') או גדולות ('A'). לאחר קבלת הכתובת, הדפסו "Valid" אם הכתובת הינה כתובות Ethernet תקינה, ו-"Invalid" אם הכתובת אינה תקינה.

בחנו את עצמכם עם הכנסת הקלטים הבאים:

- 11:22:33:44:55:66 (כתובת תקינה)
- FF:FF:FF:FF:FF:FF (כתובת תקינה)
- AB:12:cd:34:31:21 (כתובת תקינה)
- 11:22:33:44:55:66:77 (כתובת שנייה תקינה)
- 11-22-33-44-55-66 (כתובת שנייה תקינה עבור סקריפט זה)
- 11:22:33:44:55 (כתובת שנייה תקינה)
- H:22:33:44:55:661 (כתובת שנייה תקינה)

רמז: העזרו במתודה `split`. לחופין, ניתן לקרוא על regular expressions. (<https://docs.python.org/2/library/re.html>) ולהשתמש בהם.

2. במידה שהכתובת תקינה, הדפסו את מזהה היצן. אין צורך להדפיס את שמו של היצן, אלא רק את המזהה (למשל: 11:22:33).

מבנה מסגרת Ethernet

מסגרת Ethernet נראית כך:

Preamble	Destination Address	Source Address	Type	Data	CRC
					32

- Preamble - רצף קבוע מראש של שמונה בתים (bytes) שנועד לסייען את שני הצדדים על כך שמתחליה מסגרת חדשה. שימו לב - לא רואים את שדה זה ב-Wireshark.
- Destination Address - כתובות היעד של המסגרת. שדה זה מכיל שישה בתים (bytes), בפורמט עליון למדנו קודם לכן.
- Source Address - כתובות המקור של המסגרת. שדה זה מכיל שישה בתים (bytes), בפורמט עליון למדנו קודם לכן.
- Type - סוג המסגרת. שדה זה מכיל שני בתים (bytes) שמעידים על סוג ה-Data. כך למשל, מסגרת שבה שדה זה מכיל את הערך 0x800 הינה מסגרת מסוג IP. כך כרטיס הרשות יודע להפנות את המידע של המסגרת (במקרה זה - חבילת IP) אל הגורם שיודע לטפל במידע זה⁷³.
- Data - המידע עצמו של החבילה. על המידע להיות באורך של 64 בתים (bytes) לפחות. אם המידע קצר יותר, נוסף רצף של אפסים (00) בסוף המידע.
- CRC32 - מגנון Checksum לגילוי שגיאות. על משמעותו של Checksum למדנו במסמך פרק שכבת התעבורה מה זה? בפרוטוקול Ethernet, אורך ה-CRC הוא 32 ביטים (bits), שהם ארבעה בתים (bytes). גם שדה זה לא נראה ב-Wireshark, שכן הוויידוא שלו מתרחש אצל כרטיס הרשות עוד לפני ש-Wireshark "רואה" את המסגרת.

תרגיל 8.7 מודרך - התבוננות בבקשת HTTP



פתחו את Wireshark והתחילה הסנה עם מסנן התצוגה "http". במקביל, פיתחו את הדפדפן שלכם וגילשו אל www.ynet.co.il. מיצאו את החבילות הרלוונטיות בהסנה. באופן ספציפי, מיצאו את חבילת ה-GET, והבטו בה:

```

# Frame 23: 1102 bytes on wire (8816 bits), 1102 bytes captured (8816 bits) on interface 0
# Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)
# Internet Protocol Version 4, Src: 192.168.14.51 (192.168.14.51), Dst: 81.218.31.168 (81.218.31.168)
# Transmission Control Protocol, Src Port: 54671 (54671), Dst Port: http (80), seq: 1361, Ack: 1, Len: 1048
# [2 Reassembled TCP Segments (2408 bytes): #22(1360), #23(1048)]
# Hypertext Transfer Protocol
# GET /home/0,7340,L-8,00.html HTTP/1.1\r\n
Host: www.ynet.co.il\r\n
Connection: keep-alive\r\n
Cache-Control: no-cache\r\n
Pragma: no-cache\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n

```

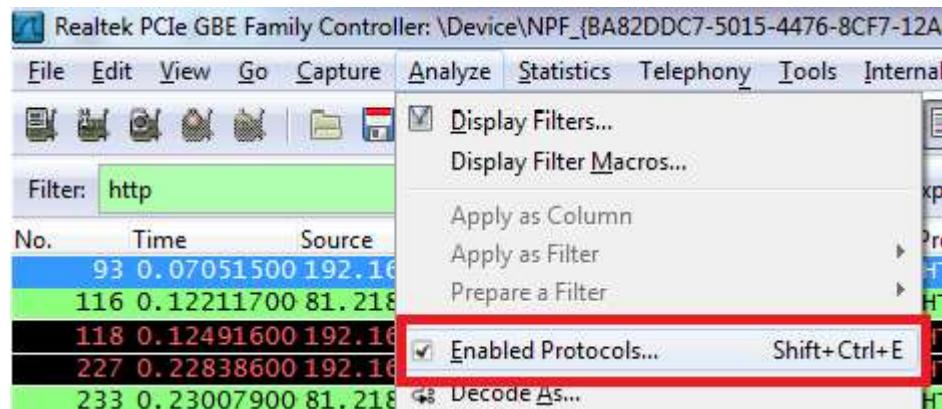
⁷³ במקרים מסוימים, שדה זה גם יכול להיעיד על האורך הכלל של המסגרת. עם זאת, במספר זה ניתן מאפשרות זו. אתם מוזמנים לקרוא עליה באינטרנט.

שימוש לב למודל השכבות:

- בשכבה השנייה, שכבת ה-Ethernet - פרוטוקול Ethernet.
- בשכבה השלישית, שכבת הרשות - פרוטוקול IP.
- בשכבה הרביעית, שכבת התעבורה - פרוטוקול TCP.
- בשכבה החמישית, שכבת האפליקציה - פרוטוקול HTTP.

הזכירו במושג **הכימוס (Encapsulation)** אותו הכרנו בפרק Wireshark ומודל חמש השכבות/ כיצד בנויה TCP פקטה? מוגרתת ה-Ethernet שלנו מכילה "בתוכה" את שכבת ה-IP, שמכילה "בתוכה" את שכבת ה-TCP. שמכילה "בתוכה" את שכבת ה-HTTP.

הסנהה של תעבורת HTTP ביצענו כבר בפרק שכבת האפליקציה, והפעם איננו מעוניינים בשכבת ה-HTTP. אי לכר, היכנסו ב-Wireshark לתפריט Analyze בסרגל הכלים, ובחרו באפשרות Enabled Protocols:



כעת, הורידו את הסימון מהפרוטוקול 4vIP. לחזו על OK. הסירו את מסנן התצוגה "http", שכן Wireshark כבר אינו מכיר אותו. כעת Wireshark יציג בפנינו רק את השדות של שכבת ה-Ethernet, TCP, וכל השאר יראה כ-Data. ממש כמו שכרטטי הרשות שלנו רואה את המסרת:

No.	Time	Source	Destination	Protocol	Length	Info
23	0.11431400	Dell_d6:0c:2a	Bewan_a5:16:63	0x0800	1102	IP
24	0.11555100	Bewan_a5:16:63	Dell_d6:0c:2a	0x0800	62	IP
25	0.11560200	Dell_d6:0c:2a	Bewan_a5:16:63	0x0800	54	IP
26	0.11636800	Bewan_a5:16:63	Dell_d6:0c:2a	0x0800	62	IP
27	0.11670600	Dell_d6:0c:2a	Bewan_a5:16:63	0x0800	54	IP

Frame 23: 1102 bytes on wire (8816 bits), 1102 bytes captured (8816 bits) on interface 0

Ethernet II, Src: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Bewan_a5:16:63 (00:0c:c3:a5:16:63)

Destination: Bewan_a5:16:63 (00:0c:c3:a5:16:63)

Source: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)

Type: IP (0x0800)

Data (1088 bytes)

Data: 4500044000bc40008006b59ec0a80e3351da1fa8d58f0050...
[Length: 1088]

0000	00	0c	c3	a5	16	63	d4	be	d9	d6	0c	2a	08	00	45	00c..*..E.
0010	04	40	00	bc	40	00	80	06	b5	9e	c0	a8	0e	33	51	da	@...@...3Q.
0020	1f	a8	d5	8f	00	50	af	14	cb	2c	70	52	08	4e	50	18P..	,.pR.NP.
0030	41	14	44	90	00	00	34	31	31	2d	34	31	32	2d	34	31	A.D...	41 1-412-41
0040	33	2e	33	36	38	40	2d	31	33	30	38	36	38	36	34	37	3. 368@-1	30868647
0050	36	40	30	40	34	34	39	2e	33	36	39	40	2d	32	30	31	6@0@449,	369@-201
0060	32	32	39	32	34	39	36	40	30	40	34	35	30	2e	33	37	2292496@	0@450. 37
0070	30	40	2d	31	31	39	31	39	31	33	34	37	40	30	40	34	0@-11919	1 347@0@4
0080	35	31	2e	33	37	31	40	2d	31	31	38	33	39	35	30	31	51. 371@-	11839501
0090	33	34	40	30	40	34	35	32	2e	33	37	32	40	2d	37	33	34@0@452	. 372@-73
00a0	39	33	30	33	35	30	32	40	30	40	34	35	33	2e	33	37	9303502@	0@453. 37
00b0	33	40	2d	31	38	36	35	30	36	32	37	33	35	40	30	40	3@-18650	62735@0@
00c0	34	35	34	2e	34	31	32	40	34	37	30	38	37	31	36	34	454. 412@	47087164
00d0	35	40	30	40	35	30	30	2d	35	30	31	32	35	30	32	2e	5@0@500-	501-502.
00e0	34	31	33	40	32	31	34	31	31	38	34	30	39	31	40	30	413@2141	184091@0

הסתכלו על השדות השונים ב-Header של Ethernet

- כתובות היעד - האם זהה הכתובת של netwOrk? התשובה היא לא! היזכרו בפרק שכבת הרשת, בו דיברנו על כך שבדרך כלל ישנו רכיבים רבים שמקשרים בין ישויות קצה ברשת. בהנחה ואינכם מחוברים באופן ישיר אל netwOrk באמצעות כל (או דרך אחרת), אתם עוברים בדרך אישות נוספת. הישות הקרובה ביותר לאליכם היא **הנתב שלכם**, וכותבת ה-MAC זו היא הכתובת שלהם. כמו כן, שימושם לב-sh-Wireshark יודע להגיד לנו שהיצירנית של הנתב הינה Bewan.
 - **שימוש לב:** לנ眉头 יש יותר מכתובת MAC אחת, שכן יש לו יותר כרטיסי רשת אחד. כתובות ה-MAC שמוצגת בהסנפה היא הכתובת של כרטיס הרשת של הנתב המחבר אל הרשת הביתית שלהם, ולא של כרטיס הרשת של הנתב שמחובר אל האינטרנט.
 - כתובות המקור - כתובות זו צפוייה להיות הכתובת של כרטיס הרשת שלכם. כדי שניתן לראות בדוגמה לעיל, הכתובת a0:0c:2a:d9:d6:0c/be:d4 זהה לכתובת שמצאו באמצעות הפקודה ipconfig/all. כמו כן, Wireshark יודיע להגיד לנו שהיצירנית של כרטיס הרשת הינה Dell.
 - סוג - הערך 0x800 מציין על כך שהיא כתובת IP. כך כרטיס הרשת ידע להפנות את כל מה שנמצא בשדה ה-Data אל הישות שמתפלת בחבילות IP (במקרה שלנו - מערכת הפעלה).
 - שדה ה-Data יכול את כל השכבות שנמצאות "על" ל-Ethernet, החל משכבה ה-IP, דרך שכבת TCP ועד שכבת HTTP.
 - שימוש לב שודות ה-Preamble וה-Checksum אינם מופיעים בהסנפה, כפי שציינו קודם לכן.



שיםו לב - כאן רואים באופן יפה את ההבדל בין השכבה השנייה לשכבה השלישייה. בחבילת ה-GET שמצוגת לעיל, כתובות המקור בשכבה השלישייה, שכבת הרשות, היא כתובות ה-IP של המחשב שלנו, וכתובות היעד היא כתובות ה-IP של השירות של Ynet. שכבת הרשות מציגה את כל המסלול - מאייפא החבילה נשלחה ומהו היעד הסופי שלה. עם זאת, בשכבה השנייה, שכבת הקו, כתובות המקור היא כתובות כרטיס הרשות של המחשב שלנו וכתובות היעד היא הכתובת של הנטב הקרוב, שכן שכבת הקו מדברת על קשר בין ישויות הקשורות באופן ישיר בלבד. לכן, בעוד בשכבת הרשות נראה את הכתובת ההתחלתית והסופית של החבילה, בשכבת הקו אנו נראה כל שלב בדרך.



כפי שראנו בפרק [שכבת הרשות / ניתוב](#), בשלב הבא החבילה תועבר מהנטב הקרוב למחשב שלנו אל הנטב הבא בדרך. בשלב זה, הנטב הקרוב למחשב שלנו ייצור את המסגרת בשכבת ה-[Ethernet](#) כך שכתובת המקור של המסגרת תהיה הכתובת של כרטיס הרשות שלו שמחובר לאינטרנט, וכתובות היעד תהיה הכתובת של הנטב הבא. כך, שכבת הקו מתארת נכון את ה-[hop](#) הנוכחי: מעבר בין הנטב שקרוב עליו אל הנטב הבא אחריו. עם זאת, בשכבת הרשות עדיין ישמרו הנ吐נים על נקודות הקצה של החבילה - המעברת ממחשב שלנו אל Ynet.



תרגיל 8.8 - התבוננות בתשובה HTTP



כעת הסתכלו בתשובה ה-HTTP שחרזה בתגובה לבקשת ה-GET. ענו על השאלות הבאות:

1. למי שייכת כתובת היעד של המסגרת? כיצד תוכלו לוודא זאת?
2. למי שייכת כתובת המקור של המסגרת? כיצד תוכלו לוודא זאת?
3. מהו סוג המסגרת?

בסיום התרגיל, אל תשכחו לחזור לतפריט Analyze->Enabled Protocols ב-Wireshark ולחזיר את הסימון לפוטווקול 4 IPv4.

תרגיל 8.9 - כיצד תיראה החבילה שלי?



עימנו בתרשימים הרשות הבא:

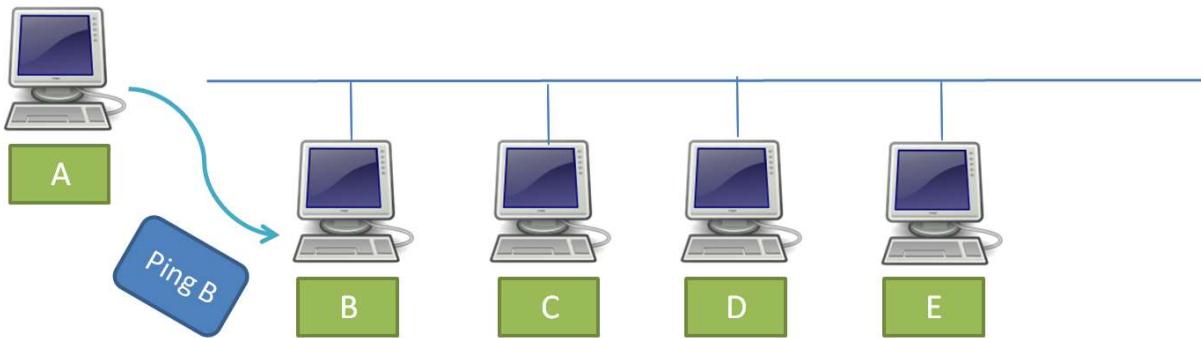


מחשב א' רוצה לשולח פקטה למחשב ב'. מחשב א' מחובר למחשב ב' דרך הנטב.
השלימו את השדות של הפקטה אותה ישלח מחשב א':

1.1.1.1	כתובת IP מטר
c:34:12:3464:20:0	כתובת MAC מטר
	כתובת IP יעד
	כתובת MAC יעד

פרוטוקול ARP - Address Resolution Protocol

עד כה תיארנו את פרוטוקול Ethernet וכייזד הוא עובד. אך עדין, משחו חסר. הבינו בשרטוט הרשות הבא:



כל המחשבים כאן נמצאים על תווך משותף. כלומר - מסגרת הנשלחת לכתובת Broadcast TAG יגיע אל כולם, ואין צורך בישות נוספת (כגון נתב) כדי להעביר הודעות ממחשב אחד למחשב אחר. במקרה לנו, המחשב שנתקרא A רוצה לשולח הודעה ping (להלן Echo-Request, כמו שלמדנו בפרק שכבת הרשות) אל המחשב B. המחשב A יודע את כתובת ה-IP של מחשב B, למשל, באמצעות שימוש בפרוטוקול DNS. אך דבר זה אינו מספיק - על מחשב A לדעת גם את כתובת ה-MAC של כרטיס הרשות של מחשב B!

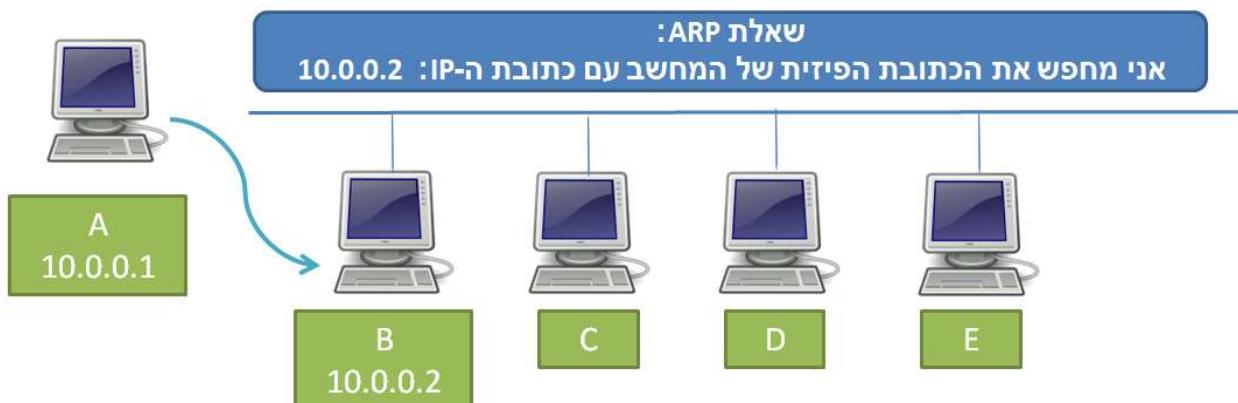


מדוע הדבר כך? מדוע לא מספיקה כתובת ה-IP?

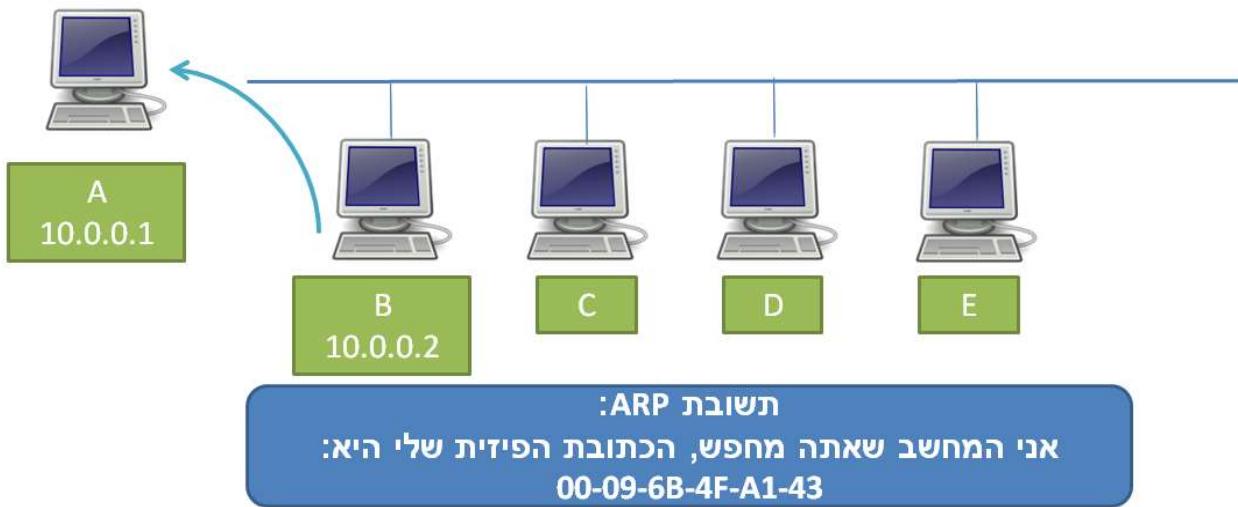
ראשית, הייצו במודל השכבות. חבילת Ping A שתשלה מחשב A צפופה להיות בניה שכבת ה-Ethernet בשכבה השנייה, מעליה שכבת IP ולבסוף שכבת ICMP. על מנת לבנות את מסגרת ה-IP, על המחשב A לדעת מה כתובת היעד של המסגרת, כלומר מה כתובת של כרטיס הרשות של B. שנית, על כרטיס הרשות של המחשב B להבין שהמסגרת מיועדת אליו. את זאת הוא עושה באמצעות הסתכלות בשדה כתובת היעד של המסגרת. **כרטיס הרשות אינו מבין כתובת IP**, הוא כרטיס מסווג Ethernet ומכיר כתובת Ethernet בלבד. על כן, הכרטיס צריך לראות כתובת MAC.

אי לכך, על המחשב A להבין מה כתובת ה-MAC של המחשב B. לשם כך נועד פרוטוקול ARP (או בשמו המלא - **Address Resolution Protocol**). פרוטוקול זה ממפה בין כתובותולוגיות של שכבת הרשות לכתובות פיזיות של שכבת הקווים.

במקרה שלנו, המחשב A מעוניין למפות בין כתובת ה-IP הידועה לו של מחשב B, לבין כתובת ה-MAC של כרטיס הרשות של המחשב B. בשלב הראשון, המחשב A ישלח שאלה לכל הרשות (להלן - לכתובת Broadcast) שמשמעותה: **מי יש את כתובת הפיזית של המחשב עם כתובת ה-IP של B?** נאמר שכותוב ה-IP של מחשב B הינה 10.0.0.2, והכתובת של המחשב A הינה 10.0.0.1:



בשלב זה, כל המחשבים מקבלים את הודעה. מי שצפוי לענות להודעה זו הוא המחשב B, אשר ידוע את הכתובת הפיזית שלו.



אם מחשב A מגלה את כתובתה MAC של כרטיס הרשות של מחשב B. אז, יש ברשותו את כל המידע שהוא צריך כדי לשלוח את חבילת Ping:

- כתובת ה-IP שלו עצמו (מחשב A) - שכן הוא ידוע מה כתובתה שלו, למשל באמצעות DHCP.
- כתובת ה-MAC שלו עצמו (מחשב A) - שכן הוא ידוע מה כתובתה שלו, שהרי היא צרובה על הkartis.
- כתובת ה-IP של מחשב B - שהוא גילה, למשל, באמצעות DNS.
- כתובת ה-MAC של מחשב B - שהוא גילה באמצעות פרוטוקול ARP.

מטען ARP (Cache)

גם עבור פרוטוקול ARP מערכת הפעלה שלנו שומרת מטען (Cache), במטרה לא לשאול שוב ושוב את אותה שאלה ARP. כפי שכבר הבנו, המחשב שלנו זוקק לתקשר עם הנット הקרוב אליו באופן תדיר. על מנת לאפשר

את התקשרות עם הנטב, עליו לדעת מה כתובת ה-MAC שלו. לא הגיוני שלפני כל חבילה שהמחשב יעביר לנטב הוא יבצע שאלת ARP ויחכה לתשובה - תהלייר זה יקח זמן רב מדי. לכן, סביר שכותבת ה-MAC של הנטב תישמר במדויק.

על מנת להביט במדויק, היכנסו לשורת הפקודה והריצו את הפקודה:

arp -a

Internet Address	Physical Address	Type
192.168.14.1	00-0c-c3-a5-16-63	dynamic
192.168.14.200	00-1b-a9-76-7d-b4	dynamic
192.168.14.255	ff-ff-ff-ff-ff-ff	static
224.0.0.2	01-00-5e-00-00-02	static
224.0.0.22	01-00-5e-00-00-16	static
224.0.0.251	01-00-5e-00-00-fb	static
224.0.0.252	01-00-5e-00-00-fc	static
239.255.255.250	01-00-5e-7f-ff-fa	static
255.255.255.255	ff-ff-ff-ff-ff-ff	static

הפקודה מציגה לנו טבלה עם שלוש עמודות:

- **באדום** - כתובת IP.
- **בכחול** - כתובת MAC המשויכת לאותה כתובת IP.
- **בירוק** - סוג הרשומה - האם היא דינמית (כלומר הושגה באמצעות פרוטוקול ARP) או סטאטית (הוכנסה באופן ידני ולא משתנה).

על מנת שהתרגיל הבא יעבד, עליו לרוקן את המטען. לשם כך, הריצו את הפקודה:

arp -d <ip_address>

לדוגמה:

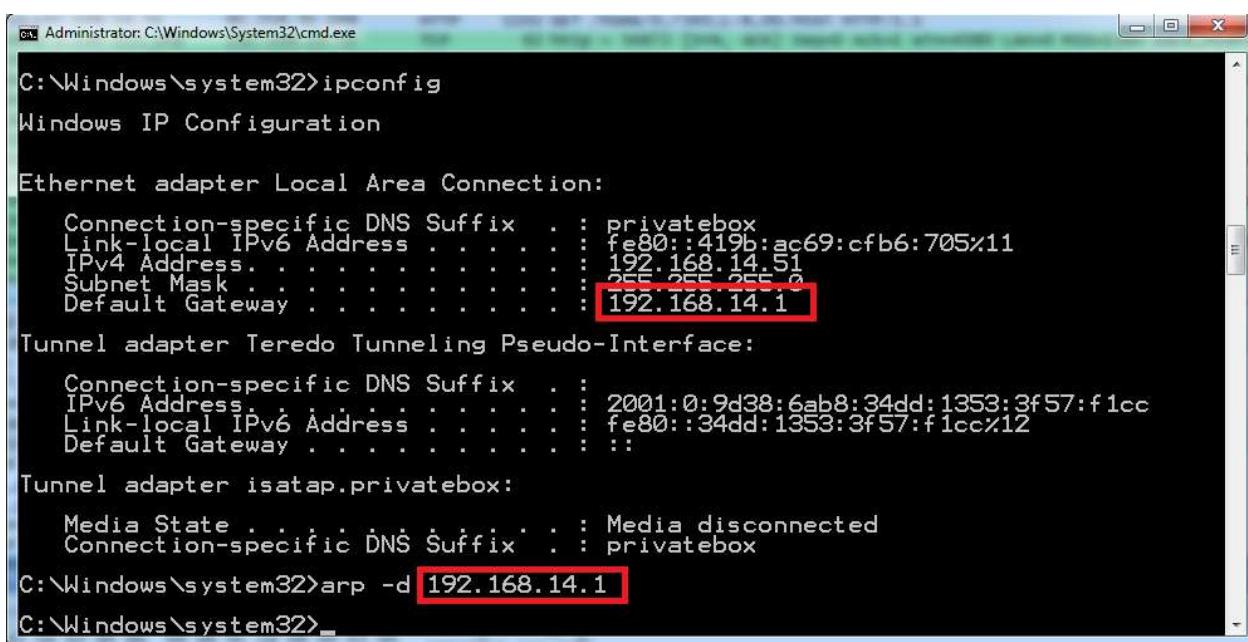
arp -d 192.168.4.1

```
Administrator: C:\Windows\System32\cmd.exe
C:\Windows\system32>arp -d 192.168.14.1
C:\Windows\system32>
```

שימוש לב שאות הפקודה יש להריץ בהרשאות גבוהות. לכן, הריצו את שורת הפקודה בהרשאות administrator.

תרגיל 8.10 מודרך - התבוננות בשאלית ARP

פיתחו את Wireshark והריצו הסנפה. כמו כן, מיחסו מה-ARP שלכם את הרשומה שקשורה לנטב שלכם (ה-default gateway). להזכירם, כדי לאגלו את כתובת ה-IP של הנטב, ניתן להשתמש בפקודה :ipconfig



```

Administrator: C:\Windows\System32\cmd.exe

C:\Windows\system32>ipconfig
Windows IP Configuration

Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix  . : privatebox
  Link-local IPv6 Address . . . . . : fe80::419b:ac69:cfb6:705%11
  IPv4 Address . . . . . : 192.168.14.51
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.14.1

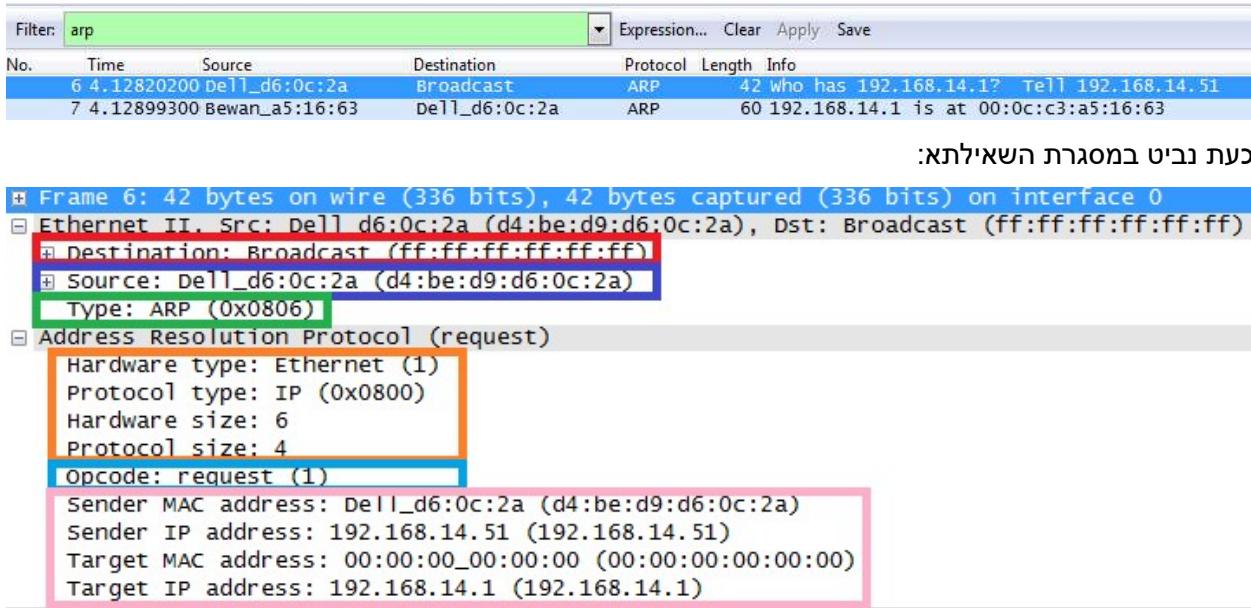
Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix  . :
  IPv6 Address . . . . . : 2001:0:9d38:6ab8:34dd:1353:3f57:f1cc
  Link-local IPv6 Address . . . . . : fe80::34dd:1353:3f57:f1cc%12
  Default Gateway . . . . . :

Tunnel adapter isatap.privatebox:
  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . : privatebox

C:\Windows\system32>arp -d 192.168.14.1
C:\Windows\system32>

```

כעת, גילשו אל <http://www.ynet.co.il>. כפי שכבר הבנו, על מנת לתקשר עם Ynet, המחשב יצרך למסת אל הנטב. תוכלו להשתמש במסנן התצוגה "arp" כדי לسانן את החבילות הרלוונטיות:



cutet.net במסגרת השאליתא:

+ Frame 6: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
 Ethernet II, Src: Dell d6:0c:2a (d4:be:d9:d6:0c:2a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 + Destination: Broadcast (ff:ff:ff:ff:ff:ff)
 + Source: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
 + Type: ARP (0x0806)
 - Address Resolution Protocol (request)
 Hardware type: Ethernet (1)
 Protocol type: IP (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: request (1)
 Sender MAC address: Dell_d6:0c:2a (d4:be:d9:d6:0c:2a)
 Sender IP address: 192.168.14.51 (192.168.14.51)
 Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
 Target IP address: 192.168.14.1 (192.168.14.1)

נתחיל מלהביט בשדות של שכבה ה-Ethernet :

- **באודיו** - כתובת היעד של המסגרת. הכתובת היא כתובת Broadcast, כלומר ff:ff:ff:ff:ff:ff.
- **בכחול** - כתובת המקור של המסגרת. זהה הכתובת של כרטיס הרשות שלהם.
- **בירוק** - סוג המסגרת. מדובר במסגרת מסוג ARP.

כעת נביט בשדות של שכבה ה-ARP :

- **בכתום** - נתונים המתארים כי המיפוי הוא מכתובת IP לכתובת MAC של Ethernet. שדות אלו נוחצים כיוון שרARP יכול למפות גם מכתובות לוגיות אחרות לכתובות פיזיות אחרות.
- **בתכלת** - קוד חבילת ה-ARP. הקוד הינו 1, והוא מצין שאילתא (Request).
- **בורוד** - השדות שקשורים לכתובות:
 - כתובת ה-MAC של היפוטזה השולחת, כלומר של המחשב שלנו ששלח את השאלה.
 - כתובת ה-IP של היפוטזה השולחת, כלומר של המחשב שלנו ששלח את השאלה.
 - כתובת ה-MAC המבוקשת. במקרה זה, הכתובת היא אפסים מכיוון שעליה אנו שואלים - איןנו יודעים מהי כתובת ה-MAC של היעד (הנתב שלנו).
 - כתובת ה-IP של היעד, כלומר כתובת ה-IP של הנתב.



תרגיל 8.11 - התבוננות בתשובה ARP

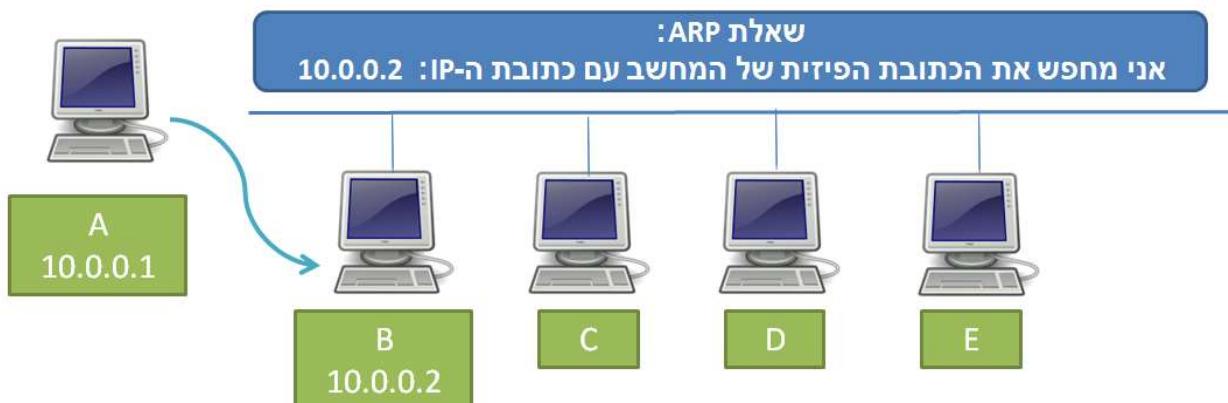
כעת, מצאו את מסגרת התשובה. ענו על השאלות הבאות:

1. בשכבת ה-Ethernet, מהי כתובת המקור של המסגרת? מי שלח אותה?
2. בשכבת ה-Ethernet, מהי כתובת היעד של המסגרת? האם היא נשלחה אל כולם (Broadcast) או רק לשוט מסוימת?
3. מהו הערך של שדה ה-eopcode בתשובה?
4. איפה במסגרת ה-ARP מופיעה התשובה לשאילתא שנשלחה, כלומר הכתובת הפיזית של הנתב?
5. התבוננו בתשובה ARP. מה היא הכתובת הפיזית של הנתב שלכם?
6. הציגו את ה-ARP Cache של מחשבכם. האם הכתובת הפיזית שמצאתם במסגרת ה-ARP הינה הכתובת הפיזית שמופיעה ב-eopcode?



למי נשלחת שאלת ה-ARP?

היות שמסגרת ARP הינה בשכבה שנייה בלבד, ולא בשכבה שלישית, היא לא מועברת להלאה על ידי נתבים. مكان שבדוגמה הבא:



אם מחשב A מעוניין לשלוח שאלה ARP למחשב B, עליו להיות בחיבור ישיר זה עם זה על מנת שהשאלה תגיע לעדשה. כדי להבין אם מחשב B מחובר אליו בחיבור ישיר, מחשב A בודק האם מחשב B נמצא אליו **באוטו-Subnet**. דבר זה אמם מזר, שכן Subnet הוא מונח של שכבת הרשות, כמו שŁmdn - כתובת IP היא כתובת לוגית שלא מלמדת בודדות על מקום פיזי של כרטיס הרשות. אף על פי כן, זו הבדיקה המתבצעת - מחשב A בודק אם כתובת ה-IP של מחשב B נמצאת אליו באוטו-Subnet. במקרה זה, הכתובת של מחשב A הינה: 10.0.0.1, הכתובת של מחשב B הינה: 10.0.0.2, ונניח כי מסכת הרשות היא: 255.0.0.0. כפי שŁmdn בפרק שכבת הרשות/מהו מזהה הישות?, משמעותה של מסכת רשת זו היא שהבית הראשוני של מזהה הרשות, ועל-כן המחשבים נמצאים באותו-Subnet. אי לכך, במקרה זה, המחשב A אכן שולח שאלה ARP עבור כתובת ה-IP של B.

אך מה היה קורה לו B לא היה נמצא באותו-Subnet כגן מחשב A? מה היה קורה לו מחשב A היה מעוניין לשולח הודעה אל Google, שכותבת ה-IP שלו הינה, לדוגמה, 3.3.3.3? במקרה זה, מכיוון שהכתובת 3.3.3.3 אינה ב-Subnet של מחשב A, החבילה צריכה להישלח אל ה-**Default Gateway** של מחשב A, אותו נתבע המיועד לטפל בחבילות היוצאות מהרשת, כפי שŁmdn בפרק שכבת הרשות/מהי טבלת הניתוב שלו?. נכון, במידה ורשומת ה-ARP ה冗余ית לא קיימת במתמונן של מחשב A, תשלוח הודעה ARP עבור כתובת ה-IP של ה-Default Gateway.

- אם כתובת היעד נמצא ב-Subnet של המחשב השולח, הרוי שניתן לשולח שאלה ARP עבור כתובת ה-IP של היעד, ואז לשולח את החבילה ישירות אל כתובת ה-MAC המוחזרת בתשובה.
- אם כתובת היעד לא נמצאת ב-Subnet של המחשב השולח, הרוי שלא ניתן לשולח שאלה ARP עבור כתובת ה-IP של היעד. במקרה זה, נשלחת שאלה ARP לגילוי כתובת ה-MAC של ה-Default Gateway, ועוד החבילה מועברת אליו להandler טיפול.



כמובן שבשני המקרים לא ישלחת שאלת ARP אם המידע הרלוונטי נמצא כבר במטמון.

שליחת מסגרות בשכבה שנייה באמצעות Scapy

כשלמדנו על Scapy, למדנו לשולח חבילות בשכבה השלישיית באמצעות הפונקציה **send**. למדנו, למשל, ליצור פקטה שמתחליה בשכבה זו, כגון פקעת ICMP Ping:

>>> my_packet = IP(dst = "www.google.com") / ICMP()

שימוש לב ש-Scapy יוצר באופן ברירת מחדל את שכבת ה-ICMP כבקשת Ping (כלומר Echo Request). ניתן לבדוק זאת:

```

C:\Windows\system32\cmd.exe - scapy
>>> my_packet = IP(dst="www.google.com")/ICMP()
>>> my_packet.show()
###[ IP ]###  

version= 4  

ihl= None  

tos= 0x0  

len= None  

id= 1  

flags= 0  

frag= 0  

ttl= 64  

proto= icmp  

chksum= None  

src= 192.168.14.51  

dst= Net('www.google.com')
\options\  

###[ ICMP ]###  

type= echo-request  

code= 0  

chksum= None  

id= 0x0  

seq= 0x0
>>>
  
```

כעת נוכל לשולח את הפקטה:

>>> send(my_packet)

Sent 1 packets.

עכשו, כשאנו יודעים יותר על רמת ה-IP, העובדה Scapy הצליחה לשלוח את החבילה אמורה לגרום לנו להרמת גבה. כיצד Scapy עשה זאת? איך הוא הצליח לשלוח פקטה בשכבה שלישית בלבד?

از ידוע לכם – אין באמת קסמיים ברשותו, Scapy הוא לא קוסם. Scapy הבין בלבד שאינו מעוניין שהוא ישלח את הפקטה מעל Ethernet, ובנה את שכבה זו בעצמו כדי לשלח את הפקטה (כשכתובות המקור היא הכתובת של כרטיס הרשת שלנו, כתובות היעד היא של הנטב, והסוג הוא IP).

אר מה אם היינו רוצים לשלח את הפקטה דווקא מכרטיס רשת מסוים? מה אם היינו רוצים שהיא תשלח מעל WiFi ולא מעל Ethernet? או מה אם היינו רוצים לשנות את השדות של שכבת-h-Ethernet באופן ספציפי?

לשם כך, Scapy מציע לנו לשלח את הפקטה בשכבה שנייה, זאת באמצעות הפונקציה **sendp**. בואו ננסה:

```
>>> my_layer2_packet = Ether()/IP(dst="www.google.com")/ICMP()
```

```
>>> sendp(my_layer2_packet)
```

.

Sent 1 packets.



```
C:\Windows\system32\cmd.exe - scapy
>>> my_layer2_packet = Ether()/IP(dst="www.google.com")/ICMP()
>>> sendp(my_layer2_packet)
Sent 1 packets.
>>>
```

שים לב לא להתבלבל בין הפונקציות **send** ו-**sendp**. שילוח פקטה שלא מכילה שכבה שנייה באמצעות **sendp** תגרום, מן הסתם, לשילוח של פקטה לא תקינה. כך גם שימוש ב-**send** לשילוח פקטה שמכילה כבר שכבת Ethernet, או כל פרוטוקול שכבה שנייה אחרת.

תרגיל 8.12 - שילחה עם שיליטה בשדות ה-Ethernet



השתמשו בכתובת ה-IP של ה-Default Gateway שלכם שמצאתם קודם לכן. השתמשו בפקודה **ping** כדי לשלחו אליו בקשה Echo Request על השדות של הבקשה. באופן צפוי, בשכבה ה-Ethernet, כתובות היעד צפויות להיות כתובות ה-MAC של הנטב.

כעת, נסו לעשות דבר אחר. שלחו בקשה ICMP Echo Request אל ה-IP של הנטב, אך בשכבה ה-Ethernet. השתמשו בכתובת היעד FF:FF:FF:FF:FF:FF. האם הנטב ענה לשאלתך? מדוע?

רכיבי רשת בשכבה הקרו ובסכבה הפיזית

از למדנו רבות על שכבת הקרו, והבנו איך היא פועלת. למדנו על פרוטוקול Ethernet, וכן על פרוטוקול ARP שמאפשר לנו למפות בין כתובות לוגיות לנכתבות פיזיות. אך איך כל הדבר הזה עובד? איך, למעשה, יישויות שונות (כמו מחשבים) מחוברות באותו שכבה שנייה? האם כל המחשבים ברשת מחוברים על אותו הכלל?

לשם כך, علينا להכיר רכיבי רשת שמחברים את היחסיות השונות זו לזו (בנהנזה שמדובר בתווך קווי ולא אלחוטי).

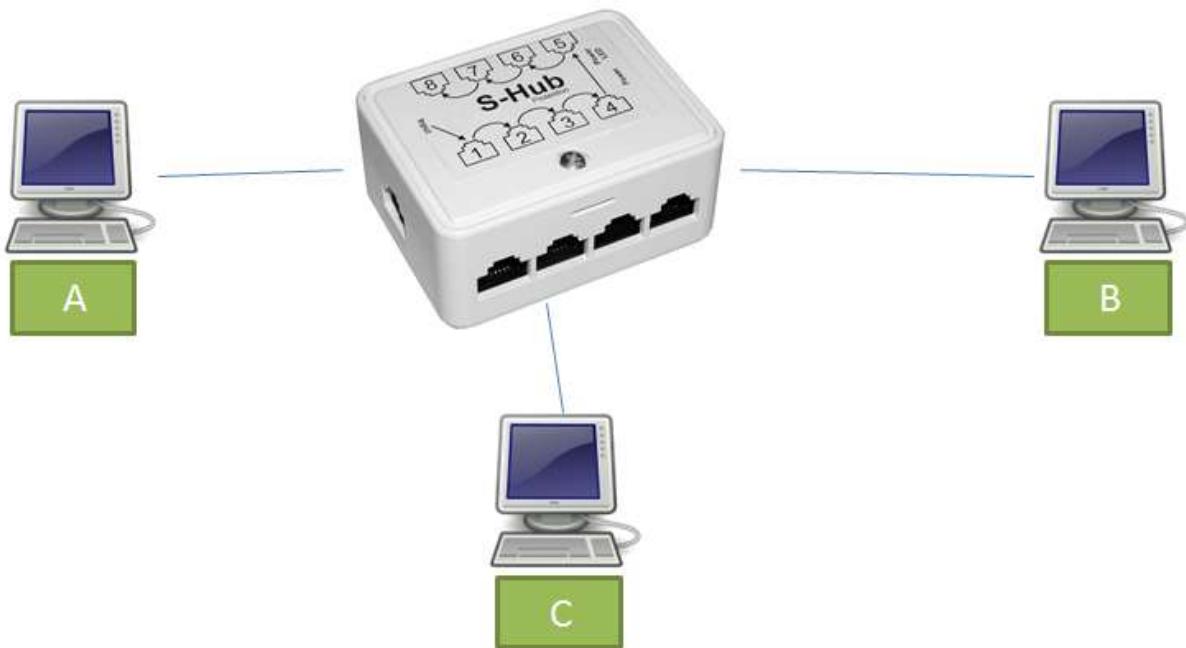
Hub (רכזת)

Hub (ובעברית - רczת) הינו למעשה רכיב של שכבה הפיזית, כלומר השכבה הראשונה במודל השכבות. ה-Hub הוא "קופסה" שאליה מתחברים מספר כבלי רשת. הוא לא יודע איך גראות מסגרות Ethernet, לא מבין מה זה כתובת MAC ולא יודע לחשב checksum. הוא רק מחבר כמה יישויות זו לזו.



כל "כניסה" ב-Hub אליה ניתן לחבר כבל רשת נקראת **פורט** (באנגלית - **Port**)⁷⁴. כאשר מחשב שנ לחבר ל-Hub שולח מסגרת, ה-Hub מעתיק את המסגרת ושולח אותה לכל הפורטים שלו, מלבד זהה שמננו המסגרת נשלחה. כך למשל, בדוגמה הבאה:

⁷⁴ על אף שהוא אותו השם, שימוש לב לא להתבלבל בין פורטים פיזיים לבין הפורטים עליהם למדנו בפרק שכבת התעבורה.



המחשבים B ו-C מחוברים זה לזה באמצעות Hub. במקרה שהמחשב A ישלח מסגרת אל B, המסגרת תגיען אל המחשב B והן אל המחשב C. אם המחשב A ישלח הודעה אל המחשב C, המסגרת גם תגיען אל המחשב B והן אל המחשב C. אם המחשב B ישלח מסגרת המיועדת אל המחשב A, היא תגיען למחשב A והן למחשב C, וכך הלאה.

במקרה שהמחשב A שלח מסגרת אל המחשב B, המסגרת תגיע כאמור הן אל המחשב B והן אל המחשב C. בשלב זה, כרטיס הרשות של כל מחשב צריך להבין האם המסגרת מיועדת אליו, על פי כתובות היעד של המסגרת בשכבה השנייה (למשל - כתובת היעד של Ethernet עלייה למדנו). אם המסגרת מיועדת אל כרטיס הרשות הרלבנטי (בדוגמה שלנו - מחשב B), הוא יಡאג לשולח את המידע למי שצריך לטפל בו (למשל - מערכת הפעלה). אם לא (בדוגמה שלנו - מחשב C), המסגרת נזרקת.

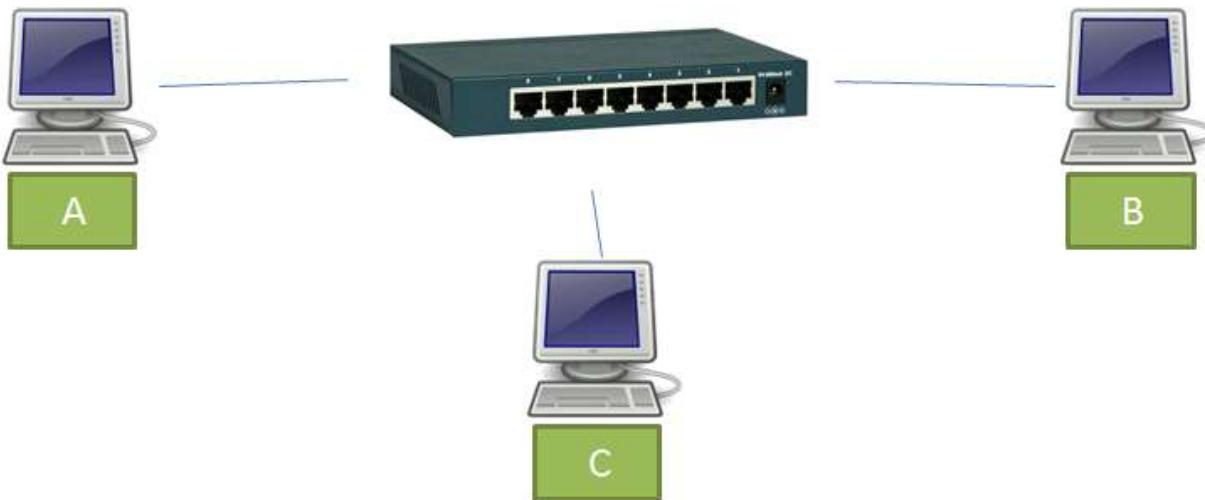
השימוש ב-Hub מאפשר אמנון לחבר מספר מחשבים זה זהה, אך יש בו בעיות רבות. ראשית, העובדה שככל המסגרות מגיעות לכל המחשבים עשויו לפגוע בפרטיות של המשתמש, שכן כרטיס רשות שלא אמור לראות את המסגרת מקבל אותה. שנית, העובדה שהמסגרות מגיעות תמיד לכל הישויות מעמיסה בצורה משמעותית על הרשות. היא מעמיסה הן על החיבורם (שבהם יכולה להשליח מסגרת אחת בלבד בכל פעם), והן על כרטיסי הרשות של כל ישות שצרכיהם לטפל בכל מסגרת. שלישיית, השימוש ב-Hub לא מונע התנגשויות, בעיה עלייה נלמד בהמשך הפרק.

מכל סיבות אלו, השימוש ב-Hub אינו מספיק טוב. על מנת להתגבר עליה, נמצא ה-Switch.

Switch (מתק)

בניגוד ל-Hub, עליו למדנו קודם, ה-Switch (בעברית - מתק) הוא כבר רכיב שכבה שנייה לכל דבר. ה-Switch מכיר פרוטוקולים של שכבת הקו (לדוגמה - פרוטוקול Ethernet) ומכיר כתובות MAC. חלק מה-Switchים גם יודעים לחשב checksum ו"לזרוק" מסגרות עם שגוי.

בחינה חיצונית, Switch וה-Hub הם דומים: שניהם נראים כמו קופסה עם כמה פורטים אליו ניתן לחבר כבל רשת. עם זאת, הפקנציונליות שלהם שונה מאוד. לאחר שה-Switch למד את הרשת, הוא מעביר מסגרת מהפורט בה הוא קיבל אותה אל הפורט הרלוונטי בלבד. בדוגמה הבאה:



המחשבים B ו-C מחוברים ל-Switch. אם המחשב A שלח מסגרת למחשב B, המסגרת תגיע אל המחשב B בלבד - ולא תגיע למחשב C או תוחזר אל המחשב A. באותו אופן, אם המחשב B שלח מסגרת למחשב C, המסגרת תגיע אליו בלבד ולא תגיע אל המחשב A (וכМОון לא תוחזר אל המחשב B).

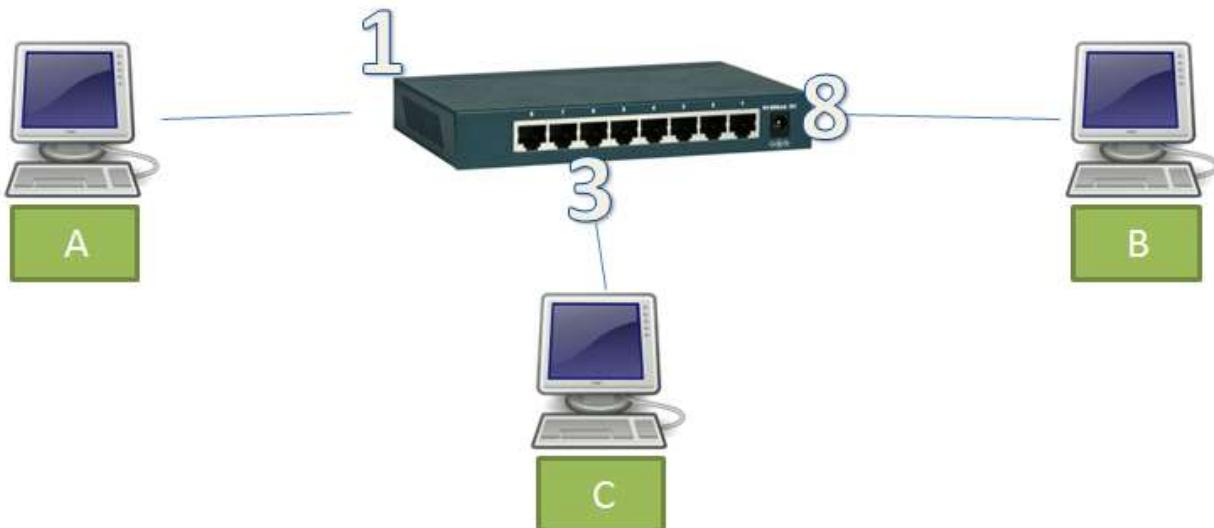
אי לכך, ל-Switch יש יתרונות רבים על פני ה-Hub. היה שכל מסגרת מגיעה רק אל היעד שלה, אין פגיעה בפרטיות המשתמשים. בנוסף, הוא חוסך את העומס הרב שהוא נוצר לו היו משתמשים ב-Hub. כמו כן, מסיע במניעה של התנגשויות, עליו נלמד בהמשך.



كيفية عمل Switch؟

הבנו שה-Switch יודע להעביר כל מסגרת אל הפורט המיועד אליה בלבד. איך כיצד הוא עושים זאת? איך הוא יודע היכן כל ישות רשת נמצאת?

ובכן, ל-Switch יש טבלה שעלייה למלא בזמן ריצה. הטבלה תמפה בין כתובת MAC לבין הפורט הפיזי הRELVENTI. לכל פорт פיזי יש מספר המאפשר לזהות אותו. לדוגמה, ב-Switch של הרשות שהציגו לעיל, יש שמונה פורטים, שממוספרים מפורט 1 ועד פорт 8. אם נביט שוב בדוגמה לעיל, אך הפעם נסתכל גם על מספרי הפורטים:



נראה שפורט מס' 1 מקשר למחשב A, פорт מס' 3 מקשר למחשב C, ופורט מס' 8 מקשר למחשב B. אי' לך, על ה-Switch לבנות אצלו טבלה שתיראה בסופו של דבר כך:

MAC Address	Port
A	1
C	3
B	8

כמובן שה-Switch לא יודע שלמחשב קוראים A, אלא הוא שומר את כתובת ה-MAC של כרטיס הרשות שלו (לדוגמה: D4:BE:D9:D6:0C:2A). לצורך נוחות הקראיה, נשאיר בטבלה את שם המחשב ולא את כתובת ה-MAC של כרטיס הרשות.

נאמר וה-Switch הינו Switch חדש ברשות, כלומר הוא עדין לא הספיק להכיר אותה. בשלב זה, הטבלה שלו תהיה ריקה:

MAC Address	Port

cut, המחשב A שולח מסגרת אל מחשב B. מה על ה-Switch לעשות? הרי הוא לא יודע לשير את כתובת ה-MAC של כרטיס הרשת של המחשב B לשום פורט פיזי. במקורה זה, מכיוון שה-Switch אינו יודע למי להעביר את המסגרת, הוא מתנהג בדומה ל-Hub ועביר אותה לכל ה포רטים מלבד זהה אליו היא נשלחה. למשל, בדוגמה הזאת, הוא יעביר את המסגרת אל מחשב B ואל מחשב C, אך לא חזרה אל מחשב A.

אך בנוספ', ה-Switch למד משהו. הוא ראה את המסגרת שהגיעה ממחשב A מגיעה מ포רט מס' 1. מעבר לכך, הוא יכול לקרוא את המסגרת, ולראות את כתובת ה-MAC שמצוינת בכתובת המקור של החבילה. בשלב זה, ה-Switch למד שככתובת ה-MAC של כרטיס הרשת של מחשב A מחוברת אל פורט 1. cut, הוא יכול לציין זאת בטבלה שלו:

MAC Address	Port
A	1

נאמר ועכשו מחשב A שוב שולח מסגרת אל מחשב B. חשבו על כך - האם הפעם תהיה התנהגות שונה? התשובה היא - לא. ה-Switch אמונם יודע איפה נמצאת כתובת ה-MAC של כרטיס הרשת של מחשב A, אך הוא אינו יודע איפה נמצאת כתובת של כרטיס הרשת של מחשב B. אי לכך, הוא נאלץ שוב לשלוח את המסגרת לכל ה포רטים מלבד לפורט המקור, למשל **למחשב B ולמחשב C**.

לאחר זמן מה, מחשב B שולח מסגרת אל מחשב A. הפעם, התהיליך הוא שונה. ה-Switch מסתכל בטבלה, וראה שהוא מכיר את כתובת ה-MAC אליה המסגרת נשלחת, והיא מקושרת לפורט מס' 1. אי לכך, ה-Switch מעביר את המסגרת רק אל פורט 1, ולא אף פורט אחר. בנוספ' על כן, הוא מסתכל במסגרת, וראה שבשדה

כתובת המקור נמצאת כתובת MAC של כרטיס הרשת של מחשב B. היות שהמsegרת נשלחה מפורט מס' 8, הוא יכול להוציא את מידע זה לטבלה:

MAC Address	Port
A	1
B	8

במצב זה, כל מסגרת שתשלוח אל מחשב A (בין אם מחשב C ובין אם מחשב B) תגיע אל פорт מס' 1 ופורט זה בלבד. כל מסגרת שתשלוח אל המחשב B (בין אם מחשב A ובין אם מחשב C) תגיע אל פорт מס' 8 ופורט זה בלבד. עם זאת, מסגרות שתשלוחה אל מחשב C, יעברו לכל ה포רטים בלבד לזה שמננו הן נשלחו, שכן ה-Switch עדין לא מכיר את כתובת MAC הרלוונטית. מצב זה ישנה כאשר מחשב C ישלח מסגרת, ואז ה-Switch יוכל ללמוד על הכתובת של כרטיס הרשת שלו, ולהשלים את הטבלה:

MAC Address	Port
A	1
C	3
B	8

 **שים לב** - המחשבים לא מודעים לכך שהם מחוברים ל-Hub, ל-Switch או לכל רcieב אחר בשכבה ה-2. במקרה ל-Router, עליו למדנו בפרק [שכבות הרשת / נתב \(Router\)](#), שהמחשב צריך להכיר בכך להצליח להפנות אליו חבילות, המחשב מחובר ישירות ל-Hub או ל-Switch ולא מודע לכך.

תרגיל 8.13 - פועלות Hub



הבטו בשרטוט הרשות שלפניכם:



כאן שלושה מחשבים ושרת (Server) מחוברים זה לזה באמצעות hub. מחשב A מחובר לפורט מס' 1, מחשב C מחובר לפורט מס' 2, השרת מחובר לפורט מס' 3 ומחשב B מחובר לפורט מס' 4. הניחו שה-hub הינו Hub חדש ברשות. כמו כן, כל שאלת מסתמכת על השאלות הקודמות (בשאלה מס' 3 ניתן להניח שהמסגרת משאלה 2 כבר נשלחה).

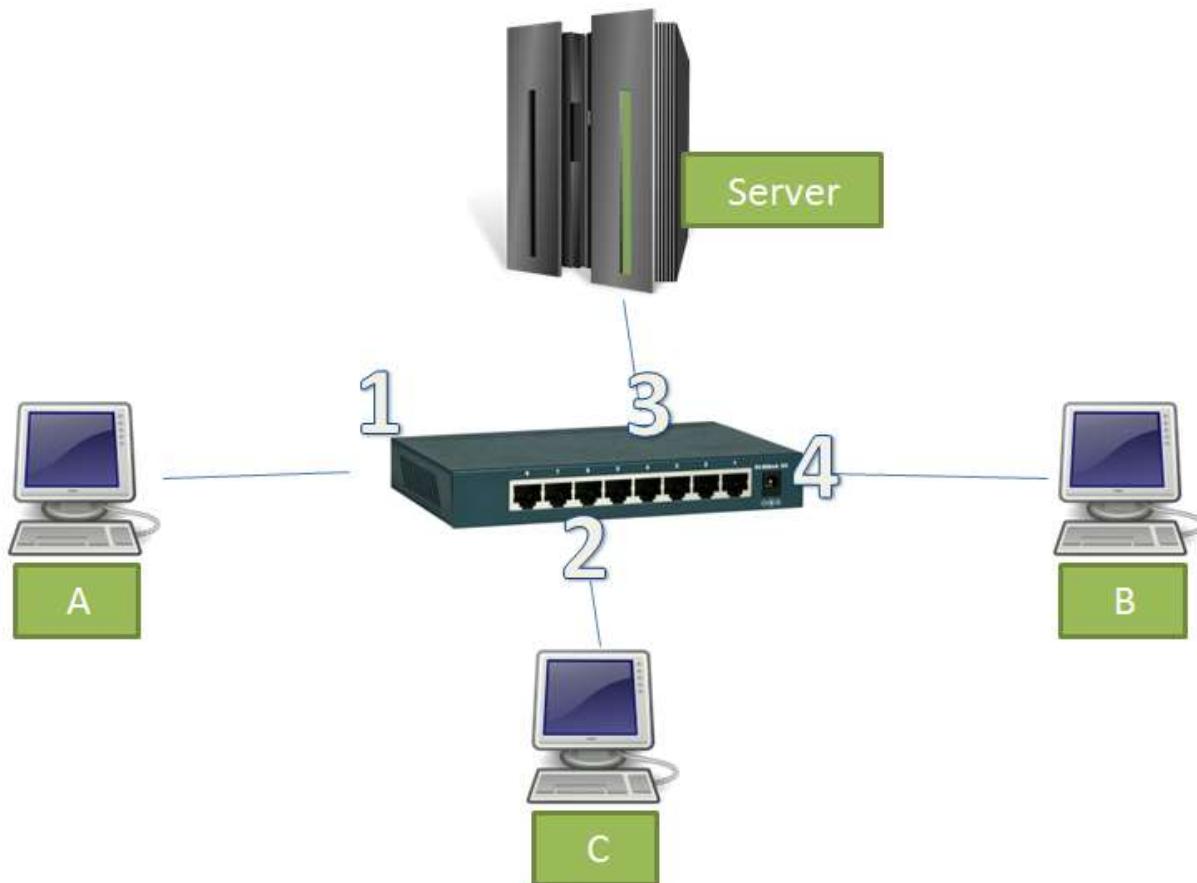
ענו על השאלות הבאות:

1. המחשב A שולח מסגרת אל המחשב B. לאילו פורטים ישלח ה-hub את המסגרת?
2. המחשב B שולח מסגרת אל השרת. לאילו פורטים ישלח ה-hub את המסגרת?
3. המחשב C שולח מסגרת אל כולם (Broadcast מסגרת). לאילו פורטים ישלח ה-hub את המסגרת?
4. השרת שולח מסגרת אל המחשב A. לאילו פורטים ישלח ה-hub את המסגרת?

תרגיל 8.14 - פועלות Switch



לפניכם שרטוט רשת זהה לשורתו שהוזג בתרגיל הקודם, אך הפעם - הישיות השונות מחוברות באמצעות :Hub, Switch



מחשב A מחובר לפורט מס' 1 של ה-Switch, מחשב C מחובר לפורט מס' 2, השרת מחובר לפורט מס' 3
ומחשב B מחובר לפורט מס' 4.

הניחו שה-Switch הינו חדש בראשת. כמו כן, כל שאלה מסתמכת על השאלות הקודמות (בשאלה מס' 3 ניתן להניח שהמוגדרת שאלה 2 כבר נשלחה).

כעת, ענו על השאלות הבאות:

1. המחשב A שולח מסגרת אל המחשב B. לאילו פורטים ישלח ה-Switch את המסגרת?
2. המחשב B שולח מסגרת אל השרת. לאילו פורטים ישלח ה-Switch את המסגרת?
3. המחשב C שולח מסגרת אל כולם (מסגרת Broadcast). לאילו פורטים ישלח ה-Switch את המסגרת?
4. השרת שולח מסגרת אל המחשב A. לאילו פורטים ישלח ה-Switch את המסגרת?

שכבה ה-2 - סיכום

בפרק זה הכרנו את השכבה השנייה במודל חמש השכבות, שכבת ה-2. בתחילת הפרק למדנו על תפקידה של השכבה וכייזד היא משתמשת במודל השכבות. לאחר מכן, הכרנו את **פרוטוקול Ethernet**. למדנו על כתובות **MAC**, איך הן בנויות, וכייזד נמצא כתובת MAC של כרטיס הרשת שלנו. למדנו איך נראת מסגרת Ethernet, והתבוננו בשדות של פרוטוקול זה בעת ביצוע פניה HTTP.

לאחר מכן למדנו על **פרוטוקול ARP**, הבנו את הצורך בו וכן את דרך הפעולה שלו. בהמשך למדנו כיצד ניתן להשתמש ב-Scapy כדי לשולח מסגרות בשכבה שנייה ולהשפיע על שדות בשכבה זו, וכן תרגלונו נושא זה. לאחר מכן למדנו על רכיבי רשת - הכרנו **Hub**, שהוא רכיב של השכבה הראשונה, ו-**Switch**, שהוא רכיב של השכבה השנייה, ולמדנו על ההבדלים ביניהם. בסוף, הכרנו את סוגיות ההתגשויות והתפקיד של שכבת ה-2 בהתקומות עם סוגיה זו.

בפרק הבא, נלמד על השכבה הפיזית ובכך נסימן את הカリוטינו עם מודל השכבות. כמו כן, נמשיך ללמידה על נושאים מתקדמים בתחום רשתות המחשבים.

נספח א' - התנגשויות

עד כה הסבכנו את מטרתה של שכבה ה₀, הכרנו פרוטוקול לדוגמה של שכבה זו וראינו רכיבי רשות המאפשרים לחבר ישוות שונות. עם זאת, לא הتمודדנו עדין עם בעיה שעל שכבה ה₀ לטפל בה - בעיית ההתנגשויות.

כדי להסביר את הבעיה, נדמיין מקרה המוכר לנו שאנו קשור לעולם המחשבים. נאמר לנו נמצאים בעיצומו של דיון סוער של מועצת הביטחון של האו"ם. בדיון ישנים נציגים מחמש עשרה המדינות החברות במועצת, וכל אחד מהנציגים מעוניין להביע את עמדתו. במקרה שככל הנציגים ידברו במקביל, כלומר באותו הזמן ממש, אף אחד לא יוכל להבין את דברי הנציגים האחרים. אי לכך, יש למצאו שיטה שתבטיח שרק אדם אחד יוכל לדבר בכל זמן נתון, על מנת שכולם יצליחו להבין אותו.

סוגיה זו קיימת גם ברשותות מחשבים הפעולות מעל ערוץ משותף. במקרה זה, ישנים מספר משתתפים (ישויות רשות) שרצו לשתף בו זמנית על אותו הערוץ. במידה שכמה ישויות ישדרו יחד, תיווצר התנגשות (**Collision**), והמידע לא יגיע בצורה מסודרת.

ערוץ משותף - הגדרה

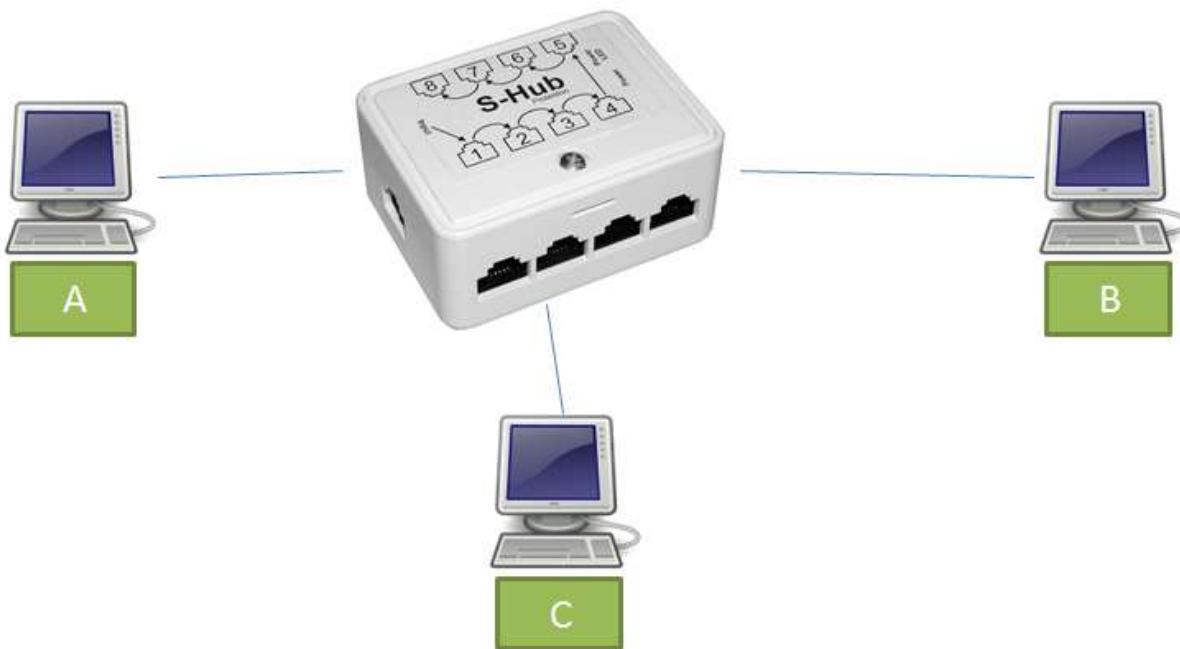
ערוץ משותף הוא קו תקשורת המחבר בין מספר ישויות ומאפשר להן להחליף מידע. המידע בערוץ מופץ ב-*Broadcast* - ככלומר, כאשר ישות משדרת, המידע מגיע לכל הישויות. בנוסף, כל ישות יכולה להאזין לערוץ בזמן שהוא משדרת - וכך לגנות האם המידע שהוא שידרה הועבר בהצלחה.

דוגמה לרשות המחברת בערוץ משותף היא רשות בה הישויות השונות מחוברות ב-*bHN*. כפי שכבר קודם לנו, במידה שברשת ישנים מספר מחשבים מחוברים ב-*bHN*, כל מסגרת שתשלוח תגיע לכל המחשבים.



מהי התנגשות?

כפי שציינו קודם לכן, כאשר שתי יישויות (או יותר) משדרות בערוץ המשותף בו בזמןית, נוצר מצב של **התנגשות (Collision)**. במקרה זה, המידע שנשלח יגיע באופן משובש - כלומר, המידע שיגיע אל הוא לא המידע שהישות התוכונה לשולח. בדוגמה הבאה:



אם המחשבים A ו-B ישדרו מסגרת באותו הזמן, עלולה להיווצר התנגשות. אם נראה למשל את הקישור בין המחשב C לבין ה-hub, נראה שבאותו הזמן אמורה להישלח עליו המסגרת שהגעה ממחשב A, כמו גם המסגרת שהגעה ממחשב B⁷⁵. במקרה זה תהיה התנגשות, והמידע שיגיע למחשב C יהיה משובש - הוא לא יהיה זהה למידע שנשלח ממחשב A ולא זהה שנשלח ממחשב B. במקרה זה, על מנת להצליח להעביר את המידע שמחשבים A ו-B רצוי לשולח, יש לשולח את המסגרות מחדש.

זמן שבו מתרחשת התנגשות נח呼 ל"זמן מת" - מכיוון שלא עבר בקוו שום מידע ממשועוט, ויש לשדר מחדש כל מסגרת שתשלוח - אין סיבה להשתמש עוד בערוץ המשותף ולשלוח עוד מסגרות. ברור למד' כי זהה תופעה שלילית, שכן אנו מבזבזים זמן בו לא נשלח שום מידע על הערוץ.

⁷⁵ ישנם Hubים חכמים שיודעים להמנע מקרים כאלה ובכך למנוע התנגשיות, אך בפרק זה נתעלם מקרים אלו.



מה תפקידה של שכבת הקו בNetworking להتنגשיות?

על שכבת הקו להגיעה לנצלות מירבית של הקו - כלומר, לצמצם את הזמן המתאים עד כמה שניתן.

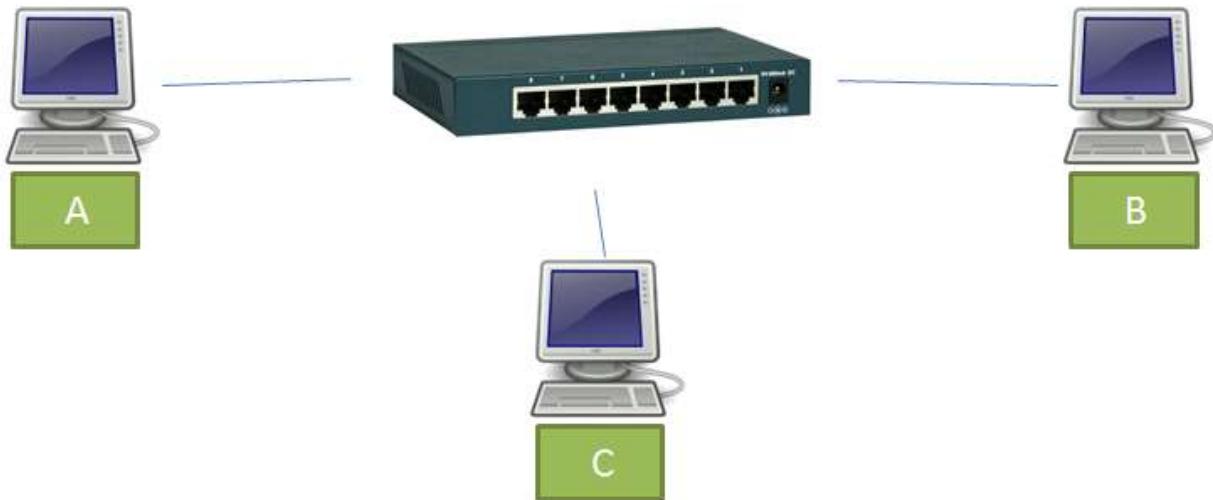
מניעת התנגשיות

ישנן דרכים רבות למנוע התנגשיות, והשיטות התקדמות עם הזמן. אחד הפרוטוקולים הראשונים שניסו להתמודד עם סוגיה זו נקרא ALOHA. לפי פרוטוקול זה, כאשר ישוט מטיימית רוצה לשדר מסגרת, היא מתחילה לשדר אותה מיד. בזמן שהתחנה משדרת, היא מזינה לערוץ ובודקת האם השידור הועבר באופן תקין. בזמן שידור המסגרת, היא בודקת האם המידע שהוא קלטה מהערוץ זהה למידע שאotta היא שידרה. אם המידע זהה - הכל בסדר. אך אם המידע שונה - הייתה התנגשות.

במקרה של התנגשויות, הישות ממתינה פרק זמן אקראי, ולאחריו מנסה לשדר את המסגרת שוב. חשוב שהישות תמתין פרק זמן אקראי, שכן אחרת גם הישות השנייה שניסתה לשדר וגרמה להתנגשות, הייתה מחייבת אותו זמן כמוותה, והייתה נוצרת התנגשויות נוספות. חשבו למשל על חמישה אנשים הנמצאים בחדר חסר לחילוטין. עליהם לנסות ולדבר, אך לא להתחיל לדבר יחד. אם שני אנשים (למשל: נגה ואופיר) מתחילהם לדבר באותו הזמן ממש, מתרחשת התנגשות. באם לאחר ההתנגשות, נגה ואופיר יჩכו חמש דקות בדיק, ויתחילו לדבר מחדש - תיווצר שוב התנגשות. לכן, כל אחד מהם ימתין זמן רנדומלי. למשל, נגה תמתין דקה ואז תנסה לדבר, בעוד אופיר ימתין שלוש דקות בטרם יתחל להשמיע את קולו. כך, השניים צפויים להצליח להעביר את המסר שלהם מבלי שתיווצר התנגשויות.

זהו דוגמה אחת בלבד לניסיון להתמודד עם התנגשיות על עroz משותף. בפרק זה לא נסקור דרכים נוספות, אך קוראים סקרנים מוזמנים להרחב את הידע שלהם [בסעיף צעדים להמשך של פרק זה](#).

במקום להתמודד עם התנגשיות כשהן מתרחשות, ניתן גם למנוע מראש מההתנגשיות. דרך משמעותית מאוד לעשות זאת ברשתות Ethernet נוצרה כאשר הומצא ה-Switch. בדוגמה הבאה:



אם מחשב A משדר מסגרת למחשב C, וגם מחשב B משדר מסגרת למחשב C, ה-Switch יודע לשלווה את המסגרת רק כאשר הערוצ פנוי. כלומר, הוא ישלח קודם את אחת המסגרות (למשל - זו שלח המחשב A), ורק לאחר מכן את המסגרת השנייה (למשל - זו שלח המחשב B). כך, ה-Switch מצליח למנוע התנגשויות מראש, **בלא ניהול מורכב**.

הפתרונות של שימוש ב-Switch אפשרי במקרים מסוימים (כמו רשתות Ethernet), אך לא תמיד. למשל, ברשות WiFi בה החיבור עבר באוויר, כל המסגרות מגיעות לכל הישויות שנמצאות באותו הקיליטה. במקרים כאלו, יש להסתמך בפתרונות אחרים.

שכבה הקרויה - צעדים להמשך

על אף שלמדנו רבות על שכבה הקרויה, נותרו לנו שאים רבים בהם לא העמכו. מניעת התנגשויות מהוות נושא מרתק, ובפרק זה נגענו רק בקצת המצלג במשמעות שלו ובדריכים שונות להשיג אותו. כמו כן, הتمקדמו בפרוטוקול Ethernet ומעט לא הזכרנו מימושים נוספים, כגון WiFi או Bluetooth. לא שאלמו את עצמנו כיצד השכבה השנייה מצליחה לבצע מסגור - הפרדה של רצף המידע למסגרות שונות, כיצד היא יודעת מתי מסגרת התחלת ומתי היא נגמרה. בנוסף, לא הסבכנו על המושג Virtual LANs.

אלו מכם שמעוניינים להעמק את הידע שלהם בשכבה הקרויה, מוזמנים לבצע את הצעדים הבאים:

קריאה נוספת

בספר המצוין Computer Networks (מהדורה חמישית) מאת Andrew S. Tanenbaum ו- J. Wetherall, הפרקים השלישי והרביעי מתיחסים במלואם לשכבה הקרויה. באופן ספציפי, מומלץ לקרוא את החלקים:

- 3.1.2 - מסגור.
- 4.2 - התמודדות עם התנגשויות בערוץ משותף.
- 4.3.2 - מבנה מסגרת Ethernet. באופן ספציפי, בפרק זה לא הרחבנו על המקירה בו שדה Type-Field מעיד על אורך המסגרת. כמו כן, לאذكرנו מודיען אורך המסגרת חייב להיות 64 בתים או יותר. התשובות לשאלות אלו נמצאות כאן.
- 4.4.1, 4.4.3, 4.4.4 - רשותות אלחותיות.
- Bluetooth - 4.6
- Virtual LANs - 4.8.5

בספר Computer Networking: A Top-Down Approach (מהדורה ששית) מאת James F. Kurose, פרק החמישי מוקדש כולו לשכבה הקרויה. כמו כן, הפרק החישי מוקדש לרשותות אלחותיות ולולריות. באופן ספציפי, מומלץ לקרוא את החלקים:

- 5.3 - פרוטוקולים בגישה לערוץ משותף.
- Virtual LANs - 5.4.4
- 6.3 - רשותות אלחותיות.

תרגיל 8.15 - זיהוי מרוחק של מחשב מסניף (אתגר)

בפרק [Wireshark ומודל השכבות](#) [/Capture Options](#), למדנו על האפשרות של הסנפה ב- Promiscuous Mode. אז, הסבירנו שהמשמעות של אפשרות זו היא להכנייס את כרטיס הרשת ל"מצב פרוץ", מה שיגרום לכך שנראה בהסנפה את כל המסגרות שראהו כרטיס הרשת, גם כאלה שלא מיועדות אליו. בעת אנו מסוגלים להבין את משמעות ה-Promiscuous Mode בצורה טוביה יותר. במצב זה, נראה בהסנפה גם מסגרות שהגיעו אל כרטיס הרשת שלנו מבלי שכותבת ה-MAC בשדה כתובת היעד של המסגרת תהיה הכתובת של כרטיס הרשת שלנו, או כתובת של קבוצה בה הוא חבר⁷⁶ (למשל מסגרת המיועדת ל-Broadcast, כלומר לכל הישויות ברשת). תיכון שמסגרות כאלה יגיעו אל כרטיס הרשת שלנו, למשל, מכיוון שברשת שלנו המחשבים מחוברים באמצעות Hub. תיכון גם שנראה מסגרות כאלה מכיוון שה-Switch עדין לא למד להכיר את הכתובות השונות.

האם נוכל לזהות מרוחק מחשבים ברשת שכרגע מסניפים? המסמך הבא: <http://goo.gl/2LZwqP> מຕאר שיטה לזיהוי מרוחק של כרטיסי רשת במצב Promiscuous Mode. מכיוון שבדרך כלל מי מסניף אכן משתמש באפשרות זו, נוכל להשתמש בשיטה המתוארת כדי לזהות מחשבים מסניפים ברשת.

קראו את המאמר, והבינו את השיטה המוצעת לזיהוי כרטיסי רשת שנמצאים ב-Mode Promiscuous. לאחר מכן, כתבו סקריפט באמצעות Scapy שմMESS את השיטה זו, ומדפיס את כתובת ה-MAC של כל כרטיס רשת שהוא כcartis שנמצא במצב Promiscuous Mode. הריצו את הסקריפט ברשת שלכם מבלי שאף מחשב מסניף, וודאו כי הסקריפט לא מתריע על כך שיש מחשבים מסניפים. לאחר מכן, הפעילו הסנפה באחד המחשבים ברשת והריצו את הסקריפט בשנית. וודאו כי הפעם הסקריפט מתריע על המחשב המסניף.

⁷⁶ על המשמעות של כתובות Ethernet של קבוצות, תוכלו לקרוא [בנוסף א' של פרק זה](#).

נספח ב' - כתובות Ethernet של קבוצות

כתובת Ethernet יכולה להיות שיכת לישות אחת (Unicast) או למספר ישויות (Multicast). כתובות השיכות למספר ישויות מתארות למעשה כתובת של קבוצה - למשל קבוצת כל הנטבים בראשת, או קבוצת כל הישויות שmariesות תוכנה מסוימת. כאשר כרטיס רשות Ethernet מסתכל על מסגרת Ethernet, הדבר הראשון שהוא רואה הוא כתובת היעד של המסגרת. אם כתובת היעד שיכת אליו - כרטיס הרשות "מעלה" את המידע בחבילה להמשך טיפול (לדוגמה, אם מדובר מחbillת מסוג IP) - הוא מעלה את המידע למי שמתפל בחבילות IP, למשל מערכת הרפעלה). ישנו שני מקרים בהם כתובת היעד שיכת לכרטיס הרשות:

- הכתובת היא של כרטיס הרשות עצמו. כלומר, כתובת Unicast.
- הכתובת היא של קבוצה אליה כרטיס הרשות שייך. כלומר, כתובת Multicast.

על מנת לדעת האם כתובת מסוימת היא כתובת Multicast או Unicast, علينا להסתכל על בית (bit) מסויים. הבית זהה נמצא בבייט (byte) העליון של הכתובת. לדוגמה, נסתכל בכתובת הבאה:

02:03:04:05:06:07

כעת, נסתכל רק על הבית העליון, כלומר 02. נimir את הבית הזה לפורמט הבינארי (כלומר, בתצוגת ביטים) שלו⁷⁷:

00000010

כעת, علينا להסתכל על הבית התיכון ביותר של הכתובת (מסומן באדום):

00000010

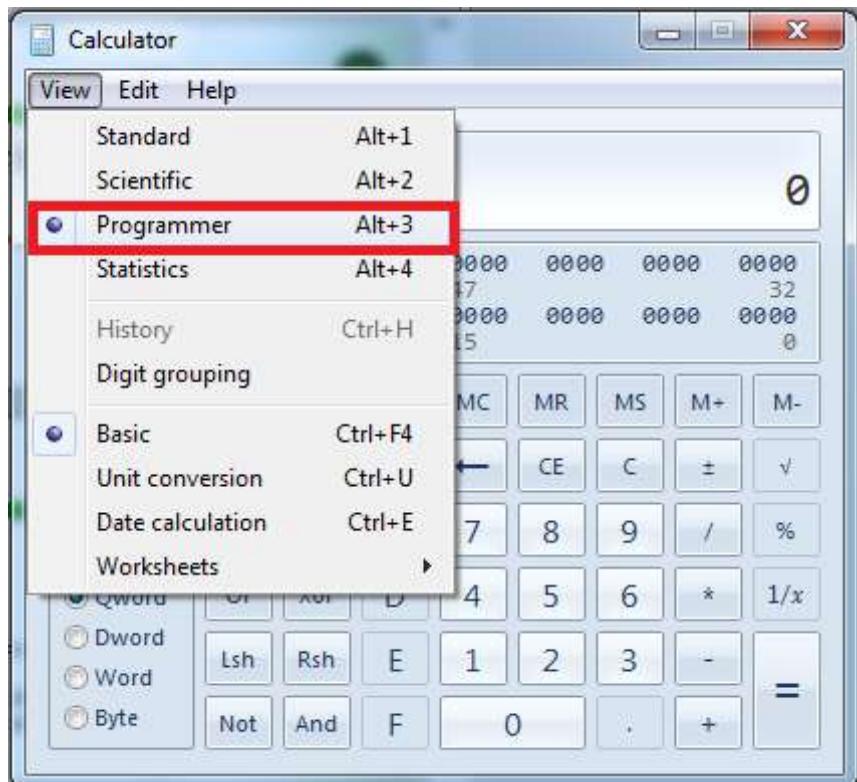
מכיון שבית זה קבוע (הערך שלו הוא 0), מדובר בכתובת Unicast.

נסתכל על כתובת כרטיס הרשות שלנו שהציגנו קודם לכן:

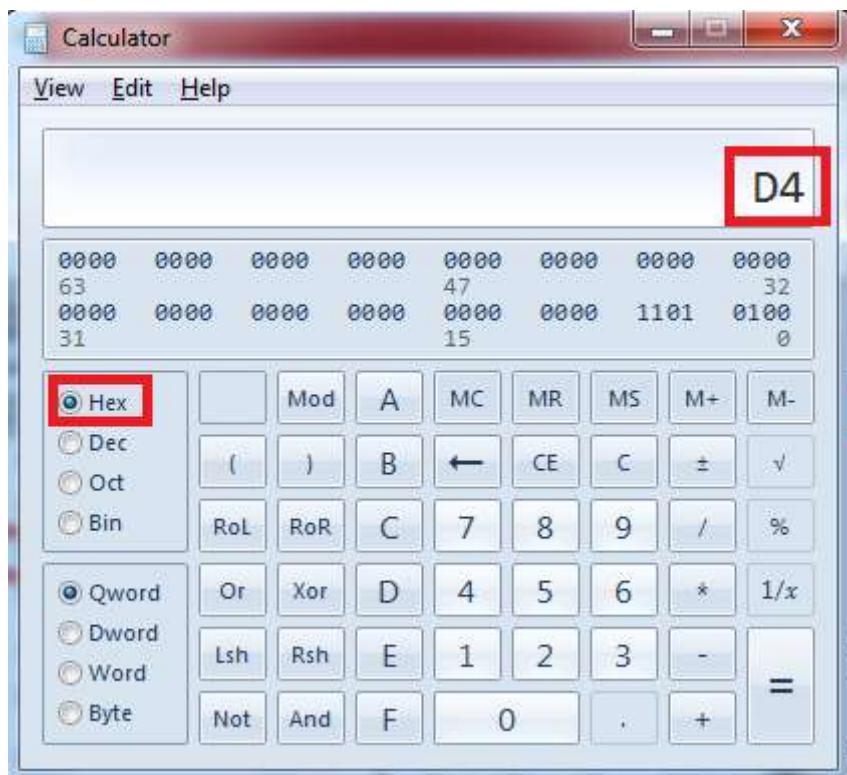
D4:BE:D9:D6:0C:2A

על מנת להבין האם כתובת זו היא Unicast או Multicast, נסתכל בbijt העליון ביותר, שהוא bijt 4. כעת, על מנת להמיר אותו לפורמט בינארי, ניעזר במחשבון של Windows. היכנסו למחשבון, ובתפरיט בחרו באפשרות View->Programmer

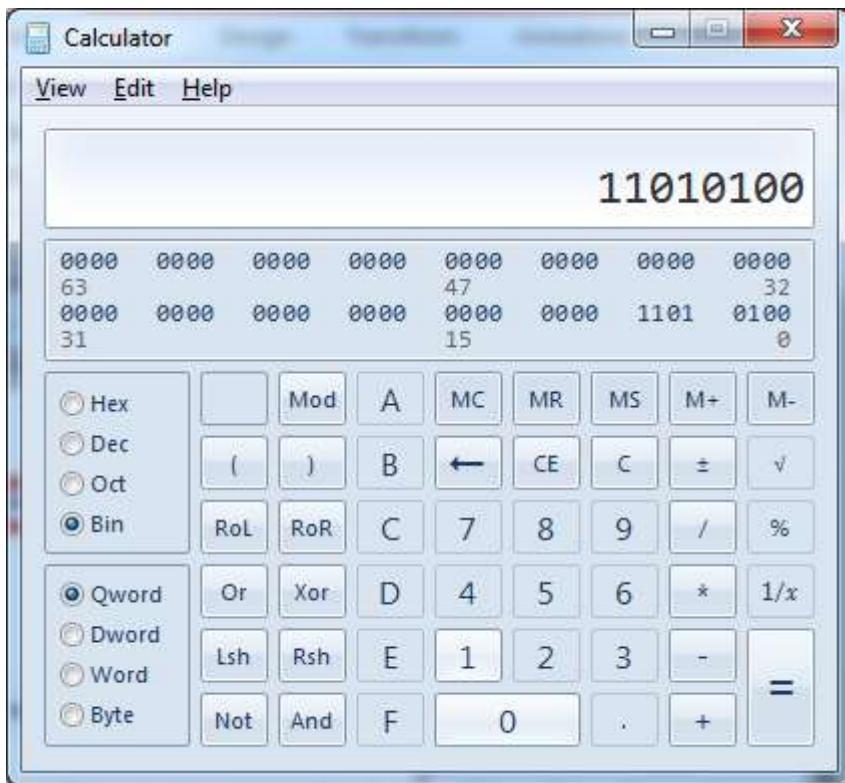
⁷⁷ אם איןכם מרגשים עדים בטוחים במונחים "bijt" ו-"bijt" או בהמרו בין הפורמטים השונים - אל תדאגו, הביטחון נרכש עם הזמן. עם זאת, קרואו לאט וודאו כי אתם מבינים את הכוונה בדוגמאות שניתנות לפניכם.



כעת, בחרו בפורמט הקסה-דצימלי (Hex), והקישו את הבית הרבוני - D4:



כעת, בחרו בפורמט בינארי (Bin). המחשבון יעשה עבורכם את המראה:



אם כן, הערך הבינארי של הבית הוא:

11010100

הבית התיכון (מסומן באדום) כבוי, ומכאן שהכתובת הינה כתובות **Unicast**. הדבר הגיוני, מכיוון שמדובר בכתובת של כרטיס רשת, וכותבת זו היא תמיד מסווג **Unicast**.

בצעו את התהליך הזה גם על כתובת כרטיס הרשות שלכם, וודאו כי הכתובת היא מסווג **Unicast.**



כעת נבחן כתובת **Ethernet** נוספת:

03:04:05:06:07:08

על מנת להבין אם הכתובת היא **Unicast** או **Multicast**, נסתכל בביית העליון - 03. נבצע המרה לבסיס בינארי:
00000011

הבית התיכון (מסומן באדום) דולק - ולכן מדובר בכתובת של קבוצה, כלומר כתובת **Multicast**.

כתובת **Multicast** נוספת הינה הכתובת **FF:FF:FF:FF:FF:FF**. כתובות זו היא כתובת **Broadcast** - כלומר הקבוצה אליה שייכות כל הيسויות ברשთ. שליחת מסגרת עם כתובת היעד **FF:FF:FF:FF:FF:FF** משמעותה שליחת המסגרת לכל היסויות שנמצאות איתנו ברשთ.



תרגיל 8.16 - זיהוי כתובות Multicast

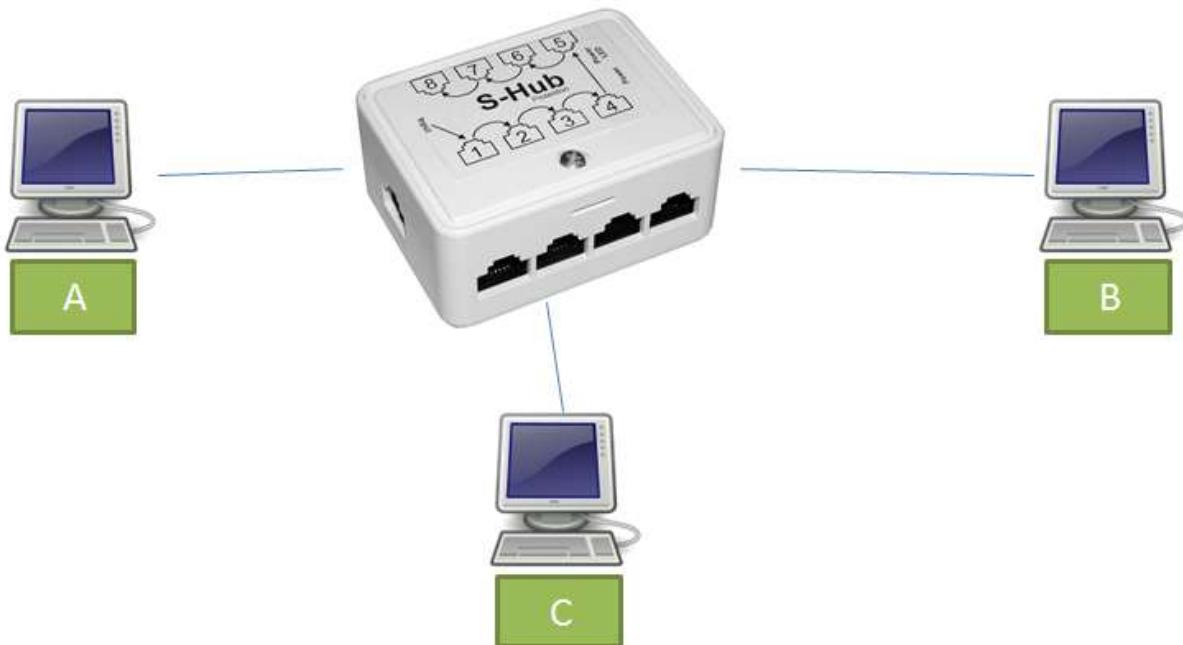
השתמשו בסקריפט שכתבתם בתרגיל 8.6 - כתובות Ethernet, אשר מבקש מהמשתמש כתובת MAC וSSID עליה מידע. ערכו את הסקריפט כך שידפיס גם האם הכתובת היא **Multicast** או **Unicast**.

פרק 9 - רכיבי רשת

ב פרקים הקודמים הכרנו מספר רכיבי רשת. פרק זה נועד כדי לעשות סדר ברכיבים עליהם למדנו.

Hub (רכזת)

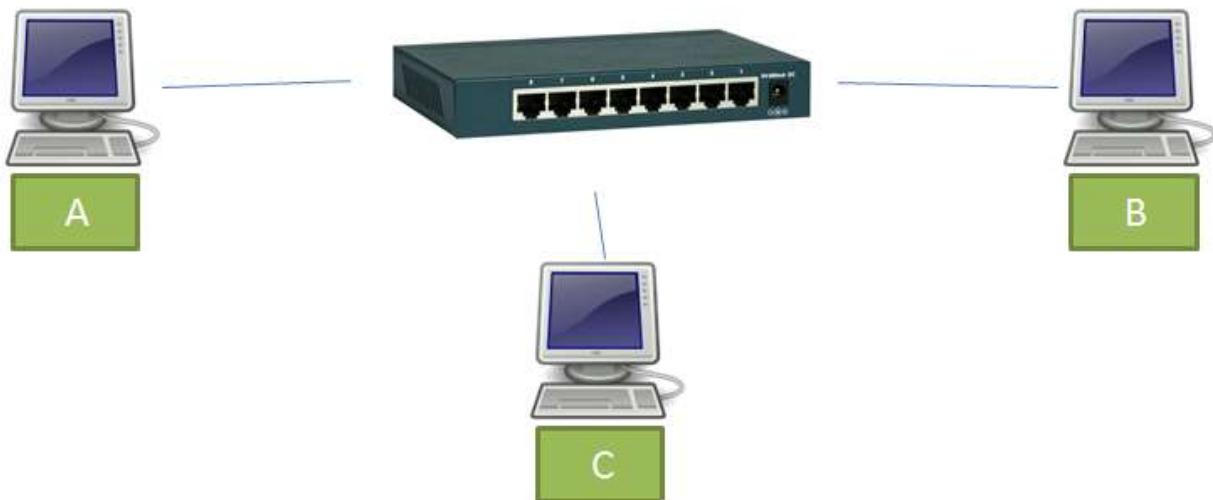
ה-Hub הינו רכיב של השכבה הפיזית, השכבה הראשונה. הוא נועד כדי לחבר כמה ישויות רשת ייחד. ה-Hub אינו מכיר כתובות IP, מבחינתו הוא רק מעביר זרם חשמלי מפורט אחד אל פורטים אחרים. כאשר מחשב שמחובר ל-Hub שלוח מסגרת, ה-Hub מעתיק את המסגרת ושולח אותה לכל הפורטים שלו, מלבד זהה שמננו המסגרת נשלה. כך למשל, בדוגמה הבאה:



המחשבים B ו-C מחוברים זה לזה באמצעות Hub. אם המחשב A ישלח מסגרת אל B, המסגרת תגיע הן אל המחשב B והן אל המחשב C. במקרה שהמחשב A ישלח הודעה אל המחשב C, המסגרת גם תגיע הן אל המחשב B והן אל המחשב C. אם המחשב B ישלח מסגרת המיועדת אל המחשב A, היא תגיע הן למחשב A והן למחשב C, וכך הלאה.

Switch (מתק)

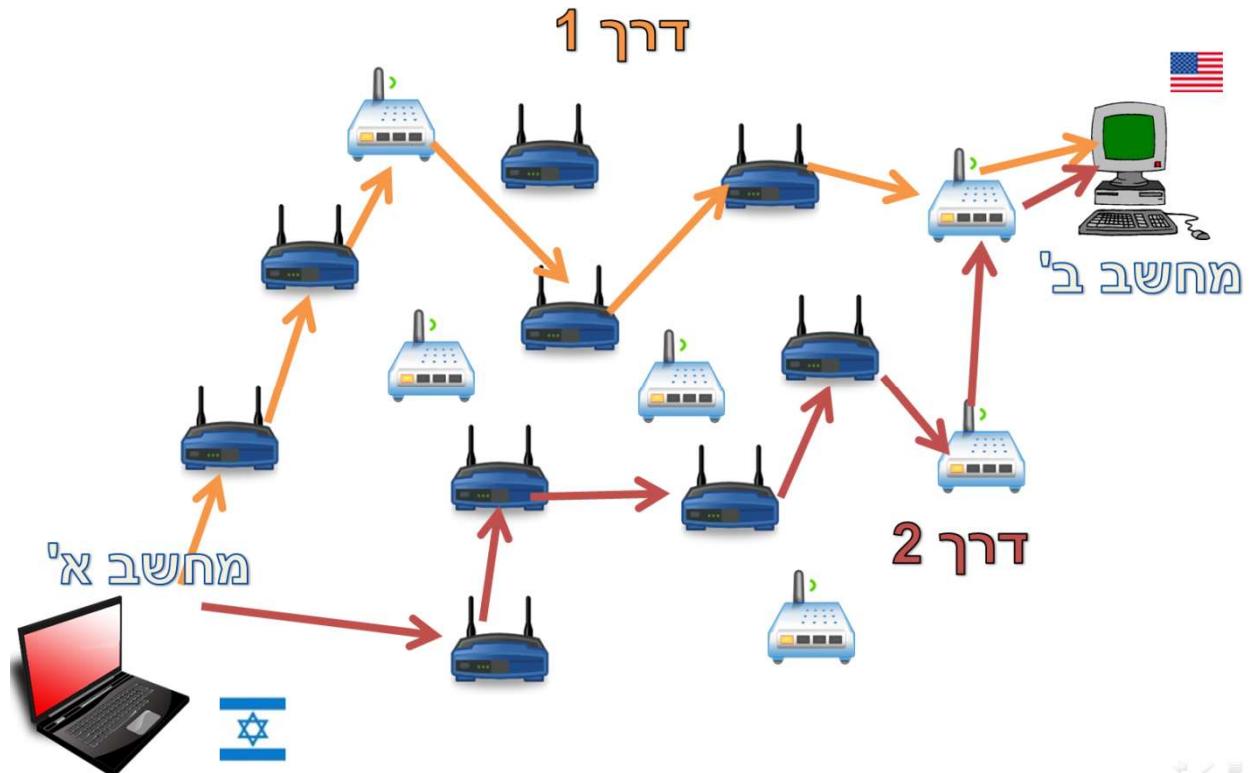
ה-Switch הינו רכיב של שכבת הקו, השכבה השנייה. אי לכך, ה-Switch מכיר כתובות MAC, מבין את המבנה של מסגרות בשכבה שנייה (למשל מסגרות Ethernet), ויודע לחשב checksum. לאחר שה-Switch למד את הרשת, הוא מעביר מסגרת מה포רט בה הוא קיבל אותה אל ה포רט הרלוונטי בלבד. בדוגמה הבאה:



המחשבים B, A ו-C מחוברים ל-Switch. אם המחשב A שלח מסגרת למחשב B, המסגרת תגיע אל המחשב B בלבד - ולא תגיע למחשב C או תוחזר אל המחשב A. באותו אופן, אם המחשב B שלח מסגרת למחשב C, המסגרת תגיע אליו בלבד ולא תגיע אל המחשב A (וכמוון לא תוחזר אל המחשב B).

(נתב) Router

הנתב (Router) הינו רכיב של שכבה הרשת, השכבה השלישי. הנתב מכיר כתובות IP, מבין את המבנה של חבילות IP ופועל על פיהן. על הנתב להבין מבנה של כתובות IP וכן מסכות רשת (Subnet Masks), ולהחליט על מסלול הניתוב הטוב ביותר עבור כל חבילה שנשלחת.



את החלטות הניתוב מבצעים הנטבבים באמצעות טבלאות ניתוב דינמיות. הטבלאות מתעדכנות באמצעות הודעות שהנטבבים שולחים אחד לשני, אם יש שינוי בראשת (למשל - נתב אחד שקרס או עומס באזורי מסוימים בראשת).

טבלת סיכום

שם	שכבה	כתובות שמכיר	מטרה ודרך פעולה
Hub (רכזת)	פיזית (1)	לא מכיר כתובות	يוצר קשר בין מספר ישויות. מעביר כל מידע לכל הישויות המחברות אליו.
Switch (מתח)	קו (2)	MAC Addresses	يוצר קשר בין מספר ישויות ברמת הקו. מעביר כל מסגרת רק למי שהמסגרת מייעדת אליו, באמצעות טבלה המיפה כתובות MAC לפורט פיזי.
Router (נתב)	רשת (3)	IP Addresses	מקשר בין רשתות ומחשבים ברמת ה-IP. מחליט על הדרך הטובה ביותר לנtbן חבילה ממקור אליעד. לרוב פועל בעזרת טבלאות ניתוב דינמיות.

פרק 10 - השכבה הפיזית (העשרה)

מבוא

עד כה, למדנו שבמודול 5 השכבות, יש תפקידים שונים לכל שכבה. הכרנו את שכבת הקוו, שתפקידיה לאפשר לשני מחשבים לדבר אחד עם השני באופן ישיר. השכבה השלישי, שכבת הרשות, תפקידה לאפשר לשני מחשבים לדבר אחד עם השני בין רשותות שונות, קרי דרך רכיבי תקשורת מתחווכים. שכבה נוספת מאפשרת לשני מחשבים, ללא תלות במרחק בין אחד לשני, להעביר מספר ערכיו מידע ביניהם. ביחיד, שכבות אלו מרכיבות את רשותות התקשרות שאנו כל כך רגילים לשימוש היומיומי בהן. למרות שנשמעו אולי כיסויו כבר הכל, מגילהה ל-Google ועד העברת מסגרות בין שני מחשבים מחוברים באופן ישיר, עדין חסירה לנו שכבה בודדת אחת - השכבה שתפקידיה להגדיר את המילים (או האותיות, אם תרצו) הבסיסיות בהן מחשבים משתמשים בשביל לדבר אחד עם השני: **סיבית** (באנגלית - **bit**, ולעיתים גם בעברית - **ביט**).

כל ארבע השכבות אשר נשענות על השכבה הפיזית, מניחות שאפשר להעביר סיביות, או ביטים (bits), בין שני רכיבי מחשב.



סיבית (Bit) - קיצור של המושג ספרה בינארית, ספרה שיכולה להחזיק אחד משני ערכים: 0 או 1. סיבית היא תרגום של המושג הלועזי **bit**, שהוא קיצור לביטוי **digit binary**. בהמשך אנו נשתמש במושג הלועזי, אך נאיית אותו בעברית: ביט או ביטם.

העברה של בית בין שני רכיבי מחשב אמונה נשמעת כמו פעולה די בסיסית, הרי בכך הכל מדובר בשתי אופציות (0 או 1). תופתעו בגלות שמאחורי פעולה זו מתחבאת שכבה שלמה, השכבה הפיזית. מגוון האפשרויות להעביר ביטים בין מחשבים הינו עצום, ולכן יש מגוון רחב של טכנולוגיות המממשות את השכבה זו.



מטרת השכבה הפיזית היא להעביר בית יחיד בין רכיבי מחשב.

אפשר להתבונן בשיטות שונות להעברת בית יחיד דרך הטכנולוגיות המלויות אותנו באמצעות התקשרות בחיננו:

- המחשב הביתי שגורש דרך מודם ה-ADSL.
- ממיר שידורי הcablim וטלפון.
- הטלפון הסלולארי.
- הדיבורית האלחוטית באותו.
- מכשיר ה-GPS.
- אפילו מנורות הרחוב, שבאופן מסוומי נדלקות בתזמון מדויק על פי השעה בה שוקעת זורחת השמש.

כל אחת מהטכנולוגיות הללו מסתמכת על שיטה אחרת שתפקידה להגדיר כיצד להעביר מידע בית או אוסף ביטים על גבי תוווכים שונים: באוויר, בכלי מתקנת ובכלי זכוכית.



תוור (Medium) - ברשותה תקשורת, תוור התקשרות הוא החומר, או האמצעי הפיזי, המשמש להעברת המידע.

לא נכנס לכל שימושי השכבה הפיזית בפרק זה, אך נשים לב שרוב המימושים מתחלקיים באופן גס על פני שלושה תוווכים:

- כבלי מתקנת (לרבות נוחות).
- אוויר ("אלחותי").
- סיבים אופטיים (שהם בעצם זכוכית או פלסטיק).

בפרק זה, ננסה להבין כיצד אפשר להעביר מידע בחומרים הנ"ל, ונפרט בקצרה את התכונות השונות של כל שיטת תקשורת. בנוסף, ננתח לעומק מספר דוגמאות מהחיים, באמצעותם נ謝ף אור על שיטות התקשרות בח'ינו ועל מאפייניהן החשובים. נתחיל מבט אל העבר בו נסדו שיטות שונות להעברת מידע, ולאחר מכן נתבונן באמצעות התקשרות הפיזית שקיימים בכל בית ממוצע בישראל. לסיום, נבחן אתגרי תקשורת גדולים יותר, כמו רשת בין יבשתי של חברת היי-טק גדולה.

עמודי התוור של התקשרות

כפי שציינו קודם, התוור הינו החומר המוחשי בו אנו עושים שימוש על מנת להעביר מידע (למשל, כבל נוחות או האוויר). כל תוור מכתיב אופן שונה להעברת ביטים, ו מגבלות שונות על קצב העברת הביטים והמרקח אליו ניתן להעברים.

העברת מידע באוויר

האם אי פעם שאלתם את עצמכם את השאלה הבאה:



איך מתקשר השלט הרחוק עם הטלויזיה?



השלט הרחוק משדר אותות לטלויזיה באור אינפרא-אדום, אבל איך זה באמת עובד? ראשית, נציין כי הרעיון של שימוש באור על מנת לתקשר למרחקים נהגה עוד בימי החשمونאים. אבותינו השתמשו באש ובעשן על

מנת לאוות ולהעיבר הודעות שונות בין גבעות מרוחקות. זה אמן נשמע מוזר, אבל השימוש באש ובעשן היעוה אמצעי לקידוד ביטים. כשייש אור (או עשן), נתיחס אליו כאילו הוא מייצג את הספרה 1, וכשאין אור (או אין עשן) נתיחס לחושך כמייצג את הספרה 0. בסוף המאה ה-18, פותחו מכשירים שונים להעיבר אור למרוחקים, ובאמצעות קידוד מסוים מראש העבירו הודעות. דוגמה נפוצה מאוד של קידוד פשוט שמאפשר העברת הודעות באמצעות איתותי אור היא קוד מורס.



קידוד (Encoding) או **Coding** - תהיליך בו מידע מתורגם לאוות מסוימים (למשל: אור או חושך). כל שיטת שימוש של השכבה הפיזית מגדרה אופן בו מתרגם את הספרות 0 ו-1 לסימנים מסוימים על גבי התווך.



קוד מורס שהזכרנו קודם לכן, מהו זה בעצם מנגנון הקידוד הוטיקות בעולמו, ושימושים נוספים נרחבים ומגוונים נעשו בו בעבר ובהווה. קוד מורס מגדר לכל אות באلف-בית האנגליקן קידום מושך משני סימנים: קו ונקודה. הטבלה הבאה מראה כיצד מתרגמים כל אות באلف-בית לרצף של קווים ונקודות.

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	• -	U	• • -
B	- - . .	V	• • - -
C	- - - .	W	• - -
D	- - . .	X	- - . -
E	•	Y	- - . - -
F	• . - - .	Z	- - - . .
G	- - - - .		
H	• • . .		
I	• •		
J	• - - -		
K	- . - -	1	• - - - -
L	- - . .	2	• . - - -
M	- - -	3	• • - -
N	- - .	4	• • • -
O	- - - -	5	• • • •
P	- - - .	6	• - • •
Q	- - - - .	7	• - - • •
R	- - . -	8	• - - - •
S	• • .	9	• - - - -
T	- - -	0	

קל להבין כיצד נוכל לתרגם בין קוו ונקודה ל-0 ו-1, ולהחליף קידוד זה בקידוד בינאר. אבל אם נעמיק ונחשוב על קידוד מorus והאופן בו משתמשים בו, נבין שיש לנו מאפיין אחד חסר.



לו הייתם חיים במאה ה-19, וברשותכם פנו המאפשר לכם להعبر איתיוטי אוור למרחוקים, כיצד הייתם מבדילים בין מקטע אוור שמייצג נקודה לבין מקטע אוור שמייצג קו?

המאפיין החסר הוא הגדרה של זמן, או יותר נכון תזמון. ההבדל בין קו לנקודה הוא האורך, או משך הזמן של הבהוב האור. אפשר לומר שהמאפיין הוא מעט "סמי", מאחר שכל אדם יבחן בהפרש המשכים בין הבבובי האור, ויסיק בעצמו שהקצר הינו נקודה והארוך הינו קו. אך יש מקום להגדירה יותר מדויקת כדי לאפשר לאנשים שונים לקודד מידע בקצב שונה (אם נקצר את משכי האור של קו ונקודה, נוכל להعبر יותר אותיות באותו הזמן. כמו כן, הגדרה מדויקת של משכי הקו והנקודה מאפשרת להפריד בין אותיות ובין מיללים בקלות רבה יותר. הסתכלושוב על הטללה וחשבו: כיצד נוכל להבדיל בין האות Q, לבין רצף האותיות MA?⁷⁸



נחזיר אל השلط הרחוק. כעת קל מאד לדמיין כיצד השולט הרחוק מתקשך עם הטלויזיה באמצעות אור (כן, אינפרא אדום הוא גם אור, רק שאינו נראה על ידי העין האנושית). בכל לחיצה על כפתור בשולט הרחוק, השולט משדר רצף הבבויים באורכים משתנים, בצורה דומה מאוד לקידוד Mors, אשר הטלויזיה קולעת ומפענחת. במקום לקודד אותיות, אפשר לקודד כפתורים כגון "העלה ווליום", "הורד ערוץ" ו-"כבה".

גלים נושא מיידע

פריצת הדרך הבאה בניצול האויר כתווך תקשורת התרחשה כמעט מאות שנים לאחר שהומצא קוד Mors, בסוף המאה ה-19, כאשר חוקרים הצליחו להعبر קוד Mors באמצעות גלים אלקטرومגנטיים למרחק של שישה קילומטרים. כדי להבין קצת מהם גלים אלקטромגנטיים, עלינו להבין ראשית מהו גל.

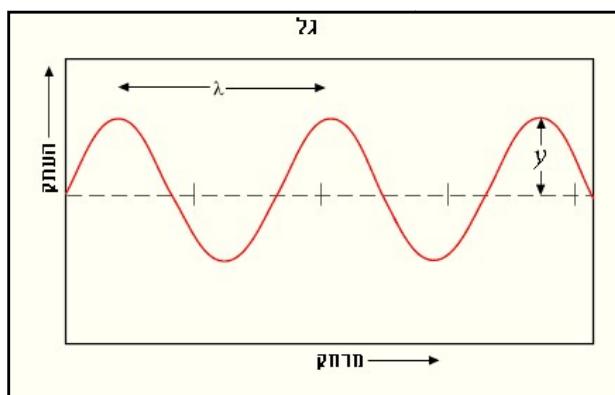


גל (Wave) - התפשטות (או התקדמות) של הפרעה מחזוריית בתווך למרחב. גל יכול לנوع בחומר (כמו גלים במים), אך גם באוויר (כמו גל קול) ואף בvakuum (כמו גלים אלקטромגנטיים, שיכולים לנوع בvakuum ובתווכים רבים אחרים).

⁷⁸ אם תרצו להרחב את היכרותכם עם קוד Mors והשימושים הרבים לו, תוכלם לקרוא עליו כאן: http://en.wikipedia.org/wiki/Morse_code



בתמונה לעיל ניתן לראות גלים המתקדמים במים. הם כMOVן לא חלקים ומושלמים כמו פונקציית הסינוס שבסרטוט התיכון, אבל יכול לדמיין שאם היו מסתכלים על גובה המים מהצד, היינו רואים גל שמציר פונקציית סינוס. לכל גל יש מספר מאפיינים מאוד חשובים שמתראים אותו. מאפיינים אלו ניתן לראות הסרטוט:



- כל גל הוא תופעה מחזורית (כמו פונקציית סינוס).
- **אורך הגל (wave length)** (λ) הוא המרחק שהגל עובר כשהוא מבצע מחזור אחד. אורך זה מסומן בשרטוט ב- λ (האות היוונית למדה).
- **זמן המחזור** הוא משך הזמן שלוקח לגל לבצע מחזור אחד.
- **תדירות הגל (Frequency)** היא כמות המוחזרים שהגל מבצע בשניה אחת. תדירות הגל נקראת גם "תדר" הגל.
- **משרעת (Amplitude)** הגל היא ה"גובה" המקסימלי אליו מגיע הגל (בשרטוט מסומן כ"העתק", ולעיתים נקרא גם **אמפליטודה** בעברית).

დפוֹס התנועה של גלים מופיע בחומרים ותוכים שונים בטבע: תנודות על פני מים (כמו הגלים בים), שינוי גלי בלחץ האוויר (הידוע גם כגלאי קול), ושינוי גלי בשדה החשמלי והмагנטי (הנקרא לרוב הgal האלקטרומגנטי).



הgal האלקטרומגנטי (Electromagnetic Wave) - הוא סוג של gal שנע בתוכים שונים במרחב (אוויר, מים, זכוכית, ריק/אקום ועוד) באמצעות שינוי של השדות החשמליים והמגנטיים. האור שמאפשר שימושו ואנו רואים הוא gal האלקטרומגנטי שהתדר שלו נמצא בתחום שהען רואה (אור gal הרלבנטי נוע בין 400 ל-800 ננומטר⁷⁹ בקירות). עצמת האור שווה לאמפליטודה של gal האלקטרומגנטי, שמעידה על חזק התנועה בשדות

⁷⁹ ננומטר אחד שווה 10^{-9} מטר.

החשמליים והמגנטיים. הgalים האלקטרומגנטיים הם תופעה מיוחדת מאוד בטבע, ותוכלו לגרום עוד עליהם בזקיפידה⁸⁰.



בחזרה להעברת קוד מורה על גבי galים אלקטרומגנטיים, כיצד אפשר לקודד נקודה וקוו באמצעות gal אלקטרומגנטי?

הפתרון הטריויאלי, בהנתן שיש לנו מוחלט galים בעוצמה (אמפליטודה) ובתדר לבחירתנו, הוא לקודד 1 ו- 0 באמצעות שידור או אי-שידור של gal. כאשר נרצה לקודד נקודה, נשדר gal למשר שנייה. כאשר נרצה לקודד קו, נשדר gal למשר שלוש שניות. בין כל קו לנקודה נפסיק את השידור. זה אכן פתרון פשוט, אבל במצבות לא משתמשים בו, והוא רחוק מהיותiesel (זכרו, היום כל טלפון בידי מקודד 100 מגה סיביות בשניה, שזה שווה למאה מיליון 0ים ו-1ים בשניה!). הסיבה לבגינה פתרון זה אינו ישים נועצה באופי ההתפשטות של galים במרחב. דמיינו עצמאם בבריכה, קופצים מעלה ומטה ומייצרים galים. אם תעצרו לשניה ואז תמשיכו, האם galים בבריכה יעצרו גם הם? מובן שלא! לא רק שיש להם קצב התקדמות מסוימים, הם גם מוחזרים מדפנות הבריכה ומשיכים לנوع הלוך ושוב למשר זמן רב. כך גם galים אלקטרומגנטיים למרחב.



אם כך, מה השיטה האמיתית באמצעותה אפשר להעביר מידע באמצעות galים אלקטרומגנטיים?

ישנן שתי שיטות שה提פחו כבר בראשית השימוש בגלים אלקטרומגנטיים: הראשונה נשענת על שינוי האמפליטודה, והשנייה נשענת על שינוי התדר. אם זה לא נשמע לכם מוכר, חישבו שוב על תחנות הרדיו האהובות علينا - רשות משתמשות בשיטת קידוד מבוססת שינוי תדר - FM = Frequency Modulation.



Modulation (אפנון) היא העברת של מידע על גבי gal נושא. הרעיון באפנון הוא להרכיב gal של מידע (כגון gal קול של מוסיקה) על גבי gal "נושא". gal נושא הוא gal "חלק" (gal שמאוד קרובה לפונקציית סינוס) בתדר גבוה יותר מגל המידע. לשימוש בגל נושא יש סיבות רבות, אך הן מורכבות מכדי להסבירן בפרק זה.

⁸⁰ [קיינה_אלקטרומגנטיות](http://he.wikipedia.org/wiki/אלקטרומגנטיות/קיינה_אלקטרומגנטיות)

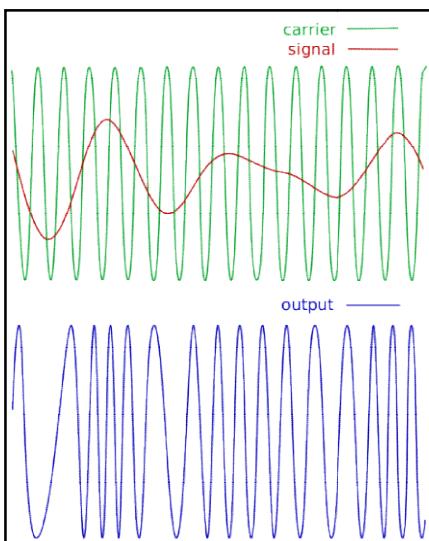
כעת נוכל להבין מה הן שתי שיטות האפנון שהתפתחו עם ראשית השימוש בಗלים אלקטرومגנטיים:

- אפנון מבוסס אמפליטודה - **Amplitude Modulation** - בשיטת אפנון זו, "מרכיבים" גל של מידע בתדר נמוך על גבי גל "נושא" בתדר גבוה וקבוע. בשיטה זו, האםפליטודה של הגל הנושא (בתדר הקבוע) משתנה לפי האמפליטודה של גל המידע.
- אפנון מבוסס תדר - **Frequency Modulation** - בשיטת אפנון זו, משנים את התדר של הגל הנושא על פי האמפליטודה של גל המידע.

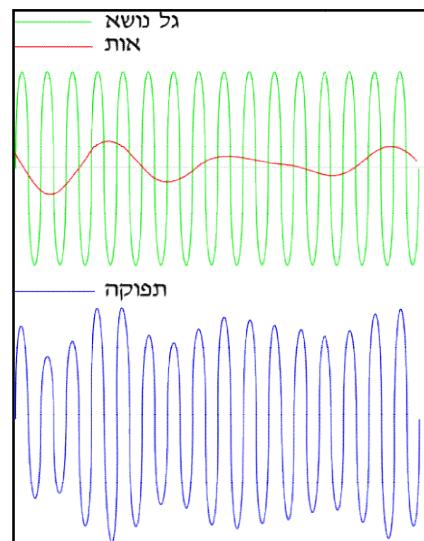


הביטו בשני התרשימים הבאים. בירוק מסומן הגל הנושא, גל בתדר גבוה וקבוע. באדום מסומן גל המידע ("אות"). בכחול, תוצאות האפנון של גל המידע בגל הנושא. איזה תרשيم מראה אפנון AM ואיזה FM?

תרשים ב'



תרשים א'



תרשים א' מראה Amplitude Modulation, לאחר שהאםפליטודה של הגל הכחול משתנה ("גובה" הגלים) בעוד התדר (המרחק בין כל פסגה של גל) נשאר קבוע. תרשימ ב' מראה Frequency Modulation, حيث שההתדר של הגל הכחול משתנה, בעוד האמפליטודה של גל הנושא נשארת קבועה.

השימוש בಗלים אלקטромגנטיים לצורכי העברת מידע הפתחה מאוד מאז סוף המאה ה-19, בה היה ניתן להעביר מספר גדול של נתונים בשניה. היום ניתן להעביר מאות מיליוני נתונים בשניה (100Mbps) וכל זאת על-ידי מכשיר סולולרי קטן. לפני שנבין את שאר ההתקפותיות, נלמד גם על תקשורת חוטית.

כלי נוחש תווור תקשורת

המאה ה-19 הייתה מרגשת מאוד מבחינת השכבה הפיזית. בתחילת המאה ה-19 נמצא הטלגרף, מכשיר פשוט מאוד שמקודד ביטים באמצעות זרם חשמלי, דרך חוט נחושת. משך הזרם (ארוך או קצר) הפריד בין 0 ל-1, וקצב התקשרות היה תלוי ביכולת של בני האדם ליציר ולפענה את רצפי הביטים. בשלב זה אתם כבר אמרו לנו שהקידוד המידע נעשה באמצעות קוד מורס.



העברת מידע באמצעות זרמים חשמליים על חוטי נחושת הניח את היסוד לתעשייה ענקית של תקשורת, שבאupon משעשע התפתחה תחת ענף הדואר. רשות הדואר בעולם פרשו חוטי נחושת בין ערים ומדינות, ואף על קרקעם הימם. אזרחים מן המניין היו מגיעים אל הדואר כדי לשולח ולקבל מברקים שעברו באופן מיידי באמצעות זרמים חשמליים שקדדו בקוד מורס אותיות ומילים על גבי כבל נחושת. בסוף המאה ה-19 נמצא הטלפון, וصاحبיו גל פיתוחים טכנולוגיים שנגעו אליום בהמשך הפרק.

הרשת בית

הבית הממוצע בישראל הוא בית מודרני מאוד מבחינה תקשורתית. זה לא מפתיע, בהתחשב בכך ש勃勃 האנושי ההכרחי לתקשר עם הקרובים והאהובים עליו. מדענים, מהנדסים ויזמים רבים זיהו זאת כהazardנות עסקית לאור ההיסטוריה, וכך הומצאו ופותחו אמצעי תקשורת רבים.



חישבו במשרך דקה ונסו למנות את כל אמצעי התקשרות שיש לכם בבית.

בואו ננסה למפות את אמצעי התקשרות השונים, ולהציגם לכל אחד את התווך בו הוא עושה שימוש:

amu	תווך	amu	תווך	amu
כלי נחושת	טלפון	אוויר	רדיו	
כלי נחושת	פקס	כלי נחושת	טליזיה בכבלים	
טליזיה בלויין	מודם ADSL או כבלים	אוויר	טליזיה בלויין	
טלפון סולארי	נתב אלחוטי	אוויר	טלפון סולארי	

קו הטלפון הביתי, או למה צריך פילטר למודם DSL?

אחרי שלמדנו קצת על סוג תווים שונים, ועל גלים אלקטרומגנטיים, הגיע הזמן שנתמקד באמצעות נוחותם הבלתי נחוצות של כל הזרים - הטלפון. הטלפון משתמש בכבלי נחושת וזרמים חשמליים בשבי להעברת קול אנושי. אם זה לא מספיק, כשהאינטרנט הגיע הביתה באמצעות שנות ה-90, הוא הגיע גם על גבי אותו כבל⁸¹ נוחות שנשא את קו הטלפון. אם אתם לא המומעים ברגע זה, אתם צריכים לחיות(!). איך כבל⁸² נוחות בעובי של מילימטר בודד יכול להעביר לא רק קול אנושי, אלא גם מוסיקה וסרטים באיכות HD? נזכיר את השאלה הזאת דרך הפילטר הקטן שנמצא בשקע הטלפון של משתמשי DSL. על אף שפירוש המילה פילטר הוא מסנן, הפילטר מפצל את השקע לשני שקעים: אחד עברו הטלפון הרגיל, ושני עברו מודם DSL.



פילטר למודם DSL



איך חתיכת פלסטיק מפצלת כבל נוחות אחד לשניים? ואם היא מפצלת, למה קוראים לה פילטר (מסנן)? מה יקרה אם נחבר טלפון ללא פילטר לשקע אחד, ומודם DSL לשקע שני?

התשובה פשוטה לשאלת האחורנה היא - כשהנחבר טלפון ומודם ללא פילטר, ברגע הראשון לא יקרה כלום. אך ברגע שנרים את הטלפון לתחילת שיחה, נוצר התנגדויות והפרעות בין שני ערוצי המידע שעוברים על כבל הנוחות הדקיק: ערוץ הקול, וערוץ ה-data. כדי להבין מדוע זה קורה, נדרש קודם כל להבין קצת בראשת הטלפוניה הביתית.

איך עובד טלפון?

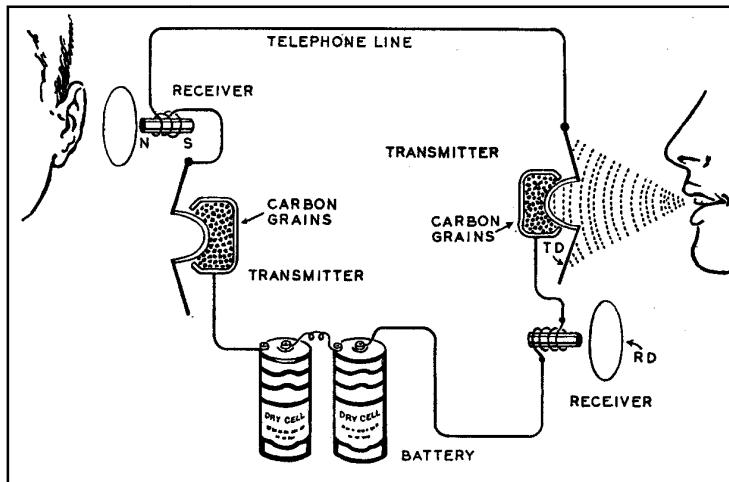
טלפון הוא דוגמה מצינית לפרק שלנו, מאחר שהוא אפליקציה לשיחה אנושית שמשתמשת ישירות בשכבה הפיזית, ללא אף שכבה מתווכת. מכשיר הטלפון ממיר גל קול לזרם חשמלי ולהיפך, באמצעות מיקרופון⁸² ורמקול. התהליך כולל את השלבים הבאים:

- כאשר אנו מדברים אל תוך שפורת הטלפון, אנו מייצרים גל קול שנע באוויר.
- גל הקול פוגע ומרעיד את משטח המיקרופון בשפורת, בעוצמה ובתדירות משתנות, על פי טון הדיבור ותוכנו.

⁸¹ לדיקנים שבינינו, כבלי הנוחות תמיד באים בזוגות. קצת כמו חיבור '+' ו '-'.

⁸² הידעת: מיקרופון בטלפון עשוי מגראג'י פחם: http://en.wikipedia.org/wiki/Carbon_microphone

- המיקרופון בשופורת הטלפון מייצר זרם חשמלי שעוצמתו ותדירותו תואמות את עצמתו ותדירותו של גל הקול.⁸³
- הזרם החשמלי מועבר לטלפון מצד השני, דרך מעגל חשמלי פשוט.
- בטלפון השני, עוצמת הזרם מתורגמת באמצעות רמקול⁸⁴ חזקה לגלי קול אותם ניתן לשמוע.



על מנת שהטלפון יעבוד, נדרש לסגור מעגל חשמלי בין שני מכשירי הטלפון, ולכן הטלפון הם כבלי זוגות, כבילים שהם זוג כבלי נחושת דקים, מלופפים⁸⁵ אחד סביב השני. כאשר מרים את הטלפון בשני צדי הכבל, נסגר המעגל המאפשר העברת הזרם החשמלי שמקודד את גל הקול. בשרטוט לעיל ניתן לראות אדם מדבר אל מיקרופון, קולו מתורגם לזרם חשמלי שעובר אל רמקול המשחזר את גל הקול באוזנו של השומע. כמובן, כל המעגל דורש חשמל ולכן יש בו גם סוללות.



חשוב להבין - הטלפון פועל בשכבה הפיזית בלבד:

- הזרם החשמלי מעביר את המידע - עוצמת גל הקול ללא המרה למידע אחר (כגון 0 ו-1).
- קצב העברת התקשרות תלוי בדוברים, כמו גם סנכרון הדיבור ("הלו?", "היא!", "באי!").
- הטלפון הינו דו כיווני - שני הדוברים יכולים לדבר בו זמן נתן.

⁸³ אם נציג גוף של לחץ האוויר במיקרופון כפונקציה של הזמן, ונשווה אותו לגורם של עוצמת הזרם בכבל הנחושת כפונקציה של הזמן, נקבל גוף שנראה מאד דומה.

⁸⁴ רמקול הוא בעצם מגנט עטוף בסיליקון ומוחובר למשטח.

⁸⁵ מטרת הליפוף היא לבטל השפעות של השראה אלקטרו-מגנטית. קראו עוד על כך: <http://goo.gl/tc1Tae>

איך עובד מודם?



מודם (Modem) - קיצור (באנגלית) של Modulator & Demodulator - מכשיר שמאפן ומשחזר ביטים על גבי ערוץ תקשורת⁸⁶.

המודם הוא הצעד הראשון בתהליך מהפכת המידע שמתחללת בשני העשורים האחרונים. עד המודם, אמצעי התקשרות העבירו בעיקר גלי קול (למשל טלפון או רדיו). בצורה "אלית" שכזו, קשה מאוד להעביר מידע שאינו נראה כמו גל. המודם מאפשר להעביר יחידת מידע הרבה יותר בסיסית: הביט. העברת ביטים למרחוקים, במקרה גבויים, פתחה מגוון רחב של אפשרויות להעברת מידע מכל סוג, כל עוד אפשר לקודד אותו בביטים. במקרה שככל פיסת מידע שתוכלן לעלה אפשר לקודד בביטים, אבל עניינינו כעת אינו בקידוד אלא באופן בו הביטים עוברים ממודם אחד אל מודם אחר.

המודמים הראשונים העבירו ביטים בקצב מאד איטי על גבי כבלי נחושת. תוכנה חשובה של כבלי הזוגות היא שהם מעבירים בצורה טובה תדרים של קול אנושי למרחוקים ארוכים. כדי שמדובר שהתדר של גל הוא כמות המוחזרים שהגאל מבעצם בשניה. תדר נמדד ב-Hz (או בקיצור Hz), כאשר 1Hz הוא תדר של פעם אחת בשניה. התדר של גלי קול אנושיים נע בין 50Hz ל-20KHz⁸⁷.

המודם מנצל תוכנה זו של כבלי הזוגות, ומשתמש בתדרי זרם חשמלי כדי לקודד ביטים. המודם הראשון קודד 0 ו-1 על פי הטלבה הבאה:

ביט	תדר צד א'	תדר צד ב'
0	1070Hz	2070Hz
1	1270Hz	2270Hz

מאחר שהמודם לא מדבר עם עצמו, וכי לא ניתן לטעות דו כיוונית, המודם הצד השני קודד 0 ו-1 באמצעות תדרים שונים.

⁸⁶ כדי להפריד בין מודם לרואוטר הביתי. הרואוטר הביתי משלב בתוכו שלושה או ארבעה רכיבים: מודם שמתקשר על גבי קו טלפון/כבלים, מתג (Switch) שמאפשר חיבור קו של מחשבים ב-LAN, (לעיטים) Access Point שמאפשר חיבור אל-חוטי של מחשבים ב-LAN, ונתב (Router) שתפקידו לנתח את המידע בין המודם ל-Switch ול-Access Point.⁸⁷ או 20,000Hz, הוא קיצור ל-Kilo.

לו הייתם מודים לקו הטלפון, הייתם שומעים רוחשים בתדרים האלו, אך מאוחר וקצב התקשרות של המודמים הראשונים היה 300 ביטים לשניה (bps - bits per second), זהה קצב מהיר מאוד עבור האוזן האנושית (אך אולי מאד למחשב), הייתם שומעים⁸⁸ רחש קבוע.



איך מודמים התקדמו ל מהירות של $C-500\text{Mbps}$ המקובלות בביטנו כיום? זה שיפור של פי מיליון!

השיפור בקצב התקשרות נועז במספר דרכים לניצוליעיל יותר של חוט הנחושת:

- הגדלת כמות התדרים שכל צד יודע לשדר בהם, ובכך להכפיל את כמות הביטים שניתן לשדר. ראה לדוגמה את הטבלה הבאה:

תדר צד ב'	תדר צד א'	בית
2070Hz	1070Hz	00
2170Hz	1170Hz	01
2270Hz	1270Hz	10
2370Hz	1370Hz	11

- הגברת קצב שיידור הביטים, דהיינו קצב ההחלפה בין תדרים. במקום 300 חילופים בשניה ל- 300bps , אפשר לבצע 600 חילופי תדר בשניה, ולהגדיל את קצב השידור ל- 600bps .
- ביטול ההד הוא שיטה נוספת לניצוליעיל יותר של כבל הנחושת. לרוב אנחנו לא חשבים על כך, אבל כשאנו מדברים בטלפון, אנחנו שומעים את עצמנו (או, את ההד שלנו). אם לא היינו שומעים את עצמנו, הייתה לנו תחושה כאילו הקו מנוקק. תכוונה זו של קו הטלפון מפריעה למודם מאחר שהביטים שהוא שולח מתערבבים עם הביטים שהוא מקבל. בלי ביטול ההד, שני המודמים בשיחה נדרשים להשתמש בתדרים שונים (כפי שראינו בטבלאות לעיל). כאשר המודם מבודד את התדרים שהוא שולח מהתדרים שהוא מקבל (באמצעות ביטול ההד), שני הצדדים יכולים להשתמש באותו תדרים, ובכך מגדילים את סך כל כמות התדרים שאפשר להשתמש בהם.
- סינון רעשיות ותיקון שגיאות - רושים בקו מייצרים שגיאות תקשורת (שגיאה היא כאשר צד אחד שולח בית 0, והצד השני מקבל בית 1, או להיפר). ללא תיקון שגיאות, עם כל שגיאה נדרש להעיבר את כל המסר מחדש. סינון רעשיות ותיקון שגיאות נעשה באמצעות שיטות מתקדמות אשר משתמשות על מתמטיקה מורכבת ועיבוד אותות מתקדם.

⁸⁸ [הקישבו כיצד נשמעו המודמים הראשונים בקישור הבא:](https://www.youtube.com/watch?v=3I2Q5-15Mic)



המודם הקלאסי המהיר ביותר הגיע ל מהירות של 56Kbps, אך איך מודם ה-ADSL המודרני מהיר פי 10,000 (100Mbps)?

הזכרנו תחילה שככל זוגות מעבירים תדרי קול אנושי (50Hz עד 20kHz) למרחקים ארוכים. קפיצת המדרגה האמיתית נעשתה כאשר חברות הטלפוניה החליטו "לקצר טוחנים", ולקrab את המרכזיה אל בתיה הלקוחות. קרבה זו אפשרה למודמים לנצל תדרים גבוהים בהרבה - 20kHz, שמאפשרים קצב העברת מידע גבוהים מאוד, וזאת מוביל לסבול מחסרון האורך המגביל של כבלי הזוגות. כך התפתח מודם ה-ADSL, שהגעתו לכל רחבי המדינה נעשתה בהדרגתיות עקב תחיליך התקנת המרכזיות⁸⁹ החדשנות קרוב לבטים.

אנלוגי, דיגיטלי ומה שביניהם

למדנו עד כה על הטלפון ועל המודם - שני אמצעים טכנולוגיים שימושיים אותן עד היום.

למדנו שהטלפון מעביר באופן ישיר את גל הקול לגל של זרם חשמלי, ולכן הוא מכשיר אנלוגי. למדנו שהמודם לוקח מידע בגיןרי, ומקודד אותו באמצעות תדרים קבועיםüber 0 וüber 1. לכן המודם הוא מכשיר דיגיטלי.

בקידוד **אנלוגי**, טווח ערכי המספרים שאפשר להעביר הוא רציף. שעון אנלוגי הוא שעון אשר מודד את הזמן באמצעות קפיץ מתוח שמשחרר בזמן מודד ועקבי. שחרור המתח בקפיץ מתבצע בזמן באופן רציף, ומה שמספריד בין שנייה לשניה הן השנות על השעון, דרך עבר המחוות.

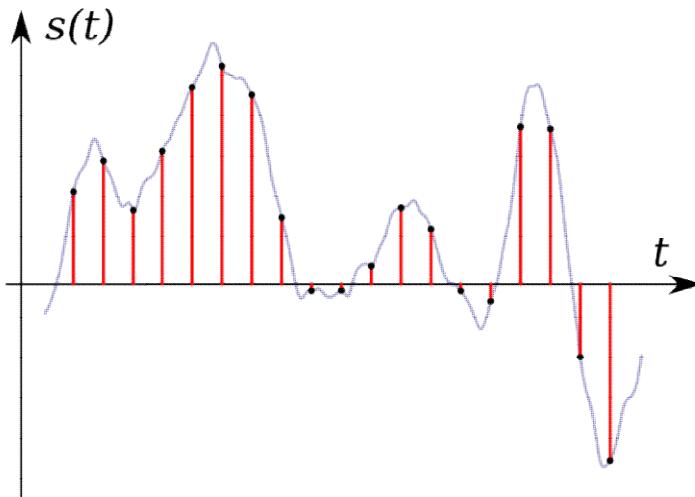
בקידוד **דיגיטלי**, טווח ערכי המספרים שאפשר להעביר הוא בדיד וסופי. שעון דיגיטלי מtabsoס על קристל שמשחרר זרם בפרק זמן קבועים ומדודים. השעון הדיגיטלי סופר כפולות של הזרמים הבודדים, ולכן מקדם את השעה בפרק זמן קבועים ובודדים (לדוגמה נאנו שנייה).

דוגמא שמחישה היטב את ההבדל בין אנלוגי לדיגיטלי היא ההבדל בין תקליטים לדיסקים.

בתקליט, גל הקול שמייצג את המוסיקה, נחרט על התקליט באותה צורה בה הוא מופיע במציאות. אילו היינו לוקחים את התקליט, ומתבוננים מקרוב בח:rightה המעלית שעליו, היינו מקבלים תרשימים של גל הקול. המסתה בפטפון עוברת מעל הח:rightות בצורת גל הקול שבתקליט, ומשחררת את הצליל המקורי. אפשר לדמיין שההתמונה

⁸⁹ מרכזיות אלו נקראות מרכזיות DSLAM - http://en.wikipedia.org/wiki/Digital_subscriber_line_access_multiplexer

הבה מייצגת תרשימים של גל הקול (הקו הסגול) כפי שהוא עבר באוויר. החיריטה בתקליט (אם נתבונן בה "מהצד") תראה בדיק אוטו דבר⁹⁰.



לעומת זאת, בדיסק (CD) המוסיקה מקודדת בביטים. משמעות הדבר היא שאחרי הקלטת המוסיקה, נדרש לקודד את גל הקול למספרים, להמיר אותם לייצוג בינארי, ולאחר מכן לצרוב את הביטים על הדיסק. קידוד מוסיקה לביטים הינו נושא נרחב, אך אם נתבונן שוב בתרשימים שלעיל, נוכל להבין במעט כיצד זה קורה. מטרתנו היא להמיר את הקו הסגול, שמתאר את גל הקול, לרצף מספרים שנייתן יהיה לרשום בביטים. כדי לעשות זאת, נבחר רצף נקודות בהן נדגום (או נמדד) את עצמת הגל ונרשום לכל דגימה את עצמת הגל שנמדודה. אפשר לראות תהליך זה בתרשימים לעיל, לפי הנקודות השחורות שהן הדגימות, והקוים האדומים שמראים את העוצמה הנמדדת בכל דגימה. לאחר שרשכנו את כל המדידות ברצף ביטים על גבי CD, נוכל לשחזר את הגל המקורי בפעולה של"תחבר את הנקודות" מחדש. כמובן שנדגים את גל הקול בקצב יותר גבוה (קרי, ככל שנמדד את הגל בנקודות יותר צפופות), יהיה יותר קל לשחזר את הגל המקורי.



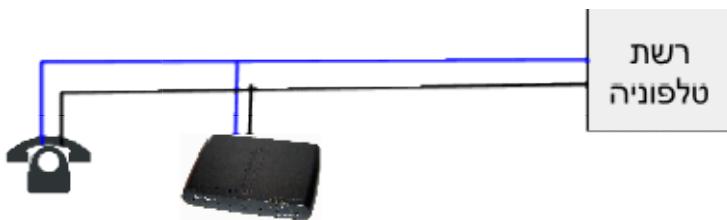
הסיבה שקובץ קול (או ידאו) בקידוד 192kbps הוא באיכות טוביה יותר מאשר קובץ בקידוד 128kbps, נעוצה ביכולת להעיבר יותר נקודות של גל הקול, ובכך לחדר את הרכבת גל הקול המקורי בעט ניגון הקובץ.



מה קורה כשמחברים טלפון לשקע אחד ומודם ADSL לשקע נוסף?

⁹⁰ ניתן לקרוא עוד על תקליטים ופטפונים בקישור: http://en.wikipedia.org/wiki/Gramophone_record

כל שקע הטלפון המחברים לאותו קו ב עצמי מוחברים לאותו מעגל חשמלי. אפשר לראות זאת בשרטוט הבא:



כאשר שני המכשירים מחוברים לאותו מעגל, הזרמיים החשמליים שהם מייצרים מגיעים לשנייהם, ולכן יש סיכוי שהם יפריעו אחד לשני.



עכשווי אפשר להבין למה צריך את הפילטר בשקע הטלפון ומה בדיק הוא מסכן!

תפקידו של הפילטר הוא להפריד ולסנן תדרים.

- הטלפון והמודם מחוברים שנייהם לאותו קו, כלומר לאותו מעגל חשמלי.
- הטלפון יכול לקבל ולהעביר רק תדרים שהאוזן האנושית שומעת ושהקהל האנושי מייצר (מ-50Hz ועד 20KHz).
- מודם ה-ADSL מקלט ומעביר רק תדרים גבוהים יותר.
- הפילטר משתמש ברכיבים אלקטרוניים על מנת לסנן עברו הטלפון רק גלים שהם בתווך הנשמע לאוזן האנושית.
- הפילטר מסנן עברו מודם ה-ADSL רק גלים בתווך התדרים שלו.

ה.filטר נדרש כדי למנוע הפרעות של המכשירים אחד לשני. כל עוד יש הפרדה מלאה בתדרים, אין בעיה. אם מחברים טלפון לשקע ללא פילטר, הטלפון יכול להכנס לקו תדרים גבוהים יותר מאשר 20KHz (לא נדע, כי האוזן לא תשמע אותם), ותדרים אלו יפריעו לסנכרון העדין בין מודם ה-ADSL לבין המרכזיה.

סיכון ביןיהם

עד כה למדנו על עמודי הטווח ההיסטוריים של השכבה הפיזית:

- תקשורת מבוססת אור (מדורות, מօרס באמצעות פנסים, שלט רחוק באינפרא אדום).
- תקשורת מבוססת גלים אלקטרומגנטיים באוויר.
- תקשורת מבוססת זרמים חשמליים בכבל נחושת.

כמו כן, למדנו מה הוא גל ומה הן תכונותיו הייחודיות (אורך הגל, זמן המחזור, התדרות, והמשרעת).

בנוסף, הכרנו מספר מושגים חשובים:

- סיבית / בית.
- תווך.
- קידוד.
- אפנון.
- אנלוגי וdigיטלי.

כדי להבין את המושגים הנ"ל, חקרנו מספר אמצעי תקשורת הקיימים כמעט בכל בית ועמדנו על האופן בו הם פועלים. הטבלה הבאה מסכמת את הרכיבים הללו:

פרוטוקול	קידוד	תווך	אמצעי תקשורת
AM, FM	אנלוגי	גלים אלקטרומגנטיים	רדיו
DVB-T, RF RF, DOCSIS	אנלוגי או דיגיטלי	גלים אלקטרומגנטיים (coax) כבל קווקסיאלי (coax)	טליזיה
AIN DECT	אנלוגי דיגיטלי	כבל זוגות גלים אלקטרומגנטיים	טלפון
V.34	דיגיטלי	כבל זוגות	פקו
ADSL	דיגיטלי	כבל זוגות	מודם ADSL
802.11	דיגיטלי	גלים אלקטרומגנטיים	נקודות גישה אלחוטיות

כעת, נמשיך ונלמד על שימושים ושימושים של השכבה הפיזית בחיי היום יום – החל מהרשות המ.rsדית, דרך מוסדים גדולים וכלה בספקיות תקשורת.

הרשות המדרדית

עד כה חקרו את מגוון רשתות התקשרות הנמצאות בبيתנו. למרות שהן רבות, הן אינן מייצגות את מרבית פתרונות התקשרות הקיימים בעולם. כדי להבין היכן עוברת מרבית מתקשרות הנתונים העולמית, علينا להסתכל במקום בו רוב בני אדם מבלים חלק ניכר מזמןם: המשרד. המשרד מהווה סביבת מידע שונה מאוד מהרשת הביתית:

- הרשות המדרדית לרוב משרותן כמות גדולה יותר של אנשים:
 - הרשות פרושה על שטח יותר גדול.
 - כמוניות המידע וקצביה התקשרות גבויים יותר.
- במשרד יש צורך בשירותי אינטרא-נט (Intranet – אינטרנט פנימי), בנוסף לשירותי אינטרנט.
- המידע הנמצא ברשות המדרדית לרוח רגש יותר (סודות עסקיים, כספים, וכו') מהמידע בראשת הביתית.

כדי להבין מעט מפתרונות התקשרות המסחריים ננסה ללמוד על חיבורו הרשות במשרדים בגודל שונה.

משרד קטן

באו נדמיין שאנו מנהלים חברת לפיתוח יחס ציבורי באינטרנט. בחברה חמשה עשר עובדים שימושיים את מרבית הזמן בעילות תקשורתית בראשות חברותיות. החברה מספקת לכל עובד שולחן כתיבה, מחשב שולחני, מסך גדול ומקלדת. כמו כן, בחברה יש שרת קבצים המ אחסון תיקית קבצים משותפת לכל החברה, ובו מסמכים אסטרטגיים ותיעוד של כל פרויקטי יחס הציבור.



מה הדרך הנכונה לחבר את כל מחשביו החברה לרשות האינטרנט?

17 מחשבים (שרת + 15 עובדים + מחשב מנכ"ל) שכולם מצויים במשרד אחד מהווים מקרה ד' פשוט עבור רשות מקומית מבוססת switch. נזכר ש-Switch הוא רכיב שמחבר מחשבים בשכבה ה-קוקו, והוא מעביר מסגרות מידע על פי כתובת MAC. על שכבת ה-קוקו ועל פרוטוקול Ethernet הרחמנו בפרק הרלבנטי. בפרק זה נתמקד בשכבה הפיזית בה עושים שימוש כמעט כל רשות מקומית בימינו.

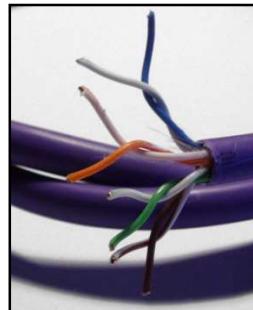
השכבה הפיזית שמתוחת לפרוטוקול Ethernet היא כבל הרשות הסטנדרטי. תשמדו לדעת שכבל זה נקרא כבל CAT 5, החיבור בקצה שלו נקרא RJ-45, והתקן שמאגד את השימוש בכבל נקרא Base-T10. הרבה מושגים לככל כמה פשוט – כתע נגידיר אותם בצורה מסוימת:



כבל CAT 5 – כבל שמאגד בתוכו 4 כבלי זוגות, כל זוגצבע שונה. אפשר לראות את כבלי הזוגות המלופפים סיבוב עצם בתמונה הבאה. אם לדיביך, ישנים מספר סוגים כבליים כאלה: CAT 3, CAT5, CAT5e, CAT6. מבחו צולם נראה אותו הדבר, אך מבפנים הם נבדלים באיכות בידוד ההפרעות החשמליות. איכות CAT6

הbidod משפיע על קצב העברת הביטים. כשאתם הולכים לחנות לקנות כבל רשת, ברוב המקרים תצטרכו כבל

⁹¹CAT5.



כבל CAT 5



חיבור RJ-45 - השקע והתקע של כבלי הרשת הסטנדרטיים, כולל צורתם וסידור הcabלים הפנימיים לфи צבע, מוגדר בטקן שנקרא Registered Jack RJ-45⁹². עבור כבלי הרשת Ethernet, התקן הוא RJ-45⁹³, אך ישנו התקנים דומים גם עבור כבלי אחרים כגון כבל הטלפון (RJ-11).



חיבור RJ-45

בדרכם נדבר על התקן T10-Base, נcir מונה נוספת:



Duplex - **דופלקס** - מאפיין של מערכות תקשורת דו כיוניות בין שני נקודות. מערכת שהיא Half Duplex מאפשרת לשני צדדים לתקשר אחד עם השני בו אופן דו כיווני אך לא סימולטני. דוגמא למערכת Half Duplex היא ווק-טוק (מכשיר קשר אלחוטי), בו רק צד אחד יכול לדבר בזמן שהצד השני מקשיב. כשהשני הצדדים מנוטים לדבר, אף אחד לא שומע את השני. מערכת שהיא Full Duplex מאפשרת לשני צדדים לתקשר אחד עם השני באופן מלא וסימולטני, כלומר שני הצדדים יכולים לדבר באותו הזמן. הטלפון הוא דוגמא למערכת Full Duplex, לאחר שהיא מאפשרת לשני דוברים לדבר בו זמנית וגם לשמוע אחד את השני.

⁹¹ ניתן לקרוא עוד על כבלי רשת בקישור: http://en.wikipedia.org/wiki/Category_5_cable

⁹² ניתן לקרוא עוד על התקן RJ-45 בקישור: http://en.wikipedia.org/wiki/Registered_jack

⁹³ יש לציין שחבר זה נקרא גם חיבור P8C8, ובמkommenות בהם קר הוא נקרא, הכוונה היא לאותו סוג לחבר כמו RJ-45.



תקן Base-T10 - תקן זה מגדיר כיצד משתמשים בcabלי CAT5 וחיבורי RJ-45 כדי להעביר בית בודד על גבי הcabל. ראשית, התקן מגדיר שנדרשים רק שני כבלי זוגות (ארבעה כבלי נחושת בסך הכל), והוא מגדיר גם בדיק באילו מככלי הזוגות להשתמש (ראו תמונה). התקן גם מגדיר כיצד להעביר בית בודד (תדרים⁹⁴ וקידוד⁹⁵), כיצד קובעים את קצב התקשרות, והאם התקשרות היא Full Duplex או Half Duplex. תקן Base-T10 הוא רק אחד משפחחת תקנים הנקראים Twisted Pair (Twisted Pair) Ethernet Over Twisted Pair המונח האנגלי לכבל זוגות).

Pin	Pair	Color	telephone	10BASE-T
1	3	white/green		TX+
2	3	green		TX-
3	2	white/orange		RX+
4	1	blue	ring	
5	1	white/blue	tip	
6	2	orange		RX-
7	4	white/brown		
8	4	brown		

טבלה⁹⁶ המפרטת את השימוש של כל תת-cabל בתווך cabל CAT5
(TX - Transmit, RX - Receive)



תקן ששמעתם פעם את המושג "cabל מוצלב", כאשר ניסיתם לחבר שני מחשבים ישירות אחד לשני עם cabל רשת. בכרטיסי רשת ישנים יותר, לו היו מחברים שני מחשבים עם cabל רשת רגיל, הם לא היו מצליחים לתקשר.



התבוננו בטבלה פירוט תתי cabלים cabל CAT5. האם תוכל לחשב על סיבת מודיען חיבור ישיר של שני מחשבים לא יעבוד?

המושג "cabל מוצלב" מרמז על התשובה. שימו לב שהcabל הירוק מסומן כcabל "שיידור" (Transmit - TX), והcabל הכתום מסומן "קליטה" (Receive - RX). אם נחבר שני מחשבים לכабל אחד, הם ינסו לשדר אותות על אותו cabל, ובcabל הקליטה לא יעברו אף מידע. כאשר מחברים cabל בין מחשב ל-Switch, Switch קורא מידע מהcabל הירוק וכותב מידע אל cabל הכתום. מכאן אפשר לנחש שגם cabל מוצלב פשטוט מצליב בין cabל הירוק לבין cabל הכתום: מצד

⁹⁴ ניתן להרחיב על התדרים בהם נעשה שימושocabלי רשת: <http://en.wikipedia.org/wiki/Baseband>

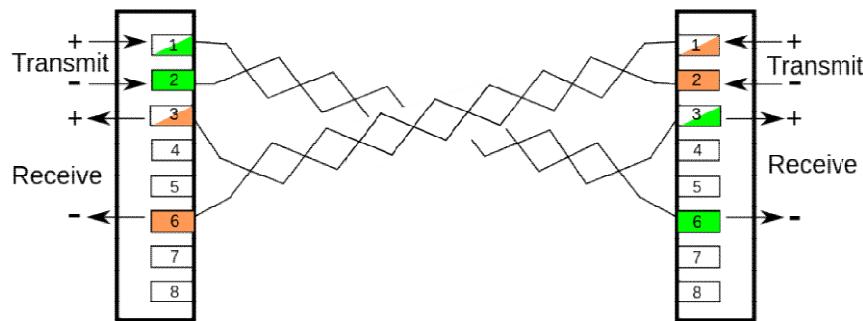
⁹⁵ כדי להבין קידוד ביטיםocabלי רשת, קראו עוד על קוד מנצ'סטר: http://en.wikipedia.org/wiki/Manchester_code

⁹⁶ התרשים המקורי: http://en.wikipedia.org/wiki/Ethernet_physical_layer

אחד של הcabל, החיבור יהיה כפי שהוא בטבלה, ובצד השני, הcabל הירוק יחבר לפינים של הcabל הכתום (פינים 3 ו-6) ולהיפך. יש לציין שכמעט רוב כרטיסי הרשת תומכים בזיהוי אוטומטי של כבלי השילוח והקבלה, ולכן כמעט אין יותר צורך בכבלים מוצלבים.



cabל רשת מוצלב (Ethernet Crossover Cable) - הינו cabל רשות בו כבלי הזוגות של השילוח וקבלת הוצלבו, דבר המאפשר לחבר שני מחשבים ישירות אחד לשני, ללא Switch ביניהם. ניתן לראות את ההצלה בין הcabלים בשרטוט שולחן:



משרד גודל

משרד יכול להתנהל עם אוסף Switchים ועם מספר Routerים כל עוד המרחק הפיזי בין רכיבי תקשורת קטן, ואפשר למשוך cabל רשות בין כל שני רכיבים. אבל מה אפשר לעשות כאשר החיבור שלנו גדול ומתפרשת על פנים מסוף בניינים? או במקרה יותר מורכב, החיבור שלנו היא בעצם מפעל שפירושו על שטח די גדול, ובתוכו נדרש להעביר תקשורת מקצת אחד לקצת השני? חברה פרטיט, שאינה חברות תקשורת כמו בזק, או הווט, אינה יכולה להרשות לעצמה למשוך cabלי תקשורת למרחקים ארוכים לאחר ומדובר בתהיליך יקר וממושך. במקרים כאלה, נדרש פתרון אחר שמחזר אותנו לתקשורת האלחוטית.

בחצי הראשון של פרק זה פגשנו בתקשורת האלחוטית בשלט הרחוק. [בנוסף א' של פרק זה](#), אתם מוזמנים להרחב על הרשת האלחוטית הביתה. טכנולוגיות אלו של תקשורת אלחוטית סובלות מרוחק תקשורת מוגבל ומקצב יחסית נמוך. כדי לחבר שני בניינים בהם שירותי או מאות מחשבים, נדרש פתרון תקשורת שייעבור מרוחק של מאות מטרים ועד קילומטרים בודדים, ויעביר קצבים גבוהים של מידע (כאלו שעומדים בקצביו רשת Gigabit Ethernet). תקשורת מיקרוגל היא הפתרון שוננה לצרכים האלו.



תקשורת מיקרוגל (Microwave Transmission) - העברת מידע באמצעות גלים אלקטרומגנטיים בטווח אורך גל שנitin למדود בסנטימטרים. כפי שלמדנו קודם, אורך הגל ותדרותו קשורים ביחס הפוך, ולכן ניתן להסיק שהגלי מיקרוגל הם גלים בטווח התדרים בין 1GHz ל-30GHz.



למי שאינו מבין מספיק בפיזיקה או הנדסה חשמל, גלי מיקרוגל יכולים להישמע כמו טווח תדרים מאוד שרירותי: למה דוקא גלים בתדר בין 1GHz ל-30GHz עוניים לנו על הבעה? יש לכך כמה סיבות מאוד מדוייקות:

- בגל אורך הגל (קטן אך לא קטן מדי), קל לבנות אנטנות כיוניות - אנטנות שמרכזות את הגלים האלקטרומגנטיים בכיוון אחד (ראו תמונה לעיל).
- גלי מיקרוגל עוברים את האטמוספירה ללא הפרעות משמעותיות.
- התדרות של גלי המיקרוגל מאפשרת אפנון עליהם ביתים בקצב גבוה.

תכונות אלו של הגלים האלקטרומגנטיים מאפשרות לבנות ציוד שידור וקליטה מאודiesel. האנטנות הכיוניות מאפשרות לחסוך אנרגיה מצד אחד, ומצד שני מונעות הפרעות בין ערוץ תקשורת מיקרוגל אחד לשני. חסכו האנרגיה מתאפשר בغال ריכוז אלומת הגלים האלקטרומגנטיים. באנלוגיה לאור נראה (שצצ'ור, הוא גם גל אלקטромגנטי), פנס ממוקד מאריך למרחק גדול יותר מאשר נורה "עגולה". באותו אנלוגיה, אפשר גם להבין מדוע ערוצי תקשורת מיקרוגל לא מפריעים אחד לשני: שני פנסים ממוקדים יכולים להאיר שתי אלומות באותו חדר מבל' שהאלומות יפריעו אחת לשניה, ולעומתם האור משתי נורות יתערבב ויהיה קשה להפריד בין מקורות האור. החסרון של גלי מיקרוגל הוא שנדרש קו ישיר ונקי מחסומים בין האנטנה המשדרת לאנטנה הקולטת⁹⁷.

לסיכום, לגלי מיקרוגל יש תכונות מיוחדות המסייעות בהעברת מידע בקצבים גבוהים ולמרחקים ארוכים (כל עוד הם "ישראלים" ולא מחסומים). שיטת אפנון הביתם בגלי מיקרוגל דומה לשיטה שלמדנו בראשית הפרק, המתבססת על שינוי תדר.



בתמונה לעיל מצולמת אנטנת "תוף", המשמשת לתקשורת מיקרוגל. אפשר למצוא אותה לרוב על עמודי אנטנות סלולריות, ולא בغال שהיא מתחברת עם מכשירים סלולריים, אלא בغال שהיא מאפשרת תקשורת בין אנטנות לבין מרכזי התקשורת של חברות הסולולר.

⁹⁷ אם תחשבו לרגע, גלי רדיו רגילים עוברים מרחקים ארוכים בהרבה, עוקפים הרם וגבועות ונכנסים לבית דרך הקירות. אין זה כך בתדרי המיקרוגל.

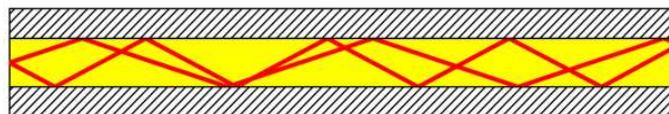
ספק תקשורת

ראינו שמשרדי קטן עד בינוי יכול להסთפק באוסף Switchים ו-Routerים, ושבורה יותר גדולה יכולה להיעזר בתקשורת מיקרגל על מנת לחבר משרדים או אתרים מרוחקים. עם זאת, התעלמנו עד כה חלק גדול מאוד בשרשראת התקשרות שבה אנו משתמשים כל יום: ספק התקשרות. ספק התקשרות כוללים גם את ספקי התשתיות (בזק, הוט, חברות הסלולר) וגם את ספקי השירותים (ISP'ים⁹⁸ לMINIHAM). חברות אלו מעבירות כמויות בלתי נתפסות של נתונים בכל שנה, ולמרחוקים בין לאומיים ארוכים מאוד. הפטרון היחיד להעברת תקשורת בקצבים גבוהים מאוד למרחוקים ארוכים מאוד הינו הסיב האופטי.



סיב אופטי (Optical Fiber) - הינו סיב עשוי זכוכית או פלסטיק, המאפשר העברת אור בתוכו למרחוקים ארוכים עם אובדן מינימלי של עצמה.

היחוד של הסיב האופטי, כפי שניתן לראות בתמונה, נעוץ בקר שהאור "כלוא" בתוך הסיב, ולא יכול להתפזר ולאבד מעוצמתו. אור שנכנס בקצה אחד של הסיב האופטי, נע לאורכו ומוחזר פנימה מדפנות הסיב (כמו מראה).



ישנם יתרונות עצומים לסיב האופטי:

- נדרש כמות מועטה מאוד של אנרגיה בשכיל להעביר אוור דרך הסיב: הבוהב קל של נורת LED בצד אחד מאפשר להעביר את האור למרחוק של עשרות קילומטרים (לשם השוואה, דמיינו כמה רוחק אתם יכולים להאיר עם פנס ה-LED החזק ביותר שלכם).
- מידע לא נכנס ולא יוצא מהסיב - אין הפרעות בתקשורת ולכן ניתן להציג נתונים גבוהים מאוד של תקשורת. לשם השוואה, קצבי המידע אליהם ניתן להגיע באמצעות סיבים אופטיים מגעילים ל-1TB/s, למרחוק של מעל 100 ק"מ. בקצב זה אפשר להעביר 12.5 Terabit לדיסקים של 1TB בשניה!

אפנון בית על גבי גל אוור בתוך סיב אופטי נעשה בצורה פשוטה, שדומה יותר להבוהב - אוור חזק מיצג 1, אוור חלש מיצג 0. כמו כן, ישן מספר דרכי יצירתיות להגברת קצב שידור הביטים. דרך אחת היא כMOVן להגברת את קצב הבהיר, ולהעביר באופן ישיר יותר ביטים בשניתה. דרך אחרת היא להעביר מידע במספר "צבעים" במקביל. "צבעים" שונים של אוור הם בעצם תדרים שונים של גל האור. אם נשימוש במספר LED'ים בצבעים שונים ובמספר גלי אוור (שריגשים לצבע יחיד), נוכל להכפיל את קצב שידור המידע. דרך שלישית היא פשוט להעביר

.ISP - Internet Service Provider⁹⁸

מספר סיבים אופטיים בלבד: אם כבר מושכים כבל למרחק, אפשר לקבץ עשרה סיבים אופטיים בלבד ביחד ולפרוש אותם בפעם אחת.

שאלה נוספת שיכולה לעלות לגבי סיבים אופטיים למרחקים ארוכים היא כיצד מושכים סיב למרחק גדול מ-100 ק"מ? התשובה לכך פשוטה, והוא דומה כמעט לכל אמצעי התקשרות: מסרים. **מסר (Relay)** הוא רכיב שמקבל אות תקשורת, מגביר אותו ומשדר אותו הלאה. מסר א/or מקבל את האותות מסיב אופטי אחד, ומיצר אותם מחדש בסיב האופטי השני. במקרה שתהיתם, מניחים מסרים גם על קרקעית הים, בשביל למשור כבליים תת-ימיים.

לסיכום, סיבים אופטיים הם תווך יעיל ביותר להעברת מידע, והם אחראים להעבירה את מרבית התקשרות האינטרנט בעולם. אמנם השתמשנו בספק תקשורת מבוססי עקרנים עיקריים של סיבים אופטיים, אבל חשוב לציין שסיבים אופטיים נמצאים בשימוש נרחב גם ברשתות משלדיותBINONIOT וגדלות, על מנת לחבר בין נתבים שמעבירים קצבים גדולים של מידע.

השכבה הפיזית – סיכום

בפרק זה למדנו על הדרכים השונות בהן אפשר להעיר את יחידת האינפורמציה הבסיסית: הבית.

למדנו על תקשורת קוית (טלגרף, טלפון, Ethernet, סיבים אופטיים) וגם על תקשורת אלחוטית (שלט רחוק ותקשורת מיקרוגל), ועל האופן בו סוג תקשורת השונים משתלבים בחיננו. כמו כן, למדנו מושגים בסיסיים שמאפשרים לנו להבין את המאפיינים של שיטות תקשורת השונות ולהשוות ביניהן.

כשמדובר בתקשורת קוית או אלחוטית, מדובר בתווך התקשורת, החומר על גביו ניתן להעיר ביטים (נחושת, אויר, אור ועוד). האופן בו מגדרים את האות המסמן 0 ואת האות המסמן 1 נקרא **קידוד**. לתווים שונים ושיטות קידוד שונות מוכתב **קצב** אחר. הקצב של ערוץ תקשורת נקבע על פי הנקודות המקסימלית של ביטים שנייתן להעביר על גבי הערז בחניה אחת. אך לא רק הקצב משתנה משיטת תקשורת אחת לשניה, אלא גם **מרחק השידור**: כמה רחוק אפשר להעביר את הביטים. מופיע נסוף של ערוצי תקשורת הוא אופן **הנסכון**: האם שני הצדדים המתקשרים יכולים לשדר בו זמינות, ואם כן איך דואגים שביטים יועברו בשני הכוונים מבלי להתנגש. כמובן שהשאיפה לערז תקשורת ללא התנגשויות כמעט ובלתי ניתנת להשגה, ולכן תקשורת נדרשים גם **لتיקון שגיאות**, שיטות המסייעות בגילוי ותיקון ביטים שהשתנו או התנגשו. בעזרתם המושגים והדוגמאות שuberנו עלייה בפרק זה, קיימים בידיכם הכלים להבין קצת יותר טוב את אמצעי התקשורת הסובבים אותנו.

במהלך הפרק, נתחנו מספר דוגמאות מהחיים. התחלנו במבט אל העבר בו נסדו שיטות שונות להעברת מידע, ולאחר מכן התבוננו באמצעות התקשורת הפיזית שקיימים בכל בית מודר בישראל. עברנו דרך הטלגרף והטלפון, השווינו בין פטיפון לדיסק, ודיברנו על רשותם במשרד קטן וגדול אף על ספקיות אינטרנט.

בפרק זה הבנו שהשכבה הפיזית מאוד שונה השכבות בכך שהיא שומרת מגוון מאוד וקיימים באמצעות רבים. אפשר לחשב על השכבה כממשק המחבר בין שכבות התקשורת המופשטות וה"נקיות" לבין אמצעי התקשורת הקיימים בעולם.

על אף שעברנו על כל חמש השכבות, עיסוקינו ברשות עדין רחוק מלסתויים. בפרק הבא נחבר את הכלים והמידע שרכשנו בפרק האחרוניים כדי הבנה טובה יותר של הדרך בה עובדת האינטרנט, וכן נכיר נושאים מתקדמים.

נספח א' - הרשת האלחוטית

אנחנו שוחים כל יום בكمות בלתי נתפסת של ביטים שזרמים באוויר בצורה גלים אלקטромגנטיים. רשת WiFi היא רק אחת מהן, ואליה מצטרפות הרשת הסלולרית, שידורי הטלויזיה (האנלוגיים והדיגיטליים - עידן⁹⁹), שידורי הלוויין, שלטי טלויזיה, דיבוריות Bluetooth ועוד. בכל רשת אלחוטית, ישנו מספר מאפיינים שמספרדים אותה מהשאר:

- טווח התדרים בו הרשת יכולה לשדר.
- עצמת האות המשודר וכיוונו למרחב.
- קצב התקשרות.
- שיטת הסyncrown והתזמון בין רכיבי הרשת.

הרשת האלחוטית הביתיית, בשמה הנפוץ WiFi, מتبוססת על פרוטוקול 802.11 אשר גדר ע"י מכון IEEE¹⁰⁰. פרוטוקול זה מכיל תתי-פרוטוקולים רבים: 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac ועוד. תתי ה프וטוקולים השונים זה זה במאפייניהם הנ"ל, אך יכולים משמשים לאותה המטרה – חיבור אלחוטי בין מחשבים שהמרחיק ביניהם לא עולה על כ-15 מטרים.

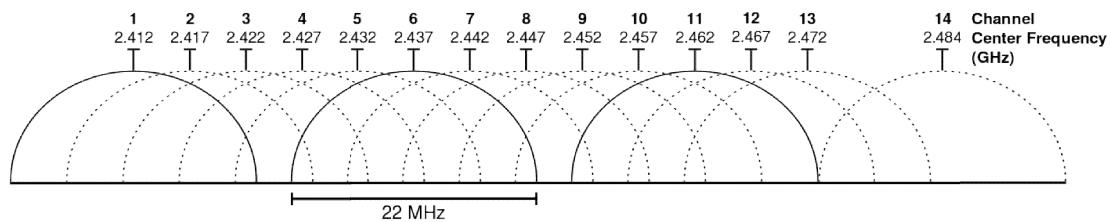
הרשותות האלחוטיות משדרות בתדר 2.4GHz (תתי-פרוטוקול a ו-g) או 5GHz (תתי-פרוטוקול a). תדר של 5GHz מאפשר קצב שידור גובה יותר, בעוד תדר 2.4GHz מאפשר מרחק שידור גובה יותר¹⁰¹.

רשת WiFi אינה משתמשת בתדר המדוייק, אלא בתדר קרוב לתדר שציין לעיל. התדר המדוייק מוגדר לפי ה-Channel (ערוץ) בה הרשת האלחוטית עשויה שימוש. לרשותות האלחוטיות הוגדרו 14 ערוצים. רשותות המשדרות בתדר 2.4GHz, בעצם משדרות בתווים התדרים בין 2.4GHz ל-2.5GHz, ומתחכו כל רשת אלחוטית עשויה שימוש בערוץ אחד אשר מכיל טווח תדרים של 22MHz. לדוגמה, רשת אלחוטית 802.11g, בערוץ מס' 6, משדרת בתדרים בין 2.426GHz עד 2.448GHz. כמו כן, בין הערוצים השונים יש חיפפה, ולכן רשותות אלחוטיות המשדרות בערוצים קרובים עלולות להפריע אחת לשניה. התרשימים הבא מראה את הערוצים של רשת WiFi בתדר 2.4GHz. ניתן לראות בתרשימים שלכל ערוץ יש תדר מרכזי, וקשת המיצגת את טווח התדרים בהם הערוץ עשויה שימוש.

⁹⁹ עידן+: שידורי טלויזיה אלחוטיים דיגיטליים. ניתן להרחיב בקישור הבא: <http://goo.gl/xtwlca>.

¹⁰⁰ מכון IEEE הוא מכון בין לאומי שתפקידו לכתוב ולתזקק תקנים המאפשרים לייצר מוצריים שיעבדו אחד עם השני בתיאום. פגשנו גם בפרק שכבת הקו את ארגון IEEE, אשר הגדרת תקן ה-Ethernet, שמספרה 802.3. ניתן להרחיב: http://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers.

¹⁰¹ כאשר התדר נמוך יותר, הגל מבצע פחות מחזורים בשניה, ולכן אורך הגל גדול יותר. ככל שאורך הגל גדול יותר, כך טובה יותר יכולתו לעבור מכשולים כגון קירות.



מגבלה נוספת על קצב השידור בשרות אלחוטיות נובעת משימוש בתדר שידור יחיד. מגבלה זו מחייבת את הרשת האלחוטית לפעול במצב **Half Duplex**. במצב זה, רק צד אחד יכול לשדר מידע, בעוד כל שאר רכיבי התקשרות נדרשים להאזין. שימושות המגבלה היא שכל שיש יותר מחשבים מחוברים ברשת אלחוטית בודדת, הקצב שלה יקטן ויתחלק בין כל המחשבים.

פרק 11 - איך הכל מתחבר, ואיך עובד האינטרנט?

בפרק הראשון של הספר, התחלנו לשאול - איך עובד האינטרנט?
ניסיינו לעשות זאת על ידי התמקדות בשאלת הבאה:



בפרק הראשון, התחלנו לענות על השאלה זו במושגים כלליים מאוד. מאז, עברנו כברת דרך ארוכה. למדנו לתכנת באמצעות Sockets, הכרנו את מודל חמש השכבות והתעמקנו בכל שכבה בו. רכשנו כלים כמו Wireshark ו-Scapy, והכרנו רכיבי רשת שונים. עכשו, מצויים בכל הידע זהה, נוכל לשאול מחדש את השאלה ששאלנו ולנסות להבין - איך כל מה שלמדנו מתחבר יחד?

בפרק זה ננסה לענות על כך ביתר פירוט, ונחבר דברים שכבר למדנו לכדי סיפור שלם - איך המחשב שלי מצליח לאלוש באינטרנט? לשם כך נשאל הרבה שאלות. נסו לחשב על התשובות, ובדקו האם אתם מצליחים לספר **את הסיפור בעצמכם**.

הסיפור שלנו מתחילה עם המחשב שלנו:



מה המחשב שלנו צריך לעשות כדי להצליח לתקשר עם האינטרנט?

במור התחלה, המחשב שלנו יצריך לדעת כל מיני פרטים. הוא צריך לדעת מה כתובת-h-IP שלו, כדי שיוכל לשЛОח אחר כך חבילות נוספת. הוא צריך לדעת מה מס' כתף הרשות שלו, כדי לדעת איזה מחשבים נמצאים איתו-ב-Subnet ואילו לא.

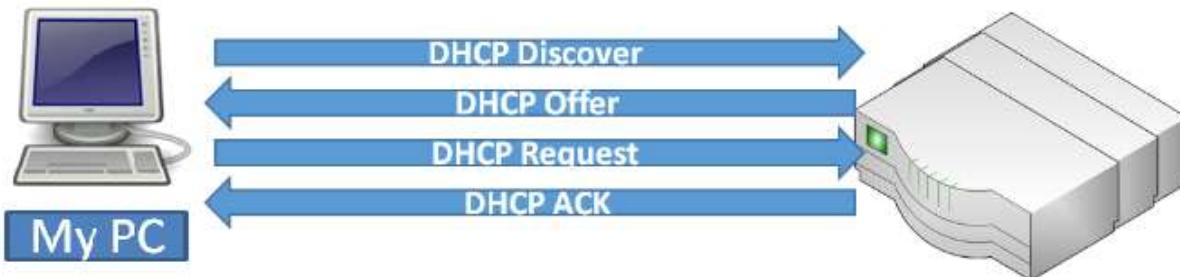


איזה מידע יש למחשב שלנו על הרשות?

בשלב זהה, המחשב יודע רק את **כתובת ה-MAC** של כרטיס הרשות שלו. הוא יודע את הכתובת, כיון שגם צרובה באופן פיזי על כרטיס הרשות.

איך המחשב שלנו מושג את כתובת ה-IP ושאר פרטי הרשות שלו?

כפי שמדובר בפרק [שכבה הרשת/DHCP](#), ישן מספר דרכים לקבל את פרטי הרשות. הדרך הנפוצה כיום היא באמצעות **פרוטוקול DHCP**. באמצעות פרוטוקול זה, כרטיס הרשות שלנו שולח הודעה הודעת DHCP Discover. ההודעה נשלחת ב-**Broadcast**, כלומר שכל הישויות ברשת יקבלו אותה. שירות DHCP רואה את הבקשה, ומהזיר DHCP Offer, הודעה בה הוא כולל את פרטי הרשות המוצעים לכרטיס הרשות שלנו: כתובת ה-IP שלו, שירות ה-DNS הרלוונטי ועוד. מכיוון שברשת שלנו יש רק שירות DHCP אחד, לא תהיה הצעות נוספות, והמחשב שלנו ישלח הודעה DHCP Request לשירות DHCP שהוא מבקש לקבל את ההצעה. לבסוף, השירות DHCP ישלח הודעה ACK, שאחריה יוכל המחשב שלנו להתחיל להשתמש בכתובת ה-IP שנימנתה לו.



מזל טוב! קיבלנו כתובת IP!

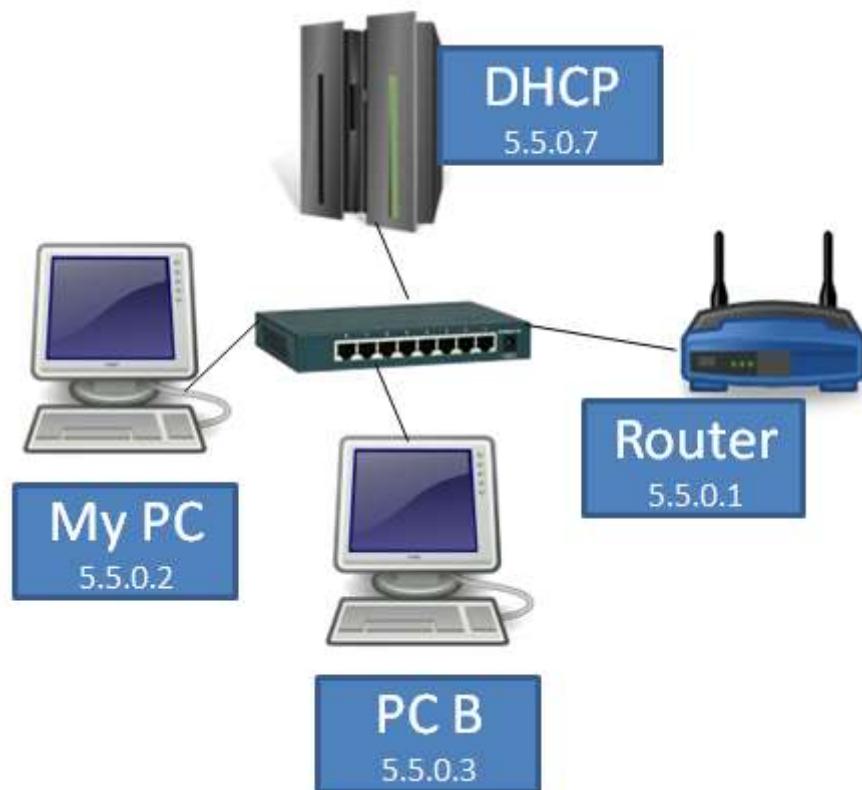
לצורך הדוגמה, נאמר שקיבliśmy את הכתובת 255.255.0.0¹⁰², כמשמעות הרשות היא 5.5.0.2. בנוסף, הנטב שלנו עבר תהליך דומה על מנת להשיג כתובת IP מסוימת. את תהליך זה הוא עבר מול השירות DHCP של ספקית האינטרנט שלנו. נאמר שהנטב קיבל את הכתובת: 1.1.1.1, ומסינכת הרשות היא 255.255.0.0.

איך ההודעה הגיע אל שירות DHCP?

על מנת שההודעה תצליח להגיע אל שירות DHCP, על הלקוח להיות איתו באותו ה-Broadcast Domain. כמובן, ההודעה צריכה להגיע מוביל לעבר אף נתב בדרך. זה הזמן להזכיר שבעצם, כפי שמדובר בפרק [שכבה ה-2/רכיבי רשות בשכבה ה-2](#), המחשב שלנו מחובר אל **Switch** (מtag), אליו מחוברים גם מחשבים נוספים בשרות (למשל המחשב בשם "PC B", או שירות DHCP¹⁰³), כמו כן - הנטב (Router) שלנו.

¹⁰² כפי שמדובר בפרק [שכבה הרשת/נספח ב' - כתובות פרטיות NAT](#), הכתובת תהיה לעיתים תקופות כתובות IP פרטיות. על מקרה זה נרchie בנספח א' [של פרק זה - תקשורת מאחורית NAT](#).

¹⁰³ במקרים מסוימים פרטיטים, הנטב הביתי הוא בדרך כלל גם שירות DHCP של הרשות. לצורך פשוטות הסביר, נניח במקרה זה שימוש שירות DHCP נפרד.



מה השלב הבא?



כעת, המחשב שלנו צריך לגלות מה היא כתובת ה-IP של www.facebook.com, על מנת שיוכל לשולח אל השירות של Facebook בקשות. כפי שמדובר בפרק [שכבת האפליקציה](#), על המחשב שלנו להשתמש בפרוטוקול DNS, וلتשאול את שרת ה-DNS שלו מהי כתובת ה-IP של Facebook.

מהו שרת ה-DNS שלנו? כיצד המחשב יודע זאת?



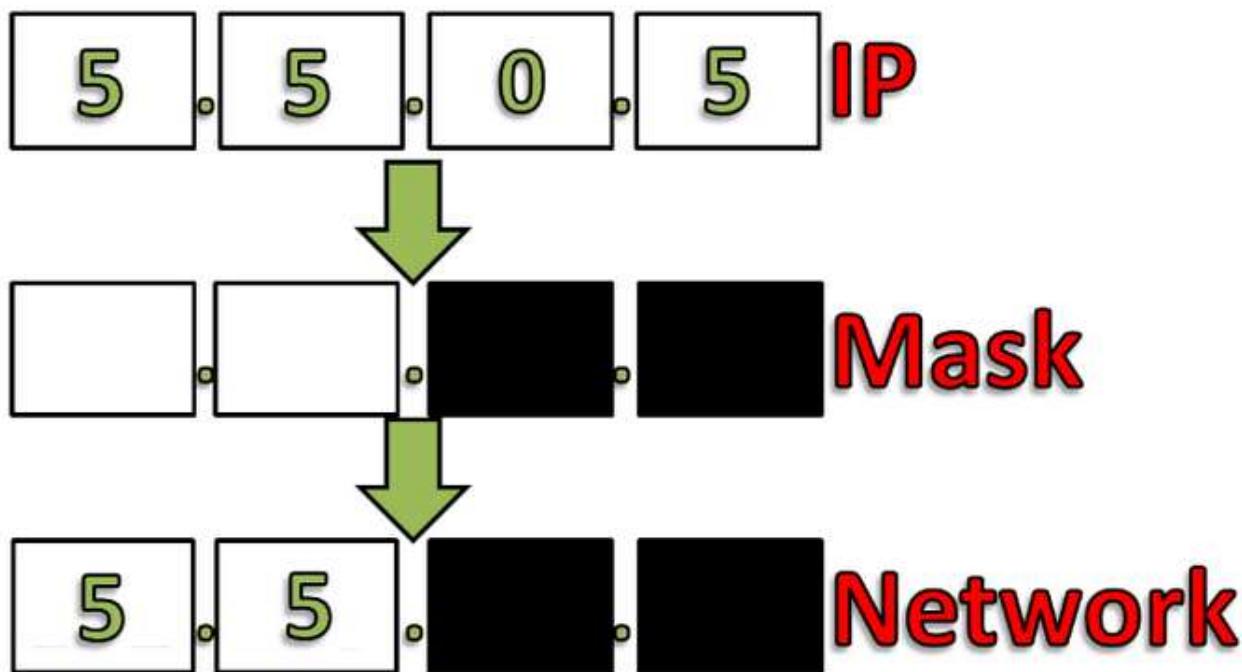
שרת ה-DNS שלנו הוא שרת ה-DNS של ספקית האינטרנט שלנו.¹⁰⁴ המחשב שלנו יודע זאת כיוון שהוא קיבל את כתובת ה-IP של שרת ה-DNS באמצעות תהליך DHCP, בו הוא קיבל גם את כתובת ה-IP שלו. לצורך הדוגמה, נאמר שכותבת ה-IP של שרת ה-DNS הינה 2.2.2.2.

¹⁰⁴ במקרים מסוימים, הנטב ישתמש כשרת ה-DNS. כלומר, המחשב ישלח שאלות DNS אל הנטב, שבתו问他 ישאל את השירות של הספקית.

? כיצד המחשב שלנו יודע לפנות אל שרת DNS?

עכשו כשהמחשב יודע שעליו לפנות אל שרת DNS ושלוח אליו שאלת DNS, איך הוא יוכל לעשות זאת?
אם הוא יפנה אל שרת DNS ישירות? אם לא, אל מי הוא יעביר את החבילה?

בשלב זה, כפי שלמדנו בפרק [שכבת הקוו/למי נשלחת שאלת ARP?](#), המחשב בודק האם כתובות ה-IP של שרת DNS נמצאת אליו באותו ה-Subnet. כזכור, כתובת ה-IP של המחשב שלנו הינה: 5.5.0.2, ומסכת הרשת היא: 255.255.0.0. כפי שלמדנו בפרק [שכבת הרשת/מהו מזזה הרשת שלי? מהו מזזה הישות?](#), משמעותה של מסכת רשת זו היא שני הביטים הראשונים שייכים למזזה הרשת:



כתובת ה-IP של שרת DNS, אותה מצאנו קודם, היא 2.2.2.2. מכיוון שני הביטים הראשונים של כתובות זו הם 2.2 ולא 5.5, הרי ששרת DNS לא נמצא באותו ה-Subnet של המחשב.

יותה שהכתובת 2.2.2.2 לא נמצאת באותו ה-Subnet כמו זו של המחשב, מערכת הפעלה מבינה שעלייה לפנות אל ה-**Default Gateway**, אותו הנטב שיאפשר למידע לצאת מהרשת המקומית אל האינטרנט. גם את הכתובת של נתב זה מצאנו קודם לכן, בתהליך DHCP.

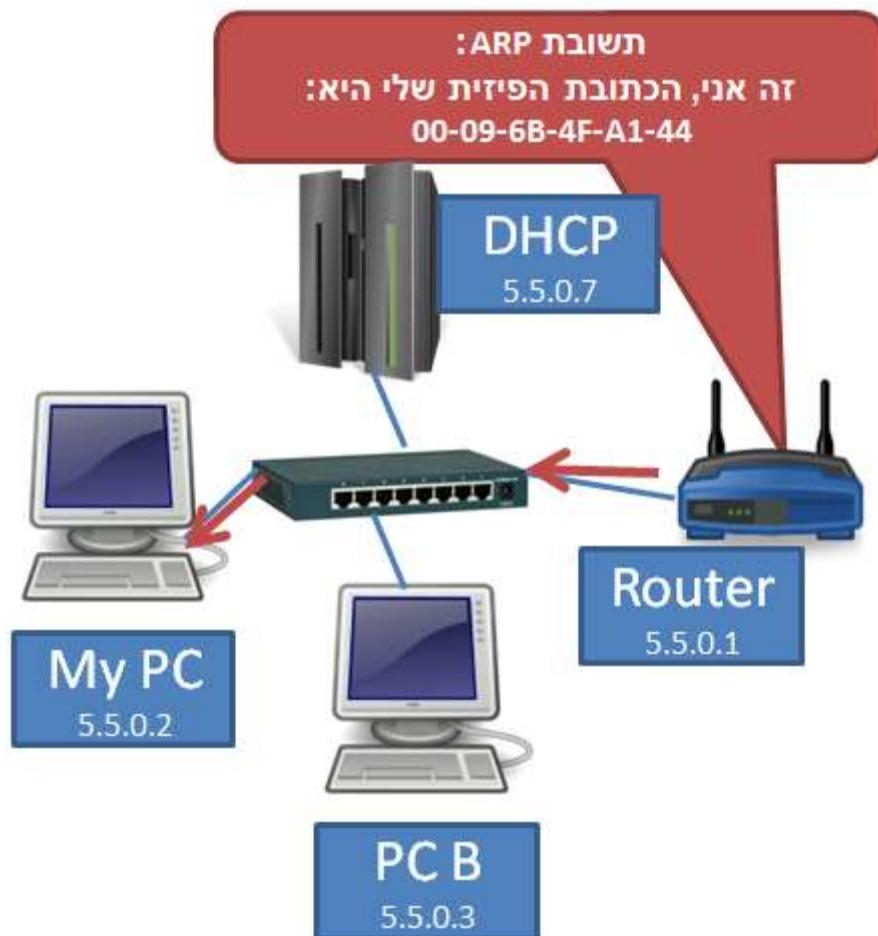
?

איך נצליח לתקשר עם הנטב?

הבנו שאנו צריכים לשלוח את חבילת DNS אל שרת-h-SNS, שכותבו 2.2.2.2. אנו גם יודעים שעליינו להעביר ראשית את החבילה אל הנטב, שכותבו 5.5.0.1. אך מידע זה אינו מספיק. מכיוון שכותבת IP היא כתובת לוגית, וכרטיס הרשות שלנו מכיר כתובות פיזיות בלבד - נדרש לגלוות את **הכתובת הפיזית** של הנטב. כפי שלמדנו בפרק [שכבה ה-2/ פרוטוקול ARP](#), תהילך זה מבצע באמצעות **פרוטוקול ARP**. בהנחה שאין רשותה עבור הנטב ב-ARP Cache של המחשב שלנו, המחשב ישלח ב-Broadcast הודעה לכל הרשות: "מי יש את הכתובת הפיזית עבור הכתובת הלוגית 5.5.0.1?" שאלת זו נקראת ARP Request:



בשלב זה, המטב רואה את ה-ARP Request, ומגיב למחשב שלו בתשובה שנקראת ARP Reply:



כעת למחשב יש את כל המידע הדרוש על מנת לשЛОח חבילת אל שרת ה-DNS! אך קודם נמשיך להתעסך בהודעה זו, ישנה שאלה קודמת עליה אנו צריכים לענות:



איך ה-Switch יודע להעביר את ההודעות?

הודעת ה-ARP Reply נשלחה מהמטב, הגיעה אל ה-Switch (מtag), שידע להעביר אותה אך ורק אל המחשב שלנו. כיצד הוא עשה זאת?

כפי שלמדנו בפרק [שכבות הקוקו/ כיצד Switch פועל?](#), ל-Switch יש טבלה אותה הוא מלא בזמן ריצה. טבלה זו ממפה בין כתובת MAC לבין הפורט הפיזי הרלוונטי. כאשר ה-Switch חובר לראשונה, הטעינה הייתה ריקה:

MAC Address	Port

בפעם הראשונה בה המחשב שלנו שלח מסגרת כלשהי, למשל את חבילת DHCP Discover, ה-Switch יראה את כתובת המקור של המסגרת, ושייר אותה לפורט הפיזי אליו מחובר המחשב:

MAC Address	Port
My PC	1

עכשו, כאשר הנטב שולח את ההודעה, ה-Switch בדק בטבלה שלו, וראה שהוא מיועד לכתובת MAC של המחשב שלנו, ומכאן שהוא מיועד לפורט הפיזי שלו. כך ידע ה-Switch למתג את ARP Reply אך ורק אל המחשב שלנו.

עת נוכל להמשיך עם החבילה שברצוננו לשלוח אל שרת DNS.

מahan הכתובות בחבילה?

בטרם נתעכט על שכבת DNS ודרך הפעולה שלה, علينا להבין כיצד נראה חבילה שנשלחת אל שרת DNS- באופן כללי. ננסה להשלים את השדות הבאים של הפקטה:

	כתובת MAC מקור
	כתובת MAC יעד
	כתובת IP מקור
	כתובת IP יעד



טרם תמשיכו בקריאה, נסו להשלים את הטללה בעצמכם.

כתובות MAC

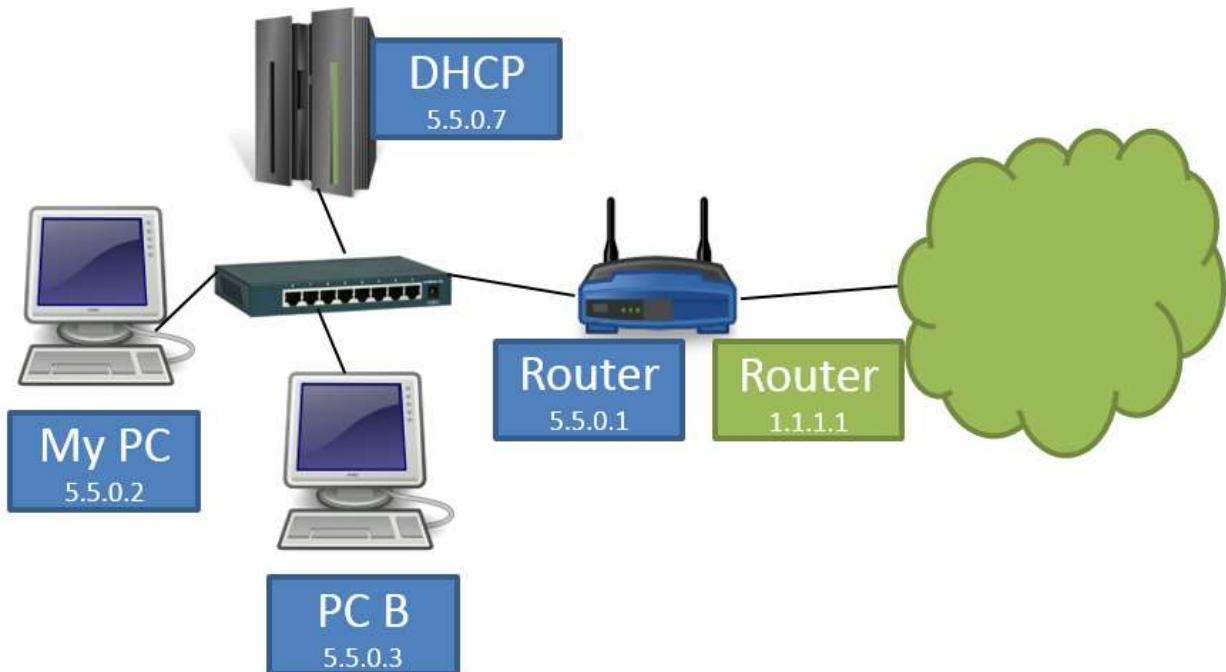
הכתובת הפיזית של המקור שייכת לכרטיס הרשת של המחשב שלנו - הרי הוא זה ששולח את החבילה. המחשב יודע את הכתובת זו שכן היא שייכת לכרטיס הרשת שלו עצמו.

הכתובת הפיזית של היעד תהיה של הנטב, שכן הוא התחנה הקורובה בדרך אל היעד. כאמור, כתובות פיזיות שייכות לשכבה הנקו ומצוינות את התחנה הקורובה בכל פעם. את הכתובת הפיזית של הנטב השגנו קודם לכן באמצעות פרוטוקול ARP.

נמלא את השדות הרלוונטיים בטבלה:

המחשב שלנו (כתובת ידועה)	כתובת MAC מקור
הנטב (הושגה באמצעות ARP)	כתובת MAC יעד
	כתובת IP מקור
	כתובת IP יעד

שימוש לב Ci כתובת MAC של הנטב משוייך לפורט הפיזי שמחובר אל ה-Switch ברשת שלנו, ולא לפורט פיזי אחר. לכל פорт פיזי של הנטב יש כתובת MAC משלו. אם נביט בשרטוט הרשת:



נראה כי לנטב יש שני פורטים פיזיים: הפורט הראשון מחבר אותו אל הרשת הביתית שלנו. עבור פורט זה, כתובות ה-IP היא: 192.168.0.1. לפורט זה יש כתובות MAC מסוימות. הפורט השני מ לחבר את הנטב אל הרשת של הספקית, המוצגת כענן יירוק. עבור פורט זה, כתובות ה-IP היא: 1.1.1.1. לפורט זה יש כתובות MAC שונה מהכתובות של הפורט הפיזי שמחבר את הנטב לרשת שלנו. בהודעה שהמחשב ישלח, כתובות ה-IP בדוחה כתובות היעד תהיה הכתובת של הפורט הפיזי המחבר לרשת שלנו.

כתובות ה-IP

כתובת ה-IP של המקור תהא אף היא של המחשב שלנו, זאת מכיוון שאנו שולחים את החבילה. את כתובות זו השגנו באמצעות תהליך ה-DHCP.

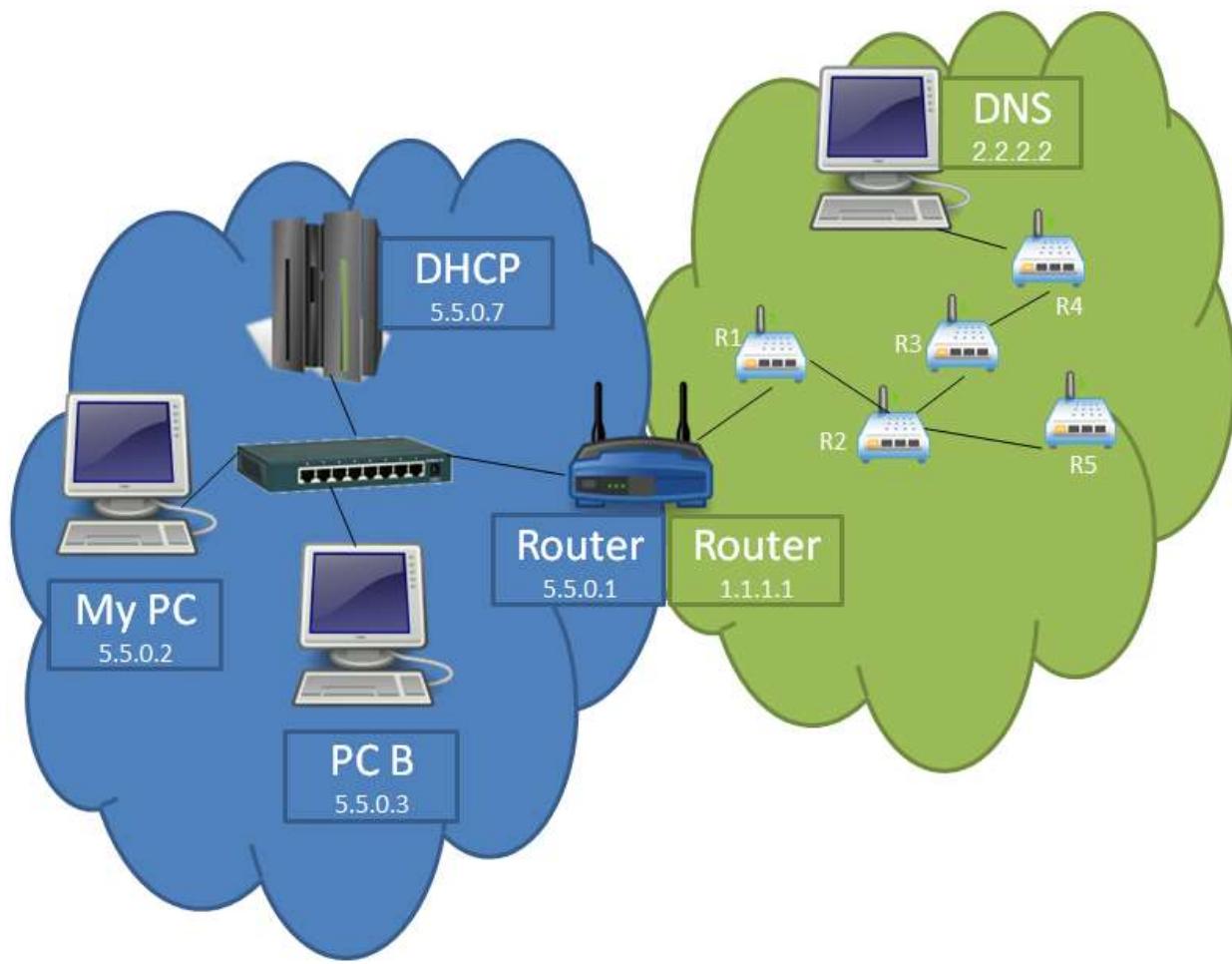
כתובת ה-IP של היעד תהיה הכתובת של שרת DNS. זאת מכיוון שבשכבה הרשת, כתובות היעד מצביעה על היעד הסופי - אל מי החבילה אמורה להגיע בסופו של דבר. את הכתובת של שרת DNS גילינו קודם לכן באמצעות תהליך ה-DHCP. הטבלה המלała תיראה כך:

המחשב שלנו (כתובת ידועה)	כתובת MAC מקור
הנטב שלנו (הושגה באמצעות ARP)	כתובת MAC יעד
המחשב שלנו (הושגה באמצעות DHCP)	כתובת IP מקור
שרת DNS (הושגה באמצעות DHCP)	כתובת IP יעד

נדגיש כי בטבלה זו נראה באופן ברור ההבדל בין השכבה השנייה לשכבה השלישית. בעוד שכבת ה-IP מצינית את הכתובות של ה-Hop הנוכחי, ככלומר שלב אחד בדרך (ומייצגת בידי כתובות MAC בשתי השורות הראשונות של הטבלה), שכבת הרשת מצינית את המקור והיעד הסופיים של החבילה (ומייצגת בידי כתובות ה-IP בשתי השורות התחתונות של הטבלה).

מהן הכתובות בתחנה הבאה?

כאשר הנטב מקבל את החבילה, ועביר אותה להלאה, כיצד תיראה הכתובות?
נסתכל על תמונה הרשת שלנו, שהתרחבה מעט.icut מופיע גם שרת DNS, שהוא חלק מהרשת של ספקית האינטרנט שלנו. בנוסף, מופיעים נתבים נוספים השייכים לספקית האינטרנט:



לצורך הבהרה, הרשת המקומית (LAN) שלנו סומנה בכחול, והרשות של הספקית סומנה בירוק. שימו לב שהנתב שלנו נמצא בשתי הרשתות, ויש לו שתי כתובות IP - אחת "פנימית", שהיא הכתובת 5.5.0.1, המשמשת אותו ברשת הביתית שלנו, והשנייה "חיצונית", שהיא הכתובת 1.1.1.1 ומשמשת אותו אל מול הספקית בפרט והאינטרנט בכלל.

נאמר שהנתב החליט להעביר את החבילה המיועדת אל שרת DNS אל הנתב R1. אילו כתובות יהיו עכשו בפקטה?

נסו למלא בעצמכם את הטבלה הבאה:



	כתובת MAC מקור
	כתובת MAC יעד
	כתובת IP מקור
	כתובת IP יעד

כתובות ה-MAC

כתובת MAC של המקור תהיה בעצם הכתובת של הנטב שלנו. זאת מכיוון שהוא זה ששלוח את החבילה - כולם כרטיס הרשת שלו הוא זה שעביר את החבילה הלאה. שימוש לב Ci כתובת MAC שייכת לפורט הפיזי של הנטב שנמצא ברשות של הספקית (הרשות הירוקה באירן לעיל), שהיא שונה מכתובת MAC שייכת לפורט הפיזי של הנטב ברשות המקומית שלנו.

כתובת MAC של היעד תהיה של הנטב R1, באופן ספציפי זו של הפורט הפיזי שמחובר אל הנטב של הרשות הבינלאומית שלו (ולא הפורט הפיזי שמחובר אל הנטב R2). הכתובת תהיה של הנטב R1 מכיוון שהתחנה הבאה של החבילה היא הנטב R2, וצורך השכבה השנייה אחראית לתקשורת בין תחנות הסמוכות זו לזו. נמלא את השדות הרלבנטיים בטבלה:

הנטב שלנו	כתובת MAC מקור
R1	כתובת MAC יעד
	כתובת IP מקור
	כתובת IP יעד

כתובות ה-IP

כתובת המקור תהיה הכתובת של המחשב שלנו. זאת מכיוון שהוא שלח את הודעה המקורית, ובשכבה השלישית אנו מצינימ את המקור והיעד הסופיים של החבילה. מסיבה זו, כתובת IP היעד תהא זו של שרת DNS.

הטבלה המלאה תיראה כך:

הנטב שלנו	כתובת MAC מקור
R1	כתובת MAC יעד
המחשב שלנו	כתובת IP מקור
שרת DNS	כתובת IP יעד

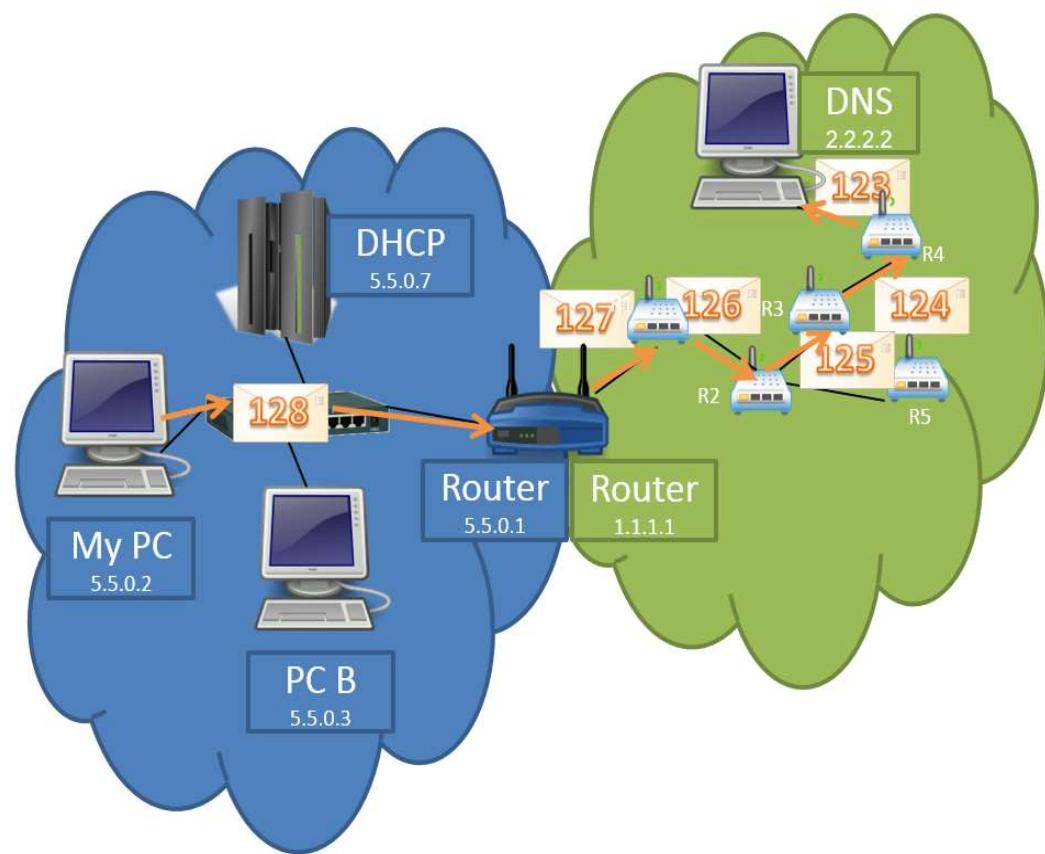
למעשה, כתובות ה-IP נותרו ללא שינוי! כך ניתן לראות שבעוד כתובות ה-MAC משתנות בכל Hop בדרך ומציניות מה השלב הנוכחי שהחbillה עוברת, כתובות ה-IP נשארות קבועות לאורך המשלוח ומציניות ממי החbillה הגיעו במקור ומה יעדה הסופי.

מה עושה כל נתב עם החbillה?

מה עוד עושה כל נתב בדרך, לפני שהוא מעביר להלאה את החbillה שהוא קיבל? מה יעשה R1 כאשר יקבל את החbillה? ומה יעשה R2?

כאשר הנתב מקבל את החbillה, עליו לcheck את ה-**Checksum** ולזוזה שהוא תקין. לאחר מכן, הוא מחסיר 1 מערך ה-TTL (Time To Live) של החbillה, כפי שלמדנו בפרק שכבות הרשת / איך Traceroute עובד?. בעקבות החסירה זו, עליו לcheck את ערך ה-**Checksum** מחדש בטרם יעביר את החbillה להלאה.

נאמר שהמחשב שולח את החbillה עם ערך TTL התחלתי של 128. נעקוב אחר ערך TTL לאורך המסלול (מושג בתווך המספר של החbillה):



כפי שניתן לראות, החבילה מגיעה אל שרת DNS כשער TTL הוא 123, כיוון שהוא חמשה נתבים בדרך. שימוש לב שהחbillה הגיעה אל הנטב הקרוב אל המחשב שלנו דרך Switch, שלא שינה את ערך TTL רק רכיבי שכבת הרשת, כגון נתבים, משנים את ערך זה, ולא רכיבי שכבת הקו לMININUM.

כמו כן, הנטב צריך **לנתב את הפקטה**, כלומר להחליט מה המסלול שעלייה לעבור. כפי שלמדנו בפרק [שכבת הרשת/יתוב](#), הנטב מבין לאן עליו להעביר את החbillה באמצעות טבלאות יתוב נשמרות אצלן, ובננות באופן דינامي.



כיצד מוצא המחשב את כתובת ה-IP של Facebook?

עת, סוף סוף, לאחר שהבנו כיצד ניתן להעביר חבילה אל שרת DNS, נוכל לחזור ולדון במה החbillה הזאת כוללת. כפי שלמדנו בפרק [שכבת האפליקציה/התבוננות בשאלתת DNS](#), החbillה הנשלחת תהיה חבילת שאלתה (Query). כפי שלמדנו, שאלות ותשובות DNS מורכבות מרשות (Resource Records (RR)). השאלתא שישלח המחשב שלנו צפוייה להיות מסוג A, כלומר תרגום בין שם דומין לכתובת IP. החbillה מכילה Transaction ID מסויים. לצורך הדוגמא, נאמר שה-ID Transaction הינו 1337. כאשר שרת DNS יענה על השאלתא, גם התשובה תכלול את הערך 1337 בשדה ה-ID Transaction. כמו כן, חבילת התשובה תכלול את השאלתא המקורי של המחשב שלנו, והן רשומות תשובה אחת או יותר.



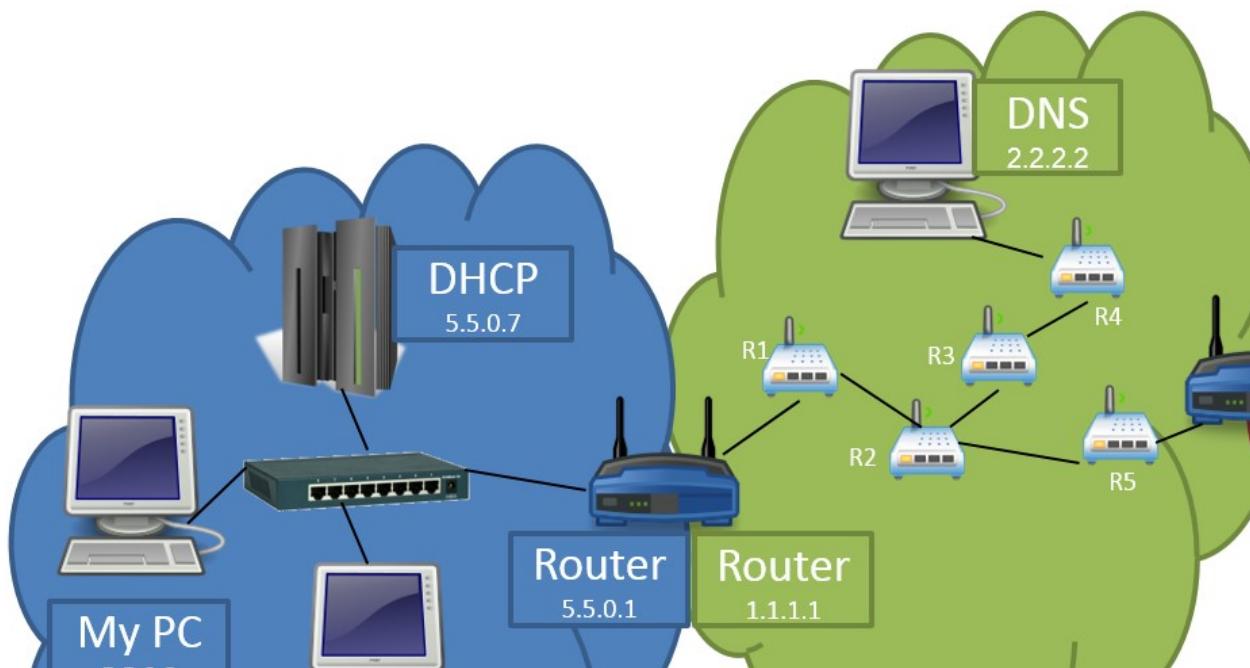
מה השלב הבא?

לאחר שמצאנו את כתובת ה-IP של Facebook, הגיע הזמן לגשת אליו. מכיוון שבממשק אנו עתידים להשתמש בפרוטוקול HTTP, علينا ראשית להרים **קישור TCP** עם השרת של Facebook.

שימוש לב שתמונה הרשות שלנו כבר גדרה מאד, ועברנו מהשלב בו הסתכלנו על מחשב יחיד:



אל שלב בו אנו רואים את הרשת המקומית, הרשת של הספקית וכן האינטרנט:



שים לב כי באיר האינטראנט אמנים נראה כמו ספקית יחידה, אך הוא מכיל הרבה ספקיות. על מנת לפחות את האיר, שלא כולל בתוכו אזוריים לכל הספקיות שנמצאות בתוכו.

בשלב זה, תמונה הרשת כבר גודלה למדי, ומכליה את המחשב שלנו, רכיבי רשת כגון Switch ונתבים, וכן שרתים. עם זאת, כפי שלמדנו בפרק שכבה התעבורה / מיקום שכבה התעבורה במודל השכבות, שכבת הרשת מספקת לשכבה התעבורה מודל של "ענן", ובכך מבטלת את הצורך שלה להכיר את מבנה הרשת. אי לכך, נוכל להציג את התמונה גם בצורה שבה שכבה התעבורה "רואה" אותה, כך:



?

בailo פורטים תבצעו התקשרות?

כפי שלמדנו בפרק [שכבה התעבורה/ ריבוב אפליקציות - פורטיפם](#), תקשורת בשכבה זו בכלל, ובפרוטוקול TCP בפרט, מתבצעת בין מזהי כתובות IP ופורטים. מה יהיה הפורט במחשב שלנו, ומה הפורט במחשב היעד?

הפורט שאלוי תבוצע הפניה, ככלומר הפורט המאזין בשרת של Facebook, יהיה הכל הנראה פорт 80. פорт זה הינו הפורט המשמש בדרך כלל ל프וטוקול HTTP. **שים לב:** אין הדבר אומר שלא ניתן לתקשר ב-HTTP מעל מספר פорт אחר, אך בדרך כלל שירותי יזינו לבקשת HTTP בפורט זה.

הפורט ממנו תבוצע הפניה, ככלומר הפורט במחשב שלנו, יהיה מספר רנדומלי שתגריל מערכת הפעלה. עם זאת, מספר זה לא יהיה רנדומלי לחוטין, מכיוון שישנם מספרי פורטים השמורים לאפליקציות מסוימות. אי לכך, רוב מערכות הפעלה מגירות מספר פорт בטוח שבין 49,152 וביין 65,535. נאמר שהפורט שהוגדר הינו 60,124. מכאן שהתקשרות תיראה כך:



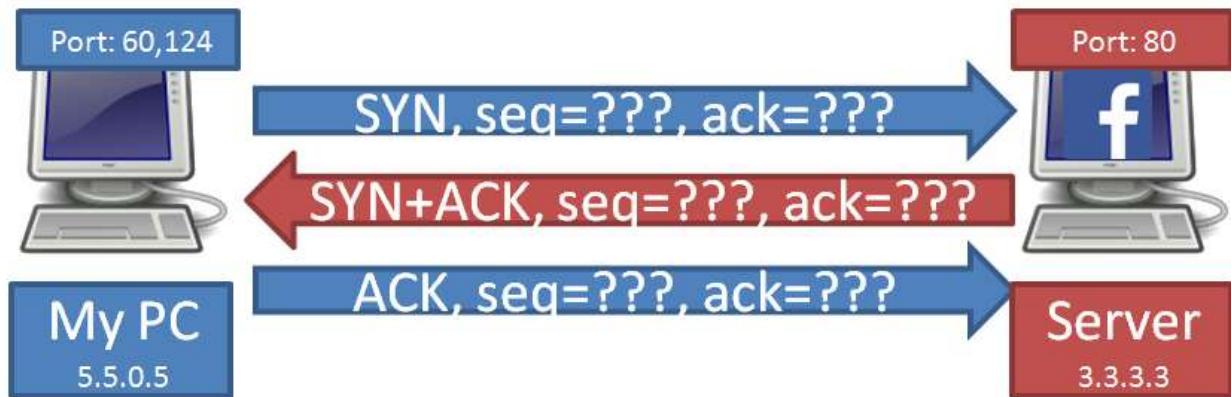
?

כיצד נראהיה הרמת הקישור?

על מנת להרים קישור TCP בין המחשב שלנו לבין השרת המרוחק, נשתמש ב-**Three Way Handshake**, כפי שלמדנו בפרק [שכבה התעבורה/ הרמת קישור בTCP](#). היות שככבה התעבורה אינה מודעת למבנה הרשת, נסתמך על מודל ה"ען" שמספקת שכבה הרשת ונתיחה לתקשורת כאילו היא מתבצעת באופן ישיר בין המחשב שלנו לבין השרת של Facebook.



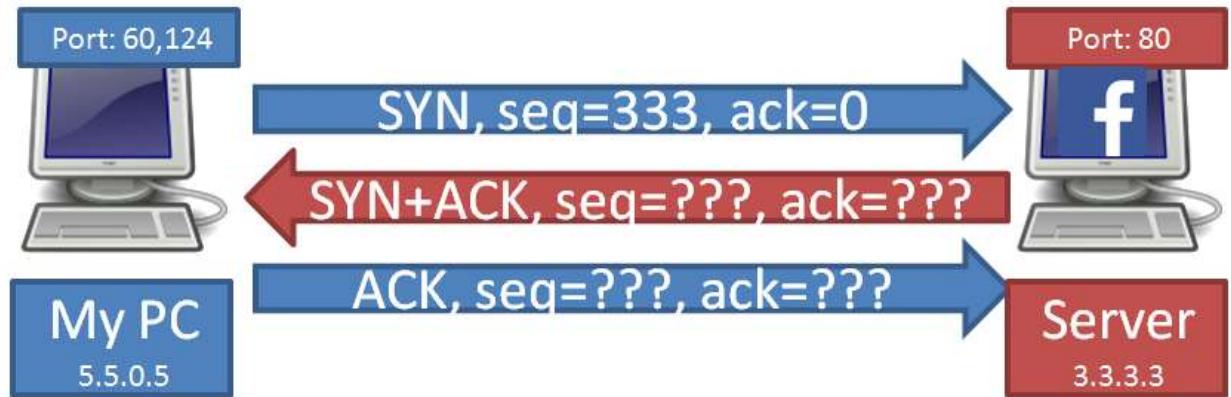
נסו להשלים בעצמכם את העריכים שבתרשים הבא, במקום סימני השאלה:



נתחיל מהחטלה הראשונה, היא חטלה SYN. זו החטלה המציינת את תחילת הקישור, ולכן הדגל SYN בה דלוק. הדגל ACK כבוי, מכיוון שזו החטלה הראשונה בקישור, ולא מתבצע אישור על קבלה של מידע קודם.

ערך ה-Sequence Number של חטלה זו הינו ערך ה-Sequence Number ההתחלתי של הקישור, ועל כן הוא יהיה רנדומלי, כפי שלמדנו בפרק שכבת התעבורה/[חטלה ראשונה SYN](#). לצורך הדוגמה, נאמר שהערך שהוגרל הוא 333.

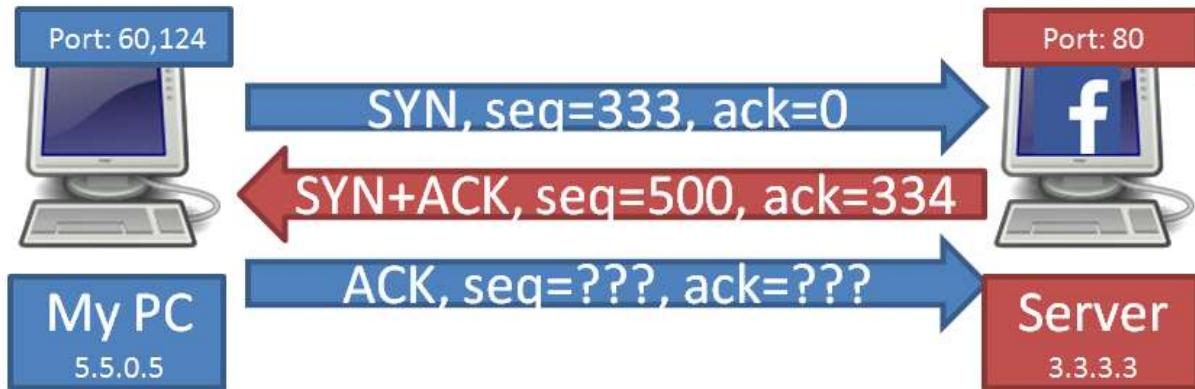
ערך ה-Acknowledgement Number של חטלה זו יהיה 0, זאת מכיוון שדגל ה-ACK כבוי. נוכל למלא את העריכים האלו בשרטוט:



cut עבור לחטלה השנייה. בחטלה זו, דגל SYN דלוק היota שמדובר בחטלה הראשונה מצד של הקישור שבין השרת למחשב שלנו. דגל ACK דלוק שכן יש לתת אישור על הגעת חטלה SYN מהלkers.

שדה ה-Sequence Number של חטלה זו יהיה אף הוא רנדומלי, מכיוון שהוא מציין את ערך ה-Sequence Number ההתחלתי של רצף המידע שעובר בין השרת לבין המחשב שלנו. לצורך הדוגמה, נאמר שהערך שהוגרל הוא 500.

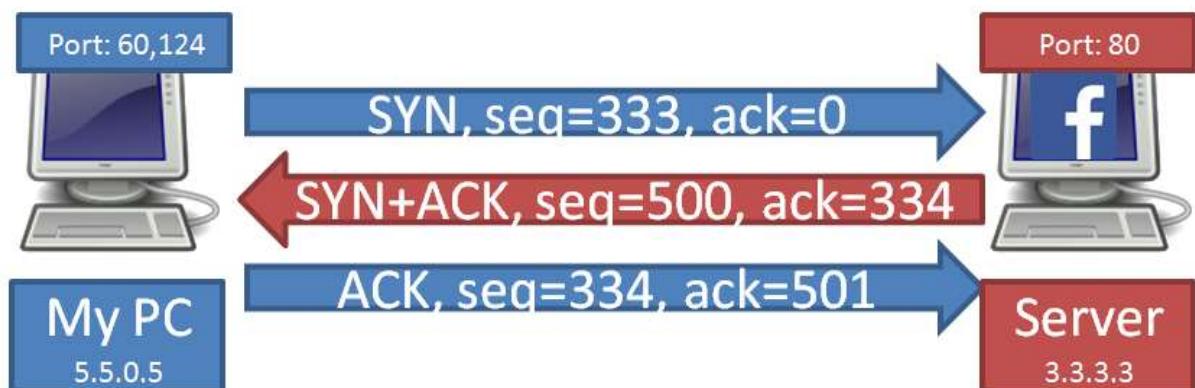
שדה ה-Acknowledgement Number-Amor להuid על כר שהחbillת ה-SYN התקבלה. כפי שלמדנו בפרק [שכבות התעבורה/Air TCP משתמש בסכבות התעבורה/Acknowledgement Numbers](#), ערך שדה זה מחושב באמצעות ערך Sequence Number של החbillת שהתקבלה (333), בנוסף לאורך המידע המועבר בה. היות שהחbillת שהתקבלה הינה חbillת SYN, גודל המידע שמחושב עבורה הוא גודל של בית (byte) אחד. אי לכר, הערך יהיה .334, כלומר $333+1$.



נותרנו עם החbillת השלישית והאחרונה לתהיליך הרמת הקישור. בחbillת זו, דגל ה-SYN כבוי, מכיוון שהוא מודיעה המעידת על יצירת הקישור. דגל ה-ACK דולק, שכן יש לאשר את קבלת החbillת הקודמת שנשלחה מהשרת.

מה יהיה ערך ה-Sequence Number? כפי שלמדנו בפרק שכבות התעבורה, ערך ה-Sequence Number זהה לערך ה-Acknowledgement Number של החbillת הקודמת (בහנחה שלא נשלחו עוד חbillות ביןתיים). אי לכר, ערך שדה זה יהיה .334.

כמו שציינו קודם לכן, שדה ה-Acknowledgement Number-Amor להuid על כר שהחbillת הקודמת התקבלה, ומוחושב באמצעות ערך ה-Sequence Number של החbillת שהתקבלה (500), בנוסף לאורך המידע, שהוא בית (bytes) אחד במקרה של חbillת SYN. אי לכר, הערך יהיה $500+1$, כלומר 501.



איך נראה בקשת ה-HTTP?

הצלחנו להרים קישור TCP. עכשו, באמצעותו, נוכל סוף סוף לבקש מ-Facebook לשלוח אלינו את העמוד הראשי שלו.

כפי שמלמדנו בפרק [שכבות האפליקציה / פרוטוקול - HTTP בקשה ותגובה](#), כאשר דפדף פונה לאתר כלשהו, הוא פונה באמצעות בקשת GET. מכיוון שהפניה מתבצעת אל העמוד הראשי של Facebook, מוביל לבקשת אפ' משאב ספציפי, היא תבצע אל המשאב שנמצא בכתובת "/". [זכור מפרק שכבות האפליקציה / מבנה פרטומי של בקשות](#), אחרי המילה GET תופיע המחרוזת "HTTP" והגירה של הפרוטוקול, למשל: 1.0. לאחר מכן יופיעו ה-HTTP Headers, עליהם לא נעמיק בפרק זה.



איך נראה תשובה ה-HTTP?

בנהנча ו-Facebook מוכן להחזיר את העמוד הראשי שלו ללא עיכובים נוספים, הוא יענה בתשובה HTTP מסווג (OK). כפי שראינו בפרק [שכבות האפליקציה / מבנה פרטומי של תשובה](#), התשובה מתחילה במחרוזת "HTTP" והגירה של הפרוטוקול. מיד אחר כך, יופיע הקוד של התגובה (200) ואז הפירוש הטקסטואלי של הקוד (OK). לאחר מכן, יופיעו כל ה-Headers הרכביים, ולבסוף - המידע עצמו.



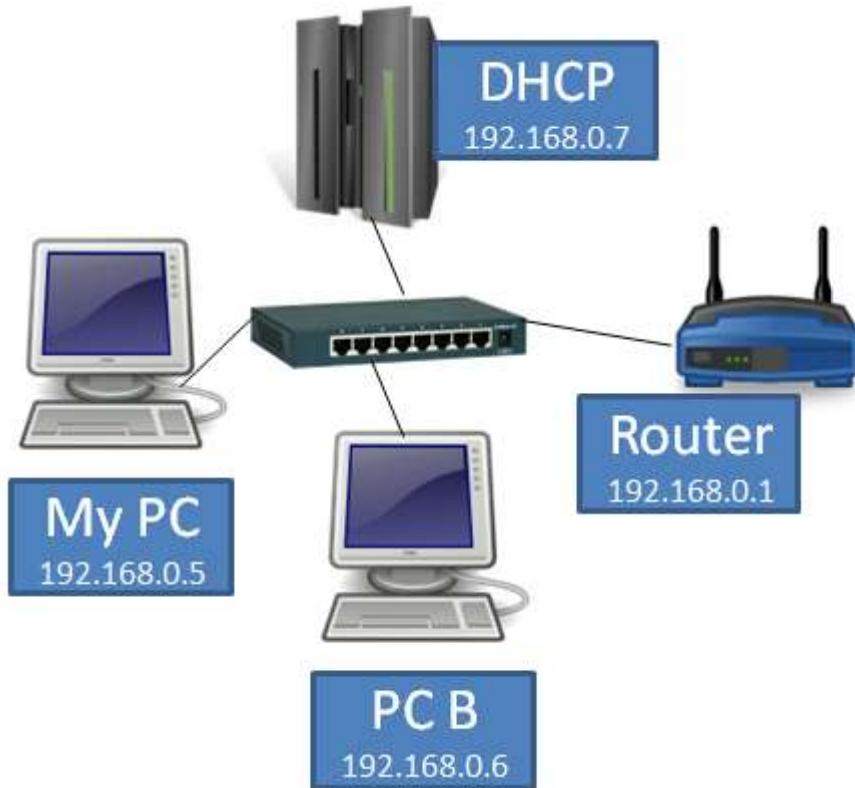
בתשובה זו הגענו לסוף התהילה, והדפדן שלנו יוכל סוף סוף לראות את העמוד הראשי של Facebook.

מה קורה כאשר המחשב שלנו נמצא מאחורי NAT?

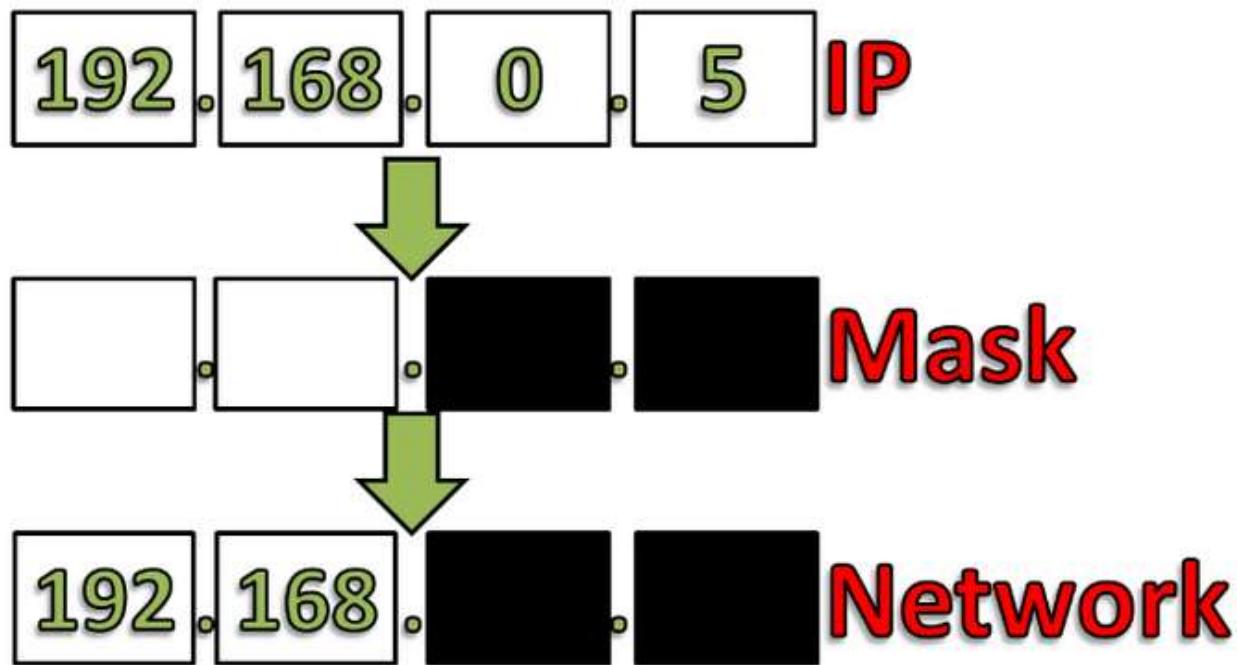


בפרק זה, עברנו על היבטים רבים הנוגעים לתהילה שקרה כאשר אנו גולשים אל Facebook. עם זאת, על אף שהמקרה שהציגו אפשרי, הוא לא המקרה הרווח ברשת האינטרנט, שכן הוא מתעלם מהשימוש בכתובות פרטיות -NAT, אותן למדנו להכיר בפרק [שכבות הרשת](#). בעת נתאר את השימוש המתחרשים כאשר המחשב שלנו נמצא מאחורי NAT. שימוש לב שנטמך בבדלים בלבד, ולא נחזור על התהילה כולה.

כפי שלמדנו בפרק [שכבות הרשת](#), הכתובת שהמחשב שלנו מקבל משרת DHCP שלו צפוייה להיות **כתובת IP פרטית**, וזאת בכך שהיא נמצאת בכתובות IP בעולם. במקרה שלנו, נאמר שקיבלנו את הכתובת: 192.168.0.5. מס'icit הרשת היא: 255.255.0.0. בנוסף לכך, על מנת לקבל **כתובת IP חיונית**, הנטב שלנו עובר תהליך דומה. כאשר הנטב מתקשר עם שרת DHCP של ספקית האינטרנט, הוא מקבל ממנו כתובת IP שאינה פרטית. נאמר שהנטב קיבל את הכתובת: 1.1.1.1, ומס'icit הרשת היא: 255.255.0.0. תמונה הรสת עשויה להיראות כך:

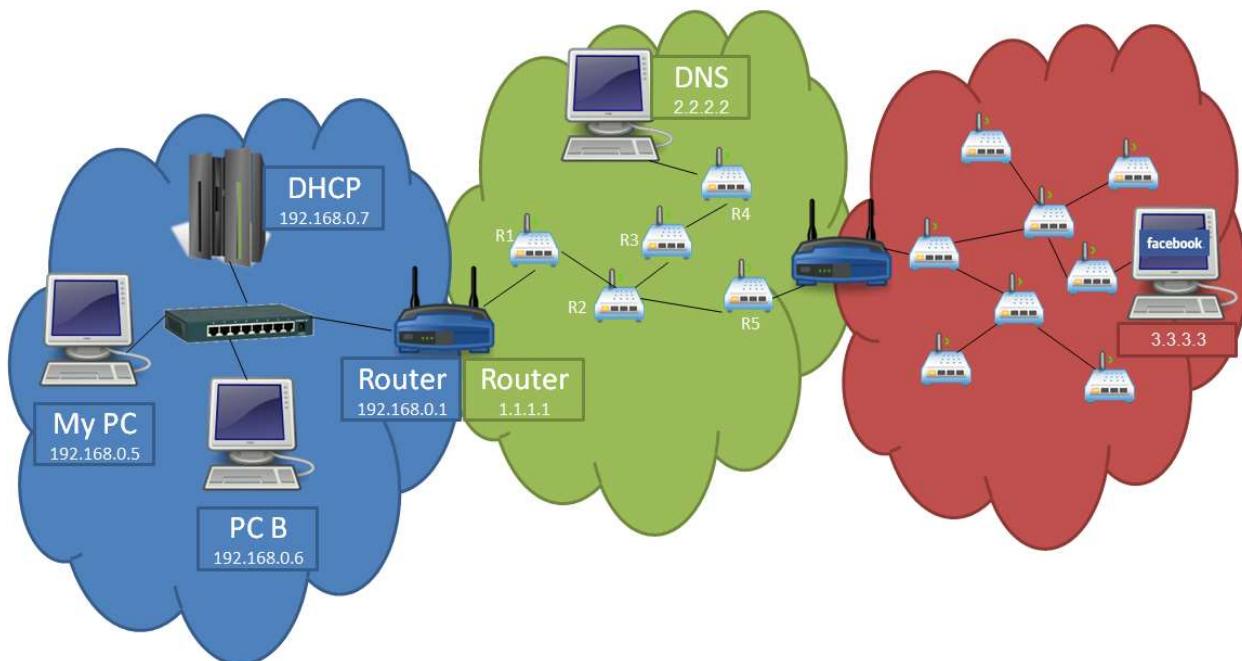


גם במקרה זה, כאשר המחשב ירצה לגשת אל שירות DNS, עליו לבדוק האם השרת נמצא באותה הרשת כשלו. לשם כך, המחשב יבצע בדיקה על מסכת הרשת שלו:



כתובת-ה-IP של שירות DNS, אותה מצאנו קודם, היא 2.2.2.2. מכיוון שני הביטים הראשונים של כתובת זו הם 2.2 ולא 192.168, הרי שירות DNS לא נמצא באותו subnet של המחשב.

המשך התהילה דומה מאוד לתהילה ללא שימוש ב-NAT, ולכн לא נהנית עליו כאן. עם זאת, עלינו להבין את ההבדל הנובע מהשימוש בכתובות פרטיות. תמונה הראית כר:



שיםו לב של נתב שלנו יש שתי כתובות IP - אחת "פנימית", שהיא הכתובת הפרטית 192.168.0.1 המשמשת אותנו ברשת הביתה שלנו, והשנייה "חיצונית", שהיא הכתובת 1.1.1.1 המשמשת אותו אל מול הספקית בפרט והאינטרנט בכלל.

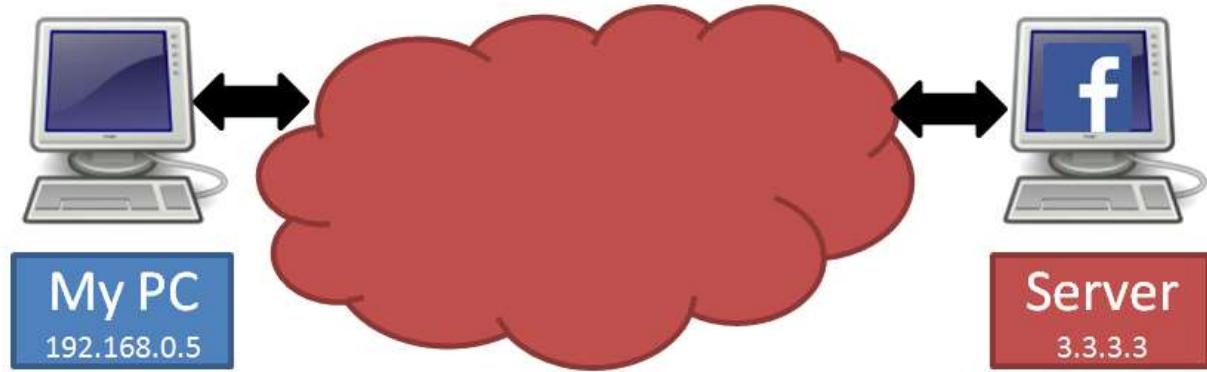
כאשר המחשב שלנו רוצה לתקשר עם Facebook, הוא לא יכול לעשות זאת ישירות. תקשורת שכזו אינה אפשרית, מכיוון שאם Facebook ינסה לענות אל המחשב שלנו, הוא לא יוכל לשולח אליו חビלה - הרי שהכתובת 192.168.0.5 הינה כתובת פרטית, והנתבים שבדרך לא יכולים לנתר אליה חבילות.

כאן נכנס לפוליה ה-NAT. כאשר המחשב שלנו ישלח את החビלה, הוא ישלח אותה אל השרת של Facebook באופן רגיל. בעת, כאשר הנתר יקבל את החビלה, הוא יחליף את כתובת המקור של החビלה לכתובת שלו, כאשר החビלה מגיעה אל Facebook, היא תיראה כאילו היא הגיעה מהכתובת 1.1.1.1. נראה זאת בשרטוט הבא (שמוצג מנוקדת המבט של השרת של Facebook):



החビלה מגיעה בשלב זה אל השרת של Facebook, שלא מודע כלל לכך שהיא נשלחה במקור מהכתובת 192.168.0.5. מבחינתו, החビלה הגיעה פשוט מישות שנמצאת בכתובת 1.1.1.1. لكن, כאשר Facebook עונה בחבילת תשובה, הוא שולח אותה אל הכתובת 1.1.1.1. בשלב זה, כאשר הנתר מקבל את החビלה, הוא מחליף את כתובת היעד של החビלה, מהכתובת 1.1.1.1 אל הכתובת 192.168.0.5. לאחר החלפה זו, הוא מעביר אותה אל המחשב שלנו.

מכיוון שהנתר מחליף את הכתובות בטרם הוא מעביר את החビלה, המחשב שלנו כלל לא צריך להיות מודע לתהליך NAT, והוא "שׁקוף" עבורי. למעשה, עברו המחשב שלנו, התהליך נראה כאילו לא היה NAT בכלל. נראה זאת בשרטוט הבא (שמוצג מנוקדת המבט של המחשב שלנו):



בדרכ זו, ה-NAT מאפשר חסוך בכתובות IP מביי לగורם לשינוי הצד של לקוח הקצה - המחשב שלנו במקרה זהה.

איןנו מתעכבים בספר זה על דרכי שונות למשת NAT. עם זאת, אתם מוזמנים להרחב את הידע שלכם בנושא בעמוד: http://en.wikipedia.org/wiki/Network_address_translation

איך הכל מתחבר, איך עובד האינטרנט - סיכום

בפרק זה חזרנו לאותה השאלה ששאלנו בתחילת הספר - איך האינטרנט עובד? הפעם, מצוידים בכלים ידועים שרכשנו לאורך הספר, יכולים לענות על השאלה בצורה מעמיקה בהרבה מאשר בפרק הראשון.

התחלנו מהמחשב הבודד שלנו, והצלהנו לקבל **כתובת IP** ואת שאר פרטי הרשות באמצעות פרוטוקול **DHCP**. על מנת למצוא את כתובת ה-IP של Facebook, הבנו שאנו צריכים לפנות לשרת ה-**DNS**. בכך לعشות זאת, הסתכלנו על **מסכת הרשות (Subnet Mask)** שלו והבנו שהרת DNS לא נמצא איתנו באותה הרשת. על כן, פנינו אל **טבלת הניתוב** והבנו שעליו להעביר את החבילה אל ה-**Default Gateway** שלו.

על מנת לפנות אל האינטרנט, היינו צריכים לגלוות את הכתובת הפיזית שלו, ולשם כך השתמשנו ב프וטוקול **ARP**. לאחר שנזכרנו בפעולה של פרוטוקול זה, חזרנו על הדרך בה ה-**Switch** פועל, וכייד הוא יודע להעביר כל מסגרת רק אל הפורט הפיזי אליו היא מיועדת. לאחר מכן, הסתכלנו על הכתובות בשכבה השנייה ובשכבה השלישית שהיו בחבילה שנשלחת אל שרת DNS, וראינו איך הן משתנות לאורך המסלול. כמו כן, הזכרנו כיצד **נתב מטפל** בחבילה שגיעה אליו.

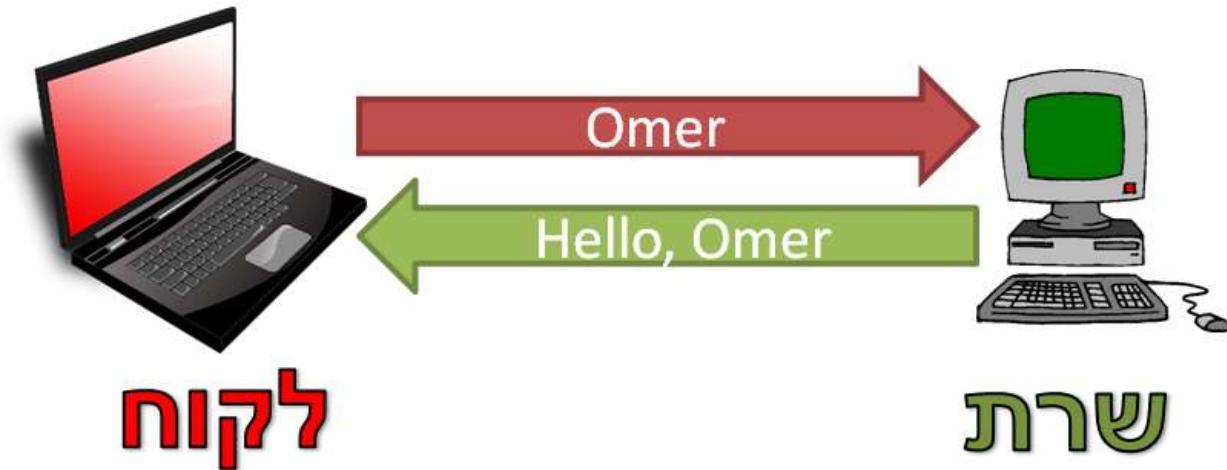
כשהבנו את הדרך שעשו החבילה באינטרנט, חזרנו על דרך הפעולה של פרוטוקול **DNS**, באמצעותו מצאנו את כתובת ה-IP של Facebook. בשלב זה, היינו צריכים להקים **קישור TCP** עם Facebook, ולכן הזכרנו כיצד נבחרים מספרי הפורטטים בהם נעשה שימוש. הקמנו את הקישור באמצעות **Three Way Handshake**, וחרנו על השדות **Acknowledgement Number**- **Sequence Number** על השדות **ACK** ו-**SEQ** של Facebook מעל קישור ה-TCP שהקמנו באמצעות פרוטוקול **HTTP**, שלחנו בקשה אל הרשת וקיבלו את התשובה.

באופן זה נגענו בנושאים רבים במהלך הפרק, ועבכנו דרך השכבות השונות של **מודל חמש השכבות**. בפרק זה הצלחנו לחבר ייחודי נושאים שונים שנסקרו לאורך כל הפרקים הקודמים, ולראות כיצד הם פועלים יחד בראשת האינטרנט. עם זאת, החסכנו נגיעה עמוקה בקונספטים רבים עליהם הרחכנו במהלך הספר. כמעט ולא נכנסנו לשדות ספציפיים של פרוטוקולים, והחסכנו התיחסות לנושאים חשובים בכל שכבה ולמבנה החבילות. אם תרצו, תוכלו לחזור אל הפרקים הרלבנטיים ולרענן את זכרונכם.

דריכנו עדין לא הסתיימה. לאחר שלמדנו את מודל חמש השכבות, והרחיבנו את הירעה על כל שכבה, ישנים נושאים מתקדמים נוספים עליהם נרחב בהמשך הספר.

פרק 12 - תכונות Sockets מתקדם: ריבוי משתמשים (הרחבה)

פרק תכונות ב-Socket למדנו מהו Socket, וכייזד לכתוב ללקוח ושרת. כתבנו מספר ללקוח ומספר שרתים, כगון שרת שמקבל מהלקוח את שמו ומחזיר לו תשובה בהתאם ([פרק תכונות ב-Sockets-תרגיל 2.3 מודרך](#) - [השרת הראשון שלו](#)):



כזכור, על מנת לטפל בבקשתו מלוקח נוסף, נאלצנו לסגור את החיבור עם הלוקח הראשון. כלומר, במקרה של לוקוח נוסף לשרת, השרת יהיה צריך לענות לבקשתו של הלוקח הראשון, ולאחר מכן לסיים את החיבור אליו. פועלה זה אמונם הגיונית עבור שירות שמציע שירות כה פשוט כגון מתן תשובה בהתאם לשם הלוקח, אך הוא לא הגיוני עבור שירות שנועד לספק שירותים למספר לקוחות בו-בזמן. חישבו למשל, עד כמה קשה היה להשתמש בפייסבוק במקרה שהשרת היה מסוגל לתת שירות רק ללקוח אחד בכל רגע נתון.



ובכן, מדובר אכן ממשקיעים פרק שלם בכך לדון בסוגיה זו? האם הפתרון לא כולל רק להוסיף לו לוקח נוסף?

הסוגיה הבועיתית הראשונה נובעת מה הצורך ליצור חיבור מול ללקוח חדש. להזכירם, כאשר רצינו לקבל חיבור מלוקח חדש בעת שמימשנו שרת שמתפל בבקשת אחת בכל פעם, השתמשנו במתודה `accept` בצורה הבאה: `(client_socket, client_address) = server_socket.accept()`

כפי שציינו בפרק תכנות ב-Sockets, המתודה **accept** הינה blocking - כלומר, הקוד "יקפא" ולא ימשיך לרווח עד אשר יתקבל בשרת חיבור חדש. מכאן שלאחר שקיבלנו חיבור מלוקה ראשון, علينا להחליט בין שתי אפשרויות:

- לטפל בבקשתו שmagiuot מהлокה הראשונית.
- לאפשר לлокה חדש להתחבר.

הסיבה לכך נועצה בעובדה, שלו מנת לאפשר לлокה חדש להתחבר, עליו לקרוא שוב למетодה **accept**, אשר עוצרת את ריצת התוכנית עד אשר יתחבר לлокה החדש.

סוגיה בעייתייה נוספת קשורה לקריאת מידע מלוקה קיימ. כאשר רצינו לקרוא מידע מלוקה בעת שמיימנו שרת שמתפל רק בבקשתה אחת, השתמשנו במетодה **recv**:

```
client_name = client_socket.recv(1024)
```

נזכיר כי גם המетодה **recv** הינה blocking - ולא מאפשרת המשך ריצת התוכנית עד אשר נקבל מידע מלוקות. מה יקרה במידה שננסה לקרוא מידע, אך הלוקה לא ישלח אלינו דבר?שוב, התוכנית תיתקע ולא יוכל לטפל בלוקות נוספים. אי לכך, כאשר נקרא ל-**recv**, לא יוכל לאפשר לлокה חדש להתחבר לשרת, או לחילופין - לקבל מידע מלוקה אחר המעניין לכתוב אליו.

הפתרון - **select**

אחת הדרכים¹⁰⁵ לפתרור את הבעיה אוטן הצגנו היא לשימוש בפונקציה **select**.

select מקבלת שלוש רשימות:

- רשימת Sockets מהם אול' נרצה לקרוא.
- רשימת Sockets אליהם אול' נרצה לכתוב.
- רשימת Sockets עבורים נרצה לבדוק מקרים של שגיאות. לצורך הפשטות, נתעלם מרגע מרשימה זו.

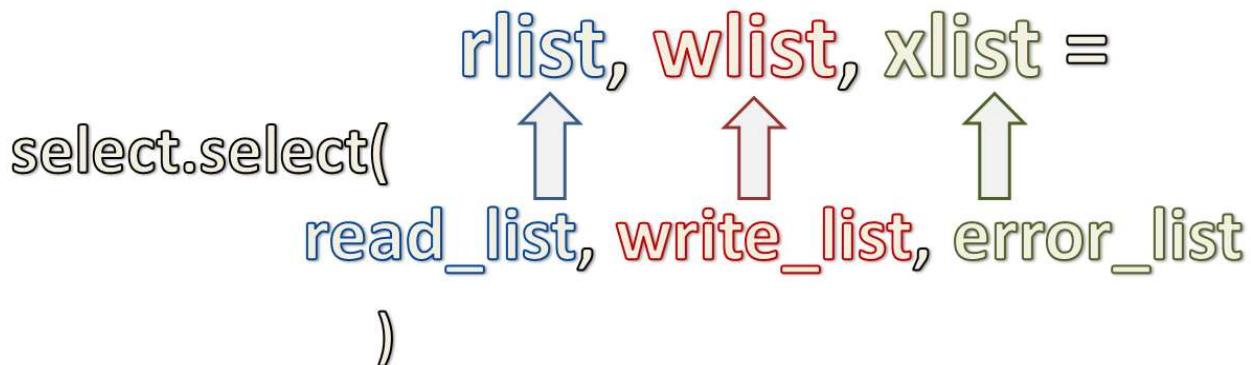
כל אובייקט **socket** יכול להיכנס לתוך אחת או יותר מהרשימות הללו.

לאחר ש-**select** מסיים את הריצה שלה, היא מחזירה שלוש רשימות:

- רשימת Sockets מהם ניתן כרגע לקרוא (באמצעות **recv**).
- רשימת Sockets אליהם ניתן כרגע לשלוח (באמצעות **send**).
- רשימת Sockets שזרקו שגיאה כלשהי.

¹⁰⁵ ישנן דרכים נוספות, כגון שימוש ב-**Threads**, אוטן לא נסקור בספר זה.

כל אחת מהרשימות הלו צוללת Sockets מתוך הרשימה התואמת שהעבರנו:



כך למשל, באם העברנו ברשימה-h Sockets מהם אול' נרצה לקרוא (בشرطוט לעיל: `read_list`) את האובייקטים `client_2_socket` ו-`client_1_socket`, יתכן שהfonקציה `select` תחזיר ברשימה-h Sockets מהם ניתן לקרוא (בشرطוט לעיל: `rlist`) את האובייקט `client_1_socket` בלבד. דבר זה מציין כי אנו יכולים לבצע `recv` על האובייקט `client_1_socket` מבלי "لتקווע" את שאר ריצת התוכנית, אך לא על `client_2_socket`.

תרגיל 12.1 מודרך - השירות מרובה המשתתפים הראשון שלו

בתרגיל זה נמשח שירות שמקבל חיבור מלוקו, קורא שם כלשהו מהלוקו, ומדפיס שם זה למסך. עם זאת, בנגד לשירות שמיישנו בפרק תכנות ב-Sockets, השירות יוכל לטפל במספר לקוחות במקביל. מעבר לכך, כל לקוח יוכל לשלוח בכל פעם שם אחר. השירות ידפיס למסך כל שם שהוא מקבל:



שימוש לב שהטיפול במקרה זה צריך להיות מקבילי - כמובן, השירות יוכל להשאיר את החיבור עם הלוקו הראשון פתוח בעודו מספק שירות ללקוח השני.

על מנת לעשות זאת, נתחילה בדרך דומה לזה שעשינו עד כה - ניצור אובייקט מסווג **socket**:

```
import socket
server_socket = socket.socket()
```

כעת, כפי שעשינו גם בשרתים הקודמים שמיימשו, נקרא למתודה **bind** על מנת לחבר את אובייקט ה-socket שיצרנו אל כתובת מקומית:

```
server_socket.bind(('0.0.0.0', 23))
```

כמו כן, נבצע **listen**:

```
server_socket.listen(5)
```

עד כה, הפעולות זהות לאלו שביצענו בשרת שmailto: בלקוקו אחד בכל פעם. כעת נתחל בשלבים שונים. ניצור רשימה שתכיל את כל אובייקטי ה-socket של הלוקוחות שיתחברו לשרת:

```
open_client_sockets = []
```

כך נראה הכל הקוד בינהי:

```
import select

server_socket = socket.socket()

server_socket.bind(('0.0.0.0', 23))

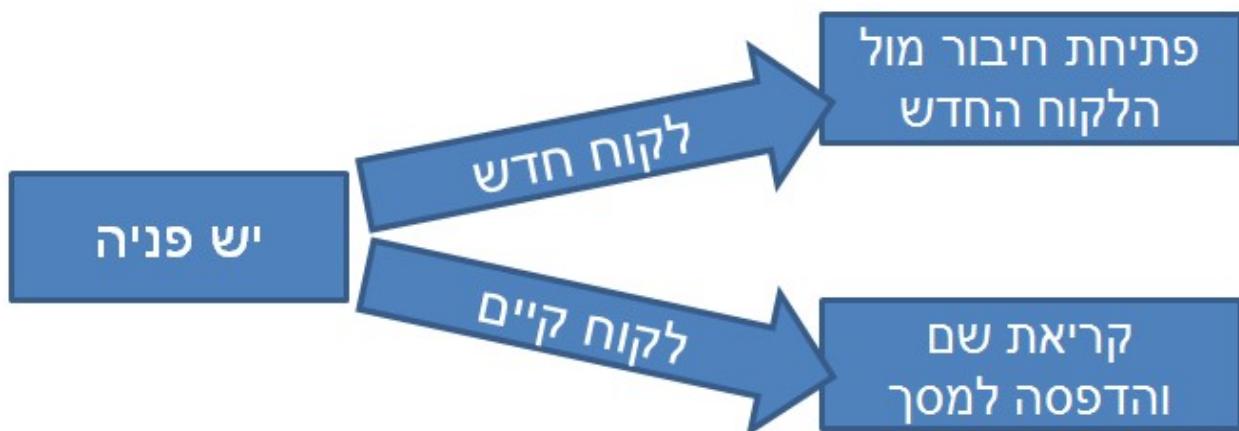
server_socket.listen(5)

open_client_sockets = []
```

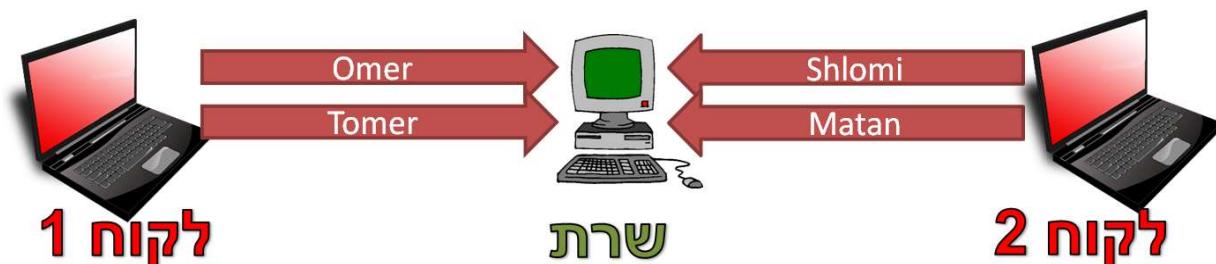
בשלב זה, علينا לטפל בפניות המגיעות אל השרת. לשם כך, ניצור לולה שככל פעם:

- אם יש פניה מלוקוח חדש - תפתח מולו חיבור (באמצעות **accept**).
- אם יש פניה מלוקוח קיים - תקרא ממנו את שמו, ותדפיס אותו למסך (באמצעות **recv**).

כלומר, בכל ריצה של הלולה, על הסקריפט לבצע את הלוגיקה הבאה עבור כל פניה:



זכoor, כל לקוח יכול לכתוב שם שונה בכל פעם, מבל' לסגור את החיבור:



הלוואה המתוארת לעיל תרוץ באופן תמידי, וכך נכתבו:

`while True:`

בתוך לולאת ה-`while`, נקרא לפונקציה בצורה הבאה:

`rlist, wlist, xlist = select.select([server_socket] + open_client_sockets, [], [])`

בכדי שנוכל להשתמש בMETHOD `select`, נדרש ראשית לייבא את המודול הרלבנטי:

`import select`

הקוד המלא יראה, אם כן, כך (השורות שהוספנו מודגשות באדום):

`import socket`

`import select`

```
server_socket = socket.socket()
server_socket.bind(('0.0.0.0', 23))
server_socket.listen(5)
open_client_sockets = []
```

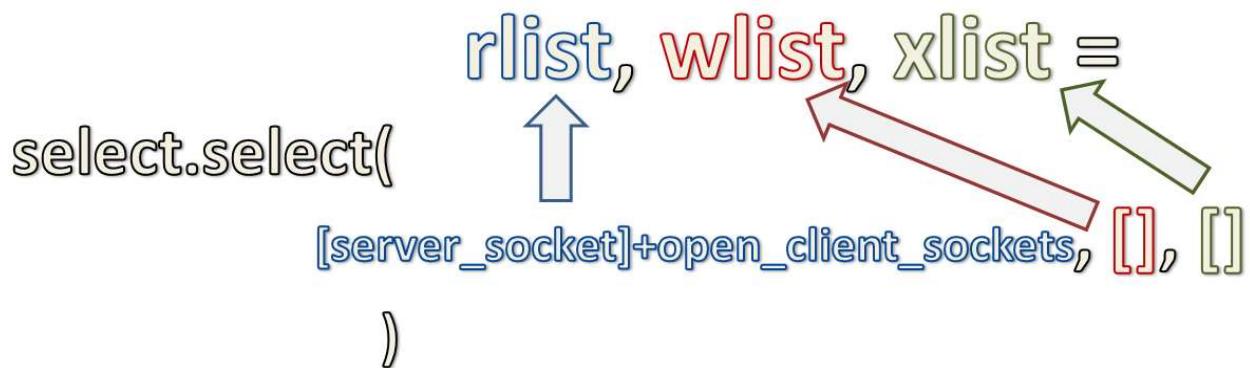
`while True:`

`rlist, wlist, xlist = select.select([server_socket] + open_client_sockets, [], [])`

זהו למעשה שורת המפתח, וכך נתעכט בכך להבין אותה. כאמור, הפונקציה `select` מחזירה שלוש רשימות:

- רשימת Sockets מהם ניתן לקרוא לקרוא - תשמר למשתנה `rlist`.
- רשימת Sockets אליהם ניתן לקרוא לשולח - תשמר למשתנה `wlist`.
- רשימת Sockets שזרקו שגיאה כלשהי - תשמר למשתנה `xlist`.

הרשימות נבנות מתוך הקלט של הפונקציה. כך למשל, אל המשתנה `wlist` ישמרו Sockets אליהם ניתן לקרוא לשולח, מתוך רשימת Sockets שניתנו לפונקציה `select` בתור הfrmater השני. להיות שהfrmater השני והשלישי הם רשימה ריקה ([]), הרי ש-`wlist` ו-`xlist` תהינהו לעולם ריקות.

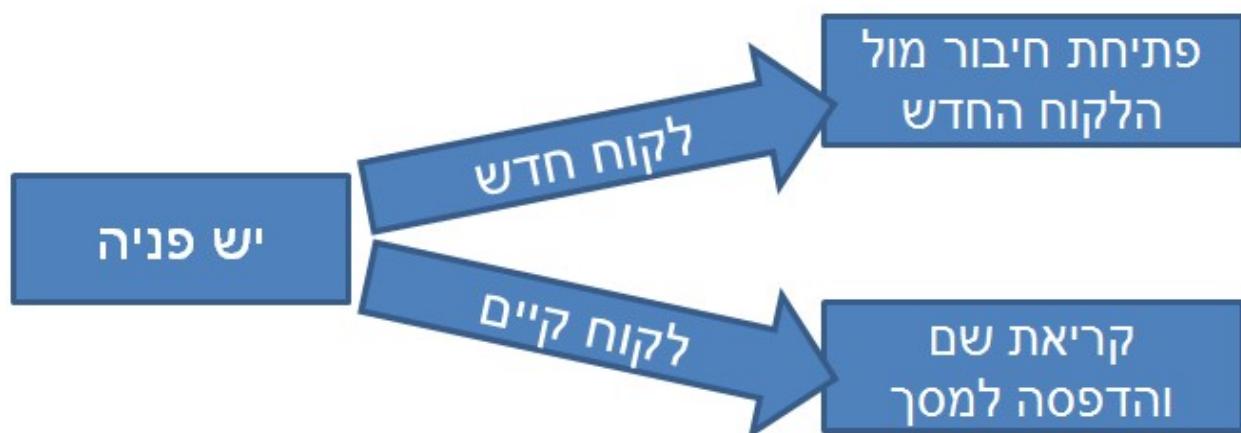


כעת נתמקד במשתנה `rlist`, אליו כאמור תשמור רשימת ה-Sockets מהם ניתן לקרוא ללקוח. אלו שולחים אל הפונקציה `select` את ה-Sockets הבאים:

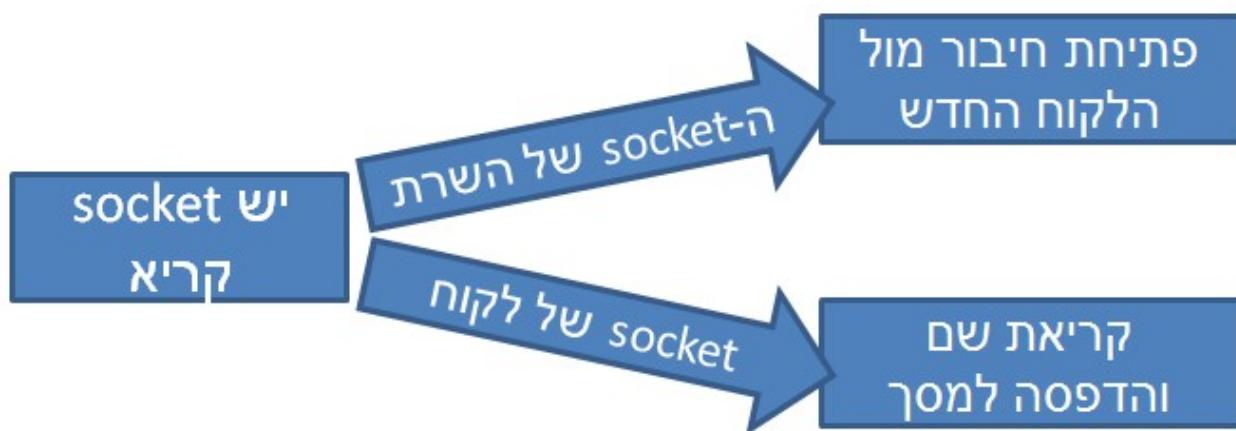
- Socket של השירות המאזין. במידה שנייתן לקרוא מ-Socket זה, המשמעות היא שיש פניה של לקוח חדש. לעומת זאת, ניתן לקרוא ל-`accept` ולהקימם חיבור עם הלוקה החדש.
- כלל ה-Sockets של הלקוחות. במידה שנייתן לקרוא מ-Socket של לקוח, הרי שהואשלח מידע. לעומת זאת, ניתן לקרוא ל-`recv` ולקבל מידע מלוקה קיימ.

שים לב כי על מנת לעשות זאת, השתמשנו בשרשור רשימות של פיטון:
`[server_socket] + open_client_sockets`

את הלוגיקה שצירפנו קודם:



ניתן למעשה לרשום גם בצורה הבאה:



כזכור, علينا להבין עבור כל Socket קריית, האם הוא ה-socket של השרת או של לקוח קיימ, ולפעול בהתאם. לשם כך, נעבור על כל ה-Sockets הקריים:

for current_socket in rlist:

 בדיקה אם ה-socket הנוכחי הוא של השרת:

 if current_socket is server_socket:

 אם כן, הרি שיש להרים מולו חיבור:

 (new_socket, address) = server_socket.accept()

 כמו כן, علينا להוסיף אותו לרשימת הלקוחות, כדי שנוכל לקבל ממנו מידע בעתיד:

 open_client_sockets.append(new_socket)

 אם לא, בשלב זה רק נדפיס שקיבלנו מידע מלקוח קיימ:

 else:

 print 'New data from client!'

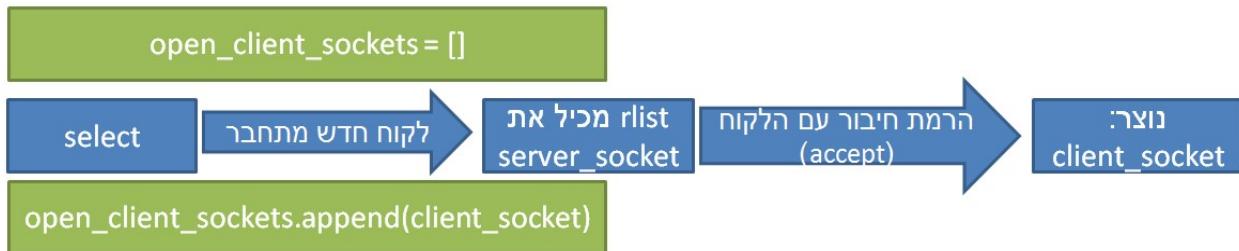
לולאת ה-while המלאה שלנו נראה这般:

```

while True:
    rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, [], [] )
    for current_socket in rlist:
        if current_socket is server_socket:
            (new_socket, address) = server_socket.accept()
            open_client_sockets.append(new_socket)
        else:
            print 'New data from client!'

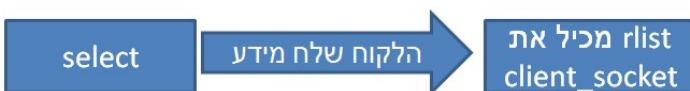
```

חישבו מה קורה כאשר השרת מופעל בפעם הראשונה. בתחילת, נקראת הפונקציה **select**, והיא ממשיכה לזרז עד אשר מגיע ללקוח חדש ומתחבר אל השרת. כאשר לקוחות יתחבר, הפונקציה **select** תחזיר אל המשתנה **rlist** רשימה שמכילה את ה-socket של השרת (**server_socket**). בשלב זה, השרת יקים את החיבור מול הלקוח באמצעות המетод **accept** ולאחר מכן יוסיף אותו לרשימה **open_client_sockets**.



בפעם הבאה שתoruן לולאת ה-`while`, הפונקציה `select` תחזיר כאשר הלוקוּט שלח מידע לשרת (בהתבהה שלא התחבר ביןתיים לוקוח אחר). הפעם, היא תחזיר אל המשתנה `rlist` רשיימה שמכילה את ה-Socket בין הלוקוּט לשרת, אותו Socket שהתווסף קודם לכן לרשיימה `open_client_sockets`. בשלב זה, תודפס למסך הודעה:

"New data from client!"



לאחר שהבנו את דרך הפעולה של הלולאה שלנו, הגיע הזמן לשפר אותה כך שהיא תטפל במידע שהגיע מהлокוּט. בטור התחלה, עלינו לקרוא את המידע:

```

data = current_socket.recv(1024)
  
```

כפי שלמדנו בפרק [תכנות ב-Sockets](#), יש להבין האם התקבלה מחזורת ריקה ("") והחיבור נסגר:

```

if data == "":
    open_client_sockets.remove(current_socket)
    print "Connection with client closed."
  
```

אם התקבל מידע תקין, ניתן להדפיס אותו למסך:

```

else:
    print data
  
```

כך נראה בשלב זה כל הקוד שכתבנו:

```
import socket
import select

server_socket = socket.socket()

server_socket.bind(('0.0.0.0', 23))

server_socket.listen(5)

open_client_sockets = []

while True:
    rlist, wlist, xlist = select.select([server_socket] + open_client_sockets, [], [])
    for current_socket in rlist:
        if current_socket is server_socket:
            (new_socket, address) = server_socket.accept()
            open_client_sockets.append(new_socket)
        else:
            data = current_socket.recv(1024)
            if data == "":
                open_client_sockets.remove(current_socket)
                print "Connection with client closed."
            else:
                print data
```



תרגיל 12.2 - ל��וח לשרת מרובה משתתפים

בתרגיל זה תכתבו ל��וח על מנת לבדוק את השרת שכתבנו בתרגיל המודרך הקודם. כתבו ל��וח אשר:

- מתחבר אל השרת שיצרתם.
- מבצע בלולה אינסופית:
 - מקבל שם מהמשתמש (באמצעות `raw_input`).
 - שלוח את השם אל השרת.

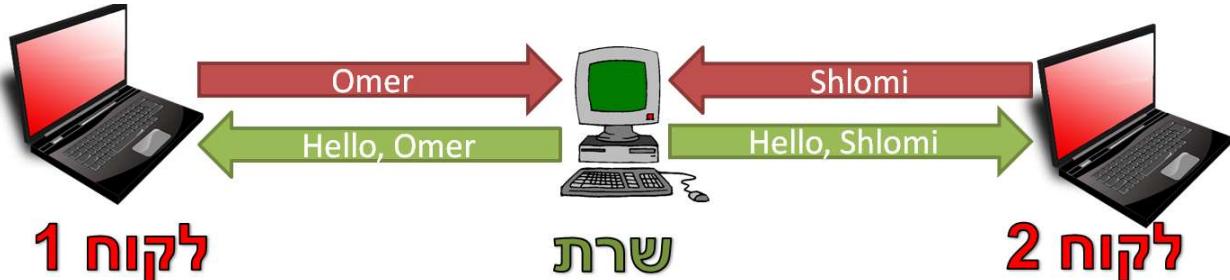
הפעילו את סקורייפט הלוקוח מספר פעמים במקביל, וכתבו אל השרתשמות שונים, בכל פעם מלוקוח אחר. וידאו כי השרת מצליח להדפיס את ההודעות מהлокוחות שלכם.



תרגיל 12.3 מודרך - שרת מרובה משתתפים עם תשובה לлокוחות

עד כה הצלחנו לקבל מידע מסווג לkokוחות מקביל. יכולת זה אمنם חשובה, אך אינה מספקת - علينا גם להצליח לתת שירות לлокוחות, כמו למשל לשלוח מידע אליהם.

בתרגיל זה נமמש שרת שמקבל חיבור מלוקוט, מקבל את שמו של הלוקוט, ועונה לו בהתאם. עם זאת, ברגע לשרת שמיימשו בפרק [תכנות ב-Sockets-תרגיל 2.3 מודרך - השרת הראשון שלו](#), השרת יוכל לטפל במספר לקוחות במקביל.



שימוש לב שטיפול במקרה זה צריך להיות מקביל - כלומר, השרת יוכל להשאיר את החיבור עם הלוקוט הראשון פתוח בעודו מספק שירות ללקוח השני.

לצורך התרגיל, נסתמך על הקוד שכתבנו בתרגיל המודרך הקודם, בו ביצענו רק קריאה של נתונים מהлокוט. נשנה את הקוד באיזור שטיפול בהודעה שהתקבלה מלוקוט קיימ. בימוש הקוד, הקוד גרם להדפסת ההודעה למסך (מסומן באדום):

```
if data == "":
    open_client_sockets.remove(current_socket)
    print "Connection with client closed."
else:
    print data
```

הפעם, נרצה לשלוח את המידע חזרה אל הלוקוט. עם זאת, אנחנו לא יכולים פשוט להשתמש בMETHOD send בשלב זה.



הodus לא ניתן פשוט לשלוח את המידע?

נסו לחשב על כך בעצמכם לפני שתקראו את השורה הבאה.
התשובה נעוצה בכך שלא בוטוח שאוטו Socket שעכשו קראתי ממנו את המידע מוכן לכך שנשלח לו את המידע. במקרה זה, הפונקציה **send** עלולה להיתקע, ולא יוכל לתת שירות לשאר הלוקוטות. אכן אנחנו רואים מקרה פשוט, בו השירות מגיב לכל לקוח בנפרד, אך בהמשך נראה מקרים מורכבים יותר בהם חשוב במיוחד לוודא שכל Socket אליו אנו מעוניינים לשלוח מידע יהיה במצב שמאכן לשילוח.

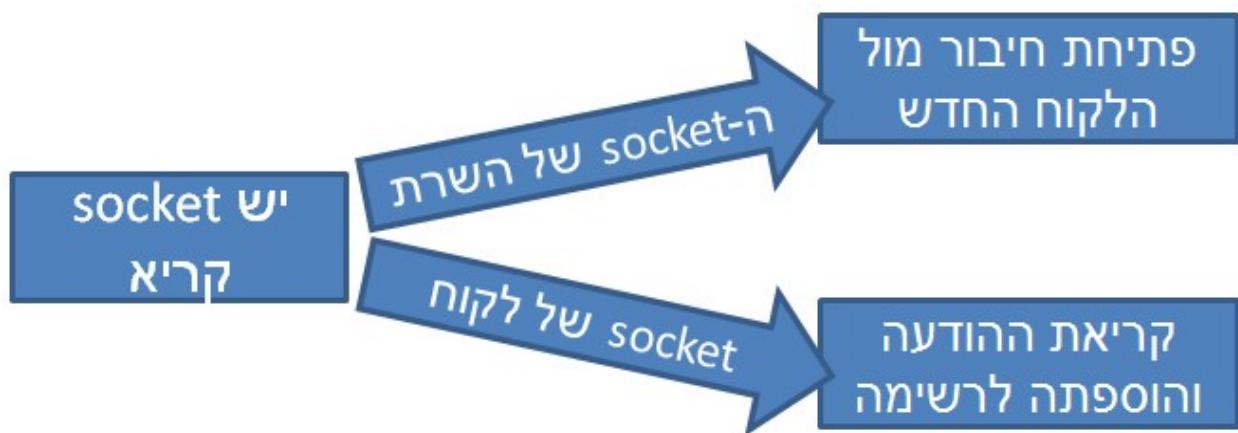
אי לכך, علينا להוסיף את הודעה הרשימה שתכיל את כל ההודעות שיש לשלוח, ולאחר מכן לשולח אותה כנתיתן יהיה לעשות זאת:

```
if data == "":
    open_client_sockets.remove(current_socket)
    print "Connection with client closed."
else:
    messages_to_send.append((current_socket, 'Hello, ' + data))
```

שים לב שהווסףנו כאן לרשימה אובייקט מסוג tuple שמכיל את ה-*Socket* שאליו יש לשולח את הודעה, ואת תוכן הודעה. מן הסתם, יהיה עליו להגדיר את הרשימה לפני שנוסיף אליו לתוכה, ולכן נעשה זאת לפני לולאת *while*:

```
messages_to_send = []
while True:
    ... (קוד הלולאה שתיארנו קודם)
```

זו הלוגיקה שמattaרת את ההתנהגות הנוכחיות של הלולאה:



בנוסף, בכל איטרציה של הלולאה, ננסה לשולח את כל ההודעות שעדיין לא נשלחו. נעשה זאת באמצעות פונקציה *send_waiting_messages*, אשר נמשך בקרוב. הפונקציה מקבלת רשימת *Sockets* שנייה לשולח אליהם מידע-current, ששמורה כזכור במשתנה *wlist*:

```
messages_to_send = []
while True:
    ... (קוד הלולאה שתיארנו קודם)
    send_waiting_messages(wlist)
```

עם זאת, כאשר מימשנו את השרת שرك קרא את המידע מהלקוחות, אמינו שהרשימה `wlist` תמיד תהיה ריקה, כיוון שהערכנו לפונקציה `select` רשימה ריקה בתור הפקטור השני:

```
rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, [], [] )
```

נשנה זאת אפוא ונעביר לפונקציה `select` את רשימת כל הלקוחות, כדי שתחזיר לנו אל המשתנה `wlist` את

רשימת הלקוחות שכרגע ניתן לשלוח אליהם מידע:

```
rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, open_client_sockets, [] )
```

בשלב זה, הלולאה שלנו נראה כך:

```
while True:
    rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, open_client_sockets, [] )
    for current_socket in rlist:
        if current_socket is server_socket:
            (new_socket, address) = server_socket.accept()
            open_client_sockets.append(new_socket)
        else:
            data = current_socket.recv(1024)
            if data == "":
                open_client_sockets.remove(current_socket)
                print "Connection with client closed."
            else:
                messages_to_send.append((current_socket, 'Hello, ' + data))

    send_waiting_messages(wlist)
```

כל שנוטר לנו לעשות הוא למשתמש את `.send_waiting_messages`

זכור, פונקציה זו מקבלת את רשימת ה-Sockets אליה ניתן לשלוח מידע, ולכן היא תוגדר כך:

```
def send_waiting_messages(wlist):
```

בתוך הפונקציה, علينا לטפל בכל אחת מההודעות שברשימה `messages_to_send`. כל הודעה היא למעשה

`tuple` המכיל את ה-`Socket` של הלקוח אליו יש לשלוח את ההודעה, וכן את התוכן של ההודעה. על מנת להקל

על העבודה, נעבור על כל הודעה ונוציא ממנה את שני האיברים שלה למשתנים נפרדים:

```
for message in messages_to_send:
```

```
    (client_socket, data) = message
```

שימוש לב שהפונקציה הצליצה לגשת אל המשתנה `message` מכיוון שהוא משתנה גלובלי¹⁰⁶.

עכשווי, علينا לבדוק האם ניתן לכתוב אל ה-`Socket`, כלומר האם הוא נמצא ברשימת ה-Sockets שניית לכתוב

אליהם:

¹⁰⁶ באופן כללי, לא טוב להשתמש במשתנים גלובליים כשבאים קוד. עם זאת, ספר זה נועד ללמד רשותות ולא תכנות נכון, ולכן לא נתעכט על אלטרנטיבות נכונות יותר מבחינתי פיתוח תוכנה.

```
if client_socket in wlist:
```

במידה שנייתן לכתוב אל ה-Socket, נשלח אליו את ההודעה:

```
client_socket.send(data)
```

כמו כן, לא נרצה שההודעה תשלוח שוב ללוקו. לשם כך נסיר אותה מרשימת ההודעות שיש לשולח:

```
messages_to_send.remove(message)
```

באם ה-Socket לא היה מוכן לכתיבה, הרוי שהפונקציה תצא מבלי לשולח את ההודעה ללוקו, וההודעה לא תמחק מרשימת ההודעות שעוד יש לשולח, ולקן תשאר בה.

כך נראה הפונקציה המלאה:

```
def send_waiting_messages(wlist):
    '''Sends waiting messages that need to be sent, only if the client's socket is writable.'''
    for message in messages_to_send:
        (client_socket, data) = message
        if client_socket in wlist:
            client_socket.send(data)
            messages_to_send.remove(message)
```

הקוד המלא של השרת נראה כך:

```
import socket
import select

server_socket = socket.socket()

server_socket.bind(('0.0.0.0', 23))

server_socket.listen(5)

open_client_sockets = []
messages_to_send = []

def send_waiting_messages(wlist):
    '''Sends waiting messages that need to be sent, only if the client's socket is writable.'''
    for message in messages_to_send:
        (client_socket, data) = message
        if client_socket in wlist:
            client_socket.send(data)
            messages_to_send.remove(message)

while True:
    rlist, wlist, xlist = select.select([server_socket] + open_client_sockets, open_client_sockets, [])
    for current_socket in rlist:
        if current_socket is server_socket:
            (new_socket, address) = server_socket.accept()
            open_client_sockets.append(new_socket)
        else:
            data = current_socket.recv(1024)
            if data == "":
                open_client_sockets.remove(current_socket)
                print "Connection with client closed."
            else:
                messages_to_send.append((current_socket, 'Hello, ' + data))

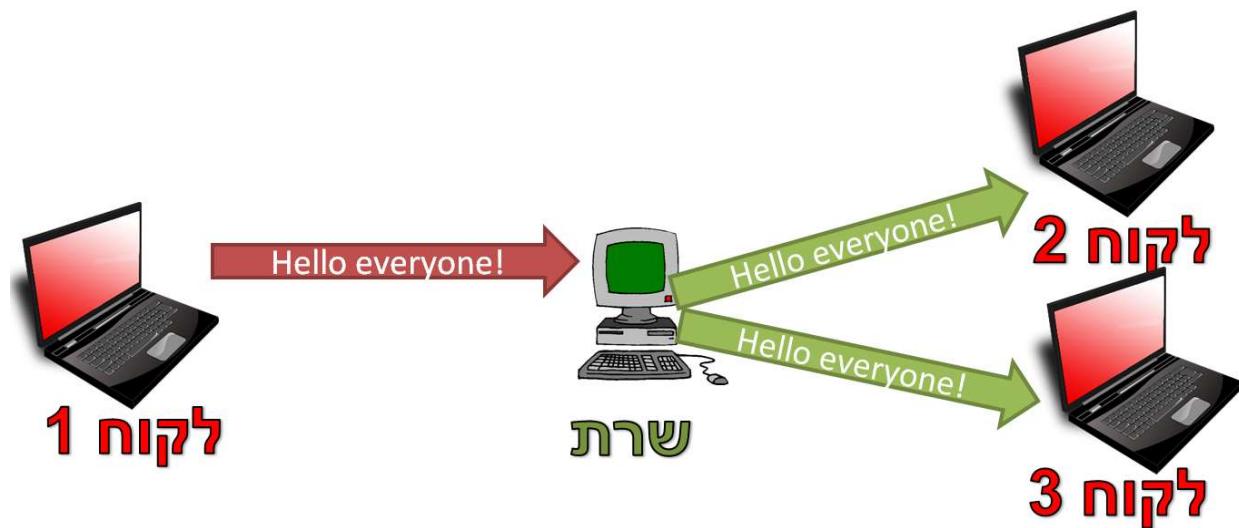
    send_waiting_messages(wlist)
```

תרגיל 12.4 - ל��וח לשרת מרובה משתתפים שקורא מידע מהשרת

כעת עליכם לבדוק את הקוד של השרת הקודם שכתבנו. היעזרו בליך שכתבתם בתרגיל הקודם שרק שולח מידע אל השרת ([פרק תכונות Sockets מתקדם: ריבוי משתמשים / תרגיל 12.2 - ל��וח לשרת מרובה משתתפים](#)).
שפרו את הלוקוח כך שגם יראה מידע שנשלח אליו מהשרת וידפיס אותו למסך.
הפעילו את סקריפט הלוקוח מספר פעמים במקביל, וכתבו אל השרתשמות שונים, בכל פעם מלוקוח אחר. וודאו כי כל לוקוח מקבל הודעה נכונה בהתאם לשם שהוא שלח לשרת.

תרגיל 12.5 - צ'אט מרובה משתתפים

בתרגיל זה תשתמשו בידע שרכשתם במהלך הפרק על מנת למש צ'אט מרובה משתתפים. עליכם למש גם את השרת וגם את הלוקוח. כל לוקוח יכול לכתוב לשרת איזה מידע שהוא רוצה. השרת יdag לשאר הלוקוחות את ההודעה שלו (ולא אליו בחזרה). כל לוקוח רואה את כל ההודעות של הלוקוחות האחרים, אך לא יכול לדעת מי הגיעו ההודעה:



הנחיות לתרגיל

יש להשתמש בספרייה `socket`, ולא בספריות עזר כגון `cgsocket`.

צד השרת

השרת צריך לשלוח את הודעה לכל הלקוחות **מלבד** זה ששלח אותה אליו. מקרה זה שונה מהשרותים שמיימנו קודם, ומורכב יותר. בכל פעם שתנסו לשלוח את הודעה, עליכם לזכור למי כבר הצלחתם לשלוח אותה, ולא לשלוח פעמיים את אותה הודעה אף לקוות.

צד הלקוח

スクרייפט הלקוח צריך להצליח גם לקרוא מידע מהמשתמש, במידה שהוא רוצה לשלוח הודעה לשאר המשתמשים, וגם לקרוא מידע מהשרת. פעולה זו צריכה להתבצע במקביל, ולכן אסור להן להיות חוסמות (blocking).

בכל הפעמים בהן מיימנו ל��, על מנת לקרוא מידע מהשרת, השימושו פשוט במתודה `recv`. עם זאת, המתודה היא `blocking`, ולכן לא תוכל פשטוט להשתמש בה. במקרה זה, יהיה עליו להשתמש ב-`select` בדומה לכך.

כמו כן, בכל הפעמים בהן מיימנו ל��, קראנו מהמשתמש באמצעות הפונקציה `raw_input`. גם פונקציה זו היא `blocking`, ועל כן לא נשתמש בה. על מנת לקרוא באופן שאינו `blocking`, קראו על המודול `msvcrt` (הכולל בהתקנה הסטנדרטית של פיתון). באופן ספציפי, תוכלם להשתמש בפונקציות `kbhit` ו-`getch` של מודול זה.

תרגיל 12.6 - צ'אט מתקדם



בתרגיל הקודם כתבתם צ'אט מרובה משתמשים. הצ'אט אמן אפשר למשתמשים לתקשר, אך לא בצורה נוחה. אף משתמש לא יכול היה לדעת מי המשתמש שכתב את הודעה או מתי כל הודעה נשלחה. מעבר לכך, חסרו לצ'אט הרבה מהיכולות שיש לצ'אט אמיתי. בתרגיל זה, תכתבו צ'אט הרבה יותר מעניין.

את הצ'אט עליו כתוב באופן מדווג. ככלומר, בכל פעם שעבדו רק על סעיף אחד. רק לאחר שהשלמתם סעיף מסוים ובדקתם כי הצ'אט פועל בהתאם למצופה, עברו לסעיף הבא.
שימוש לבן לקרוא את תיאור הפרוטוקול שמופיע לאחר רשימת הסעיפים, ולהסתמך עליו לאורך התרגיל ככל.

סעיפים

1. כל הودעה תתחיל עם שם המשתמש שכתב אותה. לדוגמה:

talmid1: Hello everyone!

2. ליד כל הודהה תירשם השעה שבה היא נכתבה. לדוגמה:

08:02 talmid1: Hello. the lesson is about to start.

3. אם משתמש שולח את המחרוזת "זעב" – על השרת לנקז אותו.

4. במקרה שימוש עזב את הציג (השרת נתקאותו, או הוא החליט לעזוב בעצמו), על השרת לכתוב לוולט שהוא יצא מבחןינו למשול.

09:45 talmid5 has left the chat!

5. אפשרו למשתמשים מסוימים להיות מנהלים. רשימת המנהלים תירשם באופן hard-coded אצל השרת (כלומר, בתוך רשימה קבועה בקוד). מנהל יכול להעיף משתמש אחר מהצ'אט. כאשר מנהל מעיף משתמש, תשליך לcoliום ההודעה:

09:47 talmid6 has been kicked from the chat!

6. ליד שם משתמש של מנהל יש להופיע הסימול '@'. לדוגמה:

10:05 @manager3: Woohoo, I am a manager.

7. דאגו לכך שימושים לא יכולים להשתמש בשם שמה המקורי ב-'@', כדי לא לגרום אחרים לחושם מנהלים למרות שהם לא.

8. אפשרו למנהל למנות באופן דינامي מנהלים גוספים. **שים לב:** מי שאינו מנהל לא יכול למסות מנהלים.

9. תננו למנהלים את האפשרות להשתיק משתמש. הכוונה היא שהמשתמש יוכל לראות מה כתוב בז'אט, אך לא יוכל לכתוב. אם הוא יקבל תשובה – "You cannot speak here – עמ' זאת, משתמש

10. אפשרו שליחת הودעה פרטיות בין שני משתמשים (במקרה של הودעה פרטית), רק שני המשתמשים הרלבנטיים יראו את ההודעה, ולא אף אחד אחר. כל הודעה פרטית תתחילה בסימול "!". לדוגמה:

09:42 !talmid2: This is a private message.

11. אם לך שולח את המחרוזת "view-managers", תוחזר אליו רשימת כל המנהלים.

טיור הפרוטוקול

צד הליכון

על כל הודעה שהליכון שולח להיות במבנה הבא:

דוגמה	סוג	שדה
6	מספר	אורך שם המשתמש
talmid	מחרוזת	שם המשתמש
1	מספר	פקודה לשימוש (פירוט בהמשך)
Hello world	משתנה	פרמטרים נוספים בהתאם לפקודה

הפקודות שניתן לשולח הן:

דוגמה	פרמטרים נוספים	משמעות	מספר פקודה
5, hello	אוריך הודעה, הודעה	הודעת צ'אט	1
7, manager	אוריך שם המנהל, שם המנהל	מיןוי מנהל	2
7, talmid2	אוריך שם המשתמש, שם המשתמש	העפת משתמש	3
7, talmid3	אוריך שם המשתמש, שם המשתמש	השתקת משתמש	4
7, talmid4, 5, hello	אוריך שם המשתמש שאליו רוצים לשולח את הודעה, שם המשתמש, אוריך הודעה, הודעה	הודעה פרטית בין משתמשים	5

דוגמה להודעת צ'אט:

4Omer118This is an example

הסבר:

- **באדום** - אורך שם המשמש של השולח. מכיוון שם המשמש הוא "Omer", האורך הינו 4.
- **בכחול** - שם המשמש של השולח. בדוגמה זו - "Omer".
- **בכתום** - הפקודה לשימוש. במקרה זה מדובר בהודעת צ'אט, ועל כן הפקודה היא 1.
- **בירוק** - אורך הודעה. מכיוון שההודעה היא "This is an example", האורך הוא 18.
- **בסגול** - הודעה עצמה, במקרה זה - "This is an example".

במקרה זה, המשמש Omer שלח את הודעה: ."This is an example" שimeo לב: על הגודל של השדה הראשון (אורך) להיות באורך מוגדר, למשל של ארבעה בתים. אחרת, כיצד נדע האם הודעה שמתחליה ב- "30" כונתה לשם משמש באורך 30, או שמא שם משמש באורך 3 תווים, כשהתנו הראשוני הוא "0"?

דוגמה למינוי מנהל:

4Omer26Shlomi

הסבר:

- **באדום** - אורך שם המשמש של המנהל הממנה. מכיוון שם המשמש הוא "Omer", האורך הינו 4.
- **בכחול** - שם המשמש של המנהל הממנה. בדוגמה זו - "Omer".
- **בכתום** - הפקודה לשימוש. במקרה זה מדובר במינוי מנהל, ועל כן הפקודה היא 2.
- **בירוק** - אורך שם המשמש שיש למנות למנהל. מכיוון שם המשמש הינו "Shlomi", האורך הוא 6.
- **בסגול** - שם המשמש שיש למנות למנהל. במקרה זה - "Shlomi".

במקרה זה, המשמש Omer מינה את המשמש Shlomi להיות מנהל.

דוגמה להעפת משטמש:

4Omer33Avi

הסבר:

- **באדום** - אורך שם המשמש של המיעף. מכיוון שם המשמש הוא "Omer", האורך הינו 4.
- **בכחול** - שם המשמש של המיעף. בדוגמה זו - "Omer".
- **בכתום** - הפקודה לשימוש. במקרה זה מדובר בהעפת משטמש, ועל כן הפקודה היא 3.
- **בירוק** - אורך שם המשמש שיש להעיף. מכיוון שם המשמש הינו "Avi", האורך הוא 3.
- **בסגול** - שם המשמש שיש להעיף. במקרה זה - "Avi".

במקרה זה, המשתמש Omer העיף את המשתמש Avi מzan הצל'אט.

דוגמה להשתקת משתמש:

4Omer43Avi

הסבר:

- **בדואן** - אורך שם המשתמש המשתמש. מכיוון שם המשתמש הוא "Omer", האורך הינו 4.
- **בכחול** - שם המשתמש של המשתמש. בדוגמה זו - "Omer".
- **בכתום** - הפקודה לשימוש. במקרה זה מדובר בהשתתקת משתמש, ועל כן הפקודה היא 4.
- **בירוק** - אורך שם המשתמש שיש להשתיק. מכיוון שם המשתמש הינו "Avi", האורך הוא 3.
- **בסגול** - שם המשתמש שיש להשתיק. במקרה זה - "Avi".

במקרה זה, המשתמש Omer השתיק את המשתמש Avi.

דוגמה לשילוח הודעה פרטית בין משתמשים:

4Omer56Shlomi23This message is private

הסבר:

- **בדואן** - אורך שם המשתמש של השולח. מכיוון שם המשתמש הוא "Omer", האורך הינו 4.
- **בכחול** - שם המשתמש של השולח. בדוגמה זו - "Omer".
- **בכתום** - הפקודה לשימוש. במקרה זה מדובר בשילוח הודעה פרטית, ועל כן הפקודה היא 5.
- **בירוק** - אורך שם המשתמש שלו נשלחת ההודעה. מכיוון שם המשתמש הינו "Shlomi", האורך הוא 6.
- **בסגול** - שם המשתמש שלו נשלחת ההודעה. במקרה זה - "Shlomi".
- **בשחור** - אורך ההודעה הפרטית. מכיוון שההודעה הינה: "This message is private", האורך הוא 23.
- **בחום** - ההודעה הפרטית עצמה. במקרה זה, ההודעה היא: ".This message is private"

. "This message is private" שלח למשתמש Shlomi את ההודעה הפרטית:

צד השירות

השירות מתקשר עם הלקוחות רק במחוזות, כלומר הוא שולח מחוזת שתוצג ללקוח. כל הודעה נשלחת בפורמט הבא:

דוגמה	סוג	שדה
11	מספר	אורק המחווזת
Hello world	מחוזת	המחווזת

דוגמה לשילוח הודעה מהשירות ללקוח:

4409:47 talmid6 has been kicked from the chat!

הסבר:

- **באדום** - אורק כל ההודעה שהשירות שולח. מכיוון שההודעה היא "09:47" .44 ."from the chat!"
- **כחול** - ההודעה עצמה שהשירות שלח. בדוגמה זו - "chat!"

תכנות Sockets מתקדם - סיכום

בפרק זה למדנו כיצד לתכנן באמצעות Sockets שרת ולקוח היכולים לטפל בכמה בקשות במקביל. התחלנו בהבנת הצורך במימוש שירותים שטיפולים בכמה ליקוחות, ולאחר מכן הבנו את האתגר שבמימוש שירותים שכolumbia. לאחר מכן, הכרנו את `select`, הפונקציה שעזרה לנו להתגבר על אותם אתגרים.

מיימנו שרת שמקבל מידע מכמה ליקוחות שונים במקביל, ומדפיס את המידע למסך. לאחר מכן, כתבנו ליקוח שיבדוק את הרשת זהה. בשלב זה רק הצלחנו לקרוא מידע מכמה ליקוחות. בהמשך, מיימנו שרת שגם עונה לכל ליקוח וליקוח בהתאם למידע שהוא שלח, וכتبנו ליקוח שיבדוק את הרשת זהה.

ماוחר יותר, כתבנו צ'אט שמאפשר לכמה ליקוחות לתקשר לתוכה זה עם זה באמצעות שרת אחד. במקורה זה, נתקלנו לראשונה בצריך למשוך ליקוח שיווכל גם לקרוא מידע מהמשתמש וגם לקבל מידע מהשרת במקביל. כמו כן, התרמודנו עם שליחת הודעה למספר רב של ליקוחות, כאשר צריך בכל פעם לזכור איזה ליקוח קיבל את המידע ואיזה עדין לא. לסירוגין, שידרגנו את הצ'אט בשלל יכולות מעניינות: החל מהוספה שם המשתמש לכל הודעה, דרך מינוי מנהלים שיכולים להעיף או להשתיק משתמשים אחרים, ועד למימוש הודעות פרטיות.

בדרכו רכשנו כל' שימושתי נוסף, שמאפשר לנו לתקשר עם מספר ישות רשת במקביל מעל ממשך ה-`Sockets`.

פרק 13 - מילון מושגים

פרק זה כולל מונחים בהם נעשה שימוש לאורך הספר, והגדורותיהם. המילוןlude שיער במהלך הקראיה. ההגדורות מובאות בהקשר שלهن לפרק ולמידע שמצוין בספר, ולא נועדו לעמוד בזכות עצמן בכך להגדיר את המושגים.

פרק 1 - תחילת מסע - איך עובד האינטרנט?

- **WWW** - ראשית תיבות של Web World Wide. אוסף עמודי האינטרנט אליום אנו גולשים בדף.
- **בקשה (Request)** - הودעה שנשלחת מהלקוח אל השרת בכך לבקש שירותו כלשהו.
- **תגובה (Response)** - הודעה שנשלחת מהשרת אל הלוקה כמענה לבקשתו.
- **כתובת מקור** - כתובת המצינית מי שלח חבילה מידע מסוימת.
- **כתובת יעד** - כתובת המצינית לאן חבילה מmoveנת.
- **ping** - כלי המאפשר לבדוק קישוריות לשוטות מרוחקת, ואת הזמן שלוקח לחבילה להגיע אליה ובחרזה.
- **traceroute** - כלי המאפשר למצוא את הדרך שעוברת חבילה בין המחשב שלו לנקודות קצה שונות.
- **GeoIP** - כלי הממפה בין כתובת IP לבין המיקום הגיאוגרפי שלה.
- **קפיצה (Hop)** - העברה של חבילה מידע בין רכיב אחד לרכיב אחר המוחברים ישירות.
- **שמות דומיין** - כתובות קריאות לפי פרוטוקול DNS, לדוגמה "www.facebook.com" או "www.ynet.co.il".
- **DNS** - מערכת המאפשרת המרה בין שמות דומיין וכתובות IP.
- **nslookup** - כלי המאפשר לבצע תשאלות DNS.

פרק 2 - תכונות ב-Sockets

- **תקשורת שרת-לקוח (Client-Server)** - סוג תקשורת בין שרת, המספק שירות כלשהו, לבין לקוח, המשמש בשירות המספק.
- **Socket** - ממשק תוכנתי להעברת מידע בין תוכנות שונות. זה API שמסופק בידי מערכת הפעלה.

פרק 3 - Wireshark ומודל חמש השכבות

- **הסנפה** - הפעולה בה אנו משתמשים על חבילות המידע בדיק כפי שנשלחו או התקבלו בכרטיס הרשות.
- **Wireshark** - תוכנת הסנפה.
- **פקטה (חבילה, Packet)** - חבילה מידע המכילה מוען, נמען ותוכן ההודעה. מונח זה מתאר גוש מידע בשכבת הרשות.

- **פרוטוקול (Protocol, תקן)** - סט מוגדר של חוקים, הקובע כלליים ברורים כיצד צריכה להיראות התקשרות בין הצדדים השונים.
- **ישות (Entity)** - כל רכיב המחבר לרשת - בין אם הוא סמארטפון, מחשב נייד, שרת של Google, רכיב רשת שנמצא בדרך בין ישויות אחרות, או רכיב בקרה של תחנת כוח המחבר גם הוא לרשת לצורך שליטה מרוחק.
- **ISO** - ארגון התקינה הבינלאומי.
- **IOS** - מודל שבע שכבות.
- **ריבוב (Multiplexing)** - התרגיל שבו מידע מספר מקורות משולב אל תווך משותף אחד.
- **הרעבה (Starvation)** - תופעה בה תחנה אחת מציפה את הקו בקצב אורך של מידע, ובכך מונעת מהתחנות אחרות גישה לשדר על הקו.
- **עטיפת מידע (Capsulation)** - עטיפת מידע בשכבות נוספות.
- **הוצאת המידע של שכבה מסוימת (Decapsulation)**
- **Header (תחלית)** - מידע שימושיפה כל שכבה לתחילת הפקטה, מכיל מידע שימוש שמשמש לשילטה ובקרה על הפקטה.
- **מסנן תצוגה (Display Filter)** - מסנן את הפקטות המוצגות למסך על פי תנאים מסוימים. מסנן זה רץ ברמת האפליקציה.
- **מסנן הסנפה (Capture Filter)** - מסנן את הפקטות הנקלטות לאפליקציה על פי תנאים מסוימים. מסנן זה רץ ברמת ה-Driver של מערכת הפעלה.

פרק 4 - שכבת האפליקציה

- **אפליקציה** - יישומים ותוכנות שנגישות למשתמשי קצה באמצעות מחשב, סמארטפון או טאבלט, ובנויות במודל שרת-לקוח, כך שהאפליקציה משתמשת לקוח (בין אם אפליקציה ייעודית ובין אם באמצעות הדפדפן), ומסתמכת על שרת שאיתו היא מתחילה באמצעות פרוטוקול אינטרנט.
- **משאב (Resource)** - ברשת האינטרנט, הכוונה היא לכל רכיב שיכול להיות חלק מעמוד אינטרנט - תמונה, טקסט, HTML, javascript ודומה.
- **URL** - כתובות לזיהוי משאב ברשת האינטרנט (ראשי התיבות: Uniform Resource Locator). האופן שבו בניין URL:

<protocol>://<host>/<resource_path>?<parameters, separated by &>

- **לדוגמה:** http://twitter.com/search?q=obama&mode=users
- מתאר גישה בפרוטוקול HTTP לשרת twitter.com, וմבוקש את המשאב /search עם הפרמטרים q ו-users (ראו: URL Parameters).
- **GET** - סוג בקשה HTTP לקבלת משאב ספציפי מהשרת - כוללת את כתובות המשאב. הבקשה לא אמורה לגרום לשינוי מצב בשרת, ולא מכילה מידע (data), מלבד פרמטרים ב-URL.

- **POST** - סוג בקשה HTTP להעברת מידע לשרת (כגון שליחת אימייל, מילוי טופס או העלאת תמונה). המבנה שלו דומה לבקשת GET, אלא שהיא מכילה גם מידע (data) בגוף הבקשה - המידע שMOVEDר לשרת.
- **HTTP Header** - מופיע הן בבקשת והן בתגובה HTTP, בין שורת הכוורת לבין התוכן. מכיל רשימה של שדות, מתוך רשימה של שדות אפשריים (כגון גודל המידע שבהודעה, סוג הלקוח, סוג השרת). חלק מהשדות ניתנים לשימוש רק בבקשת, חלקן רק בתגובה, וחלקן בשני המקרים. מנגנונים מתקדמים (כגון cache ואוטנטיקציה) לרוב עושים שימוש בשדות ה-*header*.
- **Status Code** - קוד שמתאר את מצב ההודעה שנשלחת בתגובה לבקשת HTTP. המפורטים ביותר הם OK 200 שמעיד על תגובה תקינה, וכן Not Found 404, המਸמן שהמשאבות המבוקש לא נמצא.
- **HTTP Response** - הودעה הנשלחת כתגובה לבקשת שקדמה לה, לרוב התגובה תישלח מהשרת ללקוח. מכילה קוד מצב (status code), שדות header ותוכן.
- **Content-type** - שדה header שמתאר את סוג המידע (data) שמצויר לבקשת/תגובה. סוגים תוארים נפוצים: image/jpeg, application/javascript, text/html.
- **URL Parameters** - חלק מה-URL הנשלח בבקשת ניתן לכלול פרמטרים שבהם יעשה השרת שימוש כשיטף בבקשתו. הפרמטרים מופרדים על-ידי התו &, וימצאו לאחר חלק ה-path שב-URL, מופרדים ממנו על-ידי סימן שאלה.
- לדוגמה, ב-URL הבא: <http://twitter.com/search?q=obama&mode=users>
- ישנו שני פרמטרים: הראשון בשם *q* עם הערך *obama*, והשני בשם *mode* עם הערך *users*.
- **HTTP Session** - אינטראקציה מתמשכת (כלומר, יותר מזוג בקשה-תגובה יחיד) בין משתמש קצה באפליקציית לקוח לבין שרת. session הוא stateful, כלומר הרשות "זכור" את ההיסטוריה של session, בניגוד לאופן המקורי בו פועל פרוטוקול HTTP, stateless, שמכונה session-less. לרוב יזהה ה-session באמצעות cookie שיועבר ב-*header* של הבקשות.
- **Cookie** - מחרוזת המשותפת לשרת וללקוח, הנקבעת על-ידי השרת וmovedרת על-ידי הליקוח בכל בקשה, על מנת שהשרת יוכל לזהות בקשות HTTP שישיכות לאוטו-*Session*. מנגנון זה עושה שימוש בשדות header. שימושים נפוצים: משתמש ש עבר זיהוי או אימות, לא צריך לבצע זאת מחדש (Gmail, Facebook, Amazon).
- **Cache** - מנגנון שנועד לחסוך בתעבורה של משאבי ברשת, על-ידי כך שימושים נשמרים בדיסק המקומי של הלקוח, ויובאו מחדש מהשרת רק אם הגרסה שם השתנתה. מנגנון זה עושה שימוש בשדות header.
- **Conditional-GET** - הבקשה הנשלחת על-ידי הליקוח לקבלת משאב שקיים בעברו עותק ב-*cache*. בבקשת מצורף הזמן בו נשמר המשאב ב-*cache* (בשדה *header*), ומהשאב עצמו יכול בתגובה רק אם יש גרסה חדשה יותר בשרת. אם הגרסה שב-*cache* של הליקוח עדכנית, תוחזר תגובה עם קוד מצב 304 שמסמן שהמשaab לא שונה (והמשaab לא שונה) בתגובה - כך נחסכה תעבורה "מיותרת".

- **בשודות header ו-b code status** כדי לבצע את זהויות והאימונות. המנגנון ד' חלש, משומם שהוא אינו מציין את שם המשתמש והסיסמה, אלא רק מקודד אותם.
- **שאילתת DNS (DNS Query)** - שאלת בפrotocol DNS עבור שם דומיין מסוים.
- **אזור (Zone)** - מערכת-DNS הינה הרכנית, והתו המفرد שיוצר את ההיררכיו הוא התו נקודה ("."). כך למשל, הדומיין www.facebook.com מתאר שרת בשם "www" בתוך האזור "facebook" שבתוך האזור ".com".
- **רשומה בשאילתת או תשובה DNS (Resource Record RR)**

פרק 5 - Scapy

- **Scapy** - ספרית פיתון המאפשרת לעבוד עם חבילות מידע בצורה מתקדמת. מאפשרת הסנהפה, יצירה שליחת של פקודות.
- **Resolving** - תרגום של שמות דומיין לכתובות IP, למשל באמצעות DNS.

פרק 6 - שכבת התעבורה

- **פורט (Port)** - מזהה תוכנה באורך 16 ביטים (bits). נחוץ על מנת לרabb תקשורת בין מספר תוכנות על אותה ישות רשת.
- **netstat** - כליה המאפשר לדעת על איזה פורטים המחשב מאמין, ואילו קישורים קיימים כרגע.
- **포רטים מוכרים (Well known ports)** - הפורטים מהטווין שבין 0 ועד 1023 (כולל). פורטים אלו מוקצים בידיIANA עבור אפליקציות ספציפיות.
- **תקורה (Overhead)** - מידע נוסף (יתיר) שנשלח מעבר למידע שורצים להעביר. לדוגמה, לשילוח Header יש תקורה - עובר מידע בראשת שהוא מידע נוסף נושא על המסר שרצינו להעביר.
- **פרוטוקול מבוסס קישור (Connection Oriented Protocol)** - על מנת לתקשר עם ישות כלשהי באמצעות פרוטוקול מבוסס קישור, יש ראשית "להקם" את הקישור, לאחר מכן להשתמש בקישור שהוקם ולבסוף לנתק את הקישור. מבחינות המשתמש, הוא מתייחס ל קישור כמו לשיפורת הטלפון: הוא מזין מידע (במקרה שלנו - רצף של נתונים) לenza אחד, והמשתמש השני יקבל את המידע בצד השני. פרוטוקולים מבוססי קישור מבטיחים הגעת המידע, וכן הגיעם בסדר הנכון.
- **פרוטוקול שאינו מבוסס קישור (Connectionless Protocol)** - על מנת לתקשר עם ישות כלשהי באמצעות פרוטוקול שאינו מבוסס קישור, אין צורך בהרימה וסירה של קישור, וניתן פשוט לשולח את החבילה. ב프וטוקול מסווג זה, אין הבטחה שהחביבה תגיע ליעדה. כמו כן, אין הבטחה שהחבילות תגעהה בסדר הנכון.

- **UDP (User Datagram Protocol)** - פרוטוקול נפוץ של שכבה התעבורה. פרוטוקול זה אינו מבօס קישור. דוגמה נפוצה לשימוש: פרוטוקול DNS.
- **TCP (Transmission Control Protocol)** - פרוטוקול נפוץ של שכבה התעבורה. פרוטוקול זה מבօס קישור. דוגמה נפוצה לשימוש: פרוטוקול HTTP.
- **פורט מקור (Source Port)** - הפורט של התוכנה שלחה את החבילה.
- **פורטיעד (Destination Port)** - הפורט של התוכנה שצפוי לקבל את החבילה.
- **Checksum** - תוצאה של פעולה מתמטית שנתבצעת על המידע. ה-Checksum מתיוסף לחבילה, בתור מידע יtier ומשמש לזיהוי שגיאות. הצד מקבל מחשב checksum בעצמו עבור כל חבילה, ומשווה אותו אל תוכן ה-Checksum שכתוב בחבילה עצמה. אם התוצאה יוצאה זהה - החבילה נחשבת תקינה. אחרת - התגלתה שגיאה.
- **סגןטיים (Segments, מקטעים)** - השם של גוש מידע בשכבה התעבורה.
- **Sequence Number** - מספר סידורי שנitin לחבילות או חלק מהן על מנת לעקוב אחר רצף המידע. שימוש במספר סידורי מאפשר לדעת איזה חלק מהמידע הגיע, איזה חלק לא הגיע, ולהבין מה הסדר הנכון של המידע. ב-TCP, ישנו מספר סידורי לכל בית (byte). לכל אחד מהבתים ברצף יש מספר סידורי משלהו. בכל חבילה שנשלוח, יהיה המספר הסידורי שמצין את הבית הנוכחי בחבילה.
- **ACK (Acknowledgement)** - חבילה שנועדה לאשר שהתקבל מידע מן הצד השני. שם שהמספרים הסידוריים של TCP מתיחסים לבטים (bytes) ברכף המידע, כך גם מספרי ה-ACK. מספר ה-ACK ב-TCP מצין את המספר הסידורי של הבית הבא שמצווה להתקבל.
- **Three Way Handshake (לחיצת יד משולשת)** - הדרך להרמת קישור ב-TCP. כוללת שלוש חבילות: חבית SYN, חבית SYN+ACK וחבית ACK.
- **URR (Initial Sequence Number)** - ערך ה-Sequence Number הראשוני של תקשורת TCP. ערך זה נבחר באופן רנדומלי.

פרק 7 - שכבת הרשות

- **ניטוב (Routing)** - תהליך ההחלטה על הדרכ שבה יש להגיע מנקודה א' לנקודה ב' ברשות.
- **(IP) Internet Protocol** - פרוטוקול שכבה שלישית הנפוץ באינטראנט.
- **ipconfig** - כלי המאפשר לראות מידע מידע על הגדרות הרשות שלנו. למשל, הכלי מראה מה כתובות ה-IP שלנו.
- **כתובת IP** - כתובות לוגיות של שכבת הרשות בפרוטוקול IP. כתובות IPv4 מוצגות באמצעות ארבעה בתים.
- **מזהה רשת ID (Network ID)** - חלק בכתובת לוגית (כתובת IP) המציין לאיזו רשת שייכת הכתובת.
- **מזהה ישות (Host ID)** - חלק בכתובת לוגית המציין לאיזה כרטיס רשת שייכת הכתובת, בתוך הרשות.

- **_subnet Mask (מסיכת רשת)** - מגדיר כמה ביטים (bits) מתוך כתובת ה-IP מייצגים את מזהה הרשת.
- **Broadcast** - כתובת המציינת כי החבילה צריכה להשלח אל כל הישויות ברשת.
- **Loopback** - כתובת המציינת שהחbillה לא צריכה לעזוב את כרטיס הרשת, אלא "להשאר במחשב".
- **נתב (Router)** - רכיב רשמי בשכבה שלישית. מטרתו היא לקשר בין מחשבים ורשתות ברמת ה-IP. רוב מלאכת הניתוב מתבצעת בידי נתבים.
- **טבלת ניתוב (Routing Table)** - טבלה הכוללת מזהי רשת ולאן להעביר חבילות המיועדות למזהה רשת אלו. ברוב המקרים, הטבלה היא דינמית ועשיה להשתנות בהתאם למצב הרשת. נתבים, וגם רכיבים אחרים, משתמשים בטבלאות ניתוב בכדי לדעת לאן להעביר את החבילות המגיעות אליהם.
- **route** - כל' המאפשר להסתכל על טבלאות ניתוב ולערוך אותן.
- **Default Gateway** - הנתב המשויך אל רכיב כלשהו. כל חבילה שלא התאימה על חוק ספציפי בטבלת הניתוב, תשלח אל ה-Default Gateway.
- **(ICMP) Internet Control Message Protocol** - פרוטוקול בשכבה השלישית, הנועד למציאת תקלות ברשת ולהבנת מצב הרשת.
- **TTL (Time To Live)** - שדה ב-IP Header שמצוין כמה Hops החבילה עוזר עברו לפני שהיא תעבור בטרם תיזרק. כל נתב או רכיב אחר שמעביר את החבילה הלאה מחסיר 1 מערך השדה.
- **(DHCP) Dynamic Host Configuration Protocol** - פרוטוקול המשמש להקצאה דינמית של כתובות IP וללמידה של פרטי הרשת.
- **MTU (Maximum Transmission Unit)** - מאפיין של רשת המתאר את הגודל המקסימלי של גוש מידע שיכול לעבור ברשת זו.
- **פרגמנטציה (Fragmentation)** - חלוקה של חבילת מידע למקטעים קטנים יותר, מתבצע בכך לשילוח חבילות הגדולות יותר מה-MTU.
- **כתובות IP פרטיות** - שלושה טוווחי כתובות IP שהוקצו בידי IANA על מנת לחסוך בכתובות IP בעולם. בתוך רשתות מקומיות, ישויות יכולות לקבל כתובות פרטיות, שיזרו אותן בתוך הרשת בלבד, ולא בעולם החיצוני. כתובות אלו אין ניתנות לניתוב. מכאן שנتاب באינטרנט שרוואה חבילה המיועדת לכתובת פרטית עתיד "לזרוק" אותה.
- **NAT (Network Address Translation)** - דרך להעביר מידע בין ישויות בעלות כתובות IP פרטיות לישות מחוץ לרשת. לשם כך, רכיב NAT מחליף את כתובת ה-IP של הישות בעלת כתובת ה-IP הפרטית בכתובת ה-IP של רכיב ה-NAT עצמו, לו יש כתובת IP חיונית שנייה לנtab.
- **IPv6** - הגרסה החדשה של פרוטוקול IP. כתובות בגרסה זו של הפרוטוקול הן באורך 16 בתים .(bytes)

פרק 8 - שכבת הקן

- **Ethernet** - פרוטוקול של שכבת הקן, בו משתמשים כרטיסי רשת מסוג Ethernet (המחוברים באופן קווי).
- **כטבota MAC** - כטבות פיזיות של שכבת הקן. לכל כרטיס רשת יש כטבota צזו. כטבota MAC בפרוטוקול Ethernet הינה בגודל שישה בתים (bytes).
- **מסגרת (Frame)** - גוש מידע בשכבה השנייה.
- **ARP (Address Resolution Protocol)** - פרוטוקול שנועד למפות בין כטבות לוגיות של שכבת הרשת לכטבות פיזיות של שכבת הקן.
- **פורט (Port)** - " כניסה" ברכיב רשת אליה ניתן לחבר לקבל רשות. הערה: על אף שמדובר באותו השם כמו פורטים של שכבת התעבורה, המשמעות שונה.
- **Hub (ריכוזת)** - רכיב של השכבה הפיזית, השכבה הראשונה. הוא נועד כדי לחבר כמה ישויות רשות ייחד. ה-Hub איננו מכיר כטבות Ethernet או IP, מבחינתו הוא רק מעביר זרם חשמלי מפורט אחד אל פורטים אחרים. כאשר מחשב שמחובר ל-Hub שולח מסגרת, ה-Hub מעתק את המסגרת ושולח אותה לכל הפורטים שלו, מלבד לזה שמננו המסגרת נשלה.
- **Switch (מtag)** - רכיב של שכבת הקן, השכבה השנייה. אי לך, ה-Switch מכיר כטבות MAC, מבין את המבנה של מסגרות בשכבה שנייה (למשל מסגרות Ethernet), וידע לחשב checksum. לאחר ש-*Switch* למד את הרשות, הוא מעביר מסגרת מהפורט בה הוא קיבל אותה אל הפורט הרלוונטי בלבד.
- **התנגשות (Collision)** - מצב בו שתי ישויות (או יותר) משדרות בערזץ המשותף בו זמן. במקרה זה, המידע שנשלח יגיע באופן משובש - כלומר, המידע שיגיע אל הוא לא יודע שהישות התכוונה לשולוח.
- **Multicast** - כטבota מסווג זה שיוכות ליותר מישות אחת.
- **Unicast** - כטבota מסווג זה שיוכת לישות אחת בלבד.

פרק 10 – השכבה הפיזית

- **סיבית (Bit)** - קיצור של המושג ספירה בינארית, ספרה שיכולה להציג אחד משני ערכים: 0 או 1. סיבית היא תרגום של המושג הלועזי bit, שהוא קיצור לביטוי binary digit.
- **תוֹר (Medium)** - בשרותות תקשורת, תוֹר התקשרות הוא החומר, או האמצעי הפיזי, המשמש להעברת המידע.
- **קידוד (Encoding או Coding)** - תהליך בו מידע מתורגם לאותות מוסכמים. כל שיטת מימוש של השכבה הפיזית מגדרה באופן בו מתרגמים את הספרות 0 ו-1 לסימנים מוסכמים על גבי התוֹר.
- **גל (Wave)** - התפשטות (או התקדמות) של הפרעה מחזוריית בתוֹר למרחב. גל יכול לנوع בחומר (כמו גלים במים), אך גם באוויר (כמו גל קול) ואף בvakuum (כמו גלים אלקטרומגנטיים, שיכולים לנوع בvakuum ובתווים רבים אחרים).

- **הgal האלקטרומגנטי (Electromagnetic Wave)** - הוא סוג של gal שבע בתווים שונים במרחב (אוויר, מים, זכוכית, ריק/וакום, ועוד) באמצעות שינוי של השדות החשמליים והמגנטיים. האור שmagיע מהמשש ושהותו אנו רואים הוא gal האלקטרומגנטי שהתדר שלו נמצא בתחום שהען רואה (400 עד 800 Hz). עוצמת האור שווה לאmplיטודה של gal האלקטרומגנטי, שמעידה על חזק התנוודת בשדות החשמליים והמגנטיים.
- **Modulation (אפנון)** - היא העברת מידע על גבי gal נושא. הרעיון באפנון הוא להרכיב gal של מידע (כגון gal קול של מוסיקה) על גבי gal "נושא". gal נושא הוא gal " חלק" (gal שמאוד קרוב לפונקציית סינוס) בתדר גבוה יותר מגל המידע.
- **אפנון מבוסס אמפליטודה (Amplitude Modulation)** - בשיטת אפנון זו, "מרכיבים" gal של מידע בתדר נמוך על גבי gal "נושא" בתדר גבוה וקבוע. בשיטה זו, האםפליטודה של gal הנושא (בתדר הקבוע) תשתנה לפי האmplיטודה של gal המידע.
- **אפנון מבוסס תדר (Frequency Modulation)** - בשיטת אפנון זו, משנה את התדר של gal הנושא על פי האmplיטודה של gal המידע.
- **מודם (Modem)** - קיצור (באנגלית) של Modulator & Demodulator - מכשיר שמאפן ומחזר ביטים על גבי ערוץ תקשורת.
- **Duplex (דופלקס)** - מאפיין של מערכות תקשורת דו כיוניות בין שתי נקודות. מערכת שהיא Half Duplex מאפשרת לשני צדדים לתקשר אחד עם השני באופן דו כיווני אך לא סימולטני. דוגמא למערכת Half Duplex היא ווקי-טוקי (מכשיר קשר אלחוטי), בו רק צד אחד יכול לדבר בזמן שהצד השני מקשיב. כשה שני הצדדים מנסים לדבר, אף אחד לא שומע את השני. מערכת שהיא Full Duplex מאפשרת לשני הצדדים לתקשר אחד עם השני באופן מלא וסימולטני, זאת אומרת שני הצדדים יכולים לדבר באותו הזמן. הטלפון הוא דוגמא למערכת Full Duplex, לאחר והוא מאפשר לשני דוברים לדבר בו זמנית וגם לשמוע אחד את השני.
- **כבל 5 CAT** - כבל שמאגד בתוכו 4 כבלי זוגות, כל זוג בצבע שונה. אפשר לראות את כבלי הזוגות המלופפים סביב עצם בתמונה העילונה. אם לדייך, ישנו מספר סוג כבליים אלו: CAT 3, CAT5e, CAT6. מבחוץ הם כולם נראים אותו דבר, אך מבפנים הם נבדלים באיכות בידוד ההפרעות החשמליות. איכות הבידוד משפיעה על קצב העברת הביטים. כשאתם הולכים לחנות לקנות כבל רשת, ברוב המקנים תצטרכו כבל CAT5.
- **חיבור RJ-45** - השקע והתקע של כבלי הרשת הסטנדרטיים, כולל צורותם וסידור הכבלי הפנימיים לפוי צבע, מוגדר בטקן שנקרא RJ-45. עברו כבלי רשת Ethernet, התקן הוא RJ-45, אך ישנו תקנים דומים גם עברו כבלי אחרים כגון כבל הטלפון (RJ-11). יש לציין שהיבור זה נקרא גם חיבור P8C8, ובמקומות בהם כך הוא נקרא, הכוונה היא לאותו סוג חיבור כמו RJ-45.
- **תקן Base-T10** - תקן זה מגדיר כיצד משתמשים בכבלי CAT5 וחיבור RJ כדי להעביר בית בודד על גבי הcabl.

- **כבל רשת מוצלב (Ethernet Crossover Cable)** - הינו כבל רשת בו כבלי הזוגות של השליחה וקבלת הוצלבו, דבר המאפשר לחבר שני מחשבים ישירות אחד לשני, ללא Switch ביניהם.
- **תקשרות מיקרוגל (Microwave Transmission)** - העברת מידע באמצעות גלים אלקטרומגנטיים בתווך אוויר גל שנייתן למרחקים בסנטימטרים. גלי מיקרוגל הם גלים בטווח התדרים בין 1GHz ל-30GHz.
- **סיב אופטי (Optical Fiber)** - הינו סיב עשוי זכוכית או פלסטייק, המאפשר העברת אור בתוכו למרחקים ארוכים עם אובדן מינימלי של עוצמה.
- **ממסר (Relay)** - רכיב שמקבל אות תקשורתית, מגביר אותה ומשדר אותה להלאה. תפקידו להאריך את המרחק אליו ניתן להעביראות תקשורתית.

פרק 14 - פקודות ו כלים

פרק זה כולל כלים ופקודות בהם נעשו שימוש לאורכו הספר, המטרה והשימוש בהם. הפרק נועד לשיער לקורא להתמצא כיצד להשתמש בכל מוטוים, או לקוראים המעניינים להכיר את החלופות לפקודות המוצגות מעל מערכת הפעלה מבוססת UNIX.

הרשימה

שימוש ב-UNIX	שימוש ב-Windows	הין הציג בספר	מטרת הכלי
ping -c 4 www.google.com	ping -n 4 www.google.com	<u>תחלית מסע - איך עובדהאינטרנט?/ כתובות IP</u>	לבודק קשריות, לשוט מרוחקת, ואת הזמן שלוקח לחבילה להגעה אליה ובחרזה.
traceroute -n www.google.com	tracert -d www.google.com	<u>תחלית מסע - איך עובדהאינטרנט? ענן האינטראנט</u>	למצוא את הדרך שעוברת חבילה בין המחשב שלי לנקודות קצה שונות.
nslookup www.google.com	nslookup www.google.com	<u>תחלית מסע - איך עובדDNS/האינטרנט? DNS</u>	לבצע תשאול DNS.
telnet google.com 80	telnet google.com 80	<u>שכבה האפליקציה/ התבוננותמודרכת בתגובה HTTP</u>	התחברות כלוקוט לשירות מרוחק.
nslookup set type=<TYPE> <host/address>	nslookup -t<TYPE> <host/address>	<u>שכבה האפליקציה/ תרגיל 4.15 -תשאול רשומות מסווגים שונים</u>	תשאול DNS עם סוג שאלתה מסוים.
תליי בגירסה הספציפית. הסביר ניתן למצוא כאן: http://goo.gl/AF0Ui3	ipconfig /flushdns	<u>Scapy /תרגיל 5.1 מודרך - הסופה של DNS</u>	לאפס את מתמונן רשומות ה-DNS.
netstat -na	netstat -na	<u>שכבה התעבורה/ תרגיל 6.1 מודרך -אילו פורטים פתוחים במחשב שלי?</u>	לספק מידע על חיבור רשת, ספציפית חיבור בשכבה התעבורה.
ifconfig -a	ipconfig /all	<u>שכבה הרשת/מה כתובת ה IP- של?</u>	להציג מידע על כרטיסי רשת.

שימוש ב-UNIX	שימוש ב-Windows	הין הציג בספר	מטרת הכל
route -n	route print	<u>שכבה הרשת/ מהי טבלת הניתוב שלי?</u>	להציג טבלאות ניתוב.
dhclient -r	ipconfig /release	<u>שכבה הרשת/ תרגיל 7.8 מודרנ</u> <u>-קיבלה IP באמצעות DHCP</u>	להתנתק משרת DHCP-וילוותר על פרטי הרשת.
dhclient	ipconfig /renew	<u>שכבה הרשת/ תרגיל 7.8 מודרנ</u> <u>-קיבלה IP באמצעות DHCP</u>	לקבל את פרטי הרשת משרת ה-DHCP.
arp -n	arp -a	<u>שכבה הקו/ מטמון (Cache) של ARP</u>	להציג את מטמון ARP.
arp -d 192.168.1.100	arp -d 192.168.1.100	<u>שכבה הקו/ מטמון (Cache) של ARP</u>	למחוק רשומה ממטמון ARP.

זכויות יוצרים - מקורות חיצוניים

- <http://commons.wikimedia.org/wiki/File:World%20first%20dual-core%20smartphone%20comes%20to%20europe.jpg>
- http://commons.wikimedia.org/wiki/File:Ethernet_RJ45_connector_p1160054.jpg
- <http://openclipart.org/detail/189964/pipe-by-barretr-189964>
- <https://www.flickr.com/photos/ckelly/4846654926>
- <https://www.flickr.com/photos/topgold/4399244167>
- <http://pixabay.com/en/basket-buy-order-shopping-green-156678/>
- <http://pixabay.com/en/buttons-shopping-cart-buy-24573/>
- <http://pixabay.com/en/computer-desktop-keyboard-system-98400/>
- <http://pixabay.com/en/envelope-e-mail-letter-mail-post-154134/>
- <http://www.troyjessup.com/headers/>
- [http://www.clker.com/cliparts/0/a/6/b/12065771771975582164reporter flat.svg.med.png](http://www.clker.com/cliparts/0/a/6/b/12065771771975582164reporter%20flat.svg.med.png)
- https://www.iconfinder.com/icons/23912/router_wifi_icon#size=128
- https://www.iconfinder.com/icons/47998/history_qualification_icon#size=128
- <http://www.opensecurityarchitecture.org/cms/library/icon-library>
- <http://www.clipartbest.com/free-computer-clipart>
- <http://www.iconarchive.com/show/pretty-office-9-icons-by-custom-icon-design/Magnifying-glass-icon.html>
- <http://www.iconarchive.com/show/icons8-metro-style-icons-by-visualpharm/Ecommerce-Idea-icon.html>
- <http://www.iconarchive.com/show/aerial-icons-by-chromatix/work-icon.html>
- <http://www.iconarchive.com/show/my-seven-icons-by-itzikqur/Videos-1-icon.html>
- http://commons.wikimedia.org/wiki/File%3AInternational_Morse_Code.PNG
- http://commons.wikimedia.org/wiki/File:2006-01-14_Surface_waves.jpg#mediaviewer/File:2006-01-14_Surface_waves.jpg
- http://commons.wikimedia.org/wiki/File:Amplitude-modulation_he.svg
- http://commons.wikimedia.org/wiki/File:Frequency_Modulation.svg#mediaviewer/File:Frequency_Modulation.svg
- http://commons.wikimedia.org/wiki/File:TP_Neostrada_Thomson_SpeedTouch_.546.jpg

[http://commons.wikimedia.org/wiki/File:2.4_GHz_Wi-Fi_channels_\(802.11b,g_WLAN\).svg#mediaviewer/File:2.4_GHz_Wi-Fi_channels_\(802.11b,g_WLAN\).svg](http://commons.wikimedia.org/wiki/File:2.4_GHz_Wi-Fi_channels_(802.11b,g_WLAN).svg#mediaviewer/File:2.4_GHz_Wi-Fi_channels_(802.11b,g_WLAN).svg)

<http://commons.wikimedia.org/wiki/File:Wave-he.png#mediaviewer/%D7%A7%D7%95%D7%91%D7%A5:Wave-he.png>

http://commons.wikimedia.org/wiki/File:CAT5e_Cable.jpg#mediaviewer/File:CAT5e_Cable.jpg

http://commons.wikimedia.org/wiki/File:Cat_5.jpg

http://commons.wikimedia.org/wiki/File%3AEthernet_MDI_crossover.svg

http://commons.wikimedia.org/wiki/File:Parabolic_antennas_on_a_telecommunications_tower_on_Willans_Hill.jpg

http://commons.wikimedia.org/wiki/File:Multimode_stepindex_optical_fiber.svg

© 2012 Google Inc. All rights reserved. Google and the Google Logo are registered trademarks of Google Inc.

© 2012 Google Inc. All rights reserved. YouTube™ is a trademark of Google Inc.

© 2012 Google Inc. All rights reserved. Chrome™ browser is a trademark of Google Inc.

Python and the Python logos are trademarks or registered trademarks of the Python Software Foundation, used with permission from the Foundation.