



# רשות מחשבים

עומר רוזנבוים ■ שלומי הוד



מינהל הסייבר הצבאי

[www.cyber.org.il](http://www.cyber.org.il)

# רשותות מחשבים

גרסה 2.24

## 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 הסיבר הצה"ל.

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

© תשע"ו – 2016. כל הזכויות שמורות למטה הסיבר הצה"ל.

הודפס בישראל.

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



## תוכן עניינים

3	תוכן עניינים .....
4	מבוא .....
10	מדריך לתלמידים: כיצד נכנסים לפורום רשותות הארץ .....
15	תוכן עניינים מצגות רשותות .....
17	פרק 1 - תחילת מסע - איך עובד האינטרנט? .....
33	פרק 2 - תכנות ב-Sockets .....
54	פרק 3 - Wireshark ומודל חמש השכבות .....
89	פרק 4 - שכבת האפליקציה .....
144	פרק 5 - Scapy .....
168	פרק 6 - שכבת התעבורה .....
229	פרק 7 - שכבת הרשת .....
295	פרק 8 - שכבת הקו .....
337	פרק 9 - רכיבי רשת .....
341	פרק 10 - השכבה הפיזית (העשרה) .....
367	פרק 11 - איך הכל מתחבר, ואייר עובד האינטרנט? .....
390	פרק 12 - תכנות Sockets מתקדם: ריבוי משתמשים (הרחבה) .....
411	פרק 13 - מיליון מושגים .....
420	פרק 14 - פקודות וכליים .....
422	זכויות יוצרים - מקורות חיצוניים .....

## מבוא

רשתות תקשורת הן דבר מדהים. מАЗ ומעולם, בני אדם רצו להעביר מסרים ביניהם. בעבר הרחוק, בהיעדר אמצעים אחרים, התבוססה התקשרות בעיקר עלתקשרות 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 התבוססה על תקשורת מילולית, ועד לשימוש היום יומי בגלישה באינטרנט, בדואר אלקטרוני או בתוכנות העברת מסרים כגון 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 בחיננו.

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 רשות של נתני רשתות. לכל רשות יש את המאפיינים שלה: ישן רשות קטנות (בנה 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 לכדי 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 למודע עצמאי, 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 אותנו בהחלטה אלו נושאים יכולו בספר זה ואלו ישארו מחוצה לו - הספר מתמקד באוטם נושאים אשר ניתן לישם וلتתגלג בקהלות יחסית. הספר כולל מעט מאוד נושאים שהם תיאורתיים בלבד, ורובה ככלו מתעסק במושגים אמתיים 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 עצמו. כדי לרכוש שליטה בחומר הלימוד, יש לבצע את כל התרגילים ולודאו שאתם מבינים אותם. התרגילים המודרכים נבנו כך שהם מפורקים למושימות מוגבלות המתבססות זו על זו ומונחות שהלומד מבצע בפועל את ההנחות. אל תשתפקו 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 הקריאה 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](#) אשר 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 הוספנו 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 הקריאה. בכל פעם שהיא קישור לסרטון, יופיע גם קוד QR שיאפשר להגיע אליו בקהלות. כמו כן, אתם מוזמנים לפנות לרשימת הסרטונים המלאה בכתבota:

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

## התקנות נדרשות

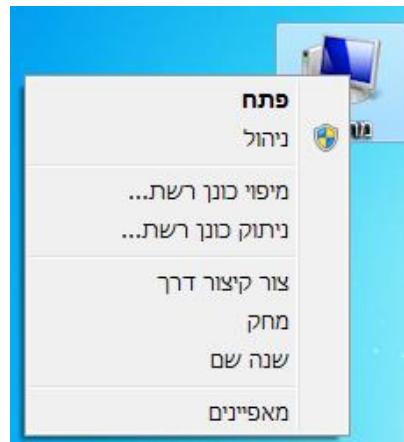
כאמור ספר הלימוד מבוסס על שפת פיתון. ישן התקנות רבות של פיתון, لكن נרצה להמליץ על סביבת העבודה ושימוש נכון בסביבת העבודה.

מומלץ להתקין פיתון חלק מסביבת התקנות של גבהים, שפותחה על ידי אורי לוי ושנמצאת בקישור:

[www.cyber.org.il/workspace/gvahim\\_64bit.exe](http://www.cyber.org.il/workspace/gvahim_64bit.exe)

[www.cyber.org.il/workspace/gvahim\\_32bit.exe](http://www.cyber.org.il/workspace/gvahim_32bit.exe)

כיצד לדעת אם מערכת הפעלה שלכם היא 32 או 64 ביט? לרבות היא תהייה 64 ביט. כדי לוודא זאת, הקליקו קליק ימני על איקון "המחשב שלי" שבdesktop ובחנו "מאפיינים" (או properties).



במסך שייפתח תוכלו לראות תחת "סוג מערכת" האם המערכת שלכם היא 32 או 64 ביט

מערכת	דרוג:
Windows 4.6	דרוג:
מעבד: Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 3.10 GHz	מעבד:
זיכרון מותקן (RAM): 4.00 GB	זיכרון מותקן (RAM):
סוג מערכת: מערכות הפעלה של 64 סיביות	סוג מערכת:
עט ומגע: אין קלט עט או קלט מגע חומניים עבור צג זה	עט ומגע:

על מנת להתקין את הסביבה יש לבצע את הצעדים הבאים:

- צרו תיקיה חדשה, מומלץ `gvahim\gvahim`:
- הריצו את הקובץ `gvahim.exe`. קובץ זה יחלץ את קבצי ההתקינה אל הספרייה שנבחרה, `gvahim\c:\`.
- בתיקייה שנוצרה נמצא קובץ `install.exe`, יש להריץ אותו כמנהל.
- קליק ימני על העכבר "הרצ כמנהל" או "run as administrator". לאחר מכן יפתח חלון cmd ויתחיל להריץ את ההתקנות באופן שקט.
- אחות ההתקנות תציג את התשובותיכם. יש ללחוץ `Next <Install>->Agree <Finish`, מכאן ואילך הסקורייפט ימשיך לבד. בסיום הריצת סקורייפט ההתקינה, הוא דואג לנקיות את הסביבה.
- הפעילו מחדש את המחשב

גרסת הפיתון של ההתקנה היא 2.7 - בהתאם לספר הלימוד. התקינה זו כוללת את כל הכלים הנוספים שנוצרו להשתמש בהם במהלך הלימוד (`wireshark`, `scapy`, `cagan`).

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

לאחר ההתקינה יהיו ברשותכם 2 שיטות שונות להריצת פיתון. נסקור אותן:

- דרך `command-line`: שיטה זו מתאימה לבדיקת דברים קטנים בפייתון, כגון פעולות מתמטיות, ביצוע `help` או `dir`. מדובר במקרה לתוכנת קוד פיתון של יותר משורות בודדות בדרך זו.
- דרך PyCharm: סביבת עבודה שמצויה ללימוד והתרנסות. היתרונות המרכזים שלה – `debugging` באמצעות `breakpoints` וחיפוי על שגיאות תכנות (כגון שכחה של נקודותים, טעות בשמות פונקציות, כמוות לא נכונה של פרמטרים לפונקציה וכו'). והוא סביבת העבודה המומלצת ללימוד ולכתיבת תוכניות. מומלץ להיעזר בלימוד `pycharm` במצגת ההדריכה מאת ברק גונן. לינקים לכל מצגות פיתון ורשאות נמצאים בהמשך.

### アイיקונים

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



שאלה.



הגדירה למונח.



הדגשת נקודה חשובה.



"תרגיל מודרני". עליים לפתרו תרגילים אלו תוך כדי קריית ההסבר.



תרגיל לביצוע. תרגילים אלו עליים לפתרו בעצמכם, והפתרון בדרך כלל לא יוצג בספר.



פתרון מודרן לתרגיל אותו היה עליים לפתרו בעצמכם.



רקע היסטורי, או מידע העשורי אחר.



הפניה לסרטון.

### אבטחת מידע

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 ליוותה את הספר מטהlixir התכנון ועד לשלביו הסופיים והשפיעה עלייו רבות. תומר גלון מימוש את הצד השרת עבור תרגילים במהלך הספר. מילל לשם סיעה באופןמשמעותי בהבאת הספר לידי גרסה להפצה. משובים שקיבלו על הספר לאורך הזמן שייפורו אותו ותרמו לו מאוד. באופן מיוחד אנו מבקשים להודות לiosis ממו וממן עבורי על העורחותם הבונות. יהוא איזנרייך כתב

פתרונותות 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 בתחום יצרת הספר – תודה רבה.

עומר רוזנבוים

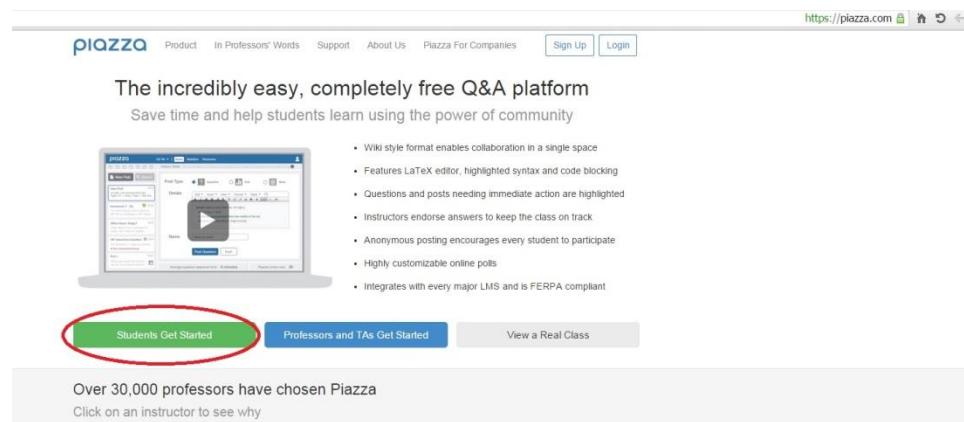
שלומי הود

## מדריך לתלמידים: כיצד נכנסים לפורום רשותה הארץ

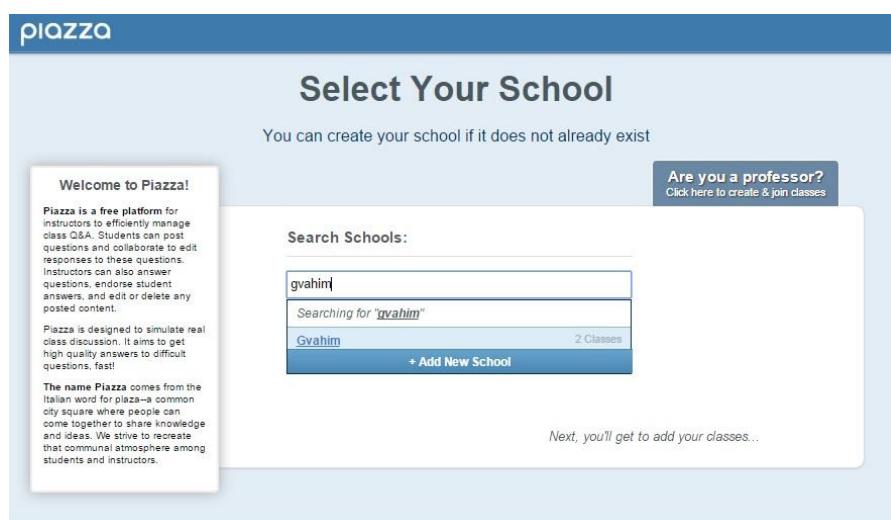
Piazza היא פורום שאלות ותשובות שמיועד לתלמידים בכל הארץ. ניתן לפתח פורום בכל נושא, לכתוב שאלה ולענות תשובות. כיוון שתלמידים בכיתות גבהים בכל הארץ נתקלים בעיות דומות, עומדת לרשותכם "אוניברסיטתה" וירטואלית, שם תוכלו לשיע אחד לשני. בנוסף, התשובות שלכם יסייעו לתלמידים בשנים הבאות!

לכינוסה ל-Piazza:

1. היכנסו לאתר [www.piazza.com](http://www.piazza.com) ובחרו "students get started"



2. בחרו באוניברסיטה "gvahim"



.3 בחרו סטטוס "other"

Welcome to Piazza!

Piazza is a free platform for instructors to efficiently manage class Q&A. Students can post questions and collaborate to edit responses to these questions. Instructors can also answer questions, endorse student answers, and edit or delete any posted content.

The name Piazza comes from the Italian word for plaza—a common city square where people can come together to share knowledge and ideas. We strive to recreate that communal atmosphere among students and instructors.

Gvahim (change school)

Selected Term: Summer 2015

Summer 2015

Class 1: Class 1:  
Class 2: Class 2:  
Class 3: Class 3:  
Class 4: Class 4:  
Class 5: Class 5:

Add Another Class

Join Classes

.4 בחרו קורס: Assembly, Python, Networks, Operation Systems

Welcome to Piazza!

Piazza is a free platform for instructors to efficiently manage class Q&A. Students can post questions and collaborate to edit responses to these questions. Instructors can also answer questions, endorse student answers, and edit or delete any posted content.

The name Piazza comes from the Italian word for plaza—a common city square where people can come together to share knowledge and ideas. We strive to recreate that communal atmosphere among students and instructors.

Gvahim (change school)

Selected Term: Other

Summer 2015

Class 1: asse  
Class 2: Searching for "asse"  
Class 3: 1: Assembly 76 Enrolled  
Class 4: Class 4:  
Class 5: Class 5:

Add Another Class

Join Classes

.5 בחרו להצטרף כ-student והכניסו סיסמה. הסיסמאות לקורסים השונים הן:

Assembly -

Python -

Networks -

OS -

Welcome to Piazza!

Piazza is a free platform for instructors to efficiently manage class Q&A. Students can post questions and collaborate to edit responses to these questions. Instructors can also answer questions, endorse student answers, and edit or delete any posted content.

The name Piazza comes from the Italian word for plaza—a common city square where people can come together to share knowledge and ideas. We strive to recreate that communal atmosphere among students and instructors.

**Gvahim** (change school)

Selected Term: **Other**

Summer 2015

**Class 1:** 1: Assembly (edit)

Instructors: Barak Gonen · 76 Enrolled

Join as:  Student  TA  Professor

**Class Access Code:** assembly

**Class 2:** [empty] X

**Class 3:** [empty] X

**Class 4:** [empty] X

**Class 5:** [empty] X

Add Another Class

Join Classes

6. הכניסו כתובת מייל, אליה ישלח קוד הפעלה של האתר, וליחסו על submit email

Please enter your email address

Email: test@test.com

Confirm Email: test@test.com

Submit Email

Unable to sign up? Email us at team@piazza.com and we'll help you get started!

.7. הכניסו את הקוד שקיבלתם ולחצו על submit

The screenshot shows the Piazza sign-up process. At the top, it says "Selected Term: Summer 2015". Below that, it lists a class: "1. 1: Assembly Instructors: Barak Gonen - 77 Enrolled" with a checked checkbox for "Joining as Student". A blue "Join Classes" button is below. In the center, there's a message: "We see you're new to Piazza! Check your inbox for your confirmation email. Enter the validation code below so you can access your classes!" Below this is a "Validation Code:" input field containing "iz9TeY9ShhD", which is circled in red. Next to it is a "Submit Code" button. To the right, there's a "Not Getting Our Email?" section with instructions and a help email address.

.8. בחרו שם משתמש וסיסמה. למטה בחרו באופציה I am not pursuing a degree ואשרו שקראותם את תנאי השימוש. לאחר מכן לחזו על Continue - זהו, ס"י מתם. ברוכים הבאים.

The screenshot shows the "Finish setting up your Piazza account" page. It has sections for "Account Information" (Full Name: test test, Choose Password: \*\*\*\*\*, Confirm Password: \*\*\*\*\*), "Contact us at team@piazza.com with any questions", and "Academic Information" (Graduate Program dropdown, Major dropdown, Anticipated Completion dropdown). Under Academic Information, there are checkboxes for "I have two majors" and "I'm not pursuing a degree" (which is checked and circled in red). Below these is a note about information usage. At the bottom, there's a "Continue to Piazza" button.

### הכנסת שאלת חדשה

על מנת לשאול שאלה, הקישו על post new. בחלון שייפתח בחרו

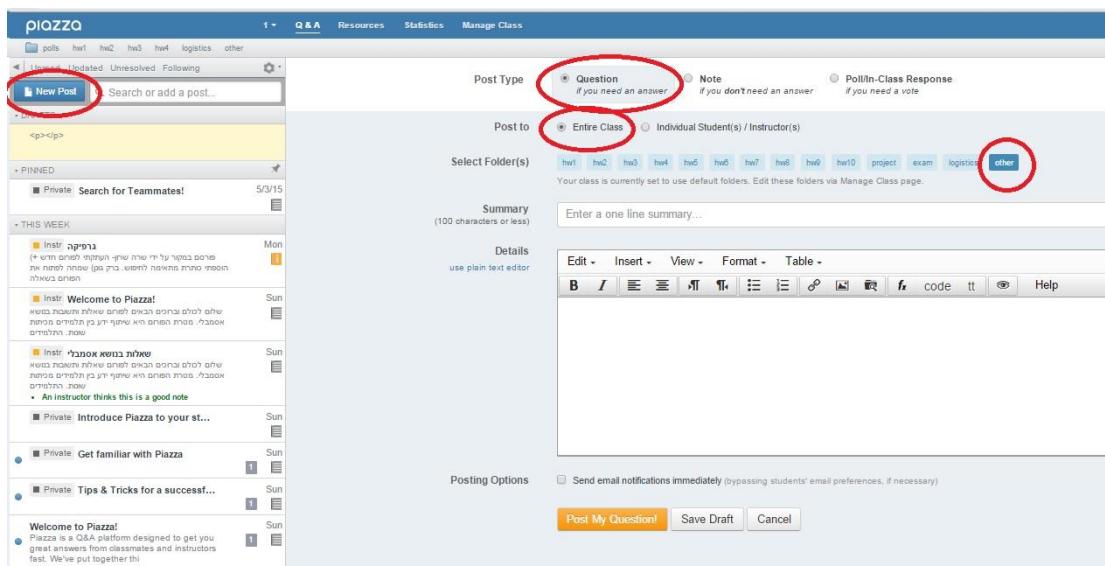
- Post type = Question

- Post to = Entire class

- Folder = Other

אל תשכו לתת כותרת לשאלת - באנגלית כדי שניתן יהיה למצוא את השאלה בחיפושים בעתיד - ולבסוף הקישו

על Post my question



## תיקון עניינים מצגות

**פתרונות:**

Before we start:

<http://cyber.org.il/python/1450-3-00.pdf>

Intro and CMD:

<http://cyber.org.il/python/1450-3-01.pdf>

Pycharm:

<http://cyber.org.il/python/1450-3-02.pdf>

Variables, conditions and loops:

<http://cyber.org.il/python/1450-3-03.pdf>

Strings:

<http://cyber.org.il/python/1450-3-04.pdf>

Functions:

<http://cyber.org.il/python/1450-3-05.pdf>

Assert:

<http://cyber.org.il/python/1450-3-06.pdf>

Files and script parameters:

<http://cyber.org.il/python/1450-3-07.pdf>

Lists and tuples:

<http://cyber.org.il/python/1450-3-08.pdf>

Dictionaries:

<http://cyber.org.il/python/1450-3-09.pdf>

Object Oriented Programming:

<http://cyber.org.il/python/1450-3-10.pdf>

Utilities and exceptions:

<http://cyber.org.il/python/1450-3-11.pdf>

Regular expressions:

<http://cyber.org.il/python/1450-3-12.pdf>

**רשומות:**

<http://cyber.org.il/networks/1450-2-00.pdf>

מבוא לשנת הלימודים:

<http://cyber.org.il/networks/1450-2-01.pdf>

פרק 1- מבוא לרשומות מחשבים:

<http://cyber.org.il/networks/1450-2-02.pdf>

פרק 2- תכנות סוקטים:

<http://cyber.org.il/networks/1450-2-03.pdf>

פרק 3א- מודל חמש השכבות:

<http://cyber.org.il/networks/1450-2-04.pdf>

פרק 3ב- wireshark :

<http://cyber.org.il/networks/1450-2-05.pdf>

פרק 4א- שכבת האפליקציה :HTTP

<http://cyber.org.il/networks/1450-2-06.pdf>

פרק 4ב- HTTP נושאים מתקדמים:

<http://cyber.org.il/networks/1450-2-07.pdf>

פרק 4ג- פרוטוקול DNS :

<http://cyber.org.il/networks/1450-2-08.pdf>

פרק 4ד- אבחון פרוטוקול SMTP :

<http://cyber.org.il/networks/1450-2-09.pdf>

פרק 5- Scapy :

<http://cyber.org.il/networks/1450-2-10.pdf>

פרק 6א- שכבת התעבורה :

<http://cyber.org.il/networks/1450-2-11.pdf>

פרק 6ב- מבוא לפרוטוקולים של שכבת התעבורה :

<http://cyber.org.il/networks/1450-2-12.pdf>

פרק 6ג- UDP :

<http://cyber.org.il/networks/1450-2-13.pdf>

פרק 6ד- TCP :

<http://cyber.org.il/networks/1450-2-14.pdf>

פרק 7א- שכבת הרשת מבוא לניטוב :

<http://cyber.org.il/networks/1450-2-15.pdf>

פרק 7ב- כתובות IP וראוטר :

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

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

## פרק 1 - תחילת מסע - איך עובד האינטרנט?



ניתן לצפות סרטון המלאה את פרק זה בכתב: <http://youtu.be/ad8EOsXFuxE>

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

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



**מה קורא לנו גולשים לאתר Facebook?**

רובנו גלשו ל-Facebook, הרשת החברתית העצומה שמונה מעל ל-מיליארד משתמשים. אך האם ערכנו לשאול את עצמנו - מה בעצם קורא מאחורי הקלעים כshawim? איך יתכן לנו נמצאים בבית, מקישים בדף (Browser) את הכתובת "www.facebook.com", לוחצים על מקש ה-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 **ל אחסון (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 תוצג רשימת החברים, היקן יציג העדכנים שלהם, איפה יציג הפרסומות ועוד. האתר Facebook גם **לאימות (Authentication)** - עליו לזהות את המשתמש שפונה אליו. לשם כך, Facebook צריך לזכור את כל המשתמשים והסימאות שלהם, ולאחר מכן לשלוחו אליהם. כמו כן, Facebook זקוק **לקשרות**.(Clomar) צריכה להיות דרך שתאפשר לאדם לתקשר עם האתר של Facebook בין אם מהמחשב שלו בישראל, ובין אם מהסמארטפון שלו כשהוא נמצא בטיחול באיטליה.

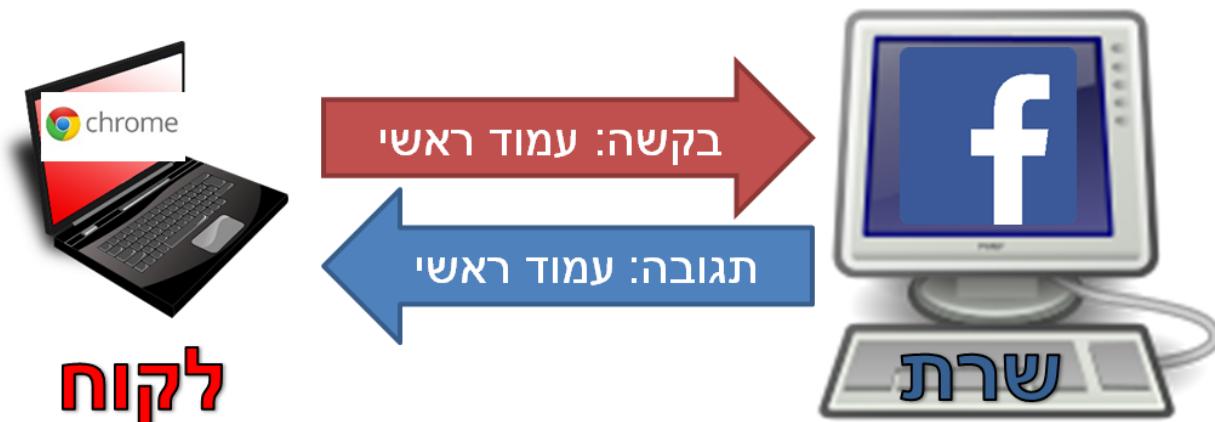
ספר זה יתמקד בתקשורת שבין רכיבים אלקטרוניים שונים ודרך העברת המידע ביניהם. בפרק הנוכחי, נתאר באופן כללי מה קורא מהרגע שאנו כתובים "www.facebook.com" בדף ולוחצים על מקש ה-Enter, ועד שמופיע דף הבית של Facebook על המסך.

### WWW – World Wide Web

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

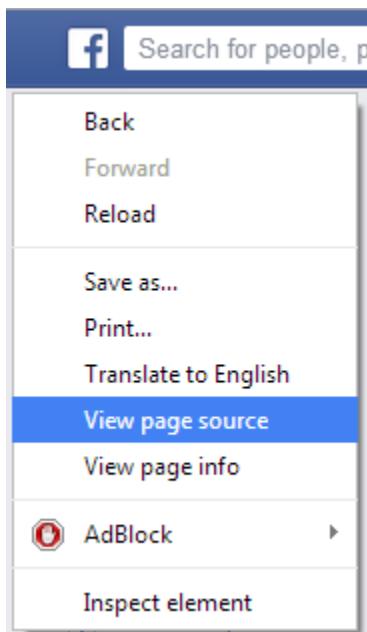
של קורי עכבי ש המקצועיים זה זהה. עמודי האינטרנט מפוזרים בכל רחבי העולם - World Wide - בעוד אחד נמצא בחיפה, שני יכול להיות בטוקיו ושלישי בניו-יורק. העמודים השונים מושרים ביניהם באמצעות לינקים.

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. בצד 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](http://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 בaczor ריק באתר האינטרנט, ולחזור באמצעות "View page source" (בעברית: "הציג מקור"):



בחילון שיפתח יוצג טקסט שמכיל את המידע ששלח השרת כתגובה לבקשת הלקוק (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 הימני של המouse, ובחרו באופציה **View page source**. מצאו בחילון שיפתח 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 מצינים על המעטפה את **כתובת היעד** (厰ען) ואת **כתובת המקור** (厰ען). באופן דומה, גם כאשר נשלח מידע ברחבי האינטרנט, יש צורך בכתובות מתאימות שיזהו את השולח ואת היעד של הודעה. 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, כולל 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-IP 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>



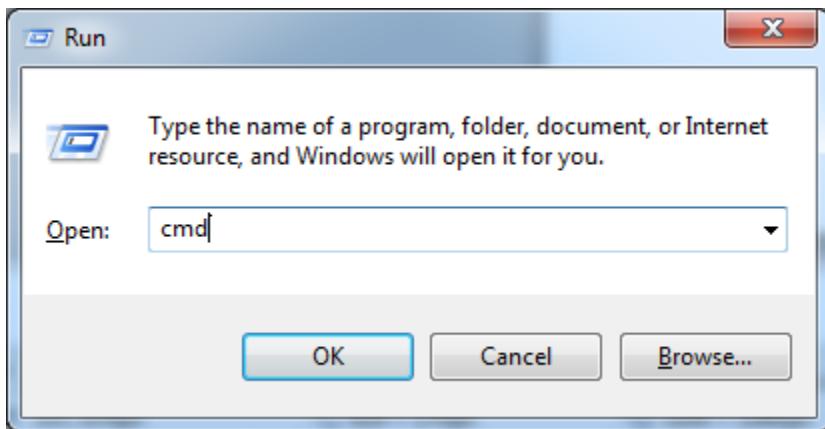
#### **תרגיל מודרך - מציאת כתובת ה-IP של אתר אינטרנט מסוים**

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

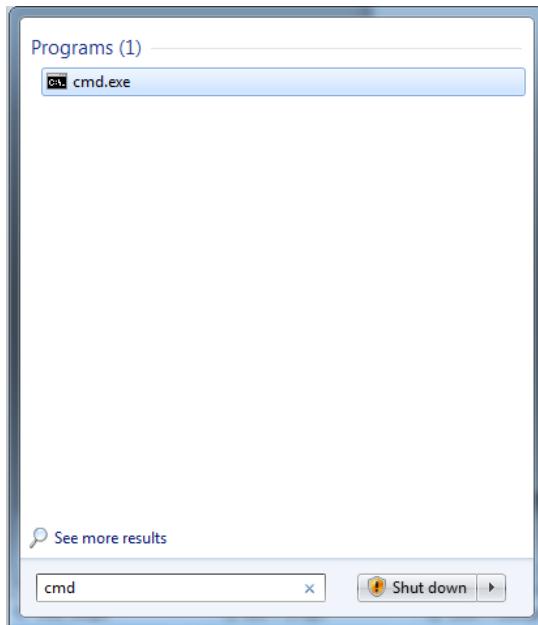
דרך אחת לגלות את כתובת ה-IP של אתר אינטרנט היא להשתמש בכלי tping. לצורך כך, הריצו את **שורת הפקודה (Command Line)**, בה נשתמש רבות לאורך הספר. לחזו על צירוף המקשיים (WinKey+R) ב כדי להגיע לשורת הפעלה. Winkey הינו המקס במקלדת שנמצא בין המקס Ctrl למקש Alt ומופיע עליו הלוגו של Windows:



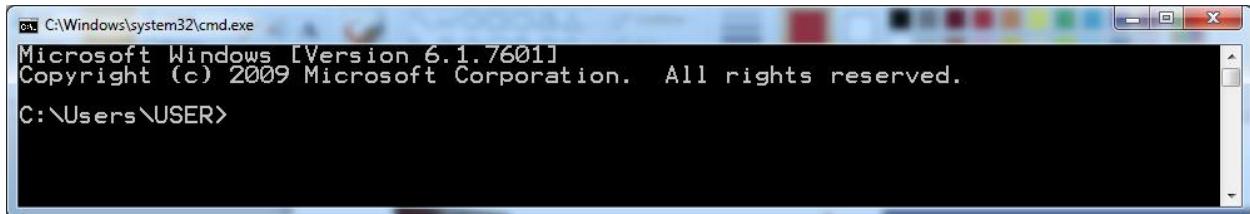
לחילופין, תוכלו לגשת אל Start (התחל) ולחזור באופציה Run (הפעל). הקישו בה את האותיות cmd ולחזו על OK:



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



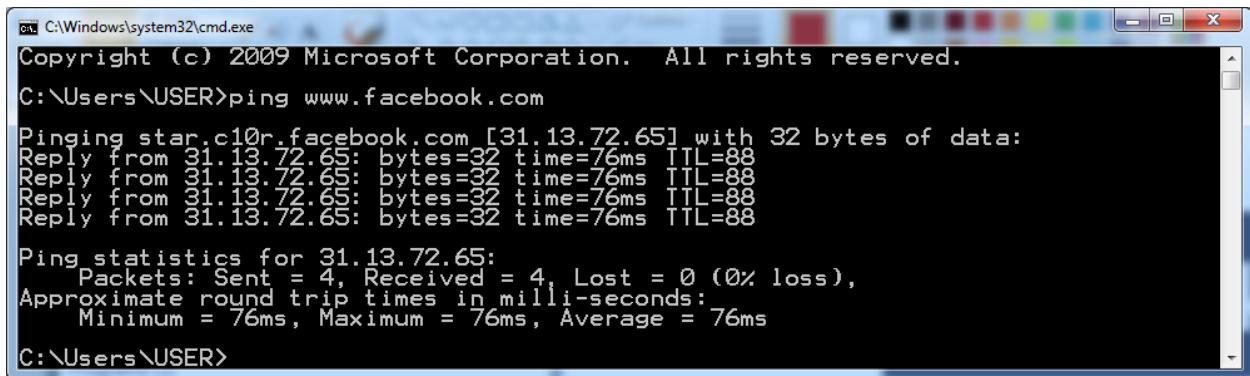
בשלב זה צפי לפתיחת חלון ה-cmd ובו שורת הפקודה:



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:

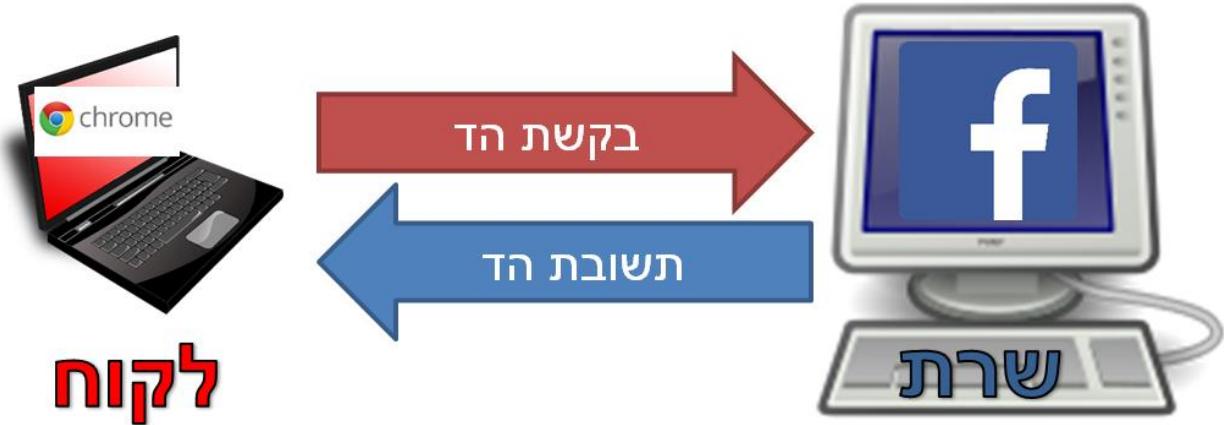


בדוגמה זו, כתובת ה-IP הינה 1.31.13.72.65



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

הכלי **ping** לא מועדף בכדי למצואות כתובת IP של אתרים. הוא מועדף 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, הכללי **ping** שולח **בקשת הד** (כמו לצעוק במערה כדי לגלוות 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 ביןו לבין אותו מחשב מרוחק. בהמשך הספר, בפרק שכבות הרשת, נלמד לעומק איך הכללי **ping** עובד - ואף תממשו אותו בעצמכם!

## GeoIP

עד כה דיברנו על שירותים, ל��וחות וכתובות. כאשר גולשים לאתר אינטרנט מסוים, הودעת הבקשה צריכה להגיע בסופו של דבר אל השירות שיטפל בה. אותו שירות נמצא במקום כלשהו בעולם. האם ניתן לגלוות 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 נתונים מענה נכון ברוב

---

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

המקרים. בגלל האופן שבו בניו האינטרנט, לא ניתן לייצר מادرר 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](#)

New tool for your Web!

City: Mountain View  
Country: United States  
IP Address: 74.125.228.80

Host / IP: www.google.com [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

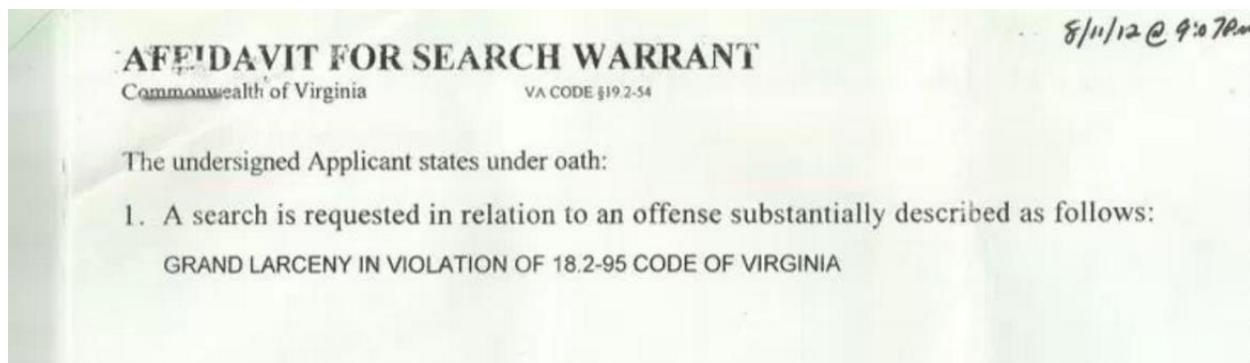
TARINGAI 16 | [Tweet](#) 227 | [g+](#) 1 | [Like](#) 1.4k | Сохранить 62 | HOSTING BY **WIROOS**

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

- [www.yahoo.com](http://www.yahoo.com)
- [www.bbc.com](http://www.bbc.com)
- [www.newtoholland.nl](http://www.newtoholland.nl)
- [www.yahoo.co.jp](http://www.yahoo.co.jp)
- [www.southafrica.co.za](http://www.southafrica.co.za)
- [www.webawards.com.au](http://www.webawards.com.au)

כאמור, המיפוי בין כתובות IP למיקום גיאוגרפי אינו מדויק. הכתבה 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 תמיים. 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 עדין יסבול מהטרדה.

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



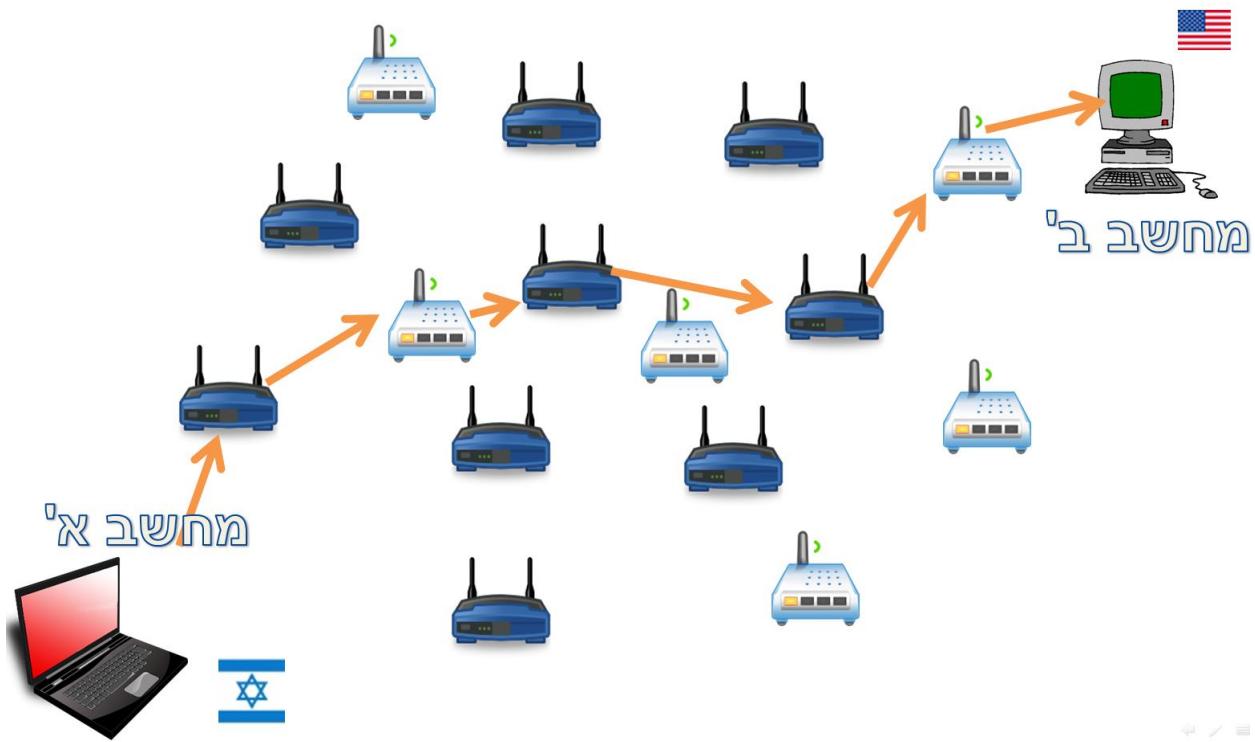
צו חיפוש משטרתי, בחשד לגנבה, שנייתן עקב מיקום IP שגוי (מתוך הכתבה)

## عن האינטרנט

אם בדקתם מה המיקום הגיאוגרפי של Google, גיליתם שהוא נמצא בקליפורניה שבארצות הברית. האם זה הגיוני? איך קורה שהודעת הבקשה המלאכות מגיעה עד לשרת שבארצות הברית, והודעת התגובה חוזרת בחזרה? ההודעות עוברות מרחוק של חצי כדור הארץ הלאן ושוב! אם היינו צריכים למן טיסה כה ארוכה עבר כל הودעה שנשלחת, השימוש באינטרנט היה עשוי להיות יקר מדי גם עבור בייל גייטס<sup>2</sup>. 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 הלאה אל הרכיב הבא, עד שהוא מגיעה אל היעד. העברה של הודעה בין רכיב אחד לרכיב אחר המוחברים ישירות נקראת **קפיצה (Hop)**.

<sup>2</sup> 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 שלל רכיבים אלו מוטלת משימה מורכבת: עליהם למצאו 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), ומעביר אותו עד לצד השני (למשל, השרת 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? הכללי **traceroute** (קיזור של מסלול - **tracert**) 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>
```

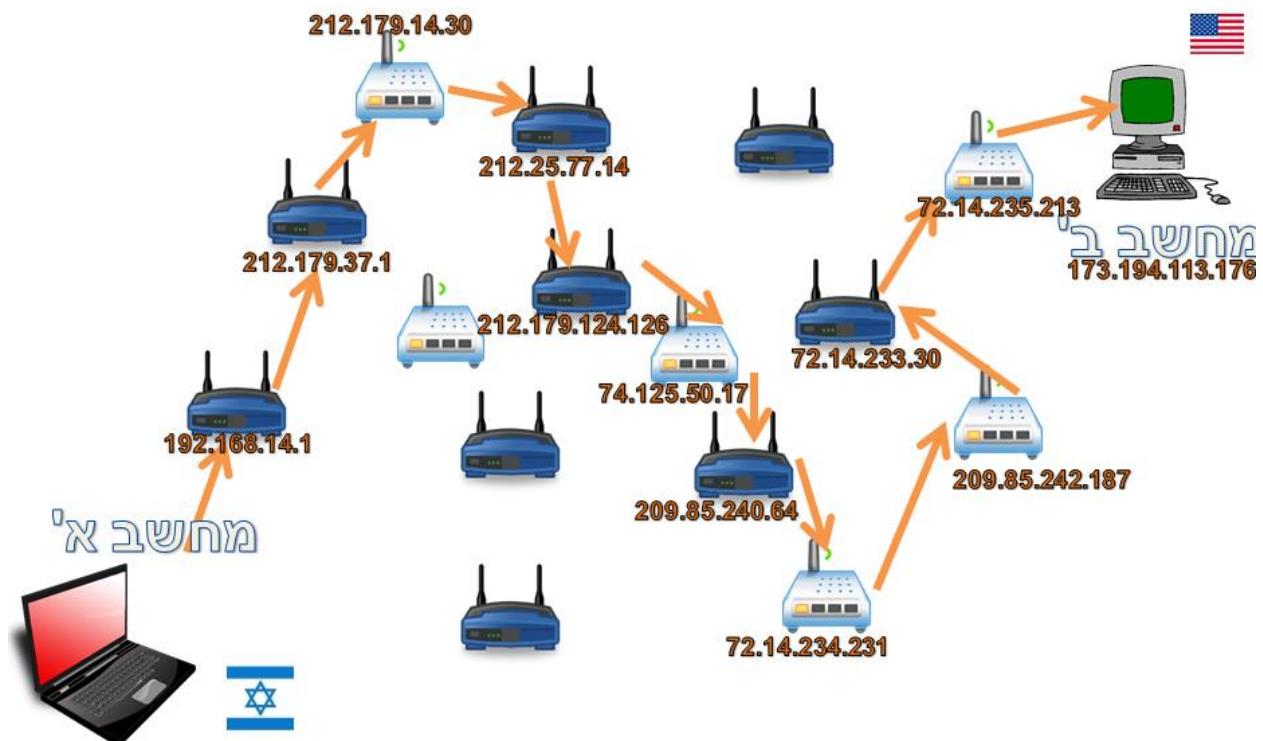
הקישו את הפקודה הבאה:

**tracert -d www.google.com**

```
C:\> 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 שדרךם עברת הודעה שלחנו אל Google.<sup>3</sup>



<sup>3</sup>בגלל הדרכו בעקבות האינטראנט, הדרך זו נכונה עבור הודעה מסוימת אך עשויה להשתנות עבור הודעה אחרת. על כן נעמיך בהמשך הספר.

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

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

נסו בעצמכם להריץ את הכל**י tracert** בכך לגלות את המסלול שהודעה עוברת מהמחשב שלכם אל  Facebook.

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

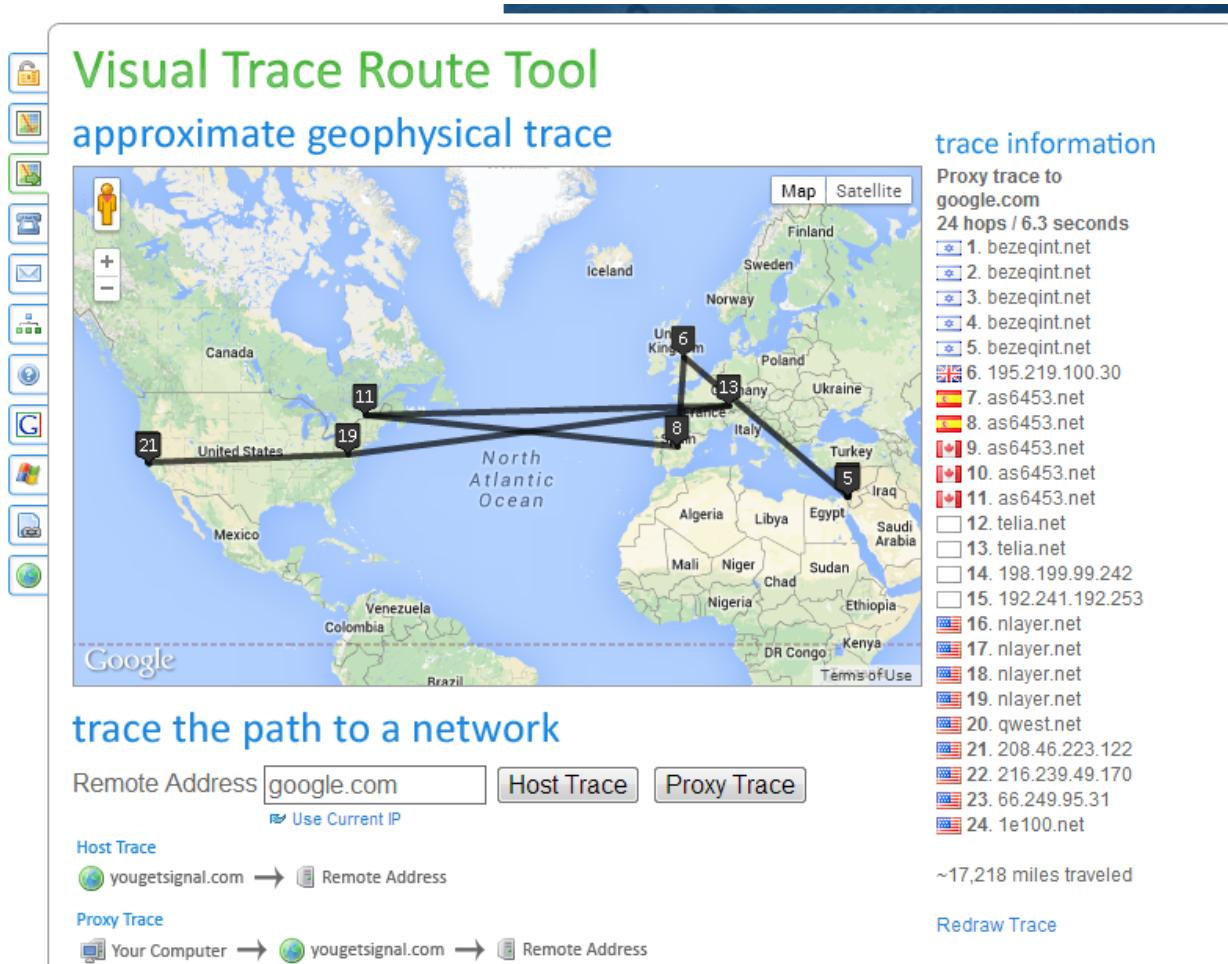


#### תרגיל מודרך - מציאת הדרכן בה עוברת הודעה על מפת העולם

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

---

<sup>4</sup> למעשה, הודעה עוברת מהמחשב שלכם הראשית אל האתר [yougetsignal.com/tools/visual-tracert](http://www.yougetsignal.com/tools/visual-tracert), וממנו נשלחת אל Google.



לאחר שהציגם את הדרכ שעברה הودעה מהמחשב שלכם אל Google, השתמשו באתר זה ומצאו



בעצמכם את הדרכ שעוברת הודעה 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)

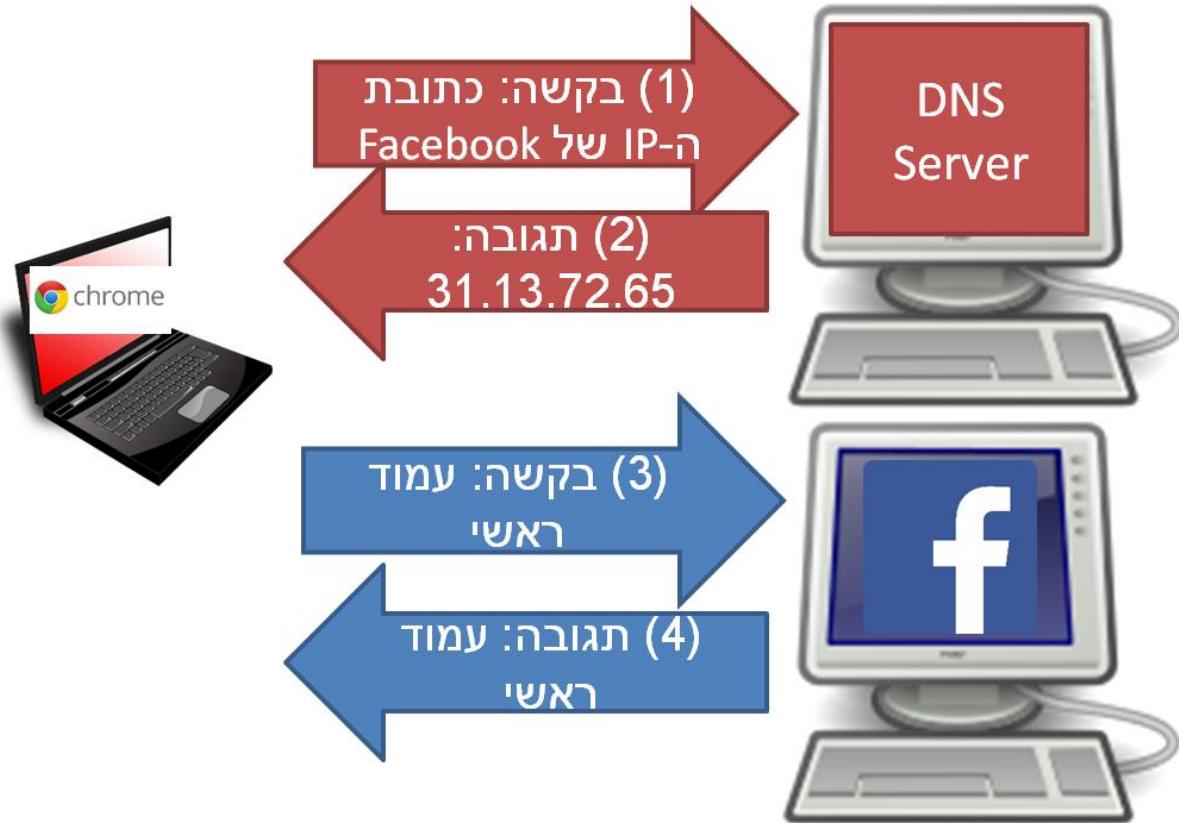
בailo מדינות עברתם בדרך לכל אתר?

## 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 המצוירה "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 לכך בה אנו שומרים מספרי טלפון: כאשר אנו מתקשרים לאדם, בדרך כלל נעדייף שהטלפון יזכיר לנו את המספר שלו, וכך נעדיף לחזיכר את המספר 054-1234567. עם זאת, שם השטלפון 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 הרלוונטית, למשל .31.13.72.65.

באינטרנט, הדרך לתרגם שמות דומיין לכתובות IP היא באמצעות המערכת **DNS (Domain Name System)**. המקרה הוא, בין שמות דומיין אל כתובות IP, מתבצעת 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 ב**אדום** בשרטוט) נשלחת אל שרת-h-DNS, בעוד בבקשת העמוד (הmoצגת ב**כחול** בשרטוט) נשלחת אל כתובת-h-IP של Facebook. על הדרכו שבה עבדת מערכת-h-DNS, נלמד בהמשך הספר.



#### תרגיל מודרך - תרגום בין שמות דומיין לכתובות IP

על מנת להשתמש במערכת 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 כתות מספר כתובות IP השויות ל-Google, למשל - 173.194.112.212.<sup>5</sup> נסן לגלוש אל כתובת ה-IP זהו בדףן. כלומר, במקומות לכתוב בשורת הכתובת "www", כתבו ?Google". אם הגעתם אל 173.194.112.212"

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

---

<sup>5</sup> 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 מוחזיר **תגובה**. למדנו כי להודעות באינטרנט, בדומה להודעות דואר, יש כתובות מקור וכתובת 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 ישירות 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** כדי להציג את הדרך הזו 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 ping, tracert, nslookup וכן את visual traceroute. עם זאת, רק התחלנו לענות על השאלות שלנו, ציירנו תמונה כללית בלבד. בעיקר, נפתחו בפנינו שאלות חדשות - איך עובדת מערכת DNS? מה זו בדיקת כתובות IP? מי הם הרכיבים האלו 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 לתפוא 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`. עליכם להקליד ולהריץ שורות קוד אלו במקביל לתהיליך הקרייה, על מנת להבין את החומר באופן מלא.



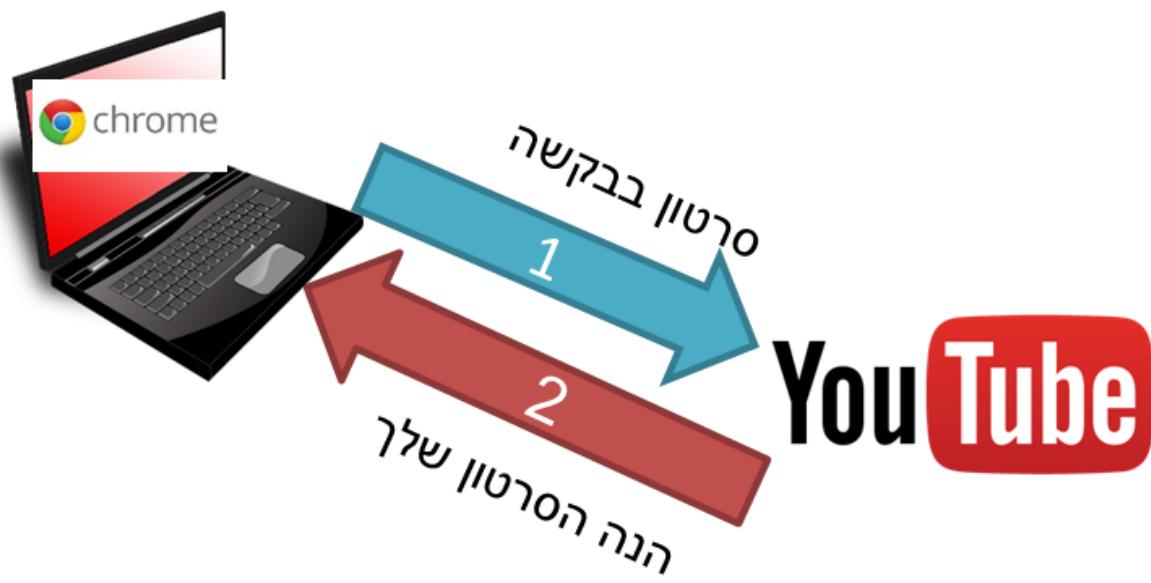
טרם תריצו את דוגמאות הקוד, מומלץ לוודא שהספרייה של `Python` נמצאת ב-`Path` של מערכת הפעלה שלכם. על מנת לעשות זאת, ניתן להיעזר בסרט ההסביר שנמצא כתובות:

<http://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:

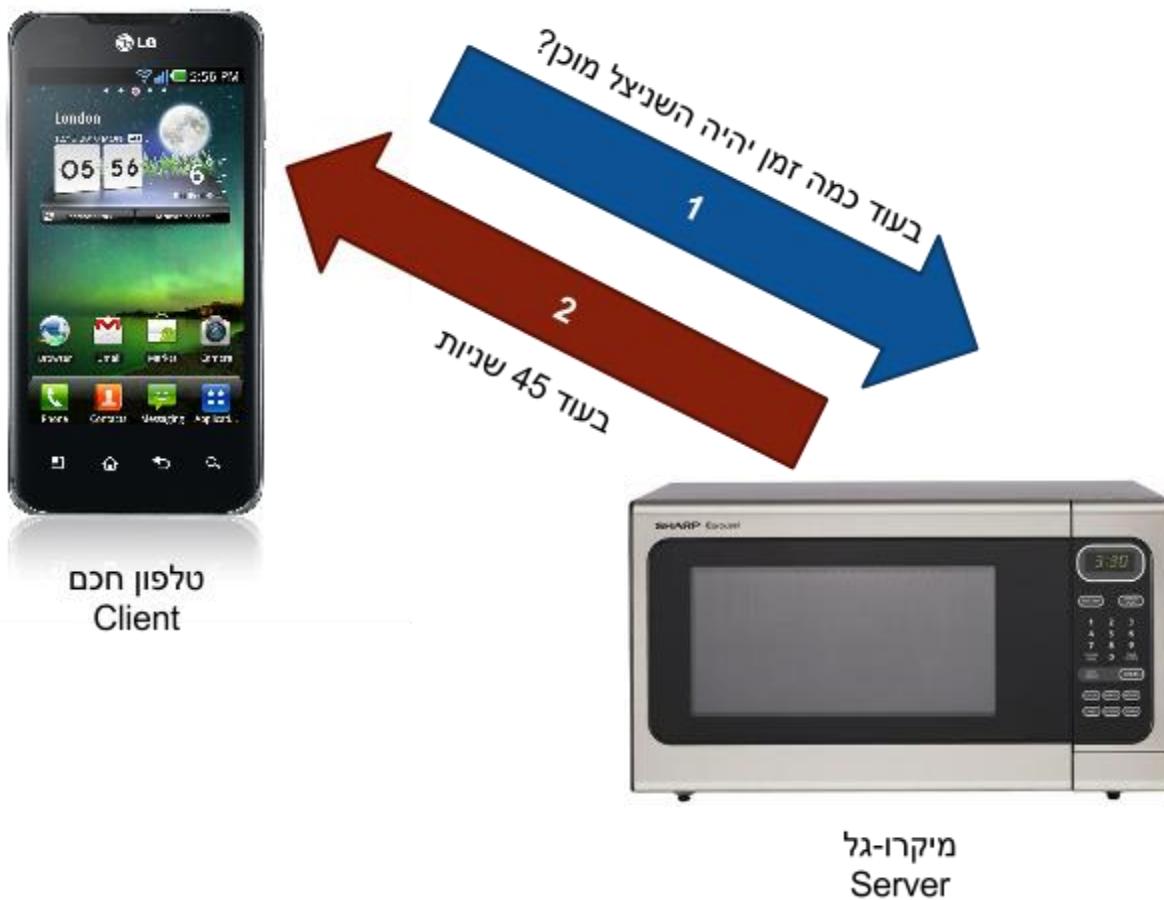


באופן דומה, במידה שאתה גולש מטלפון הנייד שלכם אל כתבה ב-[net](#), הליקוּת הימנו הטלפון שלכם (או הדףון שבטלפון שלכם), והשרת הימנו השרת של [net](#)וֹז.



**מי הליקוּת ומי השרת בדוגמה הבאה?**

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

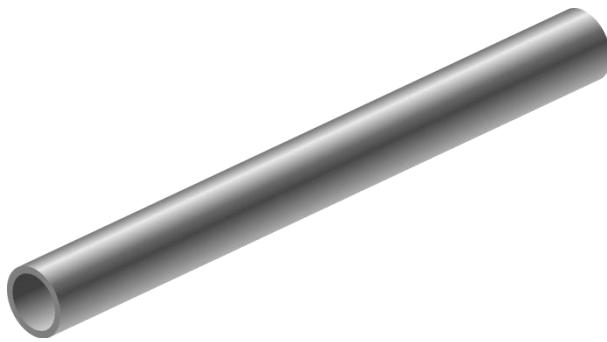


בדוגמה זו, המיקרו-גל הינו השירות, שכן הוא נותן שירות של הערכה של משך הזמן הנותר עד לתום תהליך חיים השניצל. הלוקו הינו הטלפון החכם, אשר משתמש בשירות של המיקרו-גל.

במהלך הפרק, נבנה אפליקציה בתצורת שרת-локוח וביצע זאת תוך 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). במידה שתwo תוכניות מעוניינות להעביר ביניהן מידע, נקודת הכניסה של מידע למערכת ונקודת היציאה שלו ממנה יconeו **Socket**.

על מנת שייהו לנו צינור, אנו זקוקים למספר דברים:

- לבנות את הצינור - את פעולה זו יבצעו השרת והלקוח שלנו.
- התחלה וסוף - לצינור המעביר מימין 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 קווטר ואורך, לפיהם ניתן לדעת כמה מידע יכול הצינור להכיל ברגע נתון. ה"קווטר" של הצינור שלנו, הלווא הוא **Socket**, הינו בית (byte) אחד - אנו נעביר ב-**Socket** זרם של בתים מצד אחד.

## כתובות של **Socket**

כאמור, לכל צינור יש נקודת התחלה ונקודת סיום. עלינו להגיד את נקודת 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 (התהלייר) - 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 התהלייר הוא מזהה הדירה הספציפית בבניין, למשל "דירה 23".



**מזהה הבניין:  
הרצל 1, תל אביב**

ה-Socket מבצע עבורנו את השירות של רשות הדואר - אנו, כמתכננים, לא צריכים להיות מודעים לכל הפעולות שמתבצעת בראשת בצדיה להבהיר את ההודעה מצד אחד של Socket (תיבת הדואר של משפחתי אחות) לצד השני של Socket (תיבת הדואר של משפחתי שנייה). בעוד בעולם הדואר הקישור בין משפחה לבין התיבה הינו כתובות הבניין ומספר הדירה, הקישור בין Socket ובין התוכנה שמנהל אותה (כלומר – התוכנה שאתחילה את אובייקט Socket הינו כתובות IP ומספר פורט (Port)). את המושג " כתובות 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 שלו

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](http://networks.cyber.org.il). ראשית, علينا למצוא את כתובות ה-IP של השירות. לשם כך, נשתמש בכלи **dnslookup** כדי שעשינו בפרק [תחילת מסע - איך עובד האינטרנט?](#):

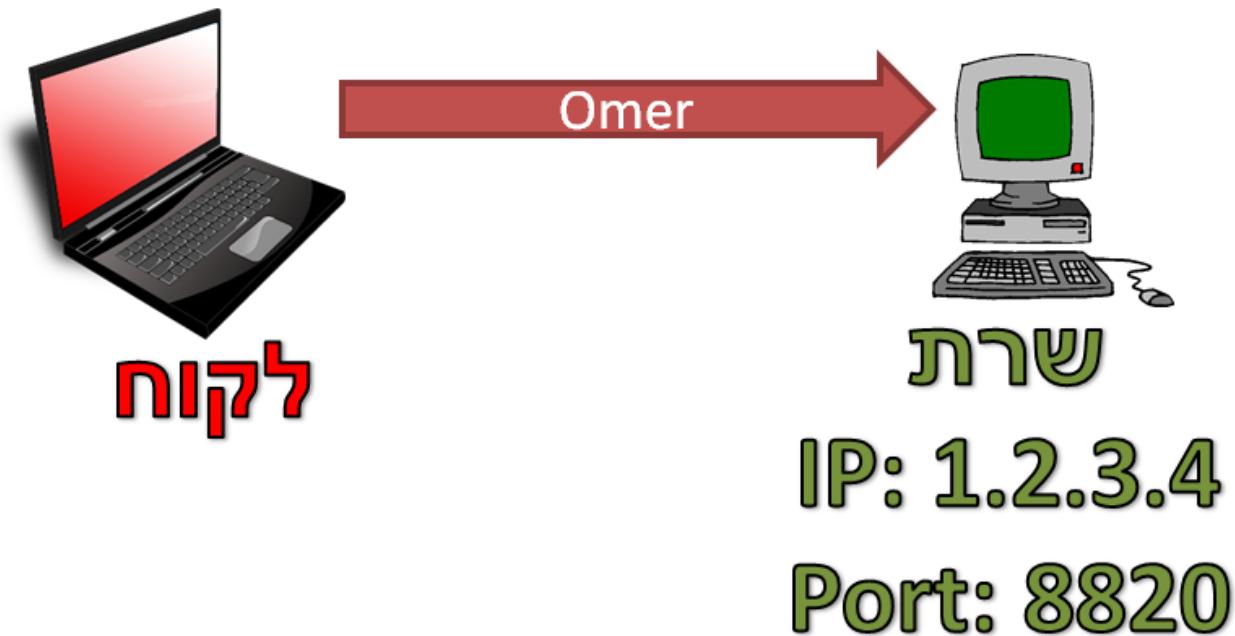
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

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 הכתובת היא הכתובת: "1.2.3.4". שימוש 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 אל השרת זהה, ונסלח לו את השםשלם (לדוגמא: "Omer"). בהמשך, נכתוב את השרת שישתמש במידע זהה.



נתחיל מילכטוּן ללקוח פשוט באמצעות פיתון. הדבר הראשון שעליינו 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 שהכתובה של Socket כוללת כתובת IP ומספר פורט. בהתאם לכך, המתודה **connect** מקבלת Tuple שמכיל כתובת IP ומספר פורט. לצורך הדוגמה, נתחבר לשרת שכתובה ה-IP שלו היא 1.2.3.4:8820

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

כעת יצרנו חיבור ביןינו לבין 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 כך כדי 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

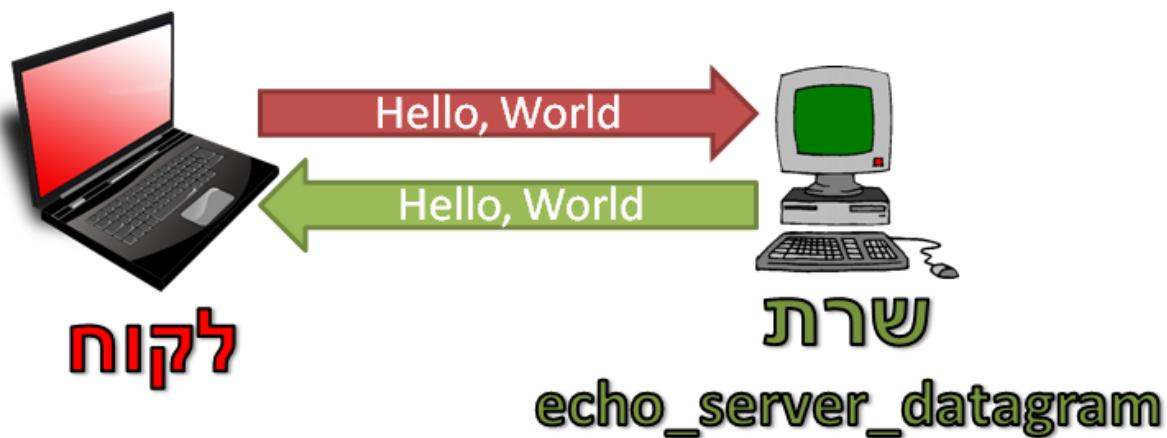
my_socket.close()
```

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 זהה בשפת פיתון.

## תרגיל 2.2 - ל�� להשתתף הדימ: שליחה וקבלת מידע



בתרגיל זה תכתבו בעצמכם לקוד, 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: <sup>6</sup>[http://cyber.org.il/networks/c02/echo\\_server\\_stream.pyc](http://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 לחברים בפורט קבוע - פорт 1729. כתבו קוד בפייתון המתחבר אל השירות, ולאחר מכן שלחו אליו הודעה. קבלו את תשובה השירות, והדפיסו אותה אל המסך.

#### שימוש לב:

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

<sup>6</sup> קובץ בסיוםת .pyc הוא קובץ שמכיל את ה-**code byte** של פיתון. לצורך עניינו, הכוונה היא שלא ניתן לראות בקלות את הקוד המקורי של הסקריפט. אנו מצרפים קבצי **.pyc** במרקם בהם נממש בעצמנו את הקוד של הסקריפט בקרוב מאוד.

<sup>7</sup> כתובת מיוחדת זו מציינת למשהה שהמידע נשלח אל כרטיס הרשת הירטואלי **Loopback** --column העברת מידע שתישאר בגדר המחשב (מערכת הפעלה) ולא יצא אל הרשת. נזכיר כתובת IP מיוחדות נוספת בהמשך הספר.

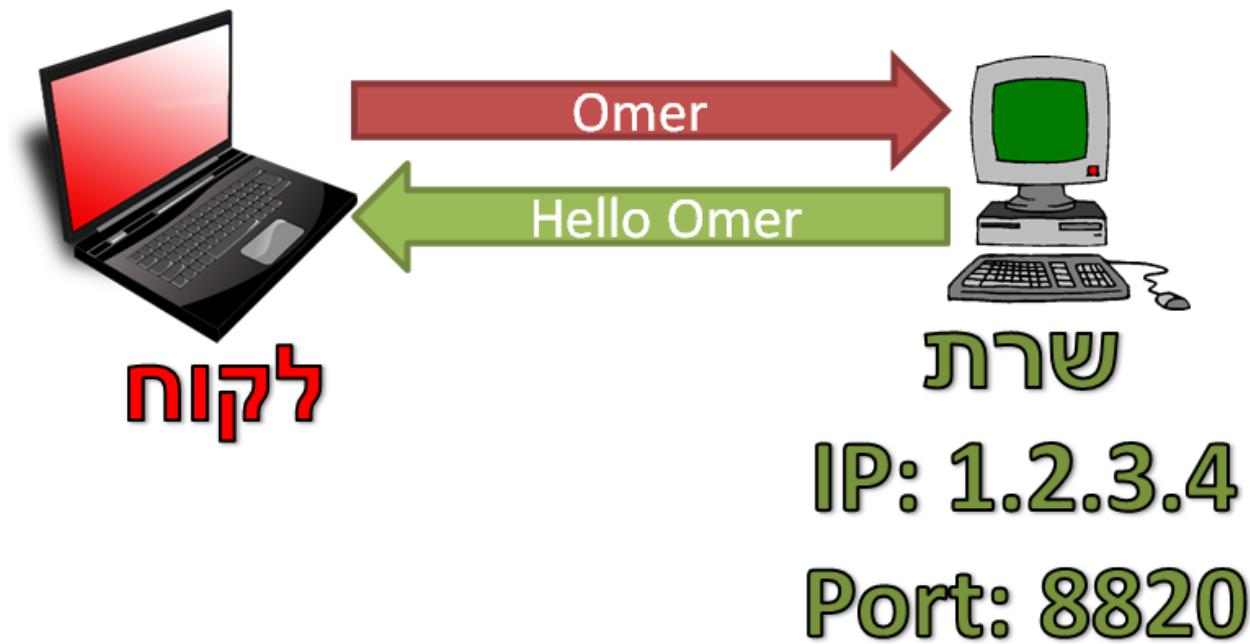
לאחר שהצלחתם לשלוח הודעה לשרת ולקבל ממנה תשובה, כתבו סקריפט פិיתון שմבוקש מהמשתמש הודעה, שלוח אותה לשרת "הדים", מקבל מהשרת את התשובה ומדפיס אותה למשתמש.

#### 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.<sup>8</sup>
- חשוב לסגור את ה-`socket` בסוף השימוש (כלומר - לקרוא `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 שלח, ולענות לו בהתאם. לדוגמה, השירות יענה במקרה זה: "Hello Omer":



<sup>8</sup> 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** לפיתון:

```
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 **socket** שייצרנו 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 ייצרנו 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 מלקוחות, נדרש להאזין לחיבורים נכנים. לשם כך נשתמש בMETHOD **listen**:

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

שימוש לב שהMETHOD **listen** מקבלת פרמטר מספרי.<sup>9</sup>

בשלב זה אנו מוכנים לחיבורים נכנים. 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 ל线索 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 אלינו. לאחר מכן יפנה אל השרת שלנו, נגייע למצב שיש לנו את OBJECT `client_socket` ובאמצעותו נוכל לתקשר עם הלוקו. לדוגמה, נוכל לקבל ממנו מידע. בהקשר זה, נרצה לקבל את השם של הלוקו:

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

ונכל גם לשלוח אליו מידע:

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

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

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

```
client_socket.close()
```

---

<sup>9</sup> פרמטר זה לא מציין כמה חיבורים ה-**socket** יכול לקבל, אלא כמה חיבורים יכולים לחתוך בוור 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 השירות:  
`server_socket.close()`



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

הנה, הצלחנו לכתוב שירות עובד. להלן הקוד המלא של השירות 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()

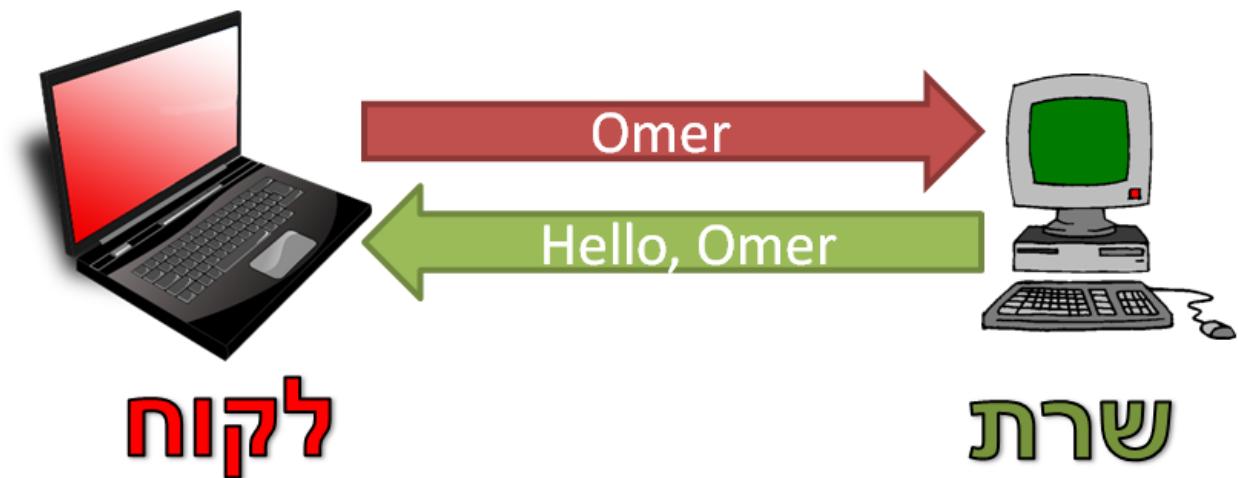
client_name = client_socket.recv(1024)
client_socket.send('Hello ' + client_name)

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



## תרגיל 2.4 מודרך - הרצת שירות ולקוח על מחשבכם

כתבנו לקוחivial לשירות את שמו, ושרת שמקבל את השם ומחייב ללקוח תשובה בהתאם, כמפורט בشرطוט הבא:



כעת נבדוק את הלקוח 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 הלקוח, علينا לבצע בו שינוי. כזכור, כאשר כתבנו את הלקוח, הניחנו שהשרות נמצא בכתובת "1.2.3.4". בהנחה שגםaina הכתובת של המחשב שלנו, علينا לשנות את הכתובת אליה הלקוח מנסה להתחבר. נוכל להשתמש בכתובת "127.0.0.1", כפי שהסבירנו קודם לכן [בתרגיל 2.2 - ליקוח לשרת הדימ.](#)

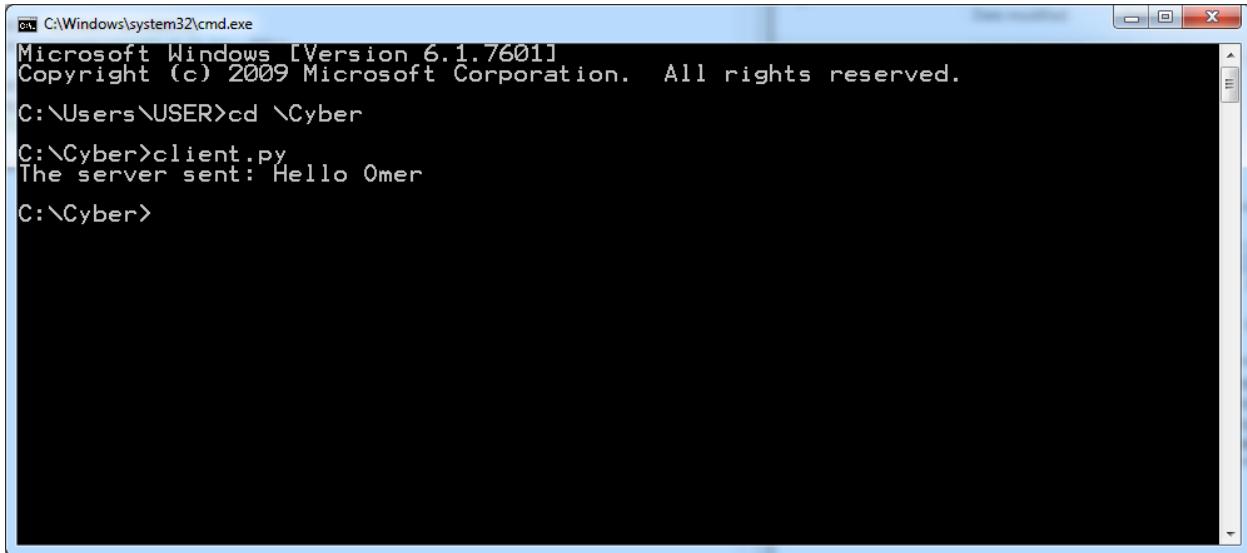
ערכו את הקובץ `client.py`. מיצאו את השורה שבה מתבצעת התחבורות אל השירות המרחוק:

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

וחליפו אותה בשורה הבאה:

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

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



```
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 אמור להיות זהה לזה שבתמונה. נסו עתה לשנות את השם אשר נשלח לשרת ("Omer") לשם אחר. הריצו את הליקוֹן מחדש, וודאו שגםם מקבלים את הפלט המתאים.

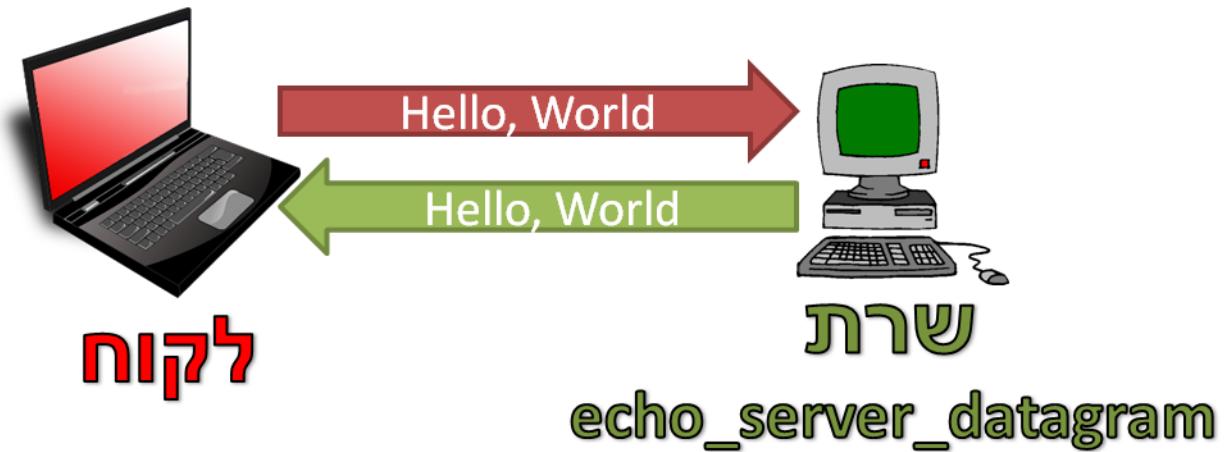
**המלצת שובבה** – כפי שנכתב במבוא לספר זה, מומלץ להשתמש בעורך PyCharm עבור עבודה עם פיתון. באמצעותו עורך זה ניתן לעבוד בצורה נוחה יותר עם קוד, וכן לדרג אותו באמצעות breakpoints, דבר הצפוי לשיער לכם רבות בפתרון התרגילים. ניתן להיעזר במצגת על PyCharm מאות ברק גון, הזמינה כתובות הבאה:  
<http://www.cyber.org.il/python/1450-3-02.pdf>

## תרגיל 2.5 - מימוש שירות הדימ



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

נזכיר 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 ה-`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 מסוימת לרצץ מבלי שהיא קראה למתודה `close`. דבר זה יכול לנבוע ממספר 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 גם, 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 קראיה 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 התחבקתם ללקוח, ובתרגיל השני התחבקתם לשרת. בתרגיל זה עליים לכתוב הן את השרת, והן את הלוקו. חלק מהאתגר בתרגיל הינו כתיבת פרוטוקול לשלוח הודעות בין השרת והלקוח.

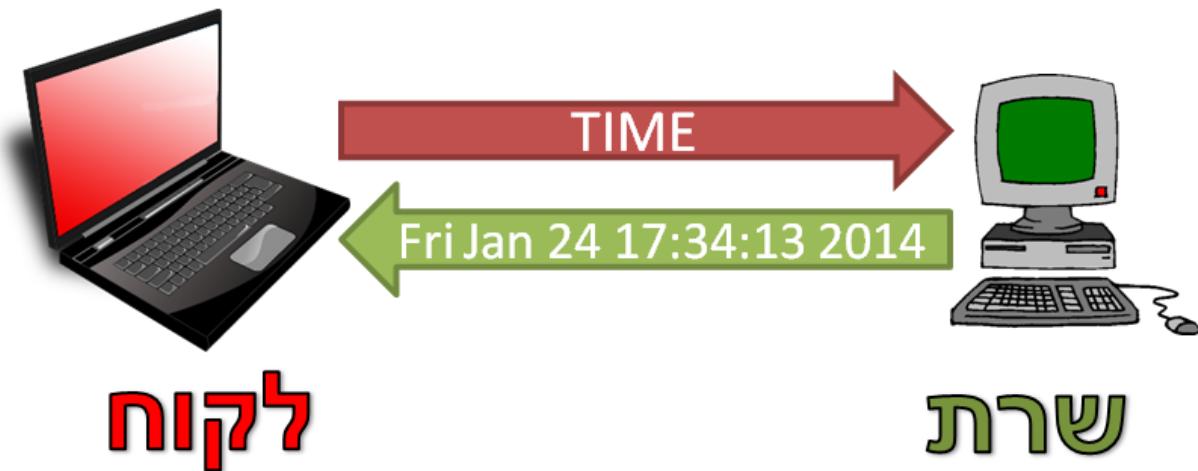
שלב 1:

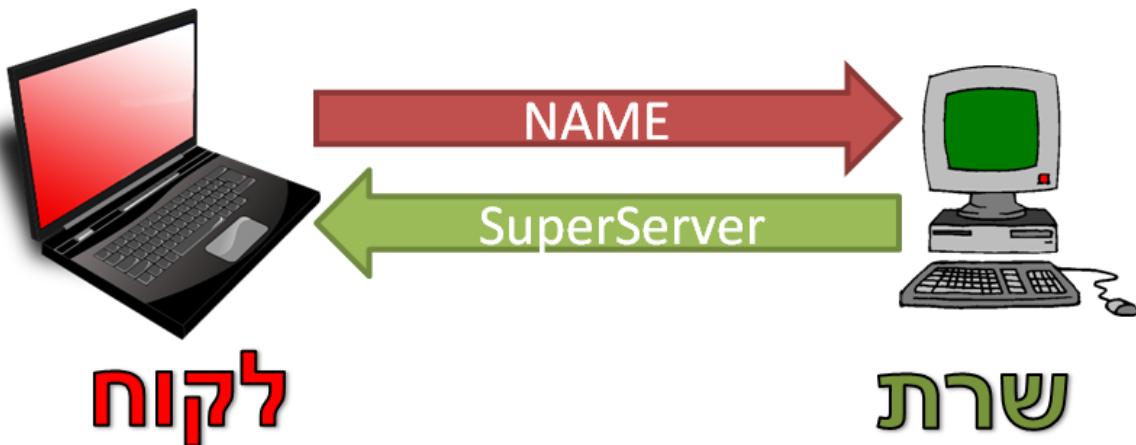
עליכם לכתוב מערכת שרת-לקוח, כאשר השרת מבצע פקודות שהלקוח שולח אליו, ומחזיר ללקוח תשובה בהתאם.

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

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

להלן דוגמאות לתקשרות:





## שלב 2:

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

- השרת ישלח ללקוח הודעה שכתוב בה מה אורך המידע שלו לקל בטור תשובה.
- השרת ישלח ללקוח הודעה מיוחדת שמשמעותה "שליחת המידע הסתיימה".

## שלב 3:

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

**הנחיות לתרגיל 2.6**

לפניהם ניגשים לכתב את הקוד, בצעו תכנון ראשוני. חשבו מה בדיק תמשו מצד הלקו ומה מצד השרת, נסנו לצפות מראש בעיות בהן תתקלו.

מצד הלקו - 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.  
מצד השרת - עליכם לקבל חיבור מהלקוח, להבין את הבקשה שלו ולהגיב אליה בהתאם. לאחר מכן, נתקו את החיבור ותוכלו לספק שירות ללקוח חדש.

בצלחה!



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

עד כה בניתם שירותי ולקוחות שתקשרו עם זה עם זה וביצעו פעולות פשוטות. בתרגיל זה עליים לתכנן מערכת שרת-לקוח, ולאחר מכן למש אתה. המערכת תאפשר לטכני לתקשר עם מחשב מרוחק ולבצע עליו פעולות שונות. הפעם, עליים 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. עליים למלא בתוכן את הפונקציות, לפי התעוד שנמצא בפונקציות ולפי הדרישה בהמשך.

[www.cyber.org.il/networks/server\\_template.py](http://www.cyber.org.il/networks/server_template.py)

שלד השירות:

[www.cyber.org.il/networks/client\\_template.py](http://www.cyber.org.il/networks/client_template.py)

שלד הלוקו:

להלן הפקודות שעליים למש:

### 2.7.1

כדי להבין מה מתורחש במחשב המרוחק, הטכני שלנו רוצה לקבל צלום מסך של המחשב המרוחק. עליים 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 עליים לשלוח אותו ללוקו. פקודה **SEND\_FILE** חד Um שם הקובץ הרצוי צריכה Lagerom 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\C:\. 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\C:, 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 ובלוק 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 עובד. הקובץ המבוקש נמצא בספריה אחרת במחשב אליו מתבצעת העתתקה. הוסיףו תמייה בפקודת COPY (לדוגמה: העתק את הקובץ C:\Cyber\1.txt אל 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 C:\1.txt אל C:\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 כזה, על השרת להגיד ללקוח האם הפעולה הצליחה או נכשלה.

EXECUTE notepad.exe

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

לדוגמה, במקרה להריץ את notepad, נוכל לבצע:

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

נשים לב שלעתים נדרש לתת את ה-path המלא של קובץ ההרצה שאנו רוצים להריץ<sup>10</sup>, למשל כך:

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

## 2.7.7

לבסוף הטכני שלו 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) שימוש לב, שכאר מבצעים בפייתון (message)socket.send(message) עבור message ריק, לא מתבצעת שלילה כלל. כלומר לא ניתן להסתמך על קבלת מסר ריק בלבד כדי לדעת שהשרת סיים את שליחת הקובץ.
- 2) שימוש לב שבפייתון, על מנת לייצג מחוזת שכוללת את התו backslash ("\"), נדרש לכתוב את התו פעמיים – כלומר "\\". זאת היות שההתו backslash יכול להיות חלק מנתויים מיוחדים, כגון "ח" שמסמל התחילת של שורה חדשה. לחיפוי, ניתן להשתמש באות z לפני הגדרת המחווזת. 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
```

---

<sup>10</sup> 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 שאתה מכירם את השימוש ב-Exceptions בפייתון, מומלץ להשתמש בהם.
- (4) מידע נוסף על המודולים בתרגיל זה עוד, תוכלו למצאו בספר ה-*Python* של מטה הס"בר הצה"ל, שנכתב על ידי זהר זילברמן ונמצא בכתובת: <http://www.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 מחוזצת 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, מחוזצת תשובה מאות המשמש. ביום ההודעה שולח הצד A' מספר פורט, לדוגמה 8071, ומתנתק.

חזר 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 התמודדים 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. כעת 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 ללקוח ראשון ושרת ראשון. בהמשך, מימשTEMם בעצמכם ללקוח ושרת "הדים", פיתחTEMם שרת פקודות בסיסי וכן שרת פקודות מתקדם. לאורך הפרק הצלחTEMם ליצור מספר מימושים במודל שרת-לקוח, ותרגלTEMם הן את יכולות הפיתוח שלכם והן עבודה מול Socket.

כעת, יש ברשותכם כל עוצמתו. אתם יכולים להשתמש ב-Socket כדי לתקשר מנקודת קצה לשניה, ולכתוב כל שרת-לקוח שתרצו. עם זאת, למדנו רק חלק מהאפשרויות שניתן 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 קונספטים חדשים, לראות דוגמאות וכן לכתוב בעצממו קוד שייממש 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 Socket הוא בסך הכל ממשק שמאפשר לנו לתקשר בקלות ומפשט לנו את כל התהליך התקשורתי שקרה בפועל (כלומר - איזה מידע עובר בראשת כשאנו משתמשים ב-Sockets). אז מה קורה בפועל כשאנו מדברים עם מחשב מרוחק? בכרע עוקב 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 לגלות 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 נדרש לתמוך בו בזמןית, והכמויות העצומות של המידע העובר בכל רגע נתון באינטרנט וצריך לחזות את הגלובוס כדי להגיע לצדי 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 שאמם 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 הלקוחות להמשיך ולרכוש מוצרים נוספים מאותה היצרנית אם הם היו רצויים לתקשר בין שני מכשירים שונים. ברור כי המצב הזה אינו רצוי, והוא אינו מאפשר למיכליים שונים באמת לדבר ב"שפה איחידה" המשמשת את כל רשת האינטרנט.

כדי לפתור את בעיה זו וליצור סטנדרטיזציה (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 התקשרות לדבר בצורה זו אחת עם השנייה, אלא מנחה את יצירניות החומרה כיצד למשתמש את מערכות התקשרות כך שתהייה איחידות בין כלם. כפי שתראו, בספר זה לא נעשה שימוש במודל שבע השכבות אלא במודל חמיש השכבות<sup>11</sup>.



### 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'ך שעובר אדם מהרגע שמנגין לשדה התעופה במדינת המקור ועד שיוצא משדה התעופה במדינת היעד? דרך אחת לעשות זאת היא לתאר את הפעולות שהוא עושה (או שעושים בשבילו) בצורה כרונולוגית: בידוק ביטחוני והפקדת

---

<sup>11</sup> ISO יצרו את מודל שבע השכבות באופן תיאורתי. מודל חמיש 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 הtgtלו כמיותרות ولكن הושמו מהמודול.

מזוודות, 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 של מזוזות (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 למטווס ועד לשער הירידה מהמטווס, משתמשים בשירות העברת המטווס 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 הדרוכנים. היא רלוונטית אך ורק לשעריהם, ורק בהזה היא "מבנה". לעומת זאת - שכבה זו אחראית על השערים בלבד, לא מכירה את השכבות שמעליה, ולא יודעת איך השכבות שמתוחתיה מתפקדות. היא אינה מעוניינת להכיר דברים אחרים שהיא לא אחראית עליהם.

- בשכבת הדרוכנים, 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 למטווס היא רק עברו מי שהפקיד את המזווידות וכבר החתים את הדרון; ההמראה היא רק עברו מי שהפקיד את המזווידות, החתים את הדרון, וכמוון – עלה למטווס. עובדה זו מאפשרת לנו לזרות 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 לדרוכנים אלקטרוניים. התהינה של החתמת הדרוכנים עדין תמלא אחריו הייעוד שלה: רישום של אדם היוצא מדינה אחת ונכנס למדינה אחרת. היתרון הגדול הוא שלאiar התהינות בשרשרת לא אכפת כיצד היא עושה את זה - העיקר שהיא תדאג לרשות שהנוסע יצא מדינת המקור כדי שאפשר יהיה להעלות אותו למטווס דרך השער, והוא נכנס למדינת היעד כדי שיוכל להמשיך ולאסוף את

המצוודות שלו. עם זאת, הדבר היחיד שכן 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 לעתים תכופות ומפתחות מעט לעת, היתרון אותו הצגנו הוא חיוני. גם אם מימוש של שכבה מסוימת ישנה, שאר המערכת לא תושפע ממשינוי זה ונוכל להמשיך לדבר עם ישות אחרות בראשת כאלו כלום לא קרה. זאת מכיוון שהשכבה עדין תספק את אותו **שירות** לרמות שמעליה (שימוש לב שמדובר על שני **במימוש השירות**, ולא בשינוי השירות עצמו). מאפיין זה של הסתרת המימוש אותו הצגנו כרגע הוא מאפיין חיוני בבניית מודול מודולרי כגון מודל חמש השכבות.

## איך זה קורה בפועל?

דברנו 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 ממש (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

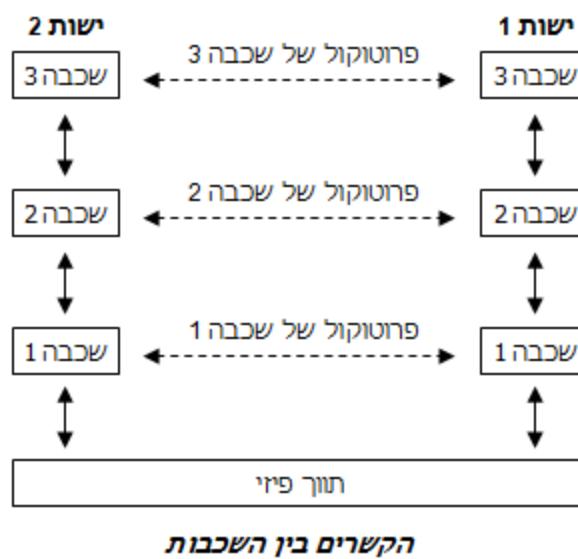
כשמדובר סוקטים: איך, לדוגמה, יודע השירות שהלך וקורא לבעץ צילום מסך? באמצעות פרוטוקול התקשורת שהגדרתם ביניהם.

גם HTTP, הפרוטוקול המשמש להעברת דפי האינטרנט אליהם אנחנו גולשים בדפדפן, הוא אכן כן. הוא קבוע כיצד דבר הדפדפן עם השירות המרוחק, ובאיו צורה יכול להשתמש את הדף שהגיש לו השירות. למעשה, בפרק הראשון בספר, שראינו שהדפדפן שלוח הודעה בקשה לשירות, והודעה זו הייתה בפועל הודעה בפרוטוקול HTTP, עליו לנלמד לעומק בהמשך הספר.

בלי פרוטוקולים היינו מקבלים סיטואציה נסוח "מגדל בבל", בו כל רכיב מדבר בשפה שלו ואף אחד 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 **ישירות**. במקומות 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` ו-`recv`) - אולם התקשורת ביניהם היא ווירטואלית, ובפועל היא משתמשת

על העברת המידע לשכבות התחתיות ושימוש בשירות שهن מספקות. המידע יורד עד לכרטיס הרשת, יוצא אל הצלב (או כל תווך פיזי אחר) ומצאת את דרכו אל היעד – שם הוא נקלט לכרטיס הרשת (כפי שנוכחנו לדעת על-פי מה שראינו ב-Wireshark) ועלה 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 התקשרות<sup>12</sup>, אבל מה זאת אומרת?

מוקדם יותר בפרק, הזכירנו שבתהליך השילחה כל שכבה מעבירה את הפקטה לשכבה שמתחתה. בנוסף 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 נבנית החבילה היא נעטפת בעוד עוד שכבה (אשר כל שכבה לא יודעת מה יש בפנים), ולאחר שבחבילה נשלחת וועברת בין המשתתפים – בכל שלב מקיפים אותה, שכבה אחר שכבה. לתהליך של עטיפת המידע בכל שכבה ושכבה לצד השולח קוראים **(כימוס)**<sup>13</sup>, ואילו תהליך קילוף הפקטה מצד מקבל נקרא **Decapsulation** (קילוף).

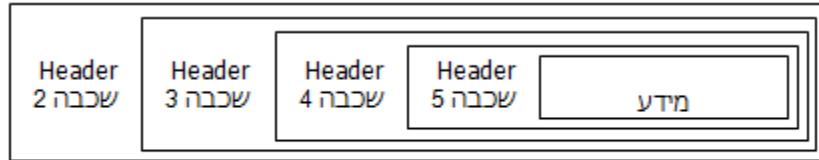
המידע שמוסיפה כל שכבה בתחלת הפקטה נקרא **Header (תחילית)**, והוא מכיל מידע שימושי לשיליטה ובקרה על הפקטה הרלוונטי לשירות שמספקת אותה שכבה (למשל: כתובת ה-IP אליה מיועדת הפקטה, בקורסות שגיאות וכו').

האיורים הבאים מתארים כיצד נראה פקטה במודל חמש השכבות, ואיופה נמצא המידע של כל פרוטוקול בפקטה השלמה<sup>14</sup>.

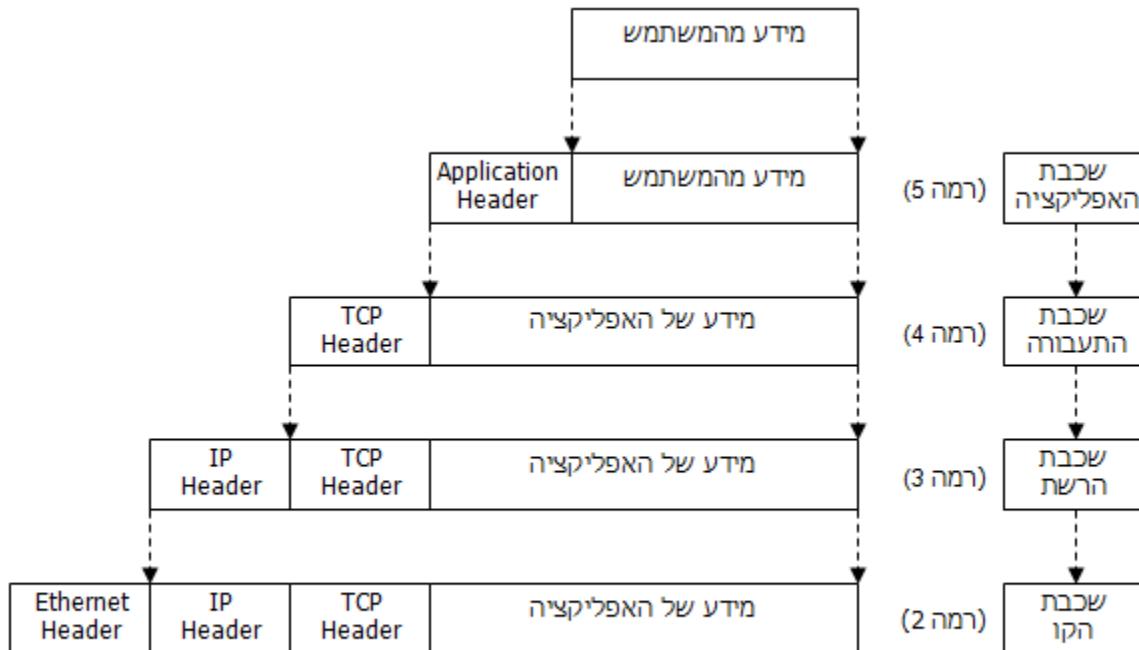
<sup>12</sup> לא כל השכבות חייבות להשתתף בתהליך התקשרות. במקרים מסוימים יש פקטות שמקילות רק את שכבות 3-1, למשל, זה הגיוני לחלוטן כשהשכבות מעליין כל לא הי רלוונטיות לתהליך התקשרות הספציפי בין שני הצדדים.

<sup>13</sup> לדוגמה Encapsulation יש שימוש נוספת נוספת בתכונות מונחה עצמים: הסתרת שימוש וחשיפת משקל תכונות. שימושם לב לא להתבלבל בין השניים.

<sup>14</sup> 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. בנוסף, לא כלנו את השכבה הראשונה (השכבה הפיזית), משומם שלרוב נתונים להתעלם ממנה בהסנה.



מבנה פקטה במודל חמ"ש השכבות #1



מבנה פקטה במודל חמ"ש השכבות #2

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

דבר מעניין שווה לשים עלי'ו דגש הוא שה-Header של כל שכבה (כלומר המידע עצמו, לא ה-Header עצמו) זהה לפקטה של השכבה שמعلיה; בתהליך השילוח כל שכבה מקבלת מהשכבה שמعلיה את הפקטה בדיק כי שהיא, מוסיפה לה את Header עלי-פי התקן (פטוטוקול) של אותה שכבה ומעבירה אותה להלאה לשכבה שמתחת. כך למשל, בשכבה השלישית, ה-Header של הפקטה כולל בין היתר את ה-Header של השכבה הרביעית (בדוגמה לעיל, ה-Header TCP). בשכבת הcano, 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 הוקן, שכבת הרשת, שכבת התעבורה ושכנת האפליקציה. כעת נסקרוו אותן מלמטה למעלה, החל ממה שכבה התחתונה ועד לשכבה העליונה:

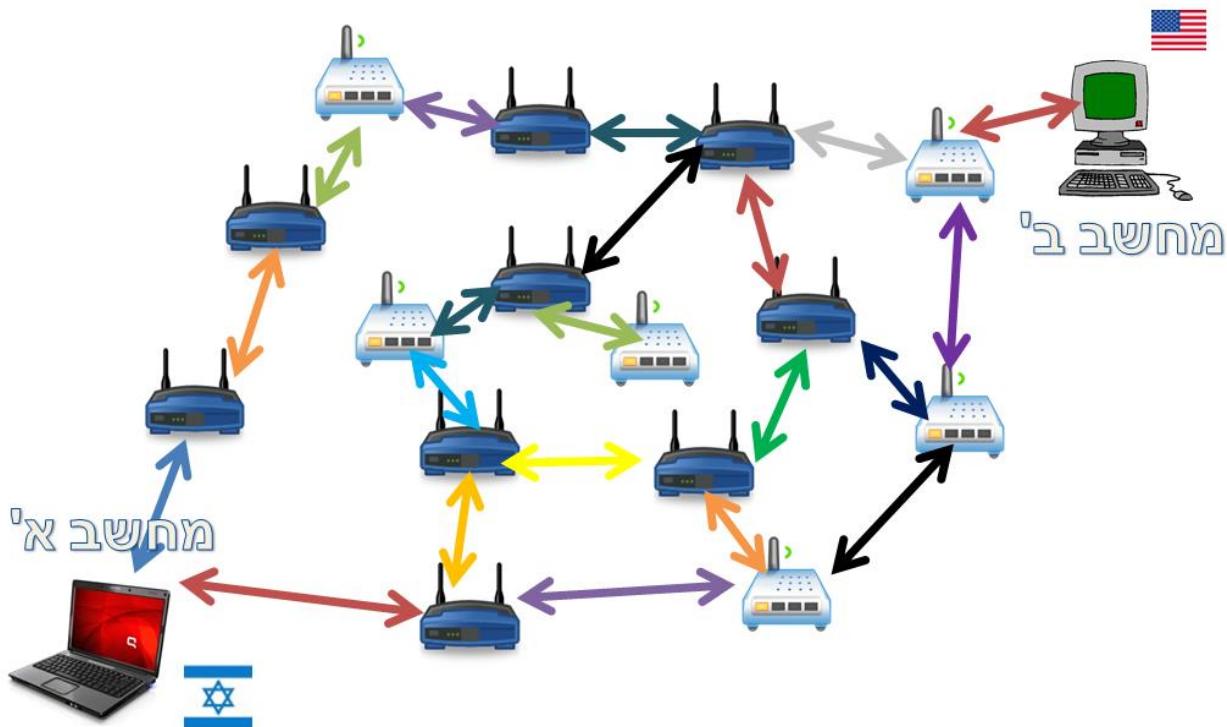
### שכבה ראשונה - שכבה הפיזית

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, פקודות או כל דבר זהה. מבחינתה עלייה להעביר בית אחד בלבד בכל פעם. העברה הפיזית יכולה להתבצע על גבי מגוון של תווים: כבלי רשת, סיבים אופטיים, באויר (גלים אלקטромגנטיים, לוין) וכו' - העיקרי שהמידע יגיע לידי.

### שכבה שנייה - שכבת הוקן

שכבת הוקן מסתמכת על העברה הפיזית של המידע שנעשה ברמה שמתתית, ומאפשרת לנו לדבר עם ישויות אחרות הקשורות אלינו.

באior שלפנים תחום האחוריות של שכבה השנייה מתבטא בכל חץ צבעוני שמקשר ישות רשת הקשורות אחת לשנייה:



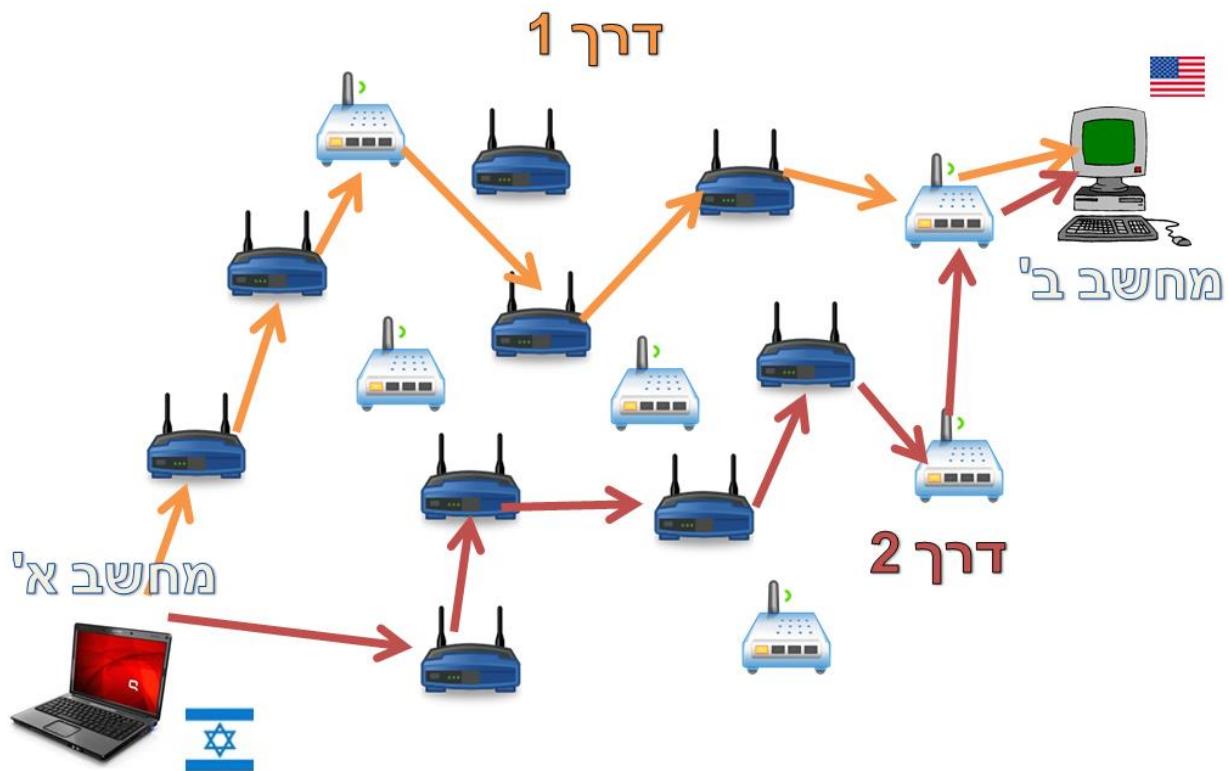
בנוסף, השכבה השנייה מוסיפה מספר יכולות חשובות:

- ארגון המידע בಗושים (המכונים **מסגרות - Frames**, כמו שראינו בפילטר 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 בקצהו השני של העולם? תפקידה של שכבת הרשות הוא למצוא את המסלול הטוב ביותר ביוטר מאייתנו אל היעד ובוחרה. שכבה זו לא מתעסקת בתקשורת בין ישותות סמוכות, אלא אחראית על המסלול המלא בין שתי נקודות קצה. ניטוב המידע מתבצע על-ידי רכיבים המכונים **רouters** (נתבים), אשר מנוטבים 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 הפקתה:

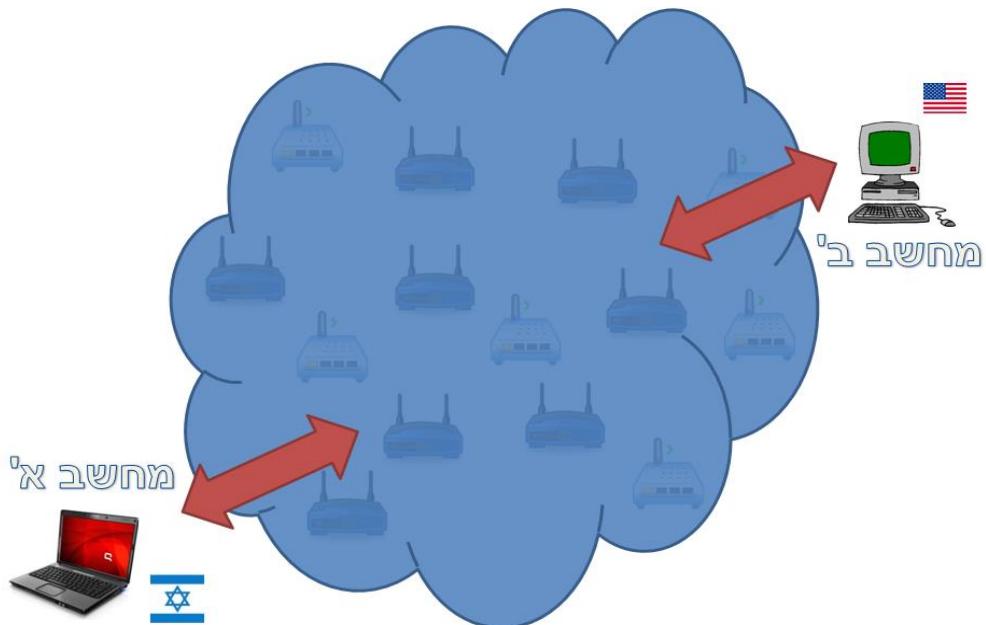


### שכבה רביעית - שכבת התעבורה

שכבת התעבורה מסתמכת על שכבת הרשת שתנתב עבורה פקודות מישות כלשהי ברשות לישות אחרת, נאמר אפילו 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 נשלוcho שתי פקודות אחת אחרי השנייה, אך הן התחלפו והגיעו לעדן בסדר הפוך? כדי להיות מסוגלים להתקיים ברשת האינטרנט אנחנו צריכים בחלק מהמקרים ליצור קישור אמין ורציף בין שתי נקודות הקצה, כזה שייתן סדר ומשמעות למועד שיז↙ל – ולא סתם אוסף של חבילות מידע שלא בהכרח קשורות אחת לשניה.

בעוד שהמטרה הקודמת היא אופציונאלית (יש מקרים בהם לא יהיה חיבורים להבטיח את סדר הפקודות שנשלחות או את הגעתן כלל), שכבת התעבורה מספקת תמיד דבר חשוב נוספת: האפשרות לפנות אל מספר שירותים הנמצאים על אותה ישות. דמיינו שעל השירות מסוים רצה גם תוכנה המספקת שירות מיילים וגם תוכנה המספקת שירות WEB (כלומר מגישה דפי אינטרנט). כיצד יוכל השירות להבדיל בין שתי הבקשות שמתכולות אליו מהלך, האחת אל שירות המייל והאחת אל שירות ה-WEB? לשם כך, השכבה הרביעית מוסיפה לנו פורטים (דיברנו על המושג כבר בפרק [תכנות ב-Sockets / כתובות 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 יצירת קישור שכזה, 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)	תקשורת בין ישויות סמוכות זו לזו	Ethernet	מסגרת (frame)
3	שכבת הרשות (Network Layer)	השלטה על המסלול שעתיד חבילת מידע בין המקור אל היעד	IP	פרקcie (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 לגוש מידע של אחת השכבות, מתכוונים לרצף המידע משכבה זו ומעלה. למשל: שימושתמשים במושג "פקטה" מתכוונים לפקטה בשכבה השלישייה, אך גם לכל המידע של שכבה הרביעית וה חמישית (שמדוילות בתחום הפקטה, כפי שהזכרנו באורח של [מבנה הפקטה](#)). המונח "מסגרת" מתאר את כל המידע השיך לשכבה השנייה, אך גם לשכבה השלישייה, הרביעית וה חמישית.

בהתאם להסביר לעיל, כל מסגרת היא גם פקטה (שהרי אין חビלה בשכבה השלישייה 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) שמספקת מערכת הפעלה כדי שאפליקציות יכולו ליצור חיבור רשמי לשירותים אחרים ברשת. הם אינם שכבה במודל חמש השכבות. האפליקציות משתמשות ב-Sockets שיצרו אצלן "צינור" להעברת המידע, ודברות מעלייהם בפרוטוקולים השונים של רמת האפליקציה (בדיקות כמו הпрוטוקול 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 שמתחתיו! בפועל, הפרמטרים שננתנו היו רלוונטיים לשירות לשכבות שמתחת ל-Socket – שכבת הרשות (השכבה השלישייה) 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))
```

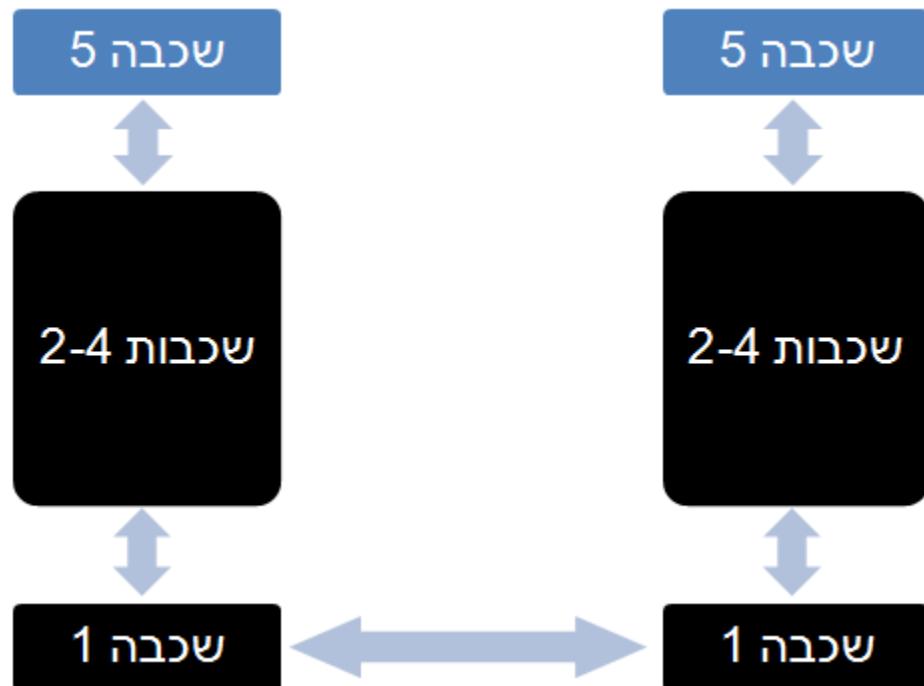
בדוגמה זו, סיפקנו את הפרמטרים של **שכבת הרשות (Network Layer)**: באיזו כתובת IP להשתמש. כמו כן, סיפקנו הפרמטרים של **שכבת התעבורה (Transport Layer)**: באיזה פורט להשתמש.

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

<sup>15</sup> למעט Raw Sockets, עליהם לא נפרט בספר זה.

בספר זה, נסקור את חמש השכבות מלמטה למעלה, 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 אליו כל " קופסה שחורה ", צו שرك מספקת שירותים ולא ברור כיצד היא פועלת (בדיקה כפי שהפרוטוקולים ברמות הגבוהות מניחים שהרמות שמתוחתיהן הן " קופסאות שחורות " וסוגיות מספקות להם שירותים שונים). ככל שנצלול לעומק ונגיע לרמות התחתונות, 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 התחתונות.



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



### למה המידע מחולק לפקודות?

למדנו על מודל השכבות, הבנו את החשיבות שליו 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 ביטים, שמתחליל כשישות אחת רוצה לשלוח מידע ליישות 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 Sağיות מוקדם יותר (לאחר שנשלח רק חלק קטן מרצף המידע השלם), ובמידה שקרתת Sağיה – לשЛОוח חדש או ורק את החלק הפוגם, במקום לשЛОוח את כל המידע מחדש.
- שילוב מספר זרמי מידע (Streams) במקביל: חלוקת המידע לפקודות מאפשרת לכמה אפליקציות לשLOWן במקביל את המידע שלהם ללא צורך להמתין קודם לשליחת תס'ים לשLOWן את המידע שלהם. חשוב על כך: כל כרטיס רשות יכול להוציא בכל זמן נתון ארוך ורך זרם נתונים אחד אל התווך אליו הוא מחובר. אם לא היוו מפצלים את המידע לפקודות, כל תוכנה הייתה צריכה לחתוך עד שההקו יתפנה, ולשלוח בתורה מידע דרך כרטיס הרשות. במצב זה לא הייתה מתאפשרה שליחת מידע במקביל בין מספר תוכנות<sup>16</sup>.
- 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 שLOWן מידע כלשהו, שאר המחשבים שנמצאים באותו קו היו מנועים מלשלוח מידע והיו צריכים לחתוך שהוא יסיים את השLOWחה. חלוקת המידע למסגרות גורמת לכך שבסוף כל מסגרת ניתנת הזדמנות לשLOWת אחרת ברשות להתחיל לשדר מסגרת משלה, ומונעת מישות אחת להשיף את הקו בקצב ארוך מאוד של ביטים<sup>17</sup>.
- מניעת בעיות סנכרון ברמת החומרה: לא ניתן לסייע הזו לעומק, אולם ניתן שברמת החומרה כל יותר לסנכרון בין מספר ישויות הקשורות על אותו קו כל עוד המידע מחולק למנוגדות, 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 יוזכו בהמשך הספר, ועל חלקן לא נדבר כלל.

## Wireshark

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

---

<sup>16</sup> התהילה' שבו מידע ממספר מקורות מסוולב אל תוך משותף אחד נקרא **ריבוב (Multiplexing)**. בדוגמה זו המידע מתקובל ממספר אפליקציות, והוא אל כרטיס הרשות (שהוא משאב יחיד המשותף לכל האפליקציות על אותה ישות).

<sup>17</sup> תופעה זו מכונה הרעבה (**Starvation**).

ונכל לגלות בדיק מה קורה ברשות ולהבין דברים שאין לנו דרך אחרת 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>  
 הפעלה שלכם.

עקבו אחר הוראות ההתקנה (במהלך ההתקנה תישאלו אם אתם רוצים להתקין *driver* בשם *WinPcap*. אשרו  
 והתקין 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 לעובדה בסיסית עם Wireshark בכתובת:  
<http://cyber.org.il/networks/videos/wireshark-basic.html>



ניתן לצפות בסרטון ההסבר על שימוש מתקדם יותר ב-Wireshark בכתובת:  
<http://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



### Interface List



Live list of the capture interfaces  
(counts incoming packets)

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



- דרך התפריט Wireshark->Interfaces ... (יש גם קיצור מקלדת: I + Ctrl)

בתפריט זה רשומים כל הכרטיסים דרכם תוכל להסניף, כתובת ה-IP שלהם, מספר הפקודות שנקלטו דרכם וקצב הפקודות שהם קולטים. כלל אכיבע – 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.

Device	Description	IP	Packets	Packets/s	
<input checked="" type="checkbox"/> Local Area Connection	Realtek RTL8168C/8111C PCI-E Gigabit Ethernet NIC	fe80::754d:3c3cae0b:8c21	509	2	<button>Details</button>
<input type="checkbox"/> Bluetooth Network Connection	Microsoft	fe80::8ac:3ed:a046:d950	0	0	<button>Details</button>
<input type="checkbox"/> VMware Network Adapter VMnet1	VMware Virtual Ethernet Adapter	fe80::d34:32c:dcf2:b3a8	0	0	<button>Details</button>
<input type="checkbox"/> VMware Network Adapter VMnet8	VMware Virtual Ethernet Adapter	fe80::a1d3:99ed:ad4d:7f37	0	0	<button>Details</button>

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 לחוץ על cptor Start ולהתחליל להסניף דרכו עם הגדרות ברירת המחדל, או לחוץ על cptor Options ולהגיע למסך של הגדרות ההסנפה.

## Capture Options

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

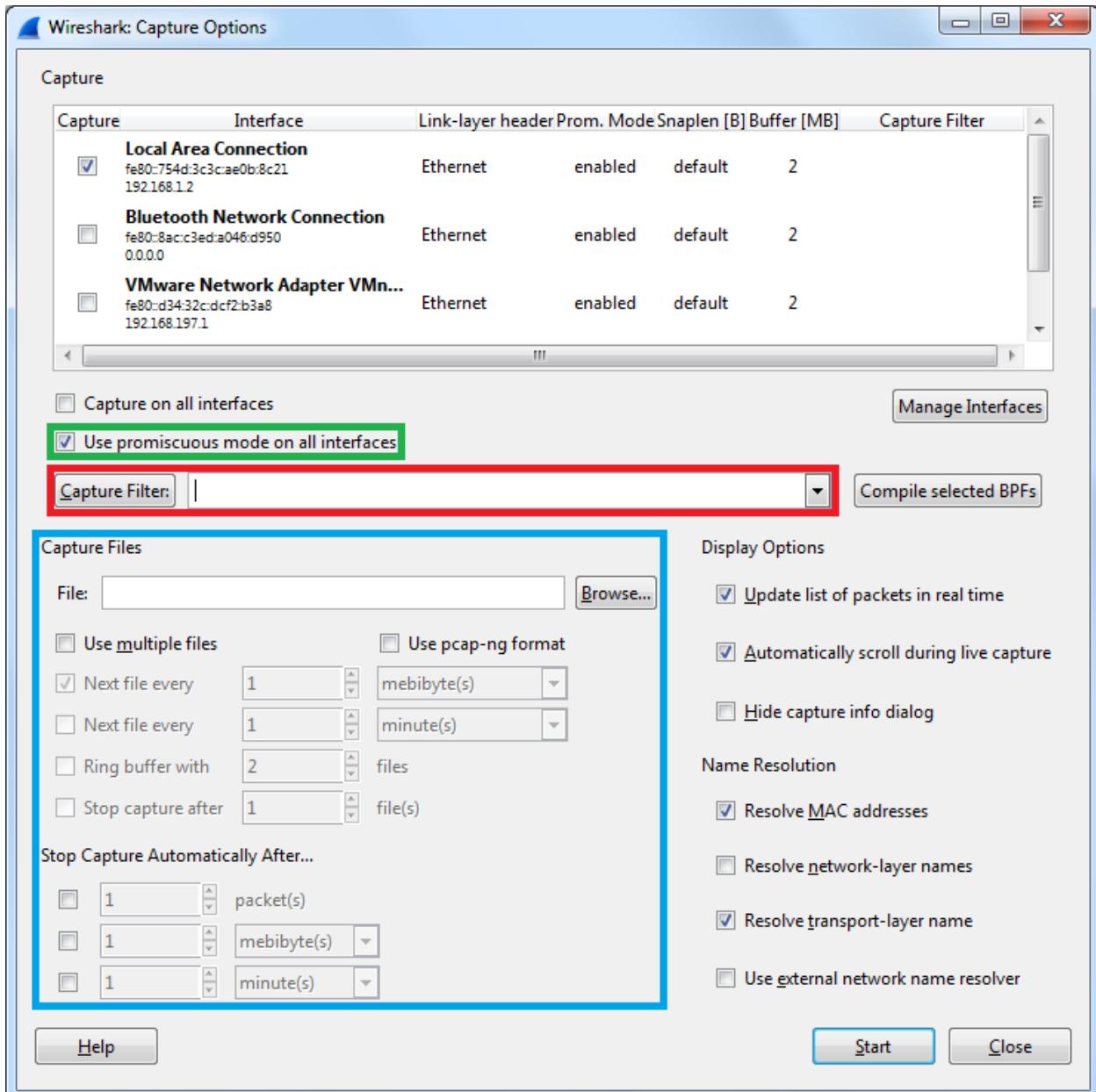
- דרך cptor Options עבור כרטיס ספציפי במסך ה-Interface List.

- 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

הaggerot bolotot b'masch zah:



- **בירוק - Promiscuous Mode:** הכנסת כרטיס הרשת ל"מצב פרוץ", מה שייגרום לכך שנראה בהසנפה את כל המסגרות שרוואה כרטיס הרשת, גם אלו שלא מיועדות אליו (את המשמעות של משפט זה נבין בהמשך הספר).
- **באדום - Capture Filter:** מגדיר מסנן של פקודות להסנפה עבור ה-Driver (נறחיב על משמעות מסנן זה בהמשך הפרק).
- **בכחול - 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 ויצירה 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 Capture -> Start (יש גם קיצור מקלדת: E + Ctrl).

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

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



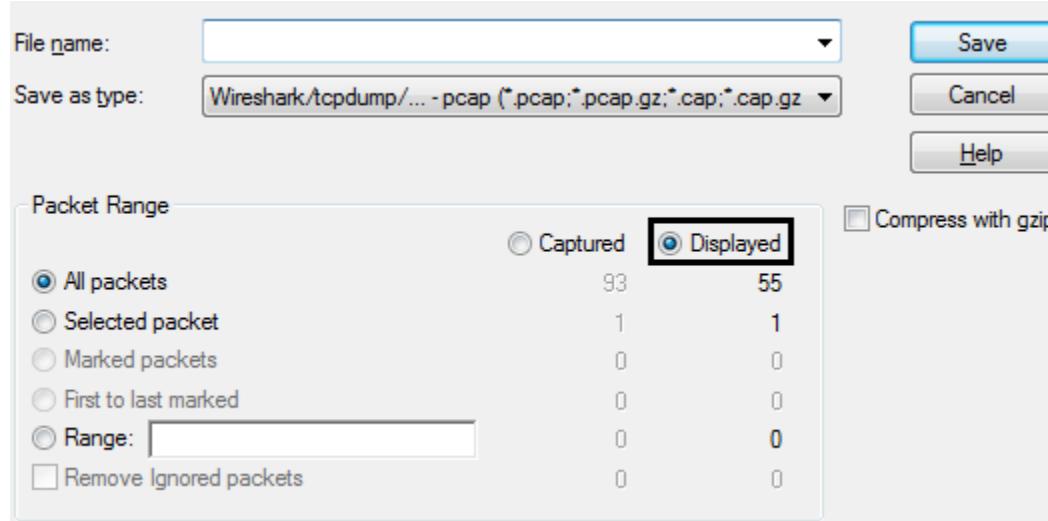
- **התחלת מחדש של ההסנפה** – ניקוי רשימת הפקודות והתחלה הסנפה חדשה על אותו interface (כרטיס רשת), עם אותם מאפיינים, ואוטו Display Filter. גם את האפשרות הזו ניתן להפעיל דרך cptutor בסרגל הכלים העליון או דרך התפריט Restart -> Capture (יש גם קיצור מקלדת: Ctrl + R):



### שמירה ופתיחה של קבצי הסנפה

כדי לשמר את הפקודות שנקלטו, נעצור את ההסנפה ונלחץ על File -> Save (קיצור מקלדת: Ctrl + S). הקובץ שישמר 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וצגות כתעט למסר), נפתח את התפריט Export Specified Packets -> File... שיציג לנו את חלון השמירה הרגיל אך יוסיף לנו את אזור ה-Range Packet שמאפשר לבחור אילו פקודות לשמר:



במידה שנרצה לשמר רק את הפקודות שעונתו 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 + I).

## מסננים (Filters)

### סוגי מסננים

כפי שכבר הוזכר, ישנו שני סוגי מסננים: מסנן תצוגה (Display Filter) ומסנן הסנפה (Capture Filter). Capture Filter, בשונה מה-Display Filter (שמיועד 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:

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

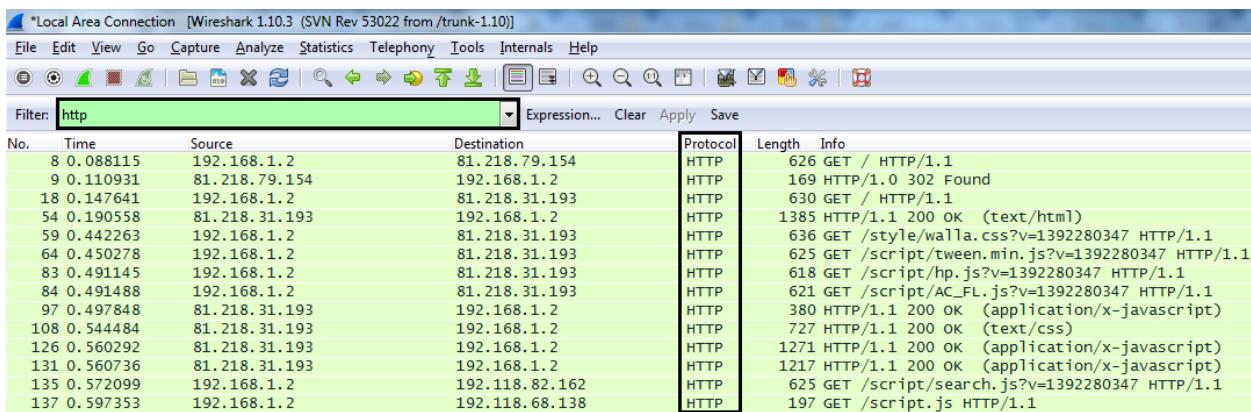
מוצומצם	עשיר ורחב	תחביר
מהירות	אייטי	

מידע נוסף על Capture Filters תוכלו 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.
  - לדוגמה: ip, arp, tcp, http



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

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

<ProtocolName>.<FieldName> Operator <Value>

- לדוגמה:

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 שאין 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, ולאחריהם בינויהם באמצעות קשר לוגי: or / and.

- לדוגמה:

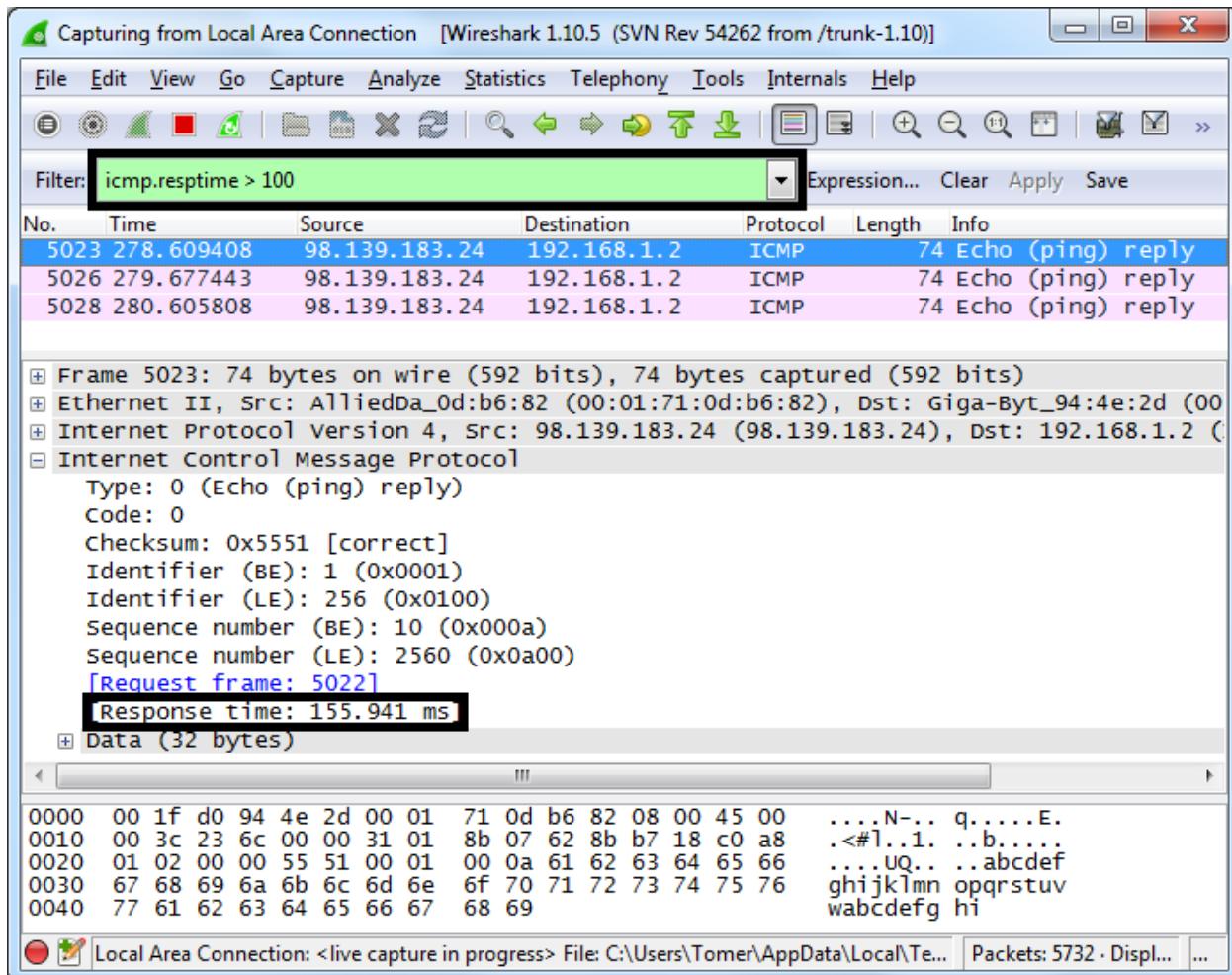
`ip.addr == 192.168.1.1 or tcp.port == 22`

(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:

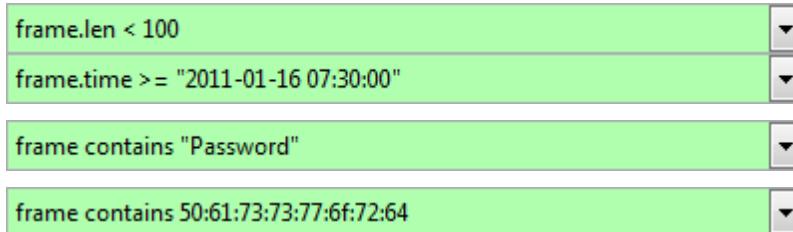
- למשל:

(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 שמווע בה במקום מסוים ("frame.time".

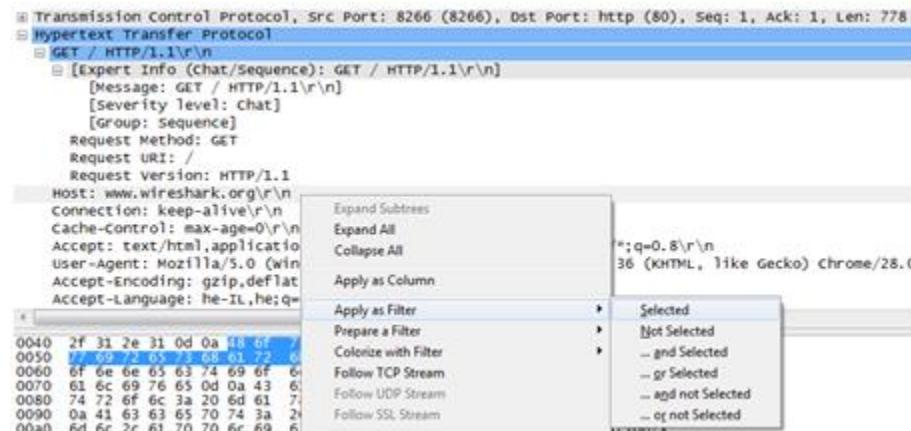


הדוגמא האחרונה זהה לזה שלפניהם, רק שהיא מאפשרת לתת ערכי הקוסה של הבטים במקום יציג ה- ASCII שלהם)

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 לקרואם לפilter שאתם מחפשים, וגם כדי ללמד בלבד על עוד פרוטוקולים ושדות עצמאכם:

1. ניתן להיעזר בתפריט ה-Expression, שעזר לנו לבנות ביטויים כאלו.
2. בערת לחיצה ימינו על שדה כלשהו בחלון ניתוח הפקטה, ובחירה ב- .Apply As Filter -> Selected.

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



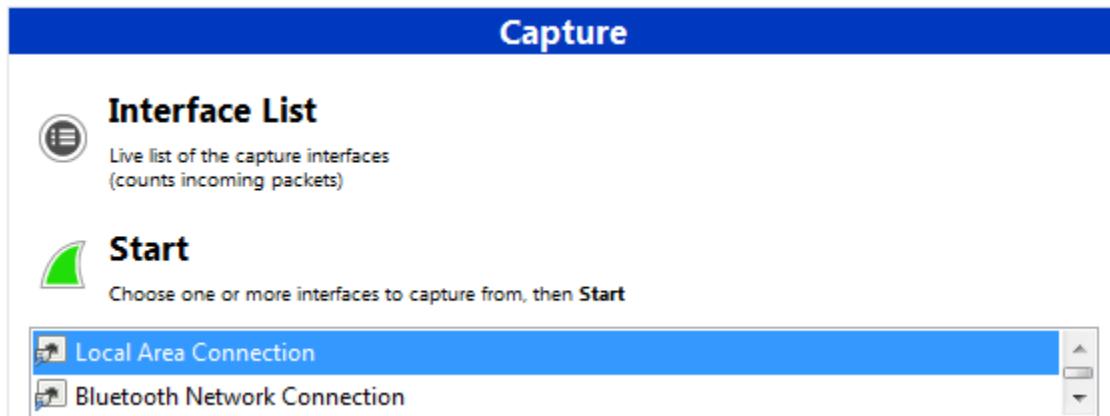
עכשו ניתן לראות את הфиילטר ש-Wireshark יצר עבור השדה הזה:

Filter: http.host == "www.wireshark.org"

שימוש לב באמצעות שיטות אלו אתם יכולים **ללמד את עצמכם** כיצד להשתמש ב-Wireshark, לגלוות פילטרים חדשים ואפשרויות שלא הכרתם עד כה.

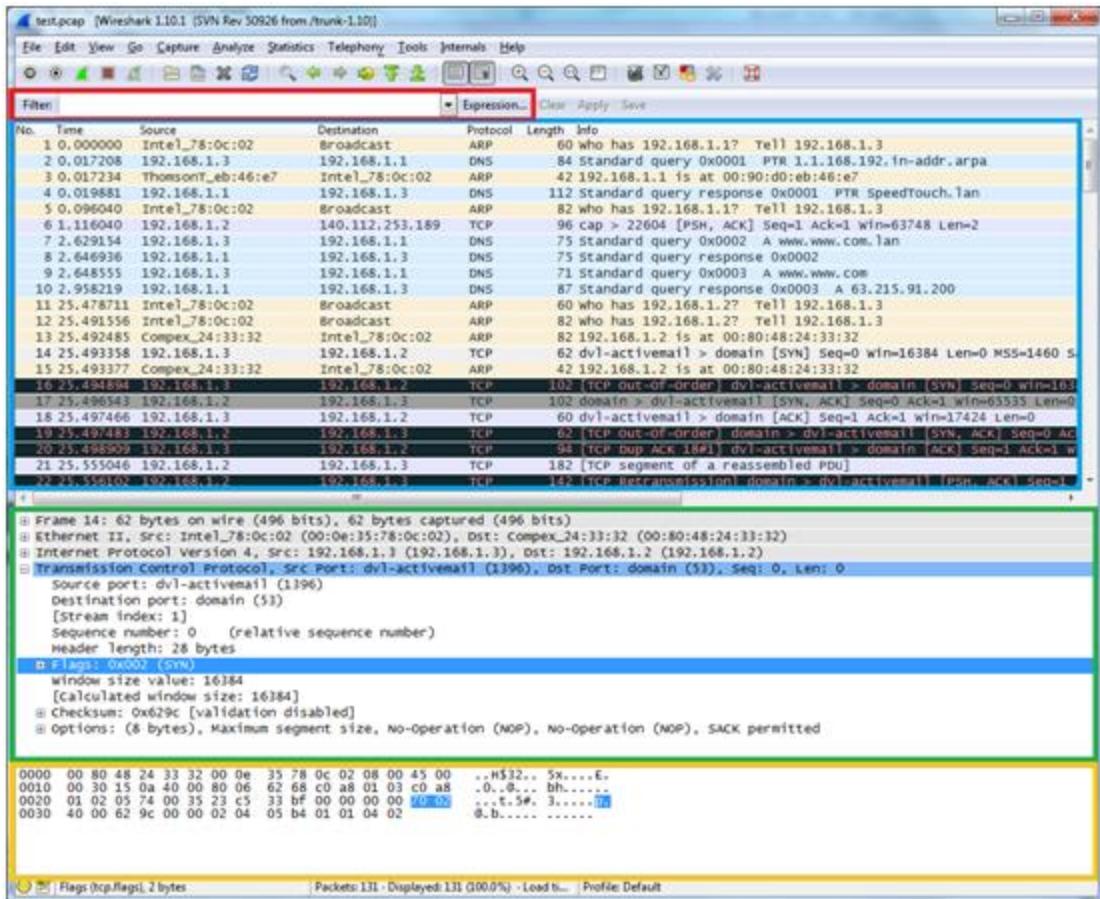
## שימוש ב-WIRESHARK לניתוח מודל חמש השכבות

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 לרשת דריך מתאם אלחוטי) ולחצו על .Start



כעת נפתח לכם מסך ההסנפה הראשי. אם תמתינו מספר שניות תוכלם לראות שבמרכז המסך מתמלאות להן שורות-שורות (אם לא – נראה שבחרתם בכרטיס הרשת הלא נכון. נסו לפתוח הסנפה חדשה על כרטיס אחר). השורות הללו מציגות **פקטות (חבילות מידע, Packets)** שכרטיס הרשת שלכם מוציא או מקבל. כזכור מההסבר אודות מודל חמש השכבות, פקטה היא מעין מעטפה שמכלילה מוען, נמען ואת תוכן הודעה – וכך מאפשרת 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 פקטות.

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 מסוימים. כרגע זו עוללה להיראות לכם כיכולת לא ממש חשובה, אבל בפעם הראשונה שתՏנינוו תראו כל כך הרבה פקודות - ותבינו שבהרבה 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 ל כתוב.

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 בתווים (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 כלי חזק, והוא מקשר לנו בין כל השדות של הפוטווקול 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 הדדים מהפרק הקודם

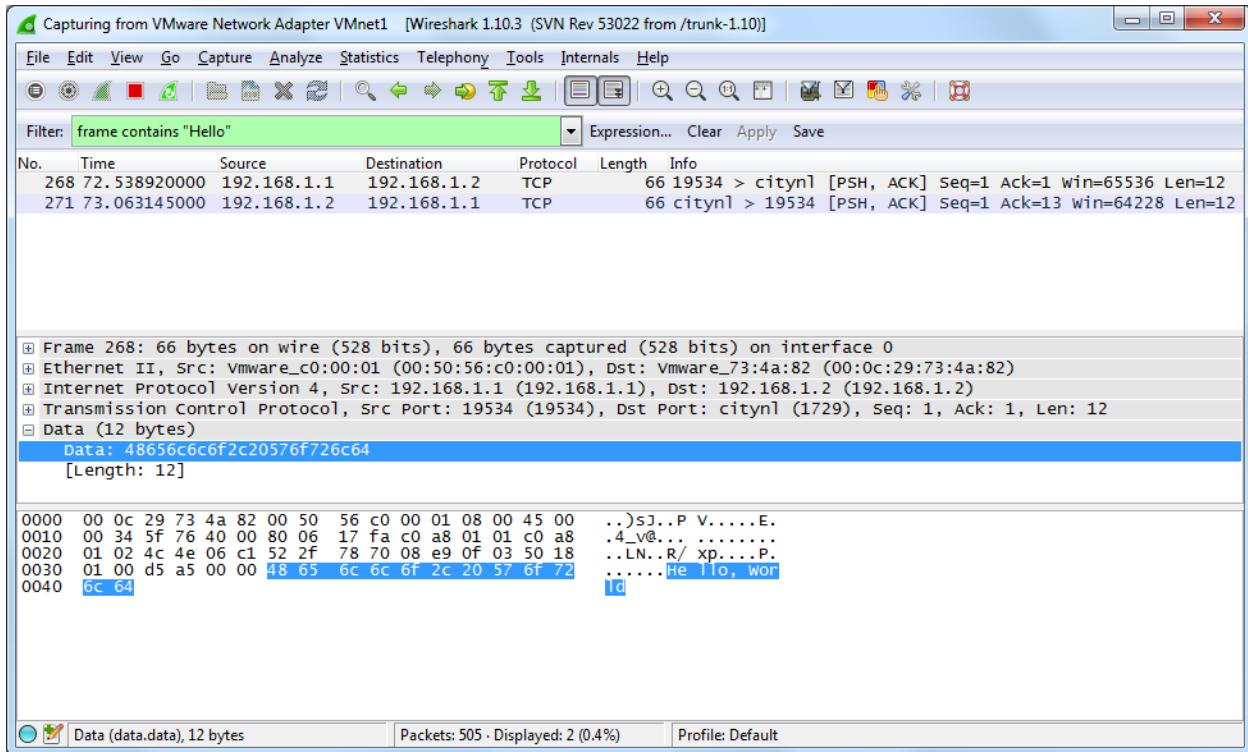
טרם נתמקד בהסנפה, הפעילו את הסكريיפטים שככובתם בתרגילים 2.2 ו-2.5 בפרק הקודם (סקריפט הלקוון וסקריפט שרת הדדים) על מחשבים נפרדים. הריצו את `uk.py` על מחשב מרוחק ואת `Client.py` על המחשב הנוכחי (הסיבה לכך שאנו מרים את סקריפט הלקוון וסקריפט השרת על מחשבים נפרדים ולא על אותו מחשב נועצה בעובדה שהרבה יותר קשה להסניף את המידע שנשלח אליו המחשב דרך הכתובת 127.0.0.1 עלייה דיברנו קודם<sup>18</sup>). לאחר שיידאתם כי סקריפט השרת והלקוון מצליחם לתקשר ביניהם על גבי מחשבים נפרדים, הריצו אותם שוב בעודם פועלם 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 שנשלח דרך כרטיס הרשות שלנו ולכן נקלט בהסנפה. אם הייתם זרים, אולי הצלחתם לזהות את הפקודות הרלוונטיות מבין כל הפקודות שהוצעו במסך ההסנפה, אך גם אם לא – אנו נשתמש באופציית `sinon` הפuktות כדי להציג רק את הפuktות הרלוונטיות בלבד. רשמו בשדה `Display Filter` את ה-`Filter` הבא:

(ה필טר הזה גורם לכך שיוצג רק הפuktות שמופיעעה בהן המילה Hello (Hello SHello))

---

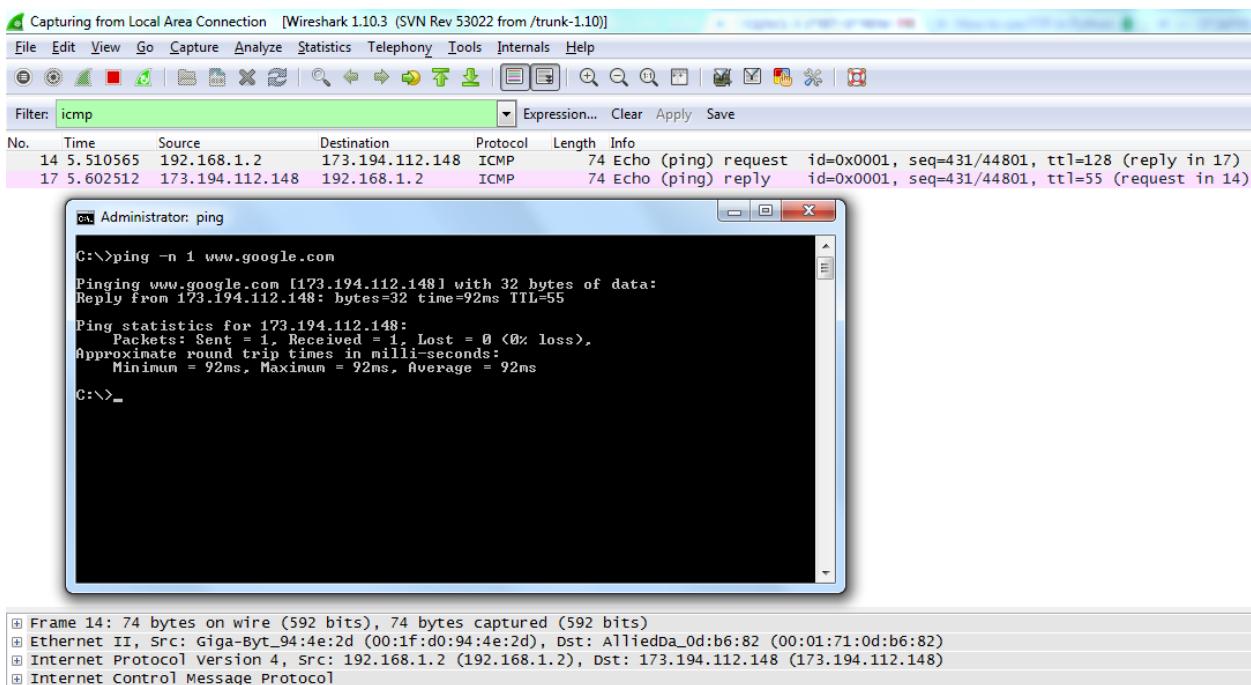
<sup>18</sup> במערכות הפעלה Linux (בשונה מ-Windows), ההסנפה על התעבורה שנשלחת למחשב עצמו דרך 127.0.0.1 היא פשוטה וזהה להסנפה על כל כרטיס רשת אחר. גם ב-Windows ניתן להסניף תעבורת שנשלחת אל 127.0.0.1, אולם התהליך מסובך יותר ודורש התקינה של תוכנות מיוחדות. אי כך, לא נעשה זאת בספר זה.



אפשר לראות ש毛病עה 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 פקחת ping, אותו ping 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.

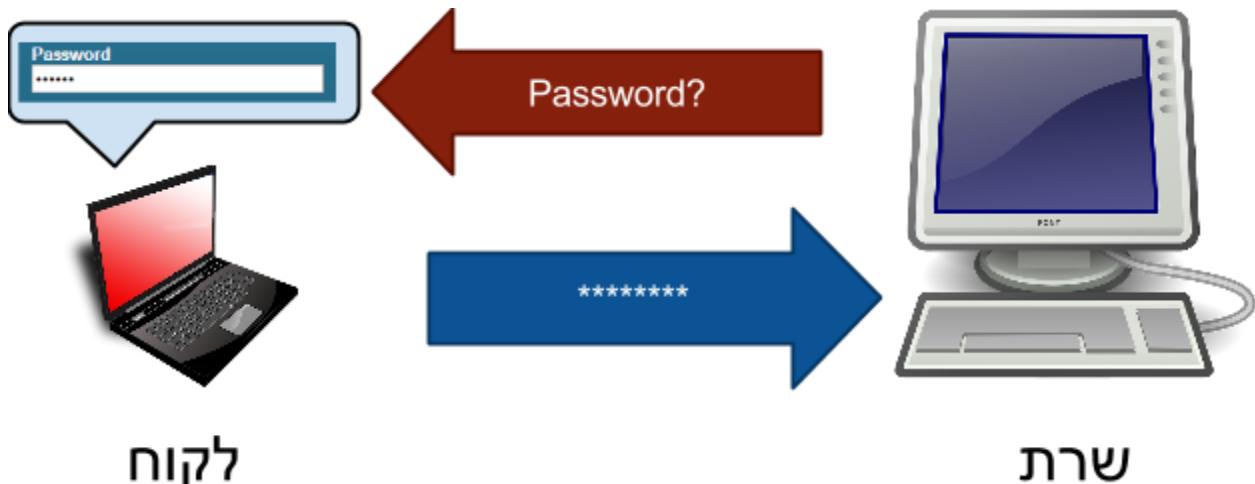


שיםו לב לתוצאה שהתקבלה: שלחנו בקשה אחת (המכונה "Echo (ping) request", ניתן לראות זאת בעמודת ה-Info) אל השרת, וקיבלנו ממנה תשובה (Echo (ping) reply) מיד לאחר מכן. שימו לב גם אל כתובות ה-IP של הפקטות – הפקטה הראשונה ובها הבקשה נשלחה מכתובת ה-IP שלנו (192.168.1.2) אל השרת של Google (173.194.112.148), ואילו 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-**ping** הוא הפילטר **icmp**, שהוא הפרוטוקול בו עוברות בקשות ותשובות **ping**. בהמשך הספר נרחיב את הדיבור על פרוטוקול זה ונלמד לעומק איך **ping** עובד. בנוסף, אפיילו נכתב כל' דמי **ping** בעצמינו!

### **תרגיל מודרך 3.2 - הסנתט 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://cyber.org.il/networks/links/plain-password.html>, היכנסו את כל הפרטים הדרושים כולל סיסמה (אל תשימו את הסיסמה האמיתית שלכם!) והירשמו לאתר. כמובן שבזמן הלחיצה על כפתור Register וודאו כי פועלת הסנפה ברקע. כתעת, היכנסו את הфиילטר הבא למסך התצוגה:  
**frame contains "txtPa"**

לחצו על Enter. צפיה להתקבל פקטה אחת שנעה על הфиילטר. הקliquו עליה ברשימת הפקטות, ועכשו תוכלו להסתכל על תוכן שלה בתחתית המסך. נסו לזהות מהו מפתח בחלק האחרון של הפקטה, ורק אחרי שתגלו – עברו לפסקה הבאה.

Screenshot of Wireshark showing a captured packet. The filter bar at the top has the expression **frame contains "txtPa"**. The selected packet (Frame 281) shows a POST request to `/BookBag/Registeration`. The packet details pane shows the following information:

- Source: 192.168.1.2
- Destination: 199.168.12.131
- Protocol: HTTP
- Length: 1476
- Info: POST /BookBag/Registeration

The packet bytes pane shows the raw hex and ASCII data. The ASCII dump highlights the password field, which is obscured by a green box. The bytes dump shows the corresponding hex values.

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

הדוגמה הקצרה זו הייתה אמורה להמחיש לכם דבר שאולי העליתם על דעתכם כבר קודם – כל דבר שנשלח לישות כלשהי ברשות יוצא מהמחשב שלכם ועובד דרך כרטיס הרשת, בין אם הוא גלוי לעיניכם (הטקסט שבחרתם לשЛОח דרך ה-Socket בפייתון, פוט שאתם מפרנסים בעמוד הפיסבוק שלכם) או סמי מהן (הסיסמה שモצתת בתור כוכיות באתר אך נשלחת בגלוי, או תזוזה של השחקן שלכם במשחק רשות אל מול שחקנים אחרים בעולם).

דרך אגב, אם הדוגמה האחרונה גרמה לכם לחשוש שהסיסמה שלכם גלויה לכל מי שייהיה נגיש למידע שיצא מהמחשב שלכם ויכול להסניף אליו – הדאגה שלכם הגיונית, אולי אין לכם ממה לחוש. ביום, הרוב המוחלט של האתרים משתמש בשיטות אבטחה שונות כדי להבטיח שהסיסמה שלכם לא תעבור בצורה גלויה ברשות האינטרנט, אלא תוצפן אצל הלוק והפוענה רק כשהתגיעה לשרת.



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

1. פתחו חלון cmd ורשמו **cmd 8.8.8.8 ping**. הסתכלו על ההסנה ונסו למצוא את הפקודות שנשלחו (תשכורת). העזרו בפילטר "icmp", שהוא הפרוטוקול שבו עוברות בקשות ה-ping. כתעת, חשבו את ה-round trip (משך הזמן החל מרגע שליחת הבקשה ועד קבלת התשובה) על-פי שדה Time-in-Wireshark (שצচוך מציג את הזמן בו עברו הפקודות בכרטיס הרשת, בפורמט של שניות מתחילה ההסנה). בדקו את עצמכם – האם הגיעتم לאותה תוצאה שהופיעה בחולון ה-cmd?

2. השתמשו ב-**ping** בשם - (SKUבע את גודל המידע שיישלח בפקחת ה-ping), והריצו את השורה הבאה:  
**ping 8.8.8.8 -n 1 -l 500**

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

3. הסתכלו על תוכן הפקטה שנשלחה בסעיף 2, בחולון המציג את תוכן הפקטה. בקשת **ping** מכילה בתוכה מידע, והוא מצפה לקבל את תשובה ה-ping עם אותו מידע בדיק (בדומה למה שעשיתם בפרק תכונות ב-Sockets כשמיימסתם שרת הדימ). ב-Wireshark, ניתן לראות את המידע הזה בשדה "Data" של פקחת ה-ping. תוכנו להזות מהו המידע שנשלח בבקשת **ping** אל השרת?

## Follow TCP/UDP Stream

אפשריה נוספת אך חשובה להפליא היא Follow TCP Stream או Follow UDP Stream (השימוש הראשון הוא הרבה יותר נפוץ). אפשרות זו שמאופעלת על פקטה, מפלטרת את כל הפקטות השייכות לוותם "Stream" (כלומר NSLCHO ותתקבלו דרך אותו Socket<sup>19</sup>), ובכך מאפשרת לראות את התעבורה "דרך העניינים של ה-Socket" באפליקציה. נדגים את השימוש באפשרות זו בעזרת התרגיל שכתבתם בפרק תכונות ב-Sockets / תרגיל 2.6.

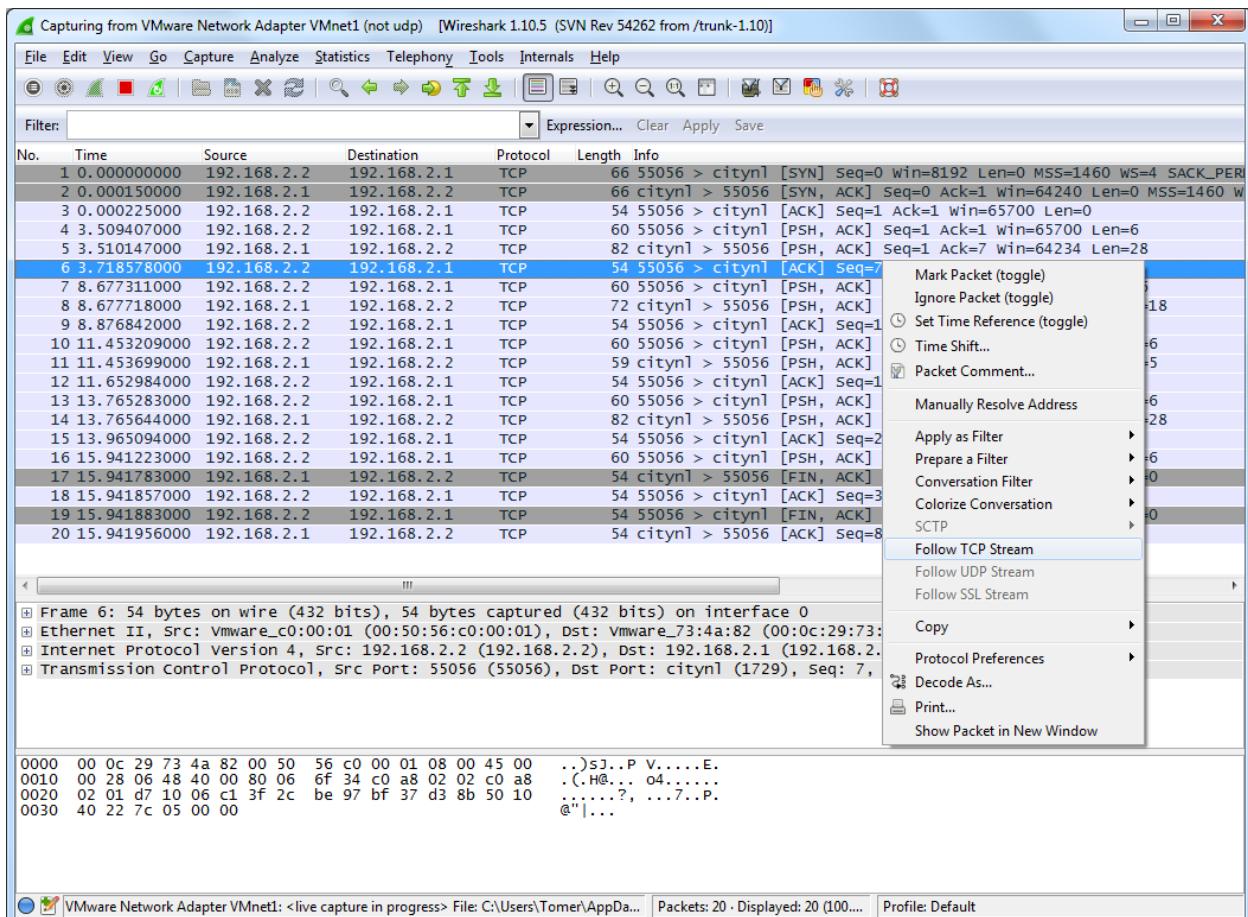
שרת פקודות בסיסי:



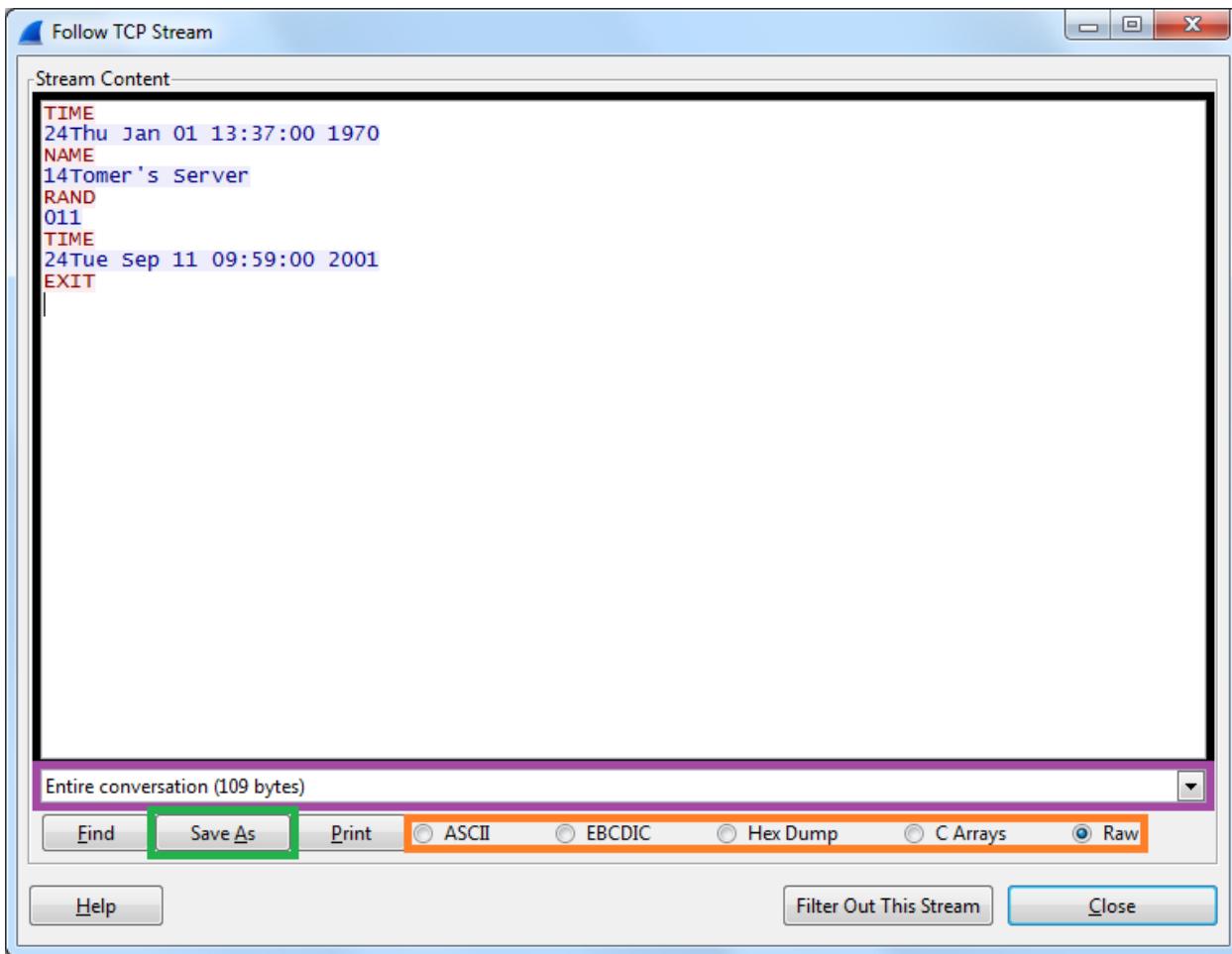
### תרגיל 3.4 מודרך - צפייה במידע של Socket בעזרת TCP Stream

- השתמשו בלקוח שכתבתם בתרגיל שרת פקודות בסיסי (TIME, RAND, NAME, EXIT).
- שנו את קוד הלקוח כך שהוא יפנה אל שרת גברים שנמצא כתובות בכתובת networks.cyber.org.il בפורט 8850 (מה כתובות ה-IP אליה יש לפנות? מיצאו אותה עצמאם בעזרת הכלים שלכם).
- התחילו הסנפה חדשה והריצו את הלקוח. פיתחו את ההסנפה.
- בחרו את אחת הפקטות שהועברו בין שני המחשבים כחלק מאותו Socket, לחזו על הceptor הימני של העבר ובחרו באופציה "Follow TCP Stream".

<sup>19</sup> למעשה, הכוונה היא לכל הפקטות שהן חלק מה-Stream בשכבה התעבורה. על המשמעות של מושג זה, וכייז Wireshark מצליח להבין אילו פקטות שייכות לאיזה Stream - נלמד בפרק שכבת התעבורה.



• ● נפתח חלון/socket המציג את כל המידע ש עבר על גבי ה-socket :

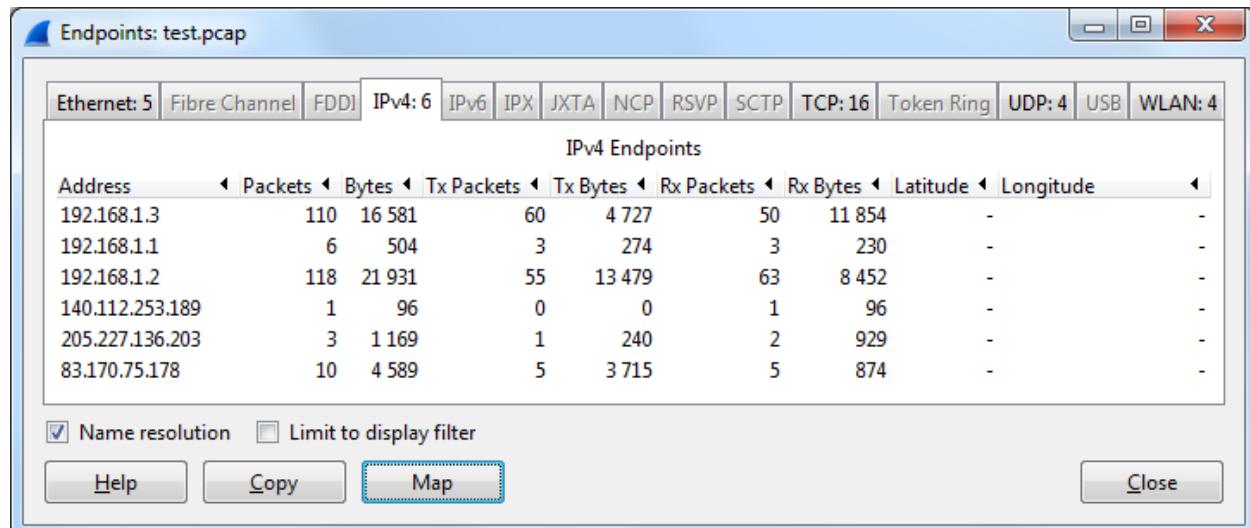


שיםו לב, כי מופיע כאן רק ה-Data השיר לשכבות האפליקציה, ללא כל ה-Header'ים והפרמטרים ששימושו לחיבור בין שתי נקודות הרצה. אם תסתכלו על המסגרת **השחורה**, תראו שככל צד בתקשורת מופיע במצב אחר - **באדום** התקשרות בין הלקוח לשרת, ובכחול התקשרות בין השרת ללקוח. ניתן להזיהות את ה프וטוקול בו בחרנו להעביר את המידע חזרה מהשרת ללקוח: שליחה של 2 ספנות המסמלות את אורך התשובה, ומיד לאחריה נשלחת התשובה עצמה.

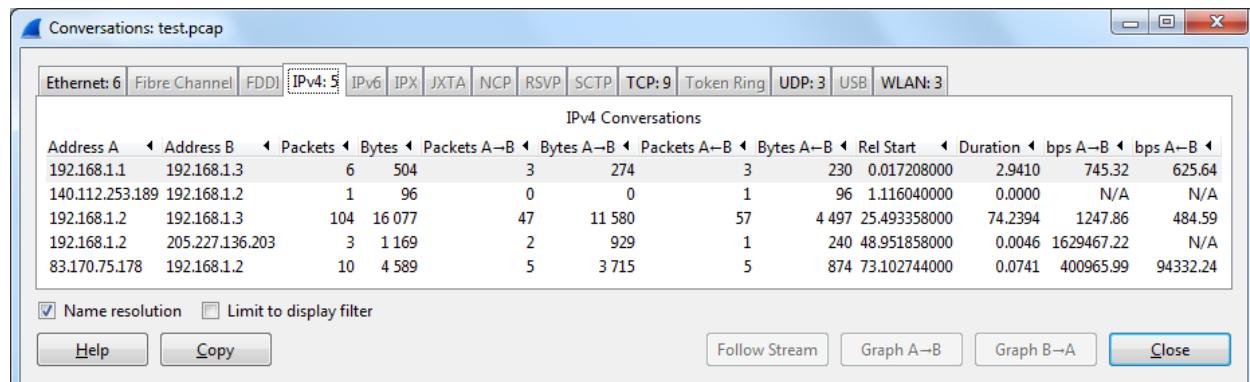
בנוסף, ניתן להגביל את הצפייה רק לאחד הצדדים של התקשרות (המסגרת **סגוליה**), לבחור את הייצוג בו נרצה לצפות במידע (המסגרת **כתומה**) ואף לשמור את המידע לקובץ ובכך לקבל את כל המידע שעבר ב-**Socket** (המסגרת **ירוקה**).

## 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 בהסכמה (Packet Lengths), גוף שמתעדכן בזמן אמיתי ומציג את כמות התעבורה (IO) וישיות רשותן שנקלטו בהסכמה (Endpoints).



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



### תרגיל 3.5 - דיהוי שרתיים בחולון ה-Endpoints וה-Conversations



השתמשו בכלי **ping** שהכרתם בפרקם הקודמים, על מנת לבדוק קישוריות אל [google.com](http://www.google.com), [ynet.co.il](http://www.ynet.co.il) ו-[walla.co.il](http://www.walla.co.il). נסו למצוא את השיחות שלכם עם אותן האטרים בחולון ה-*Conversations* (כבר למדתם כיצד להמיר בין ה-*Domain IP* לבין IP, כך שתצטרכו לזרות שיחות בין ה-IP שלכם ל-IP של השירות המרוחק). בנוסף, נסו לזרות אותן גם בחולון ה-*Endpoints*.

## מודל חמש השכבות - סיכום 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 שרת החדים שכתבתנו בפרק הקודם, הסנונו גם שימוש בכלים **ping** ו-**tracert** ואףלו סיסמאות שעוברות בגלוי בראשת האינטרנט – הכל כדי שתיווכחו כמה כוח יש ל-**Wireshark** ומה אפשר לגלוות 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 שמטהה), באופן שהוואה מפת דרכים להמשך הלימוד בספר. כתה, שנצללו לעומק 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 בראשות.

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

## פרק 4 - שכבת האפליקציה

מצויידים בכלים שלמדנו - הסנפת תקשורת בין מחשבים ותכנות Sockets בפייתון, ולאחר שלמדנו על מודל חמש השכבות, אנחנו מוכנים להתחליל ולהקhor את השכבה הראשונה שלנו - שכבת האפליקציה.



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

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



**שכבת האפליקציה היא אוסף הпрוטוקולים בהם עושים שימוש באופן ישיר, והיא מספקת הפשתה מעלה תקשורת הנתונים בראש האינטרנט.**

עד סוף הפרק, נבון בדיק מה המשפט הזה אומר. מעבר לכך, נכיר לעומק איך עובד פרוטוקול HTTP ואת היכולות שהוא מספק, ונתבונן כיצד אתרים ואפליקציות כמו book Google או המפות של Facebook 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 שכבת האפליקציה - HTTP - המשמש לגלישה באינטרנט. לפני שנכבר במיילים, נתחיל בהסתכלות על הפרוטוקול.



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



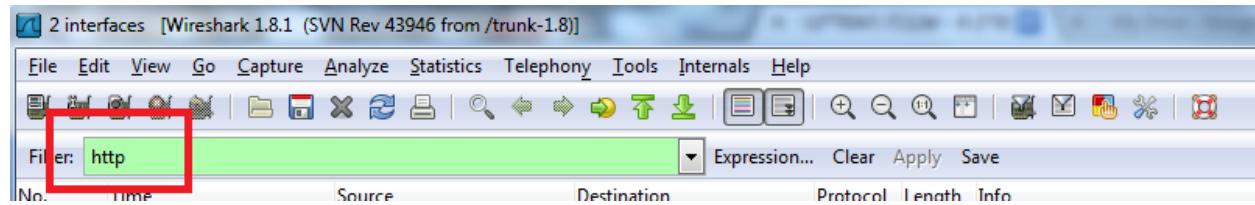
מהו מושב רשות?

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 רשות. כך גם חשבון Twitter שלו, וכן גם כתבה בפייסבוק.



#### תרגיל 4.1 מודרך - התנסות מעשית בתקשרות HTTP

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

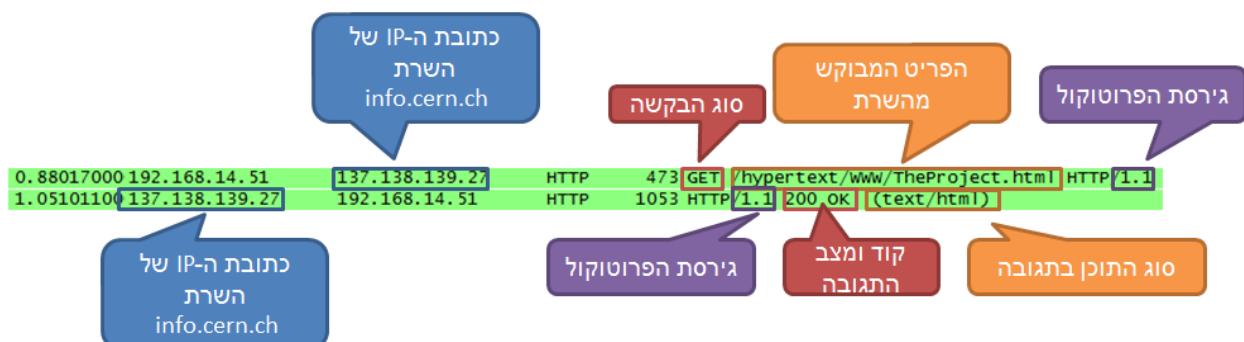


icut פתחו את הדף החביב עליהם, והכניסו את הכתובת הבאה:  
 http://info.cern.ch/hypertext/WWW/TheProject.html  
 לאחר שהדף יטען, תראו עמוד קצר ובו מספר שורות.



איך הדף נטען בדף שלנו? מה בעצם קרה כאן?

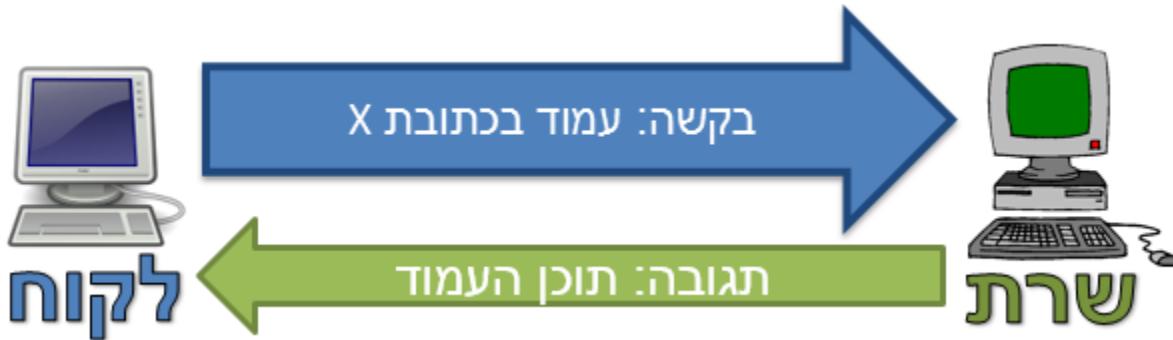
נחזיר אל Wireshark כדי למצוא את התשובה - עצרו את ההסנפה, ושימו לב שנוצרו שם שתי שורות:



**עיקרון "בקשה-תגובה"**

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

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



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

### שורות הבקשה של HTTP

הפקטה בשורה הראשונה היא הבקשה שנשלחה מהлокוח (במקרה זה - מהדפן שלכם) אל שירות באינטרנט. בשורה השנייה אנחנו רואים את התשובה שקיבל הדפן שלכם מאותו שירות, ואזותה ננתה בהמשך. נתבונן ביחיד ובין איך הפרטוקול בני - שימושו לבשבוקטת הבקשה ניתנה הכוורת:

GET /hypertext/WWW/TheProject.html

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

### שורות התגובה של HTTP

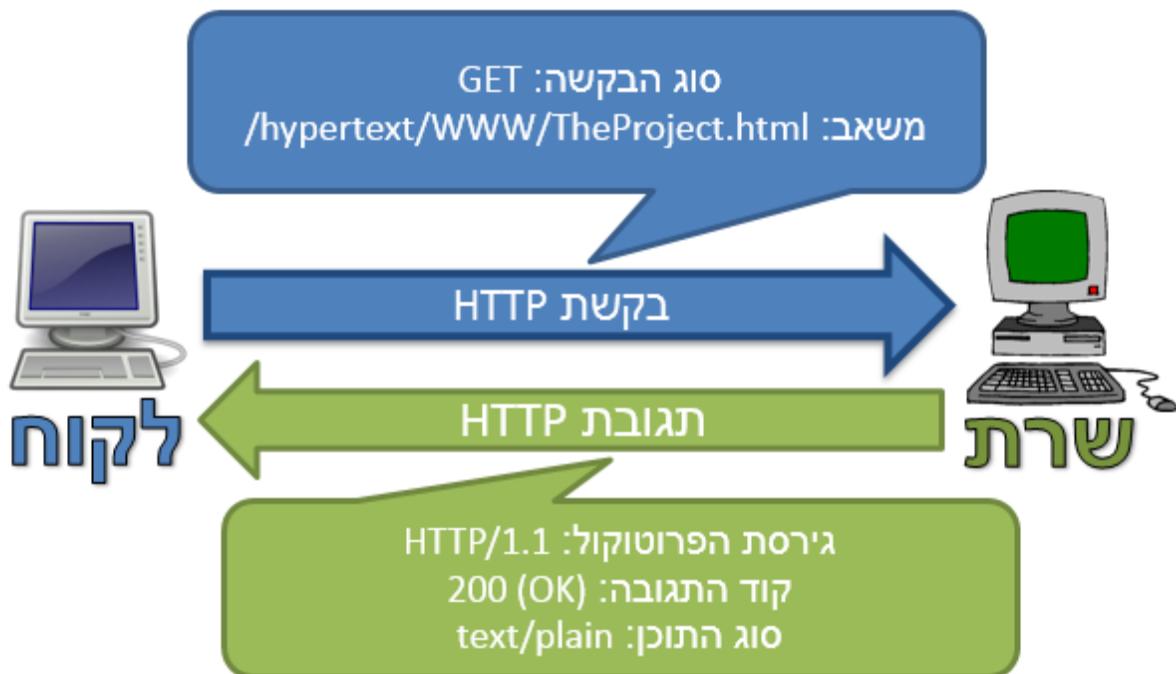
בשורות התגובה מופיעו "HTTP/1.1 200 OK" (text/plain) - למעשה יש בשורה הזו שלושה נתונים:

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

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

3. סוג התוכן שבתשובה - במקרה זה התקבל 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 שתכניות בדפדף את הכתובת:

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

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

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

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

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:d2:28:2d: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]
```

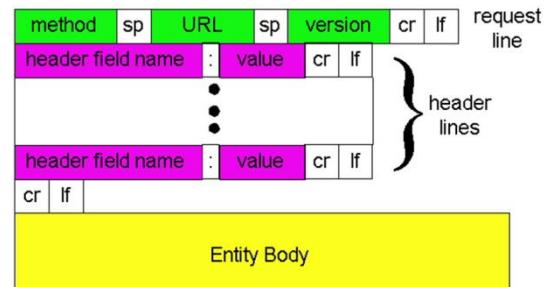
חלק משדות ה-Header יכולים להופיע גם בבקשתו וגם בתגובה (למשל: אורך התוכן), חלק יופיע רק בבקשתו (למשל: סוג התוכן שהליך מוכן לקבל בחזרה, "סוג" הלקוח - לדוגמה: דפדפן כרום) וחלק יופיע רק בתגובה (למשל: "סוג" השירות).

רשימה של שדות Header ניתן למצוא בכתובות:

[http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_header\\_fields](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 ברצף של שני תווים - ז' ומיד אחריו <sup>20</sup>.

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

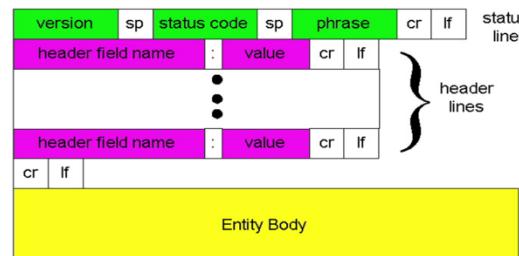
מכיוון ששורת הבקשה היא שורה אחת בלבד, ניתן לצפות שמיד עם סיום השורה (הופעת התווים ח'ז') יתחלו שדות ה-Header. כל שורה בחלק זה מכילה שדה אחד ואת הערך שלו. לאחר שדה ה-Header האחרון, יופיע כרגע "סיום שורה" (ח'ז'), והשורה שלאחר מכן תהייה ריקה - כלומר מיד יופיעו שוב ח'ז'. זהו סימן לכך שלב ה-Header הסתיים,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

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

מצד שמאל נתונה דוגמה לבקשת GET - נסו לזהות את החלקים השונים בבקשת (תזכורת: חלק התוכן יהיה ריק), את התווים המפרידים ביניהם, ונסו לזהות שדות Header שדרנו 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 שולשה של שלושה חלקים, ואחתה דרך המפריד ביניהם. עיברו על הדוגמה מצד 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 בפסקה 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 (בתוכר משתמשים) מעוניינים בתוכן עצמו. בנוסף של דבר, האפליקציה צריכה לקבל את התוכן כדי להציג לנו תמונה, הודעה או מסמך.

לכן, כשליליקציה מחליטה לבקש משאב מסוים, היא תבדוק את קוד התגובה כדי לוודא שהיא תקינה - למשל, לוודא שהתקבל קוד OK 200 (כמו שראינו בדוגמה הקודמת). אם למשל התקבל קוד 404 (משאב לא נמצא), היא תציג הודעה שגיאה מתאימה.

לאחר מכן, האפליקציה תבדוק את סוג התוכן כדי לוודא שהוא תואם את מה שהוא ציפתה לקבל - למשל, אם היא ציפתה לקבל תמונה וחזר תוכן מסוג `text`, נוראה שיש בעיה.

לבסוף, האפליקציה תציג את התוכן שנמצא בתגובה - למשל תנגן סרטון וידאו או תציג הודעה טקסט.



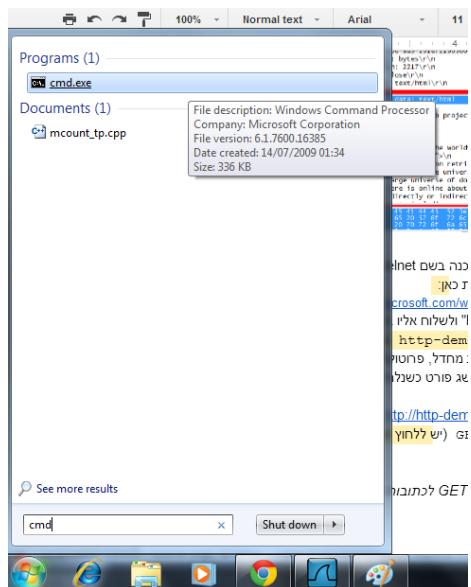
## התבוננות מודרcta בתגובה HTTP

הגיע הזמן שנתבונן בתוכן של תגובות HTTP - נחזור לדוגמה שראינו ב-Wireshark, ועכשו "נפתח" את פקעת התגובה (ונראה את מה שקרהנו לו בתרשים "תגובה HTTP ...").

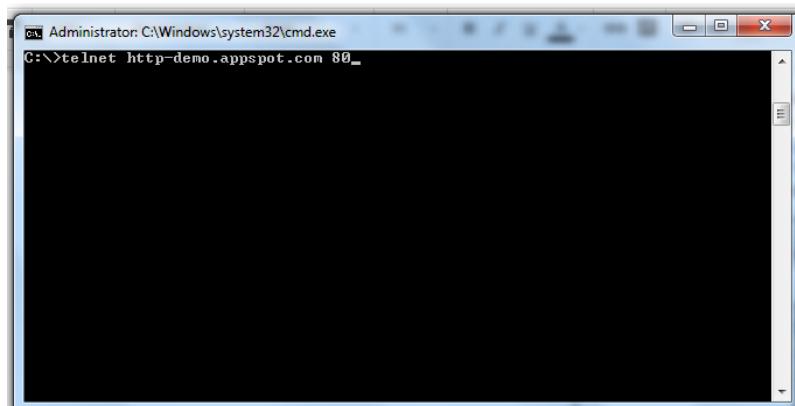
The screenshot shows a Wireshark capture window with several network packets. The third packet from the top is highlighted in green and shows an HTTP GET request to '137.138.139.27'. The fourth packet is highlighted in green and shows an HTTP response (200 OK) with the URL '/hypertext/www/TheProject.html'. The response body contains HTML code. A red box highlights the first few lines of the response header, and a red arrow points to it with the text 'שדות header'. Another red box highlights the entire HTML body, and a red arrow points to it with the text '(טוקן (data))'.

כדי לבדוק את התקשרות הזו בצורה נוחה יותר, נשתמש קלקו בתוכנה בשם `telnet` (במקום הדפסן). הנחיות כיצד להפעיל אותה על מערכת הפעלה Windows 7 נמצאות בכתבota: <http://goo.gl/Oo9DgK>

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



.telnet http://http-demo.appspot.com 80 (Command line (ה-CLI)).



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 / תרגיל 2.3 מודרך - השרת הראשון שלו](#), ולצורך מה שimes מס' הפורט).

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

`GET /a.html HTTP/1.1`

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

---

<sup>21</sup> יתכן שנצרך להזיף Header Host מסוג Host כדי שהשרת יסכים לקבל את הבקשה. לשם כך, יש להקשיב-`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 ביצוע כתובת מלאה:

GET www.google.com/ HTTP/1.1

כאשר אנו גולשים בדרך כלל לאתר אינטרנט, איןנו מציינים משאב ספציפי. אי' לכך, אנו מבקשים למשא את המשאב אשר נמצא בכתובת "/". כשהשרת מקבל פניה למשאב זהה, הוא יכול להציג את העמוד שברצונו להציג בעת גלישה של משתמש לאתר. כעת נבקש את המשאב שסקול לגלישה לכתובת:

באמצעות הדף - אותו המשאב שנמצא בכתובת "/". הפקודה שנ裏ץ: <http://http-demo.appspot.com>

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

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

The screenshot shows a Telnet session window with the following content:

```

Telnet http-demo.appspot.com
GET http-demo.appspot.com/ HTTP/1.1
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Date: Fri, 07 Feb 2014 11:23:57 GMT
Server: Google Frontend
Cache-Control: private
Alternate-Protocol: 80:quic
Transfer-Encoding: chunked

<html>
  <head>
    <title>HTTP Demo</title>
  </head>
  <body>
    <a href='/'>/</a> <200><br>
    <a href='/'>/x</a> <302><br>
    admin [Password: <b>secret</b>]<br>
    <a href='/'>/1</a> <401> [Username: <b>admin</b>] [Password: <b>secret</b>]<br>
    <a href='/'>/2</a> <403><br>
    <a href='/'>/3</a> <500><br>
    <a href='/'>/4</a> <503><br>
    <a href='/'>/something</a> <404><br>
    <a href='/'>/inc?x=4</a><br>
    <a href='/'>/add?x=1&y=2</a><br>
  </body>
</html>

```

Annotations in Hebrew are present in the screenshot:

- הבקשה (Request) points to the GET command.
- שורת התגובה (Response header) points to the status line and headers.
- Header (Header) points to the Content-Type header.
- תוקן התגובה (Response body) points to the HTML content.

### תרגיל 3



כתבו מהו התוכן ואילו שדות Header מתקבלים בתגובה לבקשת GET לנحوות 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 Sockets / שרת-לקוח](#), נבין שני סוגי של 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 היו 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 שימש 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 הגיע לדף קובץ טקסט.

נתעכט לרגע כדי להבין בניוות הכתובות הללו: לכל אתר בראש יש כתובת ייחודית, שמכונה URL - ראש תיבות של Universal Resource Locator (בעברית: "כתובת משאב אוניברסלית"). למעשה, ה"כתובות" שהכנסנו לדף בסיוף הקודם היו URLs. על משמעות המילה "משאב" נרחיב עוד בהמשך. URL בסיסי בניו הבא:



במקרה זהה, השרת שמאחוריו הכתובת `www.site.com` יփש בתיקייה `a` תי'יה בשם `b` ובתוכה יփש קובץ בשם `file_name.txt`. הפניה מתבצעת בסכמה של HTTP<sup>22</sup>. אם אכן קיים קובץ זהה, השרת יגיש אותו בתור תגובה לבקשתו.

כדי להבין טוב יותר את הרעיון, תמשכו כתעת בעצמכם שרת צזה, 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-`"/"` למשא פונה לקובץ `file_name.txt` בתיקייה `/folder_a/folder_b/file_name.txt`. אך היכן נמצאת תיקייה `a`? היא נמצאת בתיקיית השורש, `root` שבסביבה `folder_a`. aktual היכן נמצאת תיקייה `b`? היא נמצאת בתיקיית `root` אותה בתור `root`, של האתר. בפועל, התיקייה הזו היא פשוט תיקייה כלשהי במחשב שהמוכנת הגדרו אותה בתור `root` directory. בחירה נפוצה היא להגדיר את `root` directory כעט, כאשר הליקוח פונה ושולח בקשה מסוג:

GET /folder\_a/folder\_b/file\_name.txt HTTP/1.1

הבקשה תתבצע למעשה לקובץ הבא אצל השרת<sup>23</sup>:

`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 התכנית, ושימוש בדף קלוקח; היזכרו במשמעות של הכתובת `127.0.0.1` אותה הזכרנו בפרק [תכנות 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: [www.cyber.org.il/networks/webroot.zip](http://www.cyber.org.il/networks/webroot.zip). העתיקו את תוכן קובץ `ZIP` אל סדרה כלשהי (כਮון שיש לפתוח את הקובץ) והשתמשו בה בתור `root`

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

<sup>23</sup> זאת בהנחה שהשרת מרים מערכת הפעלה Windows. כאמור,מערכות הפעלה שונות ה-path עשוי להיראות בצורה שונה.

1) כתבו שרת המחברת לתוכנת נסגרת. המטרה היא שהשרת ישלח ללקוח את index.html ויתמוך באפשרויות השונות שיש בעמוד אינטרנט זה.

1) כתבו שרת המחברת לתוכנת נסגרת 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 \r - \n.

- 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" כמספרד בצוין מיקום קובץ, בעוד שב인터넷 גם בלינוק משמשים 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 הגירסה 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: בኒוגד 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, הוסיף ל-Header של התשובה את השדה Content Type בהתאם לסוג הקובץ שהתקבש. תוכלו להעזר בנתונים הבאים:

- קבצים בסויומת txt או html:

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

- קבצים בסויומת jpg:

Content-Type: image/jpeg

- קבצים בסוימת זו:

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

- קבצים בסוימת CSS:

Content-Type: text/css

(10) כתת 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 "זז". כר' למשל, משתמש שיבקש

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

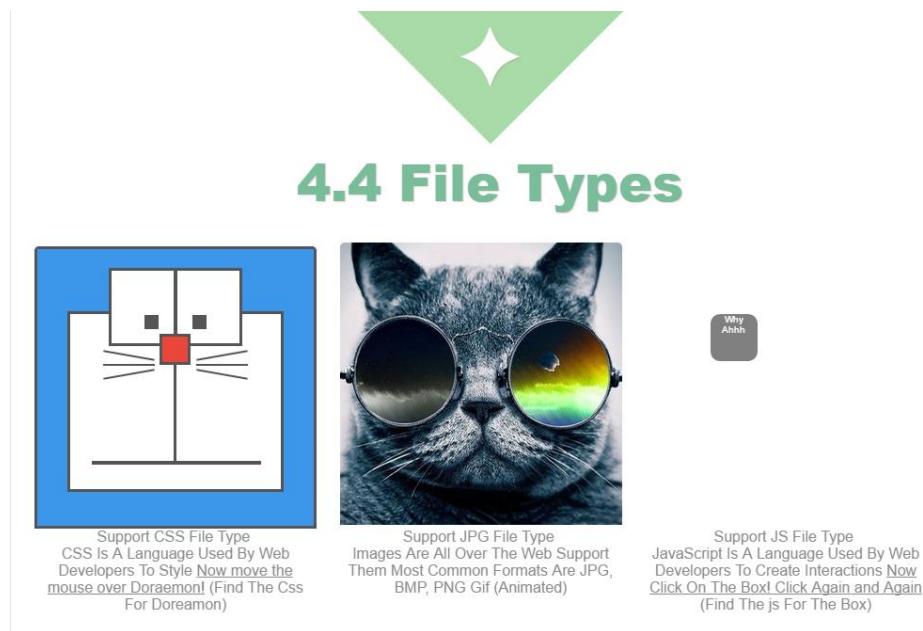
הלקוח יבקש את המאוב page2.html, ועבורי יקבל תשובה 200 OK.

.3. Internal Server Error – במקרה שהשרת קיבל בקשה שלא 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:



אתר בדיקת תרגיל שרת HTTP – קרדיט תומר טלגט

## מדריך לכתיבה ובדיקה של תרגיל כתיבת שרת HTTP

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

[http://www.cyber.org.il/networks/HTTP\\_server\\_shell.py](http://www.cyber.org.il/networks/HTTP_server_shell.py)

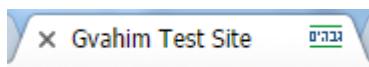
המקומות שעליים להשלים בעצמכם נמצאים תחת העלה "DO". שימוש לב שכדי שהתוכנית תעבור תצרכו להוסיף לה קבועים ופונקציות נוספות, השלד אמרור רק לשיע לכם להתמקד.

### טיפים לכתיבה

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

### איקון

מרבית אתרים האינטרנט כוללים איקון מעוצב שמופיע בלשונית של הדף. גם הדף שלכם יכולים יבקש את האיקון מהשרת. כדי למצוא הינק נמצא 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 לתוך תמונה כלשהו או להמציא איקון משלכם. לאחר שהסיטם לשבץ איקון, לחצו על כפתור icon download ושמרו אותו במקום המתאים.

### כלי דיבוג – breakpoints

שימוש breakpoints לוטבת דיבוג קוד השרת שלכם הוא הכרחי. צרו breakpoints במקומות שבו הקוד עדין מבוצע באופן תקין ועיקבו אחרי הערךם שמקבלים משתנים וփונקציות מוחזירות כדי לבדוק בעיות בקוד שלכם. באמצעות בדיקת ערכי משתנים תוכלו לבדוק, לדוגמה, מה>brequest> ששלח הלוקוט.

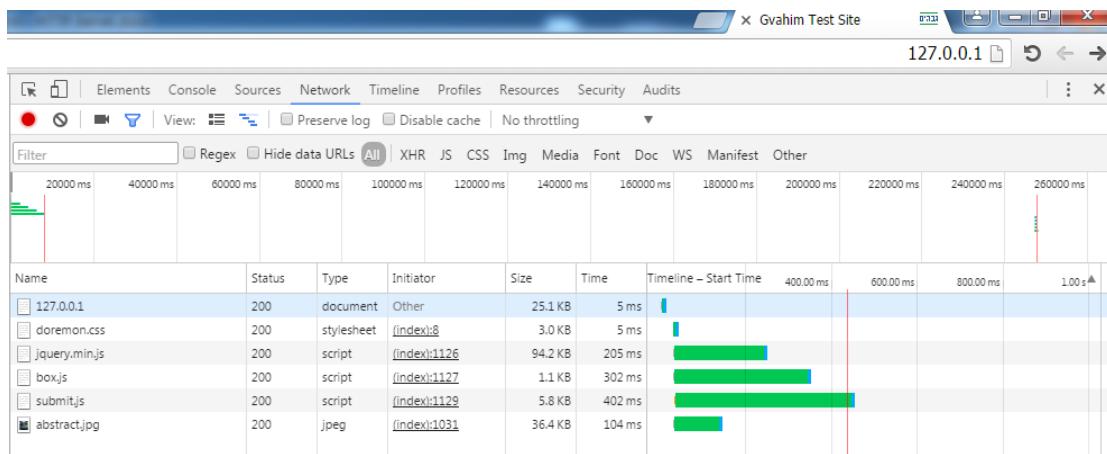
```

203     def main():
204         # Open a socket and loop forever while waiting for clients
205         server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
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)
215
216     if __name__ == "__main__":
217         # Call the main handler function
218         main()
219
220

```

### כל דיבוג - דףן כרום

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

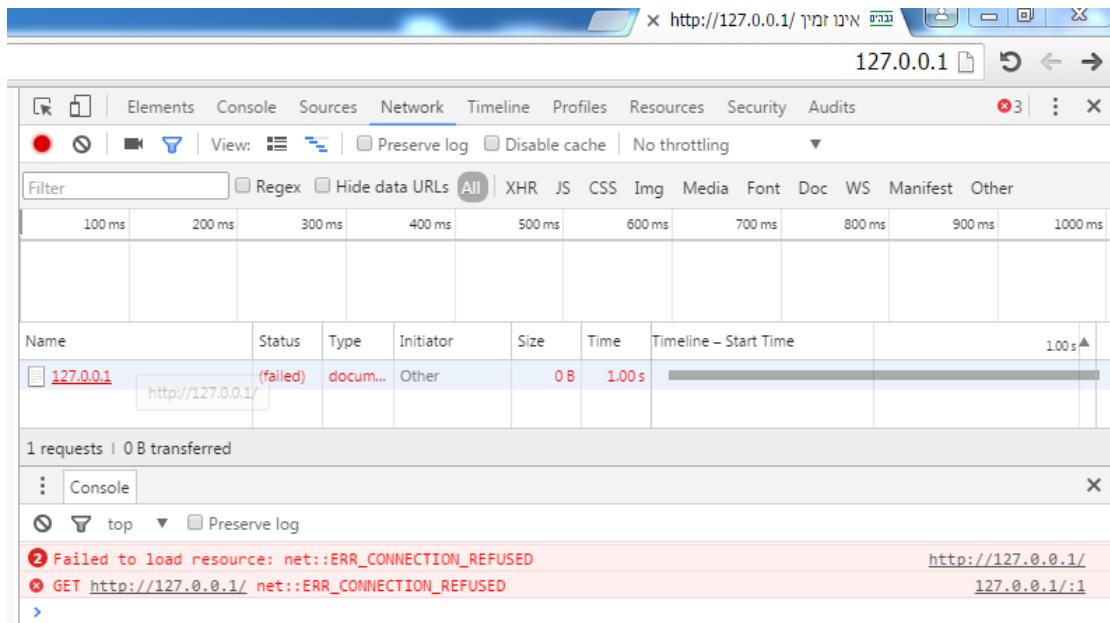


לחצו על הנקודה האדומה כדי להתחליק למסך חדש.

אם תumedו על שם של קובץ כלשהו תוכלם לקבל פרטיים נוספים בטאים החדשניים שמופיעים לצדויו - Headers, Preview, Response, Timing  
המידע המעניין ביותר לטובת דיבוג השירות שלכם מופיע ב-headers, שם ניתן לראות את הבקשות שנשלחו לשרת ואת התשובות של השירות.

Name	Headers	Preview	Response	Timing
127.0.0.1	<b>General</b> Request URL: http://127.0.0.1/ Request Method: GET Status Code: 200 OK Remote Address: 127.0.0.1:80			
doremn.css	<b>Response Headers</b> view parsed HTTP/1.1 200 OK Content-Length: 25587 Content-Type: text/html; charset=utf-8			
jquery.min.js	<b>Request Headers</b> view parsed GET / HTTP/1.1 Host: 127.0.0.1 Connection: keep-alive Cache-Control: max-age=0 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) 537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Accept-Encoding: gzip, deflate, sdch Accept-Language: he-IL,he;q=0.8,en-US;q=0.6,en;q=0.4			
box.js				
submit.js				
abstract.jpg				

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 הלוקה:



### כלי דיבוג - Wireshark

כלי נפלא זה, איתמו עשינו היכרות בעבר, יכול לספק לכם את כל המידע שעובר בין השרת ללקוח שלכם - בתנאי שתרכזו את השרת ואת הלוקוח על שני מחשבים נפרדים, שמחוברים ע"י ראוטר או switch כפוי שווודי יש לכם בבית.

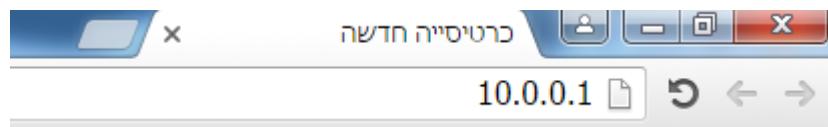
בשלב ראשון, וודאו שהשרת שלכם מאמין לכתובת IP 0.0.0.0 כתובות 0.0.0.0 (ולא 127.0.0.1). כתובות 0.0.0.0 אומrette למערכת הפעלה שלכם לשלוח לשרת שבניתם לא רק פקודות שмагיעות מתוך המחשב עצמו (127.0.0.1) אלא גם פקודות שmagיעות מחשבים אחרים - בתנאי שהם פונים לפורט הנכון. לאחר מכן בידקו באמצעות ipconfig מה כתובת ה-IP של השרת שלכם (שםו לב, כתובת IP ברשות הפנימית שלהם, 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
  Link-local IPv6 Address . . . . . : fe80::48ac:d008:caaf:7415%12
  IPv4 Address . . . . . : 10.0.0.1
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 10.0.0.138
```

לפני שנתקדם לשלב הבא, הפעילו wireshark.

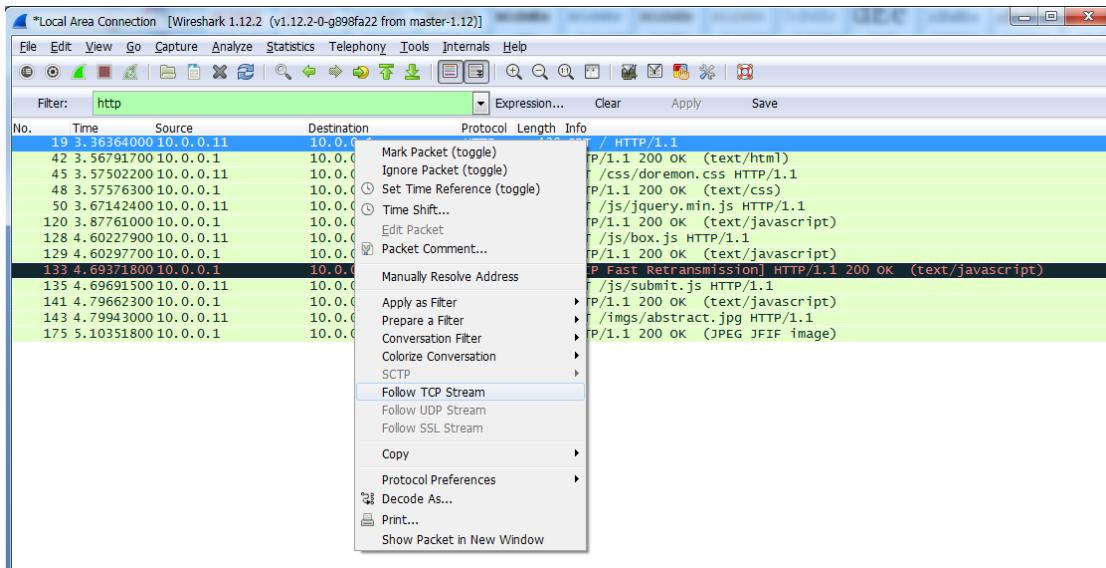
כעת עלייכם לקבוע בהתאם את כתובת ה-IP שהלך פונה אליה:



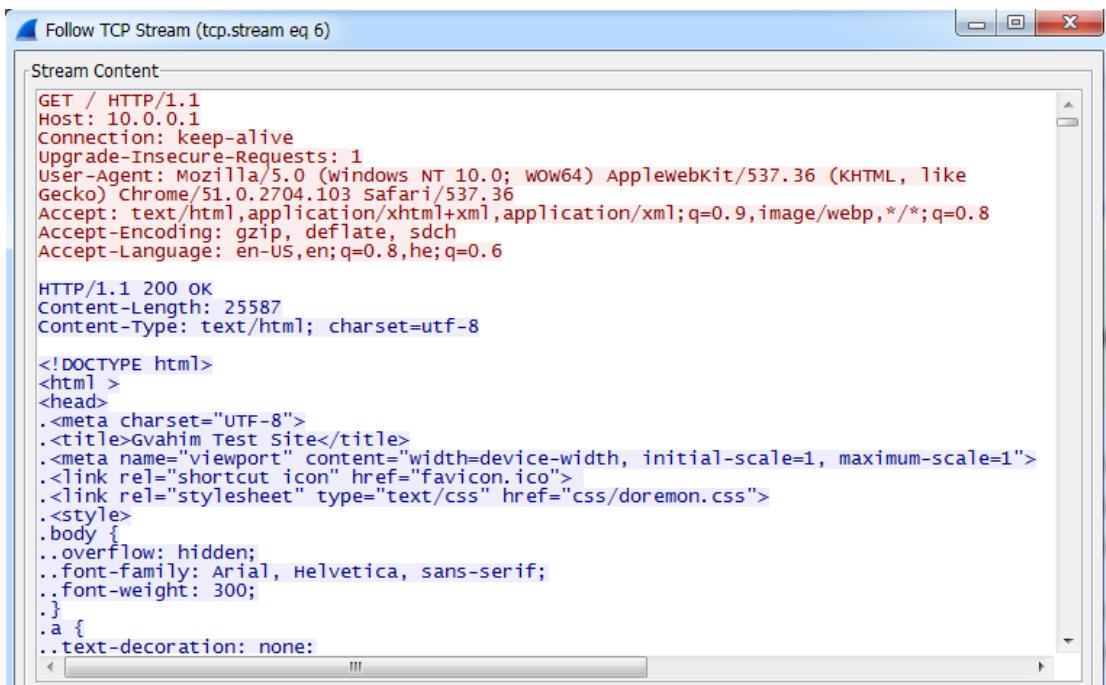
תוכלו למצאו בטור wireshark את התעבורה בין הדפןblkוות לברשות ה-HTTP שלכם. נfiltr את התעבורה לפי http://

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/css)
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/javascript)
128	4.60227900	10.0.0.11	10.0.0.1	HTTP	365	GET /js/box.js HTTP/1.1
129	4.60297700	10.0.0.1	10.0.0.11	HTTP	1212	HTTP/1.1 200 OK (text/javascript)
133	4.69371800	10.0.0.1	10.0.0.11	HTTP	1212	[TCP Fast Retransmission] HTTP/1.1 200 OK (text/javascript)
135	4.69691500	10.0.0.11	10.0.0.1	HTTP	368	GET /js/submit.js HTTP/1.1
141	4.79662300	10.0.0.1	10.0.0.11	HTTP	139	HTTP/1.1 200 OK (text/javascript)
143	4.79943000	10.0.0.11	10.0.0.1	HTTP	398	GET /imgs/abstract.jpg HTTP/1.1
175	5.10351800	10.0.0.1	10.0.0.11	HTTP	815	HTTP/1.1 200 OK (JPEG JFIF image)

nocll לראות את כל הבקשות של הלקוח ואות תשובה הרשות. בצלום מסך זה ניתן לראות שחלק מהמידע (השורה הצבועה על ידי Wireshark בשחור) שודר פעמיים מהשרת ללוקוח - Retransmission - שמייד על כך שהשרת לא קיבל אישור על המידע שלוח ללקוח (האישור הוא ברמת שכבת התעבורה, פרוטוקול TCP, עלייו נלמד בהמשך). אם נרצה לעקוב אחרי כל הפקודות שעברו בין הרשות והלקוח שלנו, כולל הודעות הקמת וסיום קשר ושליחה מחודשת של פקודות, nocll להקליק על שורה כלשהי ולבחר TCP stream follow.



ויצגו בפנינו מסכים שמראים את כל המידע בעבר, הן בשכבת האפליקציה (השרת והלקוח במצבים שונים) והן בשכבת התעבורה. מכך ניתן להשתמש בברשות כדי לאתר בעיות שאולי גרמו לכך שהשרת שלנו לא עובד כפי שתיכנו.



No.	Time	Source	Destination	Protocol	Length	Info	Expression...	Clear	Apply	Save
15	3.17640900	10.0.0.11	10.0.0.1	TCP	62	50279-80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1				
17	3.36110200	10.0.0.1	10.0.0.11	TCP	62	80-50279 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1				
18	3.36343700	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=1 Ack=1 Win=64240 Len=0				
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 reassembled PDU]				
21	3.36426500	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
22	3.37566600	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=377 Ack=1461 Win=64240 Len=0				
23	3.37570300	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
24	3.37571000	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
25	3.46497200	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=377 Ack=5841 Win=64240 Len=0				
26	3.46501300	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
27	3.46502100	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
28	3.46502400	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
29	3.46502700	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
30	3.46503000	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
31	3.46503600	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
32	3.46768100	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=377 Ack=7301 Win=64240 Len=0				
33	3.46771500	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
34	3.46772200	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
35	3.47272600	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=377 Ack=10221 Win=64240 Len=0				
36	3.47276100	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
37	3.47277000	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
38	3.47277300	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
39	3.47277500	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
40	3.56783700	10.0.0.11	10.0.0.1	TCP	60	50279-80 [ACK] Seq=377 Ack=14601 Win=64240 Len=0				
41	3.56790800	10.0.0.1	10.0.0.11	TCP	1514	[TCP segment of a reassembled PDU]				
42	3.56791700	10.0.0.1	10.0.0.11	HTTP	903	HTTP/1.1 200 OK (text/html)				

זהו, בהצלחה! ...

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



במהלך עשורים האחרונים האחرونנות, "אפליקציות web" (כינוי לאפליקציות ושרותים שעושים שימוש בפרוטוקולי אינטרנט) צברו יותר ויותר פופולריות, ובמקביל התפתחו באופן מואץ היכולות, רמת התוכום 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 שבו דברים קבועים - סרגל כלים, לינקים - והרבה דברים שנקבעים מחדש בכל כניסה - למשל הודעות על פעילות של החברים שלכם ופרסומות. למעשה מאחוריו הקളעים 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.

עת נמש שרת שמגיב בתשובות תכנותיות לבקשת GET. כמובן, השירות שלנו יבנה תשובה באופן דינامي בהתאם לבקשת שהגעה מהלקוח. בינהים, נמש את הצד השירות, ולצורך צד הליקוח נמשך להשתמש בדף.

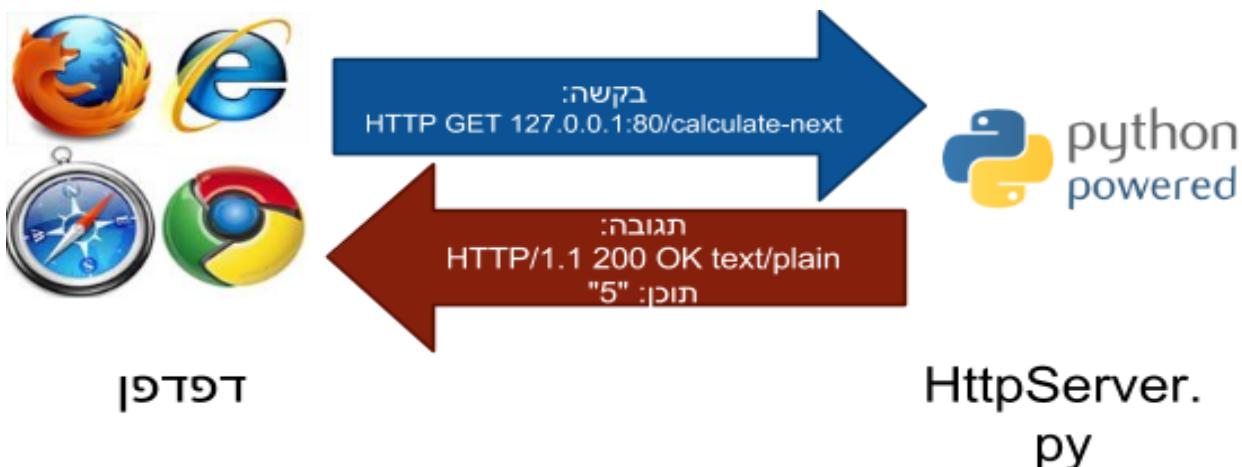
### תרגיל 4.5



עדכנו את קוד השירות שתכתבם, ככה שפנוייה לכתובת "calculate-next/"/ לא תחפש קובץ בשם זהה, אלא תענה על השאלה "מה המספר שמייע אחריו 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 רלוונטיים בתשובותיכם, ולא "סתם" Header'ים שאתם רואים בהסנפות. כדי שההתגובה תהיה תקינה, צינו קוד תשובה (כגון OK 200). עלייכם לכלול גם Content-Length (נדרש להיות גודל התשובה, בביטחון, ללא גודל ה-Header'ים של HTTP). צינו גם את ה-Content-Type (במקרה שתיחסרו קובץ – סוג הקובץ שאתם מחזירים, שאפשר להסיק לפי סיומת הקובץ). תוכלו להוסיף גם Header'ים נוספים, אך עשו זאת בצורה שתואמת את הבקשה וההתגובה הספציפיות.

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



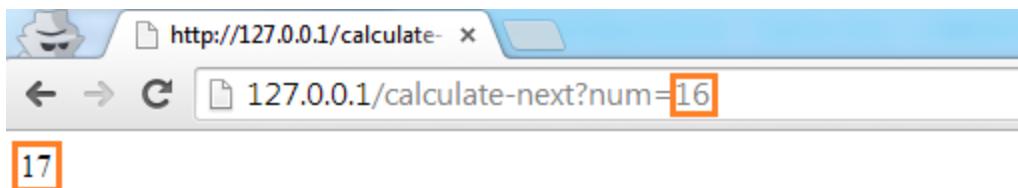
בטח כבר עלייתם בעצמכם על כך שהשרת במקורה הזה הוא מודד לא שימושי, כי הוא מניח שהמשתמש תמיד מעוניין לדעת מה המספר שבא אחריו 4. בתכניות שכתבנו בפייטון, בדרך כלל עשינו שימוש בקלט מהמשתמש כדי לעשות איתה חישוב כלשהו במהלך התוכנית. השרת שלנו היה הופך להרבה יותר שימושי, אם האפליקציה הייתה יכולה לבקש את המספר שבא אחרי מספר כלשהו; למעשה, אפשר לחשב על השרת שלנו כמו על כל תוכנית בפייטון שכתבנו עד היום, למעט העובדה שלא ניתן להעביר לתוכנית קלט, ולכן היא מחזירה תמיד את אותו הפלט. אם נוכל להעביר לה בכל בקשה קלט שונה, נקבל את הפלט המתאים לכל קלט.

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

#### תרגיל 4.6



עדכנו את קוד השרת שכתבתם, כך שפנוייה לכתובת "/calculate-next?num=4" תתרפרש כפניהם לכתובת calculate-next עם השמה של הערך '4' במשנה sum, 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 מעניינת יותר, ולאחר מכן נבחן להבין את קטע הקוד של השרת שבו השתמשנו:

הכניסו את השורה הבאה בדף: <http://www.google.com/search>

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.

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

AIR נראית הבקשה? טוב ששאלתם, היא נראית בדיקת CRC.<sup>24</sup>

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.

<sup>24</sup> ברוב המקרים ה-Host (בדוגמה זו "www.google.com") לא יכול בשורת ה-GET, אלא ב-HTTP Header נפרד. עם זאת, על מנת שהדוגמאות תהיינה ברורות, אמן נציג אותו באופן מפורש.

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.

Google search results for 'application':

- Application software - Wikipedia, the free encyclopedia**  
en.wikipedia.org/wiki/Application\_software ▾  
Application software is all the computer software that causes a computer to perform useful tasks beyond the running of the computer itself. A specific instance of ...
- Application - Wikipedia, the free encyclopedia**  
en.wikipedia.org/wiki/Application ▾  
Application may refer to: A verbal or written request: Application for employment, a form or collection of forms that an individual seeking employment must fill out ...  
Application software - Application for employment - Function application
- Web application - Wikipedia, the free encyclopedia**  
en.wikipedia.org/wiki/Web\_application ▾  
A web application or web app is any application software that runs in a web browser or is created in a browser-supported programming language such as the

## תרגיל 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](http://www.google.com/search)), שם של פרט, ורשימת ערכיהם.  
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 למחשבה:** שימושו לב להבדל המרכז בין ה-URLים הללו לבין URLים שעלייהם התבוססTEM בתרגיל שרת `http` שמייחדים בסעיף הקודם - כיום "געלו" 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 תשובה על סמך פורטרים וקוד תוכנה שמרכזיב את התשובה.

כעת נחזור לאחד החיפושים שעשינו ב-Twitter בתרגיל האחרון; שימושו לב שהתוצאות שקיבלנו מכילות ציוצים (tweets) שהכילו את המילה `obama`. מה אם הייתי רוצה לחפש את הפרופיל של אובמה? (שימוש לב שיש בצד שמאל כפתור שעושה זאת זה). האפן שבו ה"אפליקציה" (במקרה שלנו, האתר) של Twitter עושה זאת זה, הוא שהוא שולח פורטמר נוסף לשרת.

פורטמר אחד יכול את מילת החיפוש (`obama`), והפורטמר 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 הפורטמר `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 כתובות, 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 'z' עבור תחבורה ציבורית (או 'w' אם אתם מתכוונים לרכבת (ברג').

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



#### תרגיל 4.9

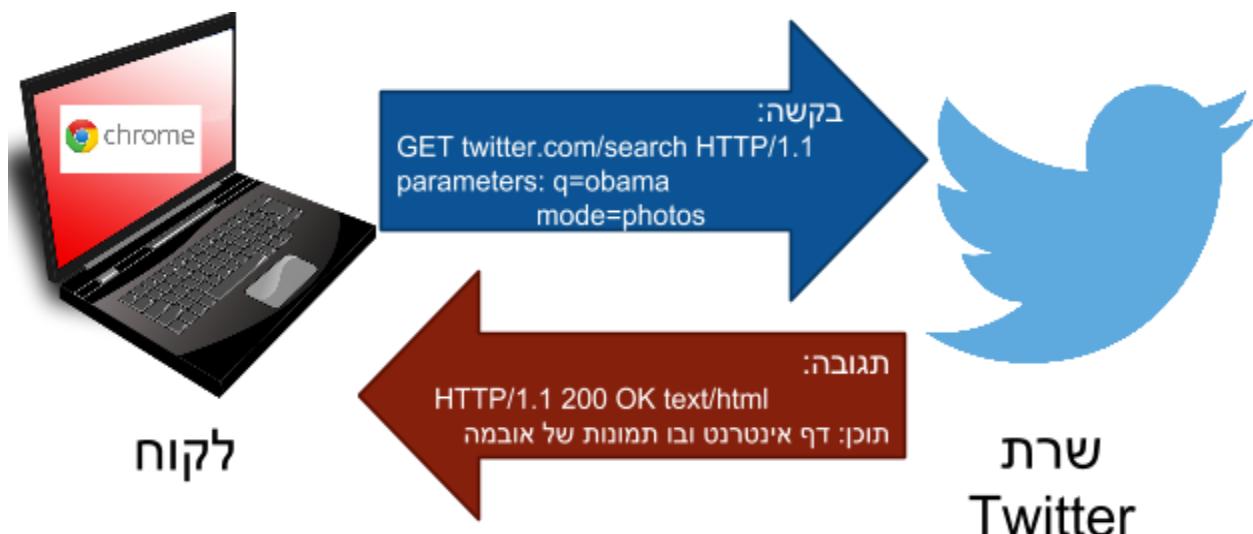
כתבו שרת HTTP פשוט, שמחשב את השטח של משולש, על סמך שני פרמטרים: גובה המשולש והרוחב שלו. למשל, עבור שורת הבקשה הבאה: `http://127.0.0.1:80/calculate-area?height=3&width=4`, יחזיר "6.0".  
בדקו אותו גם עבור קלטים נוספים.

### פרוטוקול HTTP - בקשות POST ותכנות צד לקוח בפייתון (הרחבת)

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

נתעכט 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 התיקשורת, 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 (ספקטיפית – התמונה. בטלפון ממוצע תמונה יכולה להיות בגודל של יותר מ-1MB), והיא מעוניינת להעביר את כולה לשרת.

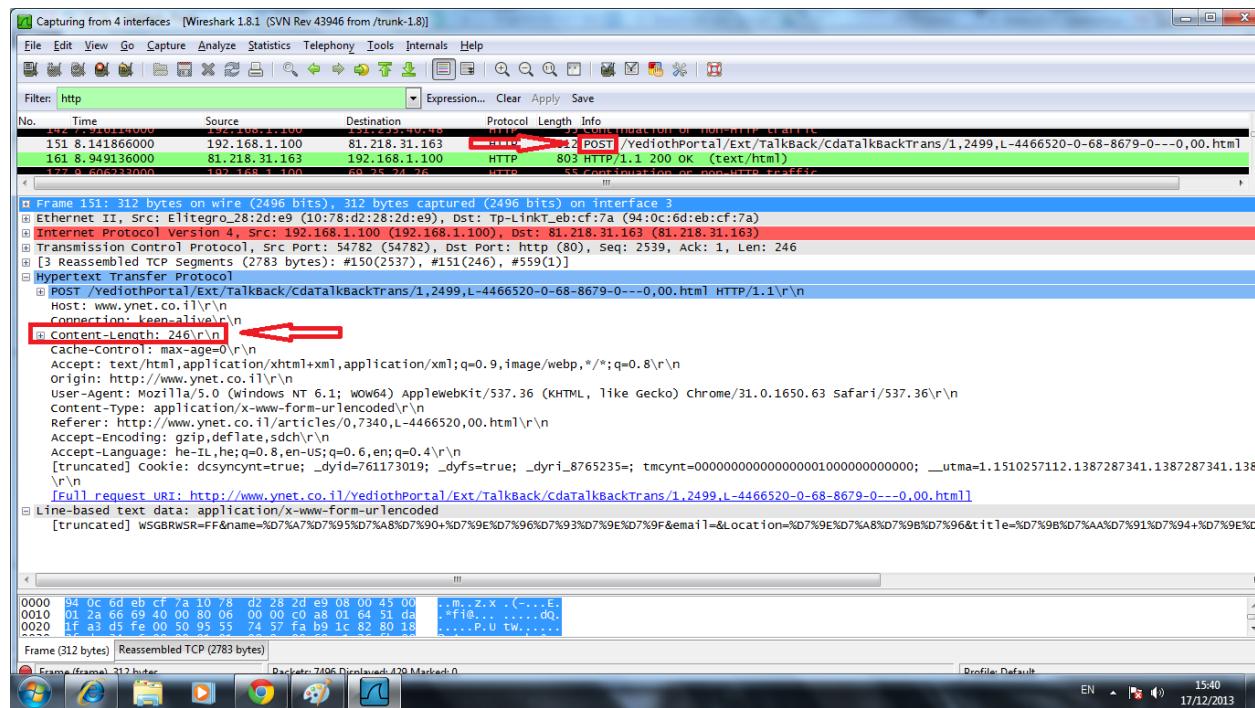


**אם ניתן להשתמש בפרמטרים של בקשה GET כדי להעביר את כל התמונה?**

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 בקשה GET.

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

כדי להתנסות בשימוש בבקשת POST, ניכנס לאתר [www.ynet.co.il](http://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 שלחנו, 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...](http://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...)

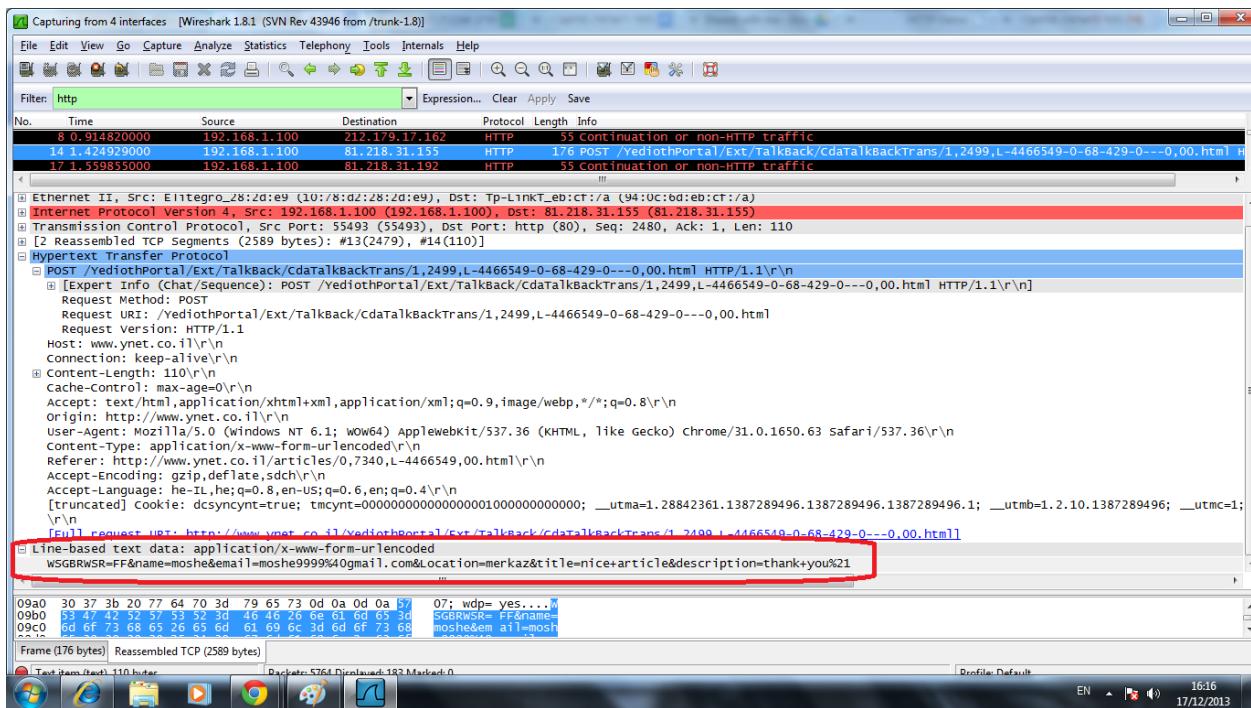
זהו אינה אופציה מציאותית - האורך המקסימלי של URL הוא 2,000 תווים, בעוד שרק התגובה עצמה עשויה להיות יותר ארוכה מכ...  
 לכן, בנוסף לפרמטרים שעוברים חלק מהכתובת (URL), בבקשת POST ניתן לזרף תוכן, שם יועבר תוכן התגובה.

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 יישלח לשרת של האתר האינטרנט שלהם באופן דומה.

אם תרצו "לראות" איך עובר התוכן עצמו בבקשת POST - לטפסים יש דרך דיאטסטנדרטית להעביר את הנתונים האלה. הדרך הזאת קצט מזיכירה 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 שמילאתם 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 הלוקה בתקשורת - כולם את האפליקציה.

כתבו תוכנית חדשה שתתפרק בתור לקוח 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 (ובצד!) שאין שם טום בשרת שרק ניתן לשמור אליו תמונות, אם לא ניתן לבקש אותן בחזרה.



### תרגיל 4.11

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

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

כעת, פתחו את הדף (בו בדרך כלל השתמשנו עד כה בתור צד ליקוח ליצירת בקשות GET), והכניסו 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 במידע שנמצא בשרת, כפי שראינו בשרת אחיזור התמונות שכתבנו בסעיף הקודם. אני יכול לבקש תמונה בשם "21-december" ולקבל הודעה שגיאיה שאומරת שהמשאב אינו נמצא (404), וביום שלאחר מכן לבצע שוב את אותה הבקשה, אלא שהפעם קיבל תמונה אחרת. מה ההסבר לכך? נראה שבינתיים מישחו ביצוע פקודת POST והעלה תמונה בשם זהה.

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

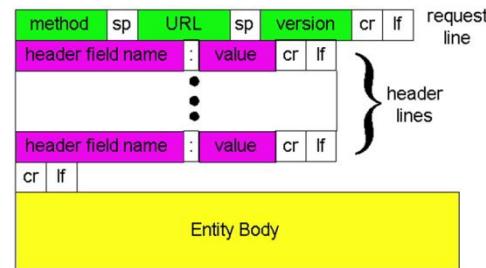
בגלל שרת HTTP המ כל כך נפוצים ופולריים, רוב האתרי האינטרנט לא ממשיכים את פרוטוקול HTTP ביצם, אלא עושים שימוש במימושים נפוצים של שירותי אלה (למשל, בפייטון ניתן להשתמש במימוש [SimpleHTTPServer](#)).  
נושאים אלו לא מכוסים בפרק זה לעת עתה, אך אנחנו מעודדים קוראים סקרנים לחקור וללמוד בעצמם, ב[חלק](#)  
[צעדים 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 בקשה 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\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
```



### (מטמון) Cache

דרך אוחת "להאיץ" את חווית השימוש באינטרנט היא באמצעות ההבנה שיש לשאבם שאוטם 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 [chetuchy](#) כדי להתעדכן בחדשות, 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 (רוחב הפס וכמות המידע שעוברת על גבי), וגם של זמן (בהתנה לשאב שיגע).

הweeney של מנגנון ה-**Cache (מטמון)** הוא לשומר משאבים אלה על המחשב של הלקוח, וכל עוד הם לא משתנים, לטען אותם מהdisk המקומי ולא על גבי הרשת.

המנגנון שמאפשר לבצע זאת בפרוטוקול HTTP נקרא Conditional-Get (בקשת GET מותנית).משמעות השם:  
בקשת-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*. הדף-פן יבקש את הלוגו של *net* עז, אבל רק אם הוא השתנה מאז הפעם האחרון שהדף הציג את האתר ושמר עצמו אצלו. ברגע המוחלט של הפעמים, הלוגו לא השתנה, והשרת פשוט יורה לדף-פן: "תשמש בעותק שקיים אצלך". וכך נחסכה העברת הלוגו של *net* עז שוב ושוב ברשות.

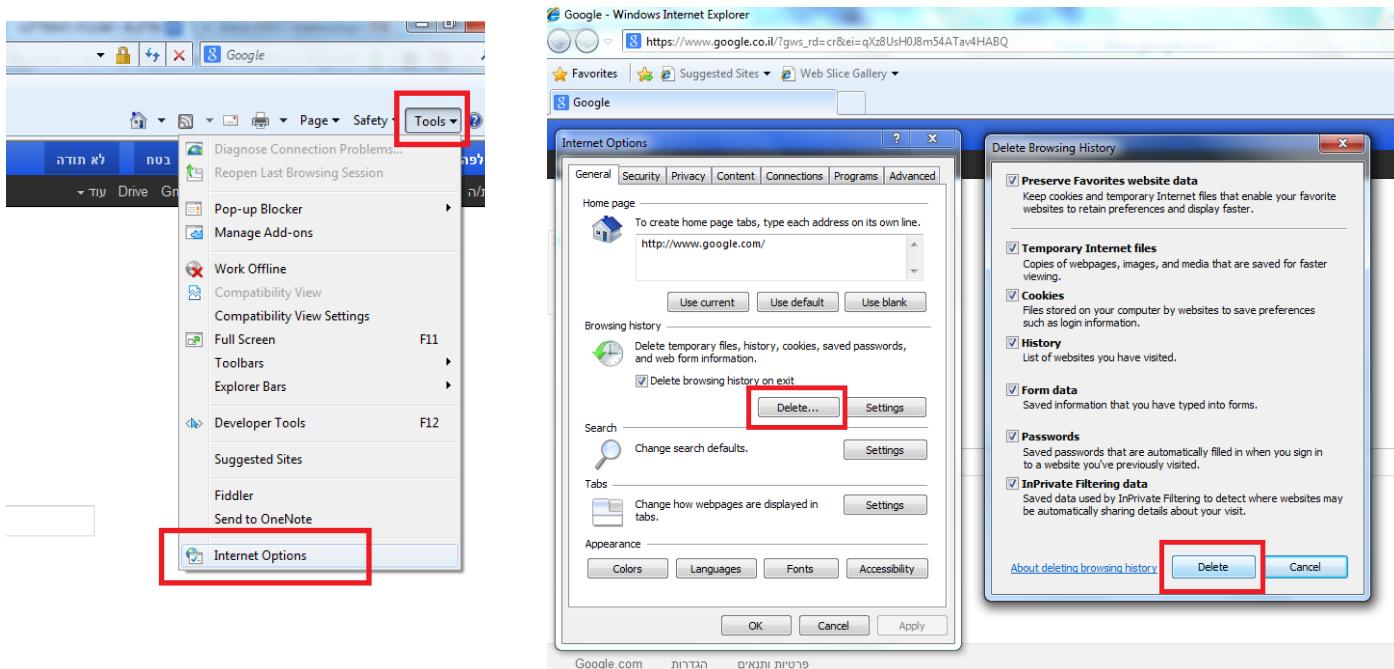
ונסה להבין את המנגנון הזה באמצעות תרגיל מעשי:

## תרגיל 4.12 מודרך - הבנת Caching באמצעות Wireshark

שיםו לב שרוב הדף-פנים כולם מוצעים ב"מטמון" דפים שהם הורידו ולא מורידים את הדף מחדש בכל פעם, אלא רק כשהם חשובים שיש בה צורך.

1. היכנסו ל-Internet Explorer ורוקנו את המתמן:

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

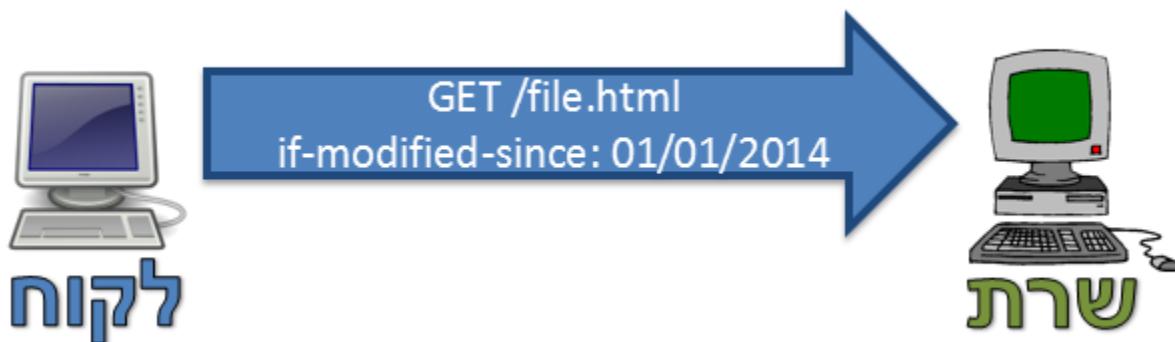


- .2. סיגרו את הדף. פיתחו אותו מחדש.
  - .3. מה, לא פתחתxs הנספה עדין? קדימה!
  - .4. היכנסו לעמוד הבא: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>
  - .5. חכו עד שהדף יסימן לטען את העמוד.
  - .6. כעת היכנסו שוב לעמוד זהה (או לחצו על **(Refresh / F5)**).
  - .7. הפסיקו את ההספנה.
  - .8. הסתכלו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



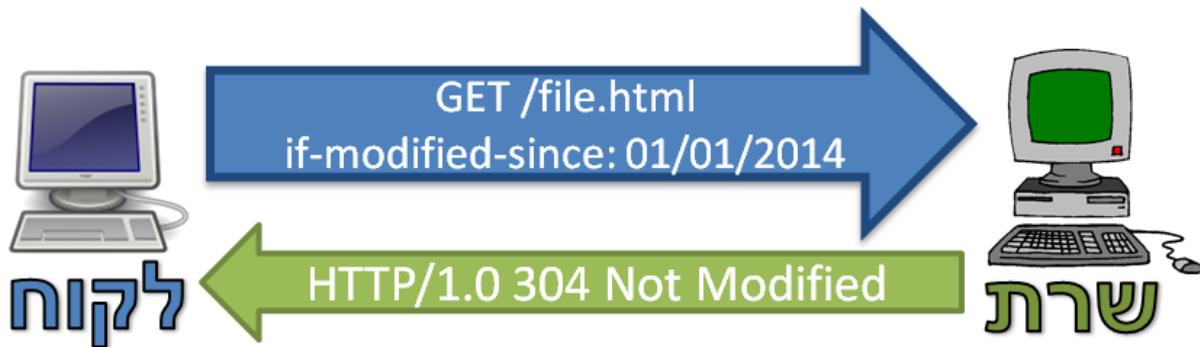
### כיצד עובד מנגנון **Cache** וביקשת-**Conditional-GET** ?Conditional-GET

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



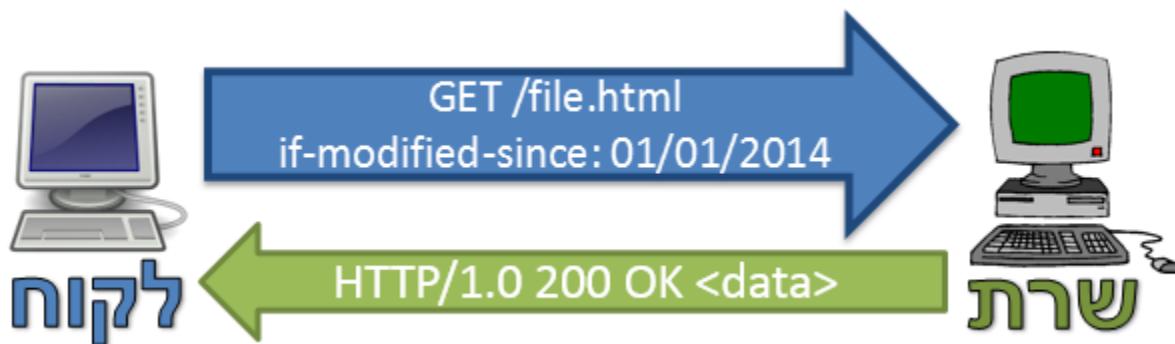
הלוקו מבקש את המשאב **file.html**. הלוקו מבקש שהשרת ישלח לו את הקובץ זה, אך ורק אם הוא השתנה מאז **01.01.2014**. הסיבה היא, שהлокו שמר בתאריך זה את הקובץ **file.html** לשלו. לכן, אם הקובץ לא

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



בשלב זה, הלוקוח יכול לטעון את המשאב מtower ה-*Cache*, ולא לקבל אותו מהשירות.

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



במקרה זה, הלוקוח מבין שהעוטק השמור אצליו אינו עדכני. הוא טוען את הקובץ מהמערת, ומחליף את הקובץ המקורי ב-*Cache* החדש.

## Cookies ("עוגיות")

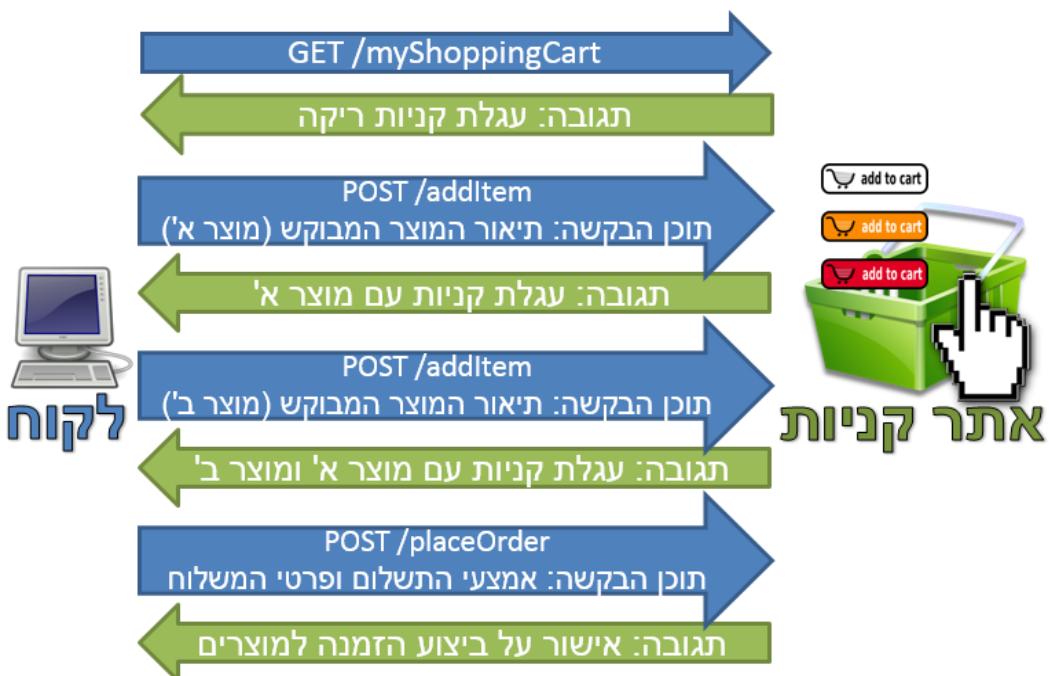
במהותו, HTTP הוא פרוטוקול "חסר-מצב" (באנגלית: stateless) או "חסר-זיכרון". המשמעות היא שכל בקשה מטופלת בפני עצמה, ללא קשר לבקשת הקודמת - בקשות מסוג GET מאפשרות גישה למשאבי אינטרנט (תמונות, מסמכים וכו'), ובקשות POST מאפשרות להעלות מידע מפליקטיבית לקוח אל השירות (טפסים, תמונות, מיילים וכו').

עם זאת, בהרבה אפליקציות ואתרי 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 השרת בשלבים, ככלומר, 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 וכתובת המשלוח.

תקשורת מהצורה זו מחייבת את השרת "לזכור" מידע בין טיפול בבקשות שונות. חשוב גם לשים לב שהשרת צריך "לזכור" בנפרד את עגלת הקניות לכל session של כל משתמש; אסור שיבלב בין עגלות של sessions שונים.



### **תרגיל לחשיבה עצמית: כיצד ניתן לעשות זאת? שימושת בקשה HTTP, כיצד "ידע" השרת לאיזה session היא שייכת?**

הפתרון לכך הוא שימוש במנגנון `cookies` ("עוגייה") היא מחורצת שימושת לשרת וללקוח; השרת קובע את המחרוזת הזו בתחילת session, ולכל אורך ה-`session` הלוקח יצרף את המחרוזת הזו לכל בקשה שלו (בשני המקרים עושים שימוש ב-`HTTP Header fields` כדי להעביר את המחרוזת הזו).

את ה-`cookie` ישמר הלוקח בדיסק המקומי, וכן היא תישמר בסיס הנתונים של השרת. ל-`cookie` נקבע אורך חיים, כך שלאחר זמן מסויים פג התוקף שלה, והשרת יקבע מחדש session חדש.

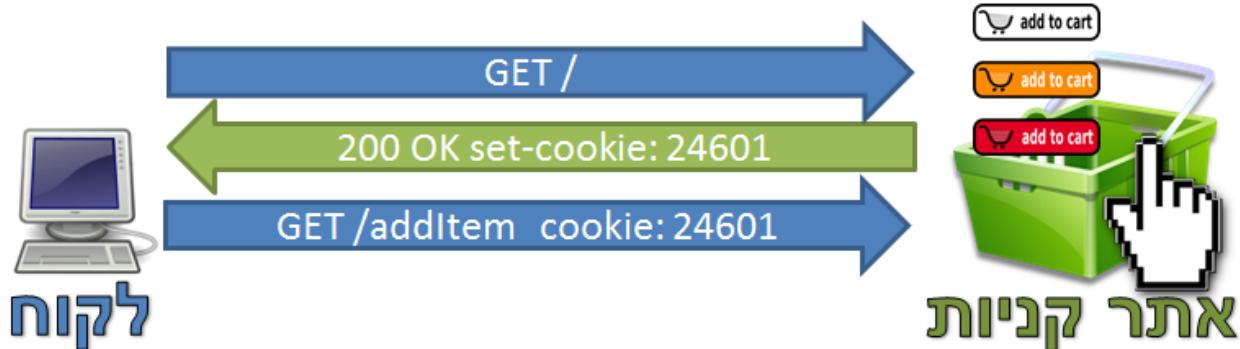
על מנת להבין את האופן בו עובד מנגנון `cookies`, נשתמש בדוגמה. נאמר שיש לנו לוקוח בשם 321client123 שניגש אל שירות קניות באינטרנט:



כעת, האתר מעוניין לשמור מידע של הלוקוח עצמו. לכן, הוא מייצר עבורו מידע. נאמר שהשרת בחר בمخזהה 24601. כעת, השירות יאמר לлокוח להשתמש מעתה בمخזהה 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 וlothות שמדובר בלקוח 24601



## - אוטנטיקציה (אימות וזיהוי)

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** - לדעת מי הם המשתמשים.
2. **אימות** - הרי לא מספיק שהמשתמש "يgid" מהי זההות שלו; כל אחד יכול לגלוש ל-Twitter ולטעון שהוא אשטן קוצ'ר. זה עדין לא יספיק כדי לקבל גישה לحسابו Twitter של אשטן קוצ'ר ולפרסם הודעות בשמו.

על מנת לקבל גישה לحسابו של זהה מסוימת, המשתמש צריך להוכיח שהוא אכן בעל זההות. בדרך הנפוצה ביותר לבצע את ההוכחה זו היא באמצעות סיסמה, שכוראה רק בעל זההות האמיתית אמרור לדעת, ושתמנעו משתמשים להתחזות לזהות שאין לה.

בפרוטוקול HTTP מוגדר מנגנון אוטנטיקציה, שמשמש לזיהוי ואימות של משתמשים. הוא די "חלש" בהשוואה למנגנונים נפוצים אחרים (כמו SSL, שלא נלמד במסגרת הפרק הזה), 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 מה 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 יתפרסמו דרכי פשטוטות לפועל לאחד שירותי הדואר האלקטרוני הפופולרי, או לאחת הרשותות החברתיות הגדולות, אפשר רק לדמיין את הבלהה שתיזכר<sup>25</sup>.

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 זההות שלו, יבוצע בצורה מוצפנת שמנעת מגורם שלישי "להאזין" לתקשורת זו ו"לගונב" את הסיסמה של המשתמש.

---

<sup>25</sup> מספיק להזכיר בבהלה שהייתה בשנת 2013, כשהתפרסמו 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, eccetera) 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 שהחפינה של הסיסמה נעשית בצורה מתחכמת יותר וקשה לפריצה, נאמר שמנגן האוטנטיקציה "חזק" יותר. לקריאה נוספת על מגנוני אוטנטיקציה, הנכם מוזמנים לפנות [לפרק "צעדים להמשך" של פרק זה](#).



**שים לב:** חשוב להבין שבטחה באינטרנט מהו נושא גדול 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, מנגן האוטנטיקציה מוצג כאן רק כדי להמחיש "על קצה המזלג" אוסף של בטחה בשכבה האפליקציה. הבנה עמוקה של נושאים בטחה והחפינה אינה הכרחית כדי למדוד את ה프וטוקול HTTP, וקוראים חלק האוטנטיקציה לא ברור להם, לא יפגעו אם ידלו לחלק הבא.

כאמור, מנגן האוטנטיקציה 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, אותו ניתן לפענוח בקלות!
8. היכנסו לאתר הבא: <http://opinionatedgeek.com/dotnet/tools/Base64Decode>, וביצעו decode למחוזת. האם הצליחם לפענוח את שם המשתמש והסיסמה?
9. האם יש בעיה באבטחה במודול ה-HTTP Authentication ?Basic Authentication. אם כן – הסבירו מדוע. אם לא – הסבירו מדוע.

אל דאגה! ישן 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 להמשך" של פרק זה](#).

## פרוטוקול DNS - הסבר כללי

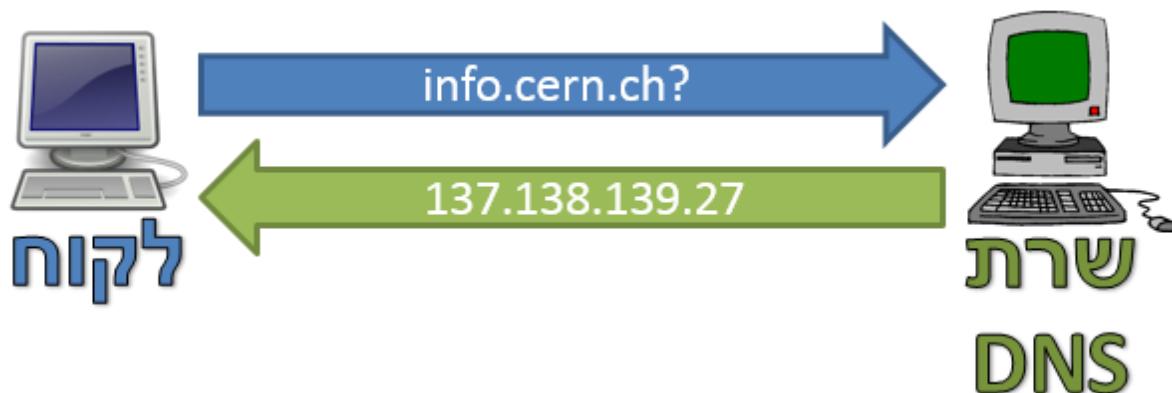
DNS (ראשי תיבות של Domain Name System) הוא פרוטוקול נפוץ נוסף בשכבה האפליקציה. תפקידו העיקרי הינו לתרגם שמות דומיינים (כגון www.google.com, www.facebook.com או http://demo.appspot.com) לבין כתובות ה-IP הרלוונטיות. לבני אדם נהוג יותר לזכור שמות טקסטואליים כגון "www.google.com" מאשר כתובות IP כגון "173.194.39.19".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

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.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 שהדומין בכתובת זהו הינו info.cern.ch, אך אם נזכיר בפרק [טכנולוגיות-ב-Sockets/כתובות-של-socket](#), נראה שמדובר בכתובת IP. 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, הוא 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 ניגש אל האתר המבוקש (info.cern.ch), על מערכת הפעלה למצוא את כתובת ה-IP הרלוונטית. שם כך, המחשב שלנו משאל שרת 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 האחראי על כל הדומיינים ותת-הdomינים (subdomains) של google.com בגן: www.google.com ו-mail.google.com, לא צריך להכיר את www.facebook.com, www.wikipedia.org ועודאי שלא את google.com.he.wikipedia.org. שירות זה מכיר רק את הדומיינים ותת הדומיינים שתחת com.

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 בשאלת DNS



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

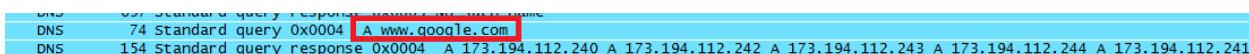
Non-authoritative answer:
Name: www.google.com
Addresses: 2a00:1450:4017:801::1010
          173.194.112.240
          173.194.112.242
          173.194.112.243
          173.194.112.244
          173.194.112.241

C:\Users\USER>

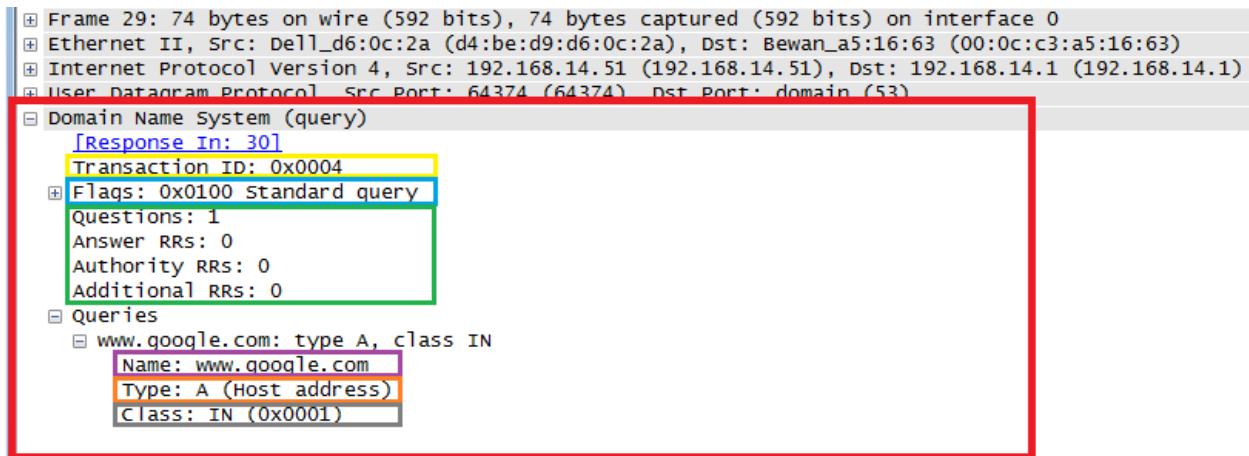
```

הערה: יתכן שתראו יותר מאוסף אחד של שאלתא ותשובות. חפשו רק את החבילה שכוללת את השאלה עבור

שם הדומין "www.google.com"



cut נטמך בחבילת השאלה :(Query)



כפי שניתן לראות, פרוטוקול DNS (שהחלק שלו בחבילה מסומן באדום) נשלח בשכבה החמישית, מעל פרוטוקול UDP בשכבה הרביעית. cut נטמך 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 והשאלה הבאה מקבלת את המזהה 5. דבר זה מקל על המחשב להפריד בין התשובות שיקבל מהשרת, ולדעת איזו מהן שייכת לאיוז שאלתא.

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 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 עברו 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 העוסה 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, ולכן לא נתעכ卜 על שדה זה.



### תרגיל 4.15 - תשאול רשומות מסווגים שונים

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

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

כך לדוגמה, עבור תשאול רשומה מסוג A, ניתן לכתוב:

**nslookup -type=A www.google.com**

כפי שציינו קודם, קיימים גם סוגי רשומות נוספים, כגון PTR - שמתאר רשומה שmaps 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 מסוג A עבור הדומיין `www.google.com`, וצפינו 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 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 ה-DNS - היכולת לקשר כתובות IP שונות אחת ליותר מכתובת IP אחת.

נתמקד באחת מרשותות 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), שכלל את שם הדומיין המלא. בדוגמה זו, התשובה היא עבור השם [www.google.com](http://www.google.com).
- **כתום** - שדה סוג (Type) הרשותה. כאמור, מדובר בDWORD בסוג A.
- **באפור** - 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](http://www.google.com), והוא יכול להשתמש בה כדי לתקשר עם השירות.



#### **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](http://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 שרתה Root, גלו מי הוא שרתה NS האחראית על ה-zone של com. מיهو שרתה NS? מה היא כתובת ה-IP שלו? מה הרצטם בכך לאלו זאת?
- באמצעות שרתה NS האחראית על ה-zone com, גלו מי הוא שרתה NS האחראית על ה-zone של google. מיho שרתה NS? מה היא כתובת ה-IP שלו? מה הרצטם בכך לאלו זאת?
- באמצעות שרתה NS האחראית על ה-zone google.com, גלו מה היא כתובת ה-IP של maps.google.com. מה היא הכתובת? מה הרצטם בכך לאלו זאת?



#### **תרגיל 4.18 – goog elgoog (אתגר)**

เครดיט: אייל אבני



הבוט הביס הצל שי הארכנ, אל פאו - החיתוף טפש מב ליחתם ליגרת לכישנים דברים שעושים בסדר הנוהג, וישנים דברים שנעשים לחלוין הפוך. אנחנו נתחיל מהסוף, ובתקווה – נצליח לשנות אותו!

בתרגיל זה ניצור שרתה DNS, שיגרום לכך שכל פניה שלנו ל-[www.google.co.il](http://www.google.co.il) תנותב לכתובת אחרת. טרם תתחילה לעבוד על התרגיל, שימו לב לקרוא היטב את ההוראות ולוזודו שאתם מבינים אותן.

הכוונות:

בחלק זה נציג הסנפה 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. שנית, הריצו את שורת הפיקודה הבאה (חשוב: פתחו CMD בהרשאות Administrator). השורה תגרום למחיקת ה-cache של 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 הסנפה עם ה-filter הבא:

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

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

עצרו ושימרו את ההסנפה ב-Wireshark.

הורידו את הקובץ

[www.cyber.org.il/networks/gvahimchallenge/elgoog.rar](http://www.cyber.org.il/networks/gvahimchallenge/elgoog.rar)

חלצו את הקבצים מתוך elgoog.rar שהורדתם לתיקיית התרגיל והריצו את הקובץ elgoog\_set.bat בהרשאות Administrator.

הסקריפט יגידיר את שרת DNS הראשי שלכם C-1.0.0.127, ואת המשני בתורו שרת הקודם שהוא לכם. שרת המשני נכנס לפעולה במידה שתרת הראשי אינו מוחזיר תשובה, וכן תוכלו להשתמש באינטרנט.

#### התרגיל:

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

מצאו בהסנפה את בקשת ה-A עבור שם הדומיין google.co.il ווואת התשובה לה.

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

בשלב הבא, צרו שרת UDP שייזין בבקשת DNS. כיון שטרם למדנו לכתוב שרת UDP, תוכלו להשתמש בשילד התוכנית הבאה. עליכם לגרום לשרת להאזין על הפורט של UDP ולכתוב את הפונקציה dns\_handler. בפרק הבא, תלמדו לעומק על פרוטוקול UDP ותבינו 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 זו מופיעה, החזירו את הכתובת 212.143.70.40 במקום את כתובות ה-IP שזרו במקור.

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

#### הערות וטיפים:

כשסימתם את העבודה, לפני שהולכים הביתה, עליכם להריץ את הקובץ המצורף: elgooG\_revert.bat בהרשאות Administrator.

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

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

לפni שאתם עוניים לבקש הספציפית שאתם רוצים לענות לה עם התשובה שלכם – מומלץ לראות שאותם מקבלים מידע כמו שצרי.

חשבו טוב לפni 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.

בכדי להכניס מידע בינהי למחוזת בפייתון, עליכם להוסיף א' לפני כל בית – אז את ערכו הhexadicלימי.

לדוגמא, במידה ונרצה להכניס את הערכים 0x37, 0x13, 0x0, נכתוב בפייתון:

```
my_binary_string = '0x13\x37'
```

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

בתרגיל אין צורך לבצע פעולות על שכבת התעבורה, שכן פרוטוקול DNS עובר בשכבה האפליקציה. לכפ' הדבר  
 היחידי 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 גליישה ל-[nana10.co.il](http://nana10.co.il) מתחילה תפנה לכתובת ה-IP של גוגל, וגלישה ל-[google.co.il](http://google.co.il) תפנה אותנו אל או. co.il. נוסף, יוכל ללא מאמץ רב להוסיף כתובות נוספות נוספות בעתיד.

## מחקר 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 מחקר של פרוטוקול שנקרא SMTP. כל מה שעלייכם לדעת בשלב זה, הוא ש-SMTP הוא פרוטוקול שהוא נפוץ מאד בעבר ושימש לשילוח דואר אלקטרוני. את יתר הפרטים אודות הפרוטוקול תסיקו בעצמכם.

בטור התחלת, **תאבחן** את ה프וטוקול, ככלומר תנתחו בעצמכם, מתוך דוגמה של שימוש בפרוטוקול, את הפקודות, השלבים והפרמטרים שלו. לאחר מכן, **תמשו** ללקוח SMTP בפייתון שיישלח בעצמו דואר אלקטרוני.

הורידו את קבצי ההסנפה smpt1.pcap מהכתובת <http://cyber.org.il/networks/c04/smtp1.pcap> ו- smtp2.pcap מהכתובת <http://cyber.org.il/networks/c04/smtp2.pcap>. קבצים אלו מכילים הסנפות של מחשב מסוים. בכל הסנפה ישנו משלוח מייל באמצעות פרוטוקול SMTP. מטרתכם היא להבין את פרוטוקול SMTP. ככלומר – להבין את מבנה ה프וטוקול, 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 במרקירים בהם או צרכיים להבין איך עובדת מערכת מביי לנו תיעוד מלא שלה. התרכו ושים דגש בצד הלקוח של הפרוטוקול (הצד **שלוח** את המייל), אין צורך להיכנס לפרטים עבור צד הרשת.

**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 פרוטוקול SMTP.
- שימו לב: לאחר הסינון יתכן ותראו גם פקודות מסווג פרוטוקול IMF. התעלמו מהן במהלך התרגיל.
- היעזרו ב- TCP Stream Follow, אותו הכרתם בפרק Wireshark ומודל חמש השכבות / Follow Stream UDP/TCP, כדי לראות את מהלך הפרוטוקול בצורה נוחה.
- נסו להבין - מה הן הפקודות שבפרוטוקול? מה הן הבקשות והתגובה? אילו פרמטרים יש לכל פקודה?
- לאחר התרגיל, זכרו את מטרת פרוטוקול SMTP – לשלווח מיילים. עצרו וחשבו כיצד אתם שולחים דואר אלקטרוני – איזה מידע אתם נדרש לספק לשם כך? איזה מידע משתנה בין מייל לאיזה 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 בקשות שונים כמו GET ו-POST, הרבה לפני שצללים לסוגי Headers הקיימים.

- במהלך התרגיל, תצטרכו להשתמש בקידוד Base64. את הקידוד זהה 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 בפעולה **:encode**:

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

שים לב כי 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 הפעולה הפוכה, משתמש בפעולה **:decode**:

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



### תרגיל 4.19 - שימוש ל Koh SMTP

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



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

על מנת לעשות זאת, השתמשו במדמה שרת SMTP, שנמצא בכתבota: [networks.cyber.org.il](http://networks.cyber.org.il). השרתamazon

על פורט סטנדרטי של SMTP, שמספרו 587.

מדמה השרת של גברים יקבל את המידע שנשלח מהלקוח שלכם. אם המידע התואם את דרישות הפרוטוקול, השרת יחזיר לכם תשובה לפי הפרוטוקול ומיתן לבקשתה. אם הלקוח שלכם שגה בימוש ה프וטוקול, שרת גברים יחזיר תשובה "הבקשה אינה לפי הפרוטוקול" וינתק את ההתקשרות.

אם מימשتم את הפרוטוקול נכון מתחילה ועד סוף, שרת גברים יחמי לכם על סיום התרגיל בהצלחה.

**dagshim limishos:**

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

- השרת אינו בודק שהסיסמה ושם המשתמש שלכם מתאימים. במקרה 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. לאור הפרק, כתבנו שירות HTTP שחלק ותפותח. בתחילת הגשו קבצים בתגובה לבקשת GET. לאחר מכן, תמכנו בסוגים שונים של תשובות, ובהמשך סיפקנו תשובה תכניתית לבקשת, בהתאם לנוטונים שהללו העביר 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 הבאים, נמשיך להתקדם במודל השכבות ולסקור שכבות נוספות - החל משכבת 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 שירות HTTP - לדוגמה SimpleHTTPServer, בכתב:
- <http://goo.gl/z2DtzF>
- מנגנוני אוטנטיקציה - ניתן לקרוא בדף: <http://en.wikipedia.org/wiki/Authentication>
  - באופן ספציפי, על המנגנון של HTTP, ניתן לקרוא עוד בדף: [http://en.wikipedia.org/wiki/Basic\\_access\\_authentication](http://en.wikipedia.org/wiki/Basic_access_authentication)
  - על המנגנון SSL/TLS, ניתן לקרוא בדף: [http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security) (עדיף לעשות זאת לאחר לימוד פרק שכבת התעבורת).

#### נושאים מתקדמים בפרוטוקול DNS

- על מנת להבין את הדרכם שבה באמצעות שאלתא, מומלץ לקרוא אחד מהמקורות הבאים:
- פרק 2.5.2 (Overview of how DNS works) בספרו Computer Networking: A Top-Down Approach (מהדורה ששית) מאת James F. Kurose .
  - 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](http://en.wikipedia.org/wiki/Post_Office_Protocol)
- העברת קבצים - פרוטוקול FTP. ניתן לקרוא כאן:  
[http://en.wikipedia.org/wiki/File\\_Transfer\\_Protocol](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 להציג לפתח עצמנו 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 לב-`sh`-`Scapy` רצה "מעל" `python`, ולכן כל הפקנציות 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 שנקלב. 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 תעבורת HTTP, כפי שעשינו בפרק [שכבה האפליקציה / פרוטוקול HTTP - בקשה ותגובה](#), ולשמור לקובץ את כל הכתובות אליהן התחבזהה גלישה? מה אם נרצה לשמור רק את הכתובות שאליהן התחבזהה גלישה והכתובות עוננות 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, ותभשו אותן בעצמכם עד סוף פרק זה. במהלך הפרק נלך יחד, צעד אחר צעד, ובכצעו לראשונה חלק קטן אך מכובד מט הפעולות Sh-`y`-`Scapy` לנו. בהמשך הספר, נשתמש ב-`Scapy` על מנת לבדוק מרוחק איזה תוכנות רצوت על מחשב מרוחק, נכתב בעצמנו כדי שדומה ל-`Ping` שפגשנו קודם לכן, נגלה מה הדרך עוברת חבילה 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 מצליחים לעקוב אחר הצעדים וمبינים 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*, *scripts* לספריה *scripts* תחת ספריית ההתקנה של Python 3 בזורה 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 לגבוי כל הערות Scapy היקר מדף למסך כשהוא עולה. מדובר בחבילות שעדין לא התקנתם כי  
הן לא נחוצות לכם, או למשל חוסר תמייה ב-IPv6, אשר לא רלבנטית לתרגום הנמצאים בספר<sup>26</sup>.  
נ hereby, עכשו Scapy פועל אפשר להתחיל לעשות דברים מגניבים.

## קבלת של פקודות (או – הסנפה)

ממש כשם 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)
```

---

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

## תרגיל 5.1 מודרך - הסנפה של DNS

כעת כשלאנו להסニア בצורה בסיסית, הגע הזמן להשתמש בידע שרכשו בכך לעשות דברים מועילים. מטרתנו עכשו היא להסニア באמצעות **Scapy**, ולהציג למסך את כל שאלות DNS שהשתמש שולח.

בפרק [שכנת האפליקציה](#), למדנו מעט על פרוטוקול DNS, אשר משמש בעיקר למיפוי בין שמות (כגון [www.google.com](http://www.google.com)) לכתובות IP, וכן על דרך העבודה של פרוטוקול זה. כעת נבנה מערכת ניתור גלישה באינטרנט, שתרוץ כל הזמן ברקע ותשמר את כל הדומיינים בבקשתות DNS. פעולה זו הייתה קשה לביצוע באמצעות Wireshark, שכן הינו צריך לאוריך זמן, לבצע סינון, לחפש כל חבילת DNS, למצוא את כתובות הדומיין מטרח השאלתא וכו'. באמצעות Scapy, יוכל לעשות כל זאת בצורה תכנית!

בutor התחלת, נסזה ליצור הסנפה ב-Scapy שתציג לנו אך ורך חבילות DNS. ראשית, הריצו את Wireshark שילווה אותנו במהלך העבודה על מנת להזכיר כיצד פרוטוקול DNS נראה.  
פיתחו את Command Line, והשתמשו בכללי **nslookup**, כפי שלמדתם בפרק תחילת מסע - איך עובד האינטרנט? DNS, כדי למצוא את הכתובת של com: www.google.com

```
C:\Windows\system32\cmd.exe - nslookup
C:\Users\USER>nslookup
Default Server: box.privatebox
Address: 192.168.14.1

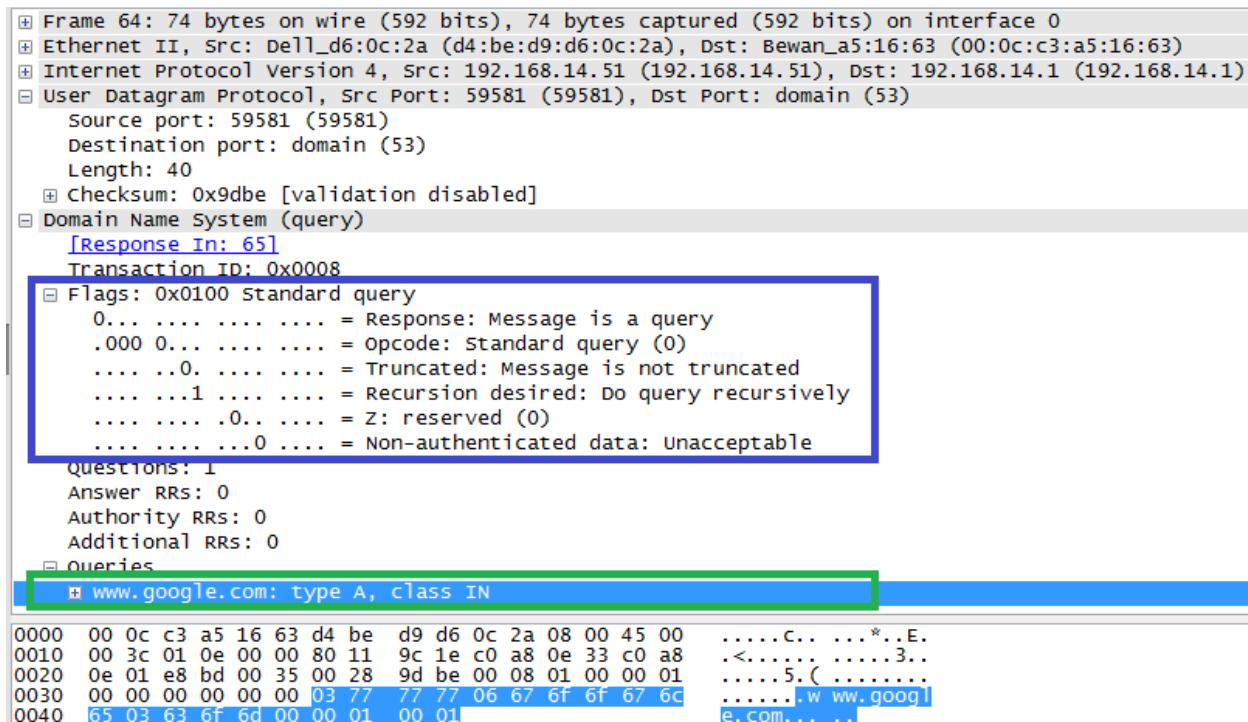
> www.google.com
Server: box.privatebox
Address: 192.168.14.1

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:



נשים לב למספר דברים:

- **בכחול** - שדה הדגלים של ה-DNS, שומרה על שאלתא.
- **בירוק** - השאלתא עצמה, מיפוי שם לכתובת, על הדומין [www.google.com](http://www.google.com)

כעת נלמד כיצד להשתמש במסנן (filter) בסיסי במהלך ההסנפה. ממש כמו שהשתמשנו במסננים בצד' לسان 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 לסנן 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.

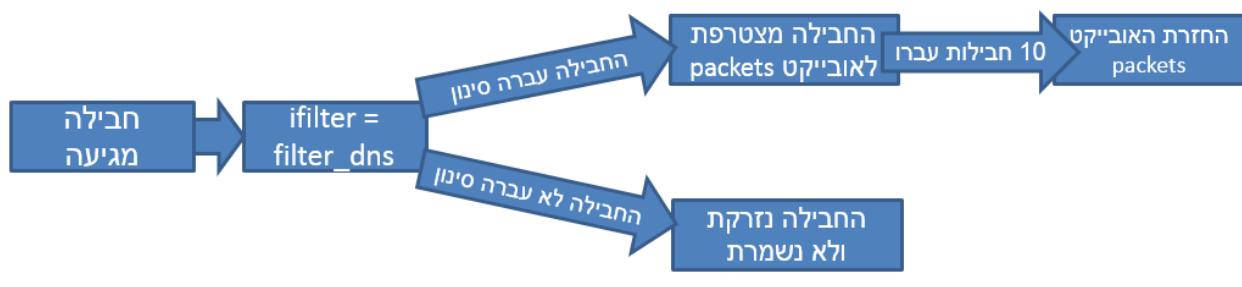
להלן הקוד הרלכנטי:

```
>>> 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 היו 10 חבילות שעברו את הסינון. שימוש לבשה הפונקציה `sniff` היא blocking, כלומר - הפונקציה לא תסיגם לרווח עד אשר ימצאו 10 חבילות שעברו את הסינון. אם החבילה לא עוברת את הסינון (`False` תחזיר `filter_dns`) - החבילה תיזרק:



הריצו שוב את הקוד 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 בשכבת DNS של החבילה. ניתן לעשות זאת באמצעות גישה לשכבת 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 עצמוו 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)
```

نبיט שוב בחבילת DNS שלנו. למעשה, ניתן לראות שככית DNS מוחולקת מבחינה Scapy לשני חלקים: החלק הכללי של DNS, והחלק שיכול את השאלות 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 הטענה השאלות:

```
>>> my_packet [DNSQR].qname
'www.google.com.'
>>>
```

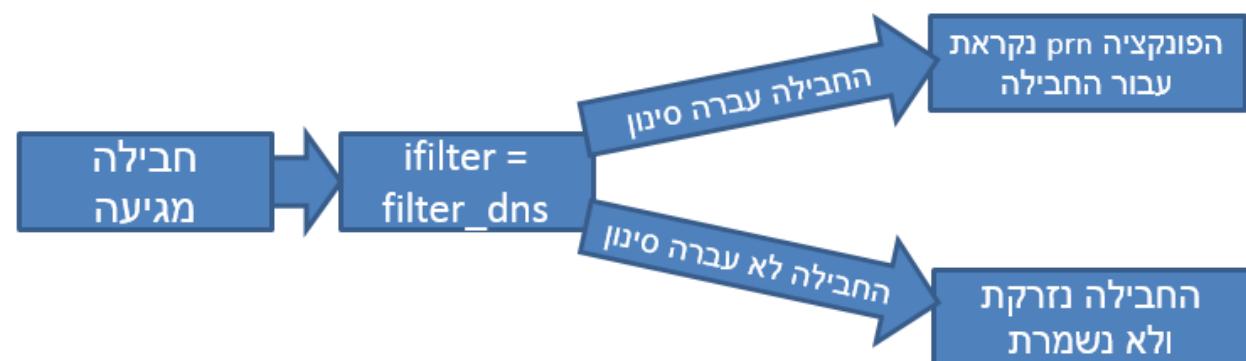
כמו כן, נרצה שהסינון שלנו יבצע רק על שאלות מסווג A. נבדוק מה הערך שנמצא בשדה qtype ומשמעותו שאלות A:

```
>>> 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)
```

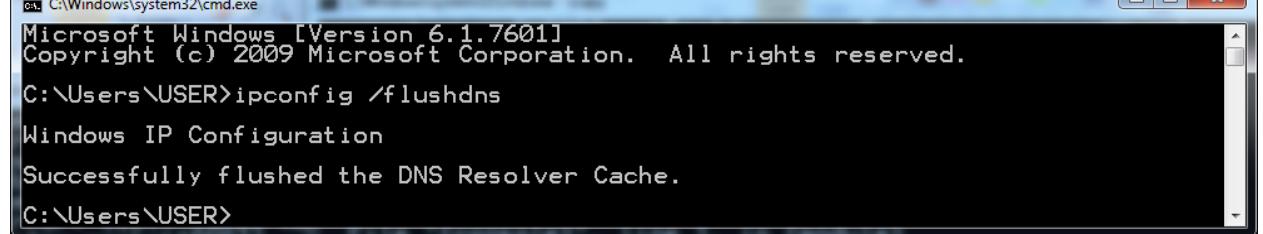
עכשו אנו מסננים אך ורק על שאלות DNS מסווג A. בשלב הבא, נרצה להדפיס למסך את השם שעליו הטענה השאלות. לשם כך, נשתמש בפרמטר של הפונקציה `sniff` שנקרא `prn`. פרמטר זה מקבל פונקציה שמבצעת פעולה על כל פקטה שעוברת את ה-`filter`. כמובן, כל חבילה שהצליחה לעבור את הסינון שהתבצע קודם לכך באמצעות `ifilter` תשלח אל הפונקציה שניתנה ל-`prn`. מכאן שרשרת הפעולות שלנו תראה כך:



ראשית, עלינו להגדיר את הפונקציה שתoruן. ברכזנו להדפיס את שם הדומיין שלו של הבדיקה השאלה. לשם כך, נגדיר את הפונקציה بصورة הבא:

```
>>> def print_query_name(dns_packet):
...     print dns_packet[DNSQR].qname
:27 ipconfig /flushdns
```

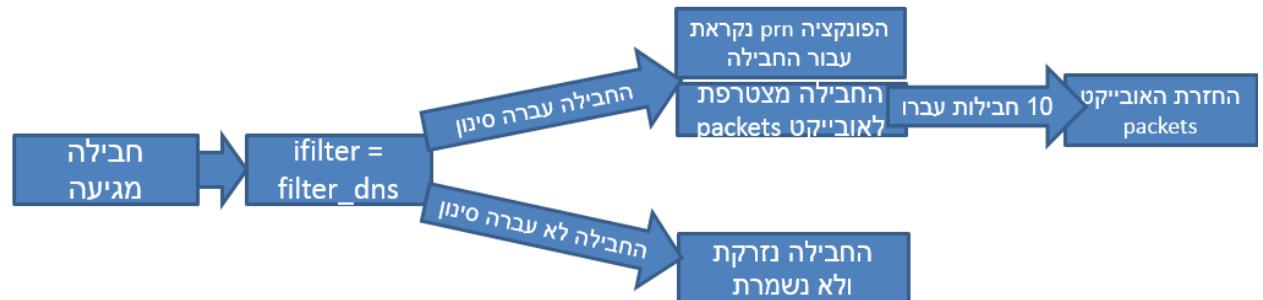
לפני ביצוע ההסנהה, נאפס את המטען של רשומות DNS באמצעות הפקודה



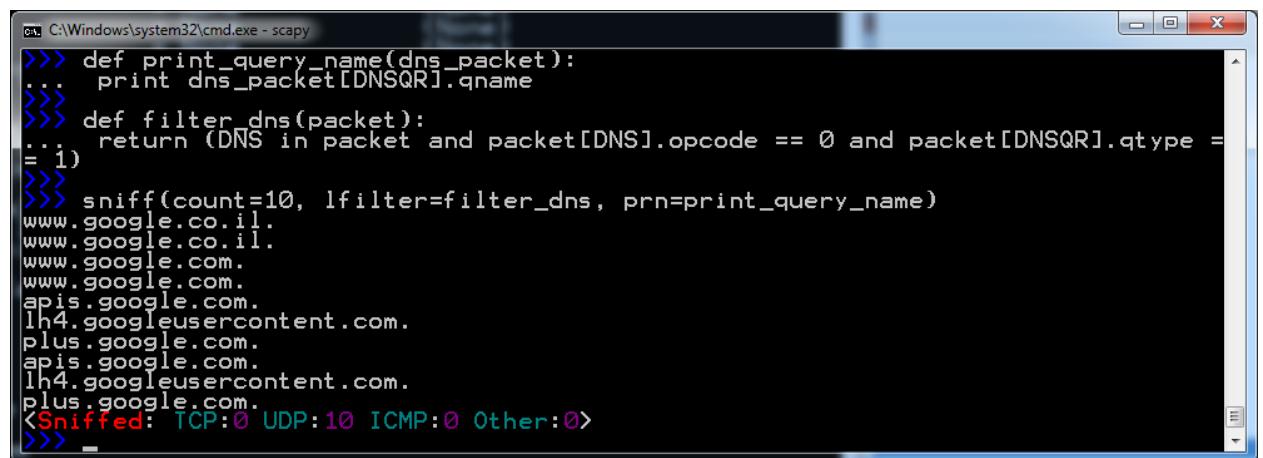
, נוכל לקרוא לפונקיה **sniff**, כאשר אנו מסננים על שאלות DNS מסוג A בלבד באמצעות **lfilter**:  
ומדפיסים את שמות הדומיינים שעליהם הבדיקה השאלתא באמצעות **prn**:

```
>>> sniff(count=10, lfilter=filter_dns, prn=print_query_name)
```

זהו השרשרת הפעולות המלאה:



נסו לגלש בדף ויראו אילו תוצאות אתם מקבלים. אתם צפויים לראות פלט הדומה לפלט הבא:



<sup>27</sup> אם לא נבצע פעולה זו, מערכת הפעלה עשויה "לזכור" את התשובות לשאלות קודמות ששאלנו, ולא לשאול עליה. כך למשל, אם נגלוש ל-www.google.com, מערכת הפעלה עשויה "לזכור" את כתובת ה-IP שנמצאה בעבר קודם לכן, ולתת אותה כתשובה מבלית בבקשת לשאול את שרת DNS.

מכיוון שהפונקציה שניתנת ל-`tzk` היא קוד פיתון לכל דבר, יוכל לבצע כל פעולה שנרצה. וכך, למשל, לשמר את הדומינינס הללו לקובץ. שימוש לב כמה קל 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 מתייחס אליו פשוט כאל מידע (כلمיר - מחוזת של תווים). אם נסניף חבילת HTTP, היא תיראה

למעשה, כל שכבת ה-HTTP שלנו הובנה על ידי Scapy בתור מידע Raw, והוא מוצג כרכץ מיידי ותו לא.

ברצוננו להשיג את כל בקשות-HTTP שפוגות לעמוד מסוים, כולל בקשות מסוג GET.



**כיצד נוכל לשנן חבילות לא מזוהות?**

העבירורה שלה היא TCP<sup>28</sup> והמידע שלה מתחילה במחזרות 'GET' הינה אכן חבילת GET.

כדי לבדוק האם המידע מתחילה ב-'GET', علينا ראשית להפוך אותו למחוזצת:

```
>>> data_string = str(my_packet[Raw])
```

cut נוכל לבדוק האם המידע מתחילה במחוזת הידועה מראש:

```
>>> data_string.startswith('GET')
```

בשלב זה, נוכל לכתוב פונקציית filter באמצעותה נסנן חבילות 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 במסנן שהגדכנו קודם לכן, והדפיסו באופן מסודר את כל ה-URLים אליום מטבחעת בקשה GET במהלך ריצת הסקריפט. שימו לב לכתוב URL מלא, כולל גם את ה-Host. לדוגמה, במקרה, בחבילה my\_packet שהציגנו קודם, תודפס למסך הכתובת הבאה:

[www.ynet.co.il/home/0,7340,L-8,00.html](http://www.ynet.co.il/home/0,7340,L-8,00.html)

<sup>28</sup> על פרוטוקול זה, והסיבה שימושים בו כפרוטוקול שכבת התעבורה של HTTP, נלמד בפרק שכבת התעבורה.

## הסניף - סיכום

ובכן, למדנו להסニアפ באמצעות Scapy. הצלחנו להסニアפ חבילות DNS ו宦官 HTTP, בנוינו מסנן באמצעות **ifilter** והצלחנו לבצע פעולות על החבילות באמצעות **hrcf**. 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. הרוי להסニアפ חבילות זה טוב ויפה, אך בהחלט לא מספיק.

## יצירת חבילות

לעתים נרצה פשוט ליצור חבילה משלנו, אולי להסתמך על חבילה קיימת שהסニアפ מהרשת. נתחיל ביצירת פקטה של <sup>29</sup>IP. 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. האם 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()
```

```
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 בונה אותם עבורנו, וגם נותן להם ערכים היגיוניים (למשל הגדרה היא 4, שכן מדובר ב-IPv4). ttl הוא 64, וכך הלאה<sup>30</sup>). מכאן ש-**Scapy** יודע ליצור עבורנו חבילות עם ערכים היגיוניים, ואנחנו יכולים לדאוג אך ורק לערכים המעניינים אותנו.

<sup>29</sup> את פרוטוקול IP פגשנו כבר בפרקיהם הקודמים, ונרחיב את ההכרות איתה בפרק שכבת הרשת.

<sup>30</sup> 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, כשניסינו להציג את `my_packet`, הוא בחר שלא להראות לנו את שדה כתובות היעד, שכן כתובות היעד לא השתנתה מערך ביריתת המחדל שלו.

כਮון ששיםינו כתובות היעד יבוא לידי ביטוי גם כשביצע `show_packet_my`. בצעו זאת בעצמכם כעת. אם ננסה לשנות יותר משדה אחד, תקלה תופעה דומה:

```
>>> my_packet.ttl = 5  
>>> my_packet
```

כאן שניסינו 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 שווודאי 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 TCP<sup>31</sup>. 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 **אדומים**, בעוד שכבת IP תחומה בסוגרים המשולשים **כחולים**. 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-Ethernet<sup>32</sup>:

```
>>> my_packet = Ether() / IP() / TCP()
my_packet
<Ether type=0x800 | <IP frag=0 proto=tcp | <TCP |>>>
```

כעת הפקטה my\_packet הינה פקטה IP וכן שכבת TCP. נוכל גם לשנות את הפרמטרים של השכבות השונות בזמן ייצירת הפקטה:

<sup>31</sup> על פרוטוקול זה נלמד לעומק בפרק [שכבת התעבורת](#).

<sup>32</sup> על פרוטוקול זה נלמד בהרחבה בפרק [שכבת הLAN](#).

```
>>> 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 (ששווה 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) יהיה 80. Scapy יודע לבצע את התרגום ולהציג אותו כפורט הייעודי של HTTP<sup>33</sup> - ממש כמו ש-Wireshark יודע לעשות.

דבר נוסף שאפשר לבצע ב-Scapy הוא להוסיף מידע "Raw", שיישמש אותנו כ-payload לשכבה הנווכחית. 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 הסנפנו חבילת HTTP קודם לכך. על אף Scapy, כאמור, לא מכיר את פרוטוקול HTTP, יוכל ליצור חבילת 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 מבפנים כתוב Raw, אלא לכתוב את השורה בצורה פשוטה:

```
>>> Ether()/IP()/TCP()/"GET / HTTP/1.0\r\n\r\n"
```

ניתן גם לבדוק ייצוג 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 00 00 00 00 00 00 00 00 45 00 :E.....@.i.....
0010  00 33 00 01 00 00 40 06 7C C2 7F 00 00 01 7F 00 :3.....P..P.
0020  00 01 00 14 00 50 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
```

<sup>33</sup> נדון בנושא הפורטים בפרק שכבת התעבורה.

## Resolving

בפרקים הקודמים הזכירנו את התרגום של שמות דומיין (כגון "www.google.com") לכתובות IP (כגון "172.15.23.49"). תהילך זה נקרא Resolving, ו-Scapy יודע לבצע אותו בשביינו. لكن, אם נכתבת את השורה הבאה:

```
>>> my_packet = IP(dst = "www.google.com")
```

Scapy יציב בשדה כתובת היעד 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 רשם את כתובת ה-IP אלא את הדומיין שציינתם. עם זאת, כאשר נשלח את החבילה (כמו שנלמד לעשוות בהמשך הפרק), Scapy יdag להבין את כתובת ה-IP הרלוונטייה ולהציב אותה בשדה הרלוונטי.

## שליחת פקודות

עד כהן למדנו ליצור עצמנו פקודות בשכבות שונות, וכן להציג אותן על המסך. כעת נלך צעד אחד קידימה, ונלמד גם **לשלוח** את הפקודות שיצרנו.

Scapy מציע, בגדול, שני דרכים לשילוח 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 הסנפה.

כעת נוכל לשלוח את החבילה:

```
>>> send(my_packet)
```

Sent 1 packets.

נסו לזרוח את החבילה בהסנפה שלכם ב-Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	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 b e d9 d6 0c 2 a 08 00 45 00	.....C.. ..*..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 - קבוע בעל הסימפת עך, ולא עבודה מתוך ה-Interpreter), ניתן להשתמש ב- Scapy לצרכים אלו באופן דומה לשימוש בחבילות אחרות ב- `os`.python. לשם כך, יש לבצע import בצורה הבאה:

```
from scapy.all import *
```

כעת ניתן להשתמש באובייקטים של Scapy.

לדוגמא, נשתמש בקוד שכתבנו מוקדם יותר בפרק בצד' להציג את כל הדומיינים שעבורם מתבצעת שאלתת 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)
```

היכולה ליצור ולשלוח חבילות תשמש אותן רבות בהמשך הספר.

## תרגילי 5.4 -Scapy

1. צרו חבילת IP באמצעות Scapy, ודאו שכותבת הידע שלה תהייה 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**. שימושם לבן 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 עady, Scapy, ולאחר מכן למדנו לבנות באמצעותו חבילות, לשולח אותן ולהסניף חבילות המתקבלות בראשת. בתקווה, הצלחתם להבין את דרך העבודה עם הכללי הנהדר הזה, וגם **קצת** לספוג את ההתלהבות מהשימוש בו והכוח הרב שהוא נותן לכם ול老子ך אצבעותיכם (בצירוף המקלדת, מן הסתם).

כפי שנכתב בתחילת הפרק – Scapy הוא כל'י. בדומה לקריאה וכתיבה, הוא לא נותן המון **כשלעצמו**. אך שמחברים אותו עם דברים נוספים (כמו הידע הנרכש שלכם בראשות), בהחלט אפשר להגיע רוחק. Scapy ימשיר ללוות אותנו לאורך הספר כולו. ניעזר בו כדי לתרגם נושאים קיימים, וכן נעמיק את היכרותנו עימנו.

## Scapy - צעדים להמשך

אלו מכם שמעוניינים להעמק 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 לשרת<sup>34</sup>.

#### שלב ראשון - שרת עצמאי

- על השרת להאזין לשאילתות DNS נכנסות בפורט 53, ולענות עליהן.
- השרת צריך לתמוך רק בסוגי הרשומות A ו-PTR<sup>35</sup>.
- על השרת לשמור קובץ TXT שיכיל את מסד הנתונים שלו. בכל רשומה יהיה רשום סוג הרשומה, 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).

---

<sup>34</sup> באופן תאורטי, יכולנו לעשות זאת מעל设备 loopback – כולם מעל הכתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של Scapy בשילוח וקבלת מסגרות מעל Windows loopback device ב-Windows, השתמש בשני מחשבים.

<sup>35</sup> 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 חברתית**

בשלב הקודם, השרת שלכם פועל לבדו. אם הוא לא הכיר שם דומיין מסוים - הוא לא הצליח לתת שירות טוב ללקוח. כדי如此, שרתים ייעודים לפעול בצורה חברתית מסוגלים לתת שירות טוב יותר. שפרו את השרת שתכתבם בשלב הקודם:

- במידה שהשרת לא מכיר שם דומיין עליון 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 לקוח.
- 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 הזה, 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://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 בגירסה 2.6. תוכלו להוריד אותה

מהכתובת: <http://goo.gl/2LIOYq>

2. אם מותקנת לכם יותר מגרסת אחת של Python – עדיף שתטיסו את הגירסאות האחרות ותשאירו רק גירסה 2.6. אחרת, שימו לב שככל ההתקנות הבאות מתיחסות לגירסה 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 לתיק'ית 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. וודאו כי Wireshark מותקן אצלכם. הוא מכיל גם את ה-driver הנחוץ לשם הריצת Scapy (driver Winpcap).

6. התקינו את Pyreadline, אותו ניתן למצוא בכתובות: <http://goo.gl/COFL9o>

7. אם ברשותכם מערכת הפעלה מסוג Windows 7 64bit, בצעו את השלבים הבאים:

א. הורידו את הקובץ <http://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>. שמו לב כי זהה גרסה מיוחדת עבור Scapy. הערה: תחת Windows 7 / Vista, 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, לחצו על כפתור ימני בעט התוכנה ובחירה "administrator".

שםו לב שמדריך ההתקנה הרשמי נמצא באתר הפרויקט, בכתב:

<http://www.secdev.org/projects/scapy/doc/installation.html>  
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 קורה? 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,علינו 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 מטרת אופציונאלית, 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:



הדבר הגיוני בהנחה והשרת מרים שירות של אימייל. עם זאת, יתכן שהלקוח ישלח יותר מבקשת אחת לשרת. למשל, יתכן והשרת 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 מילוקוחות 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 / כתובות של Sockets](#), 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 בעיר תל 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" מתיחס לפורט שתוכנה כלשהי משתמש עליו. למשל, אם יפנו אל הפורט זהה, תהיה תוכנה שמכונה לקבל חיבור. באם יש לנו שרת שMRIIZ 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

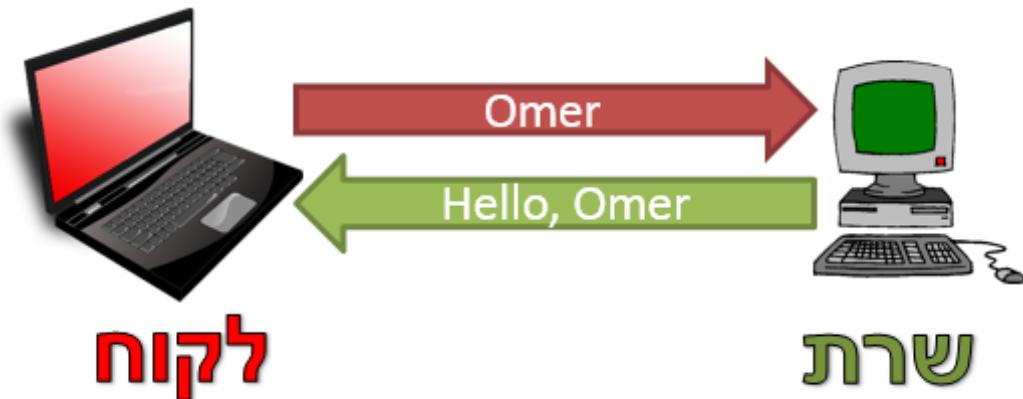
בואו נבחן את הפלט של הפקודה **netstat**:

- 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 מבצע אזין. הכתובת כתובה בפורמט של "IP:Port" ("IP:Port"). כך לדוגמה בשורה הראשונה, כתובת ה-IP הינה 127.0.0.1, והפורט הינו 5354. התעלמו מכתובת ה-IP בשלב זה, נלמד להכיר אותו בהמשך הספר.
- בכחלון** - הכתובת הרחוקה אליו המחשב מחובר. במידה שאנו 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 רצתה 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 כשריצץ את השרת שכתבנו בפרק תכנות-*s-sockets*. הריצו את השרת הראשון

שכתבנו בפרק [תכנות ב-Sockets / תרגיל 2.3 מודרך - השרת הראשון שלו](#), זה שמקבל שם מהלך ומחזיר לו:

תשובה בהתאם:



להזכירם, השרת בתרגיל האzin על פורט 8820.

הרכזו את השרת:

```
PS C:\Windows\system32\cmd.exe - server.py
C:\Users\USER>cd \Cyber
C:\Cyber>server.py
```

אפשרו לשרת להפסיק לרגע. כתע, הריצו שוב את הכלי **netstat**. איןכם צפויים לראות את ההאזנה. דבר זה נובע מכך שבאופן ברירת 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 **ל-netstat להציג בכל זאת את החיבור שלנו**, נשתמש בדגל **-a**<sup>36</sup>, ככלمر נרץ את הפקודה בצורה הבאה:

**netstat -a**

כעת אם נביט בפלט, נוכל לראות את ההאזנה שאנו מבצעים:

---

<sup>36</sup> דגל (באנגלית flag) בהקשר זהה הינו פרמטר לפקודה. כך למשל, הפרמטר "a" לפקודה netstat 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

למעה, ניתן לראות את ההאזנה כבר בשורה הראשונה!

משמעותו של State LISTENING הוא של המחשב מוכן לפתיחת חיבור. State ESTABLISHED מציין חיבור רצוי.

בואו נבחן זאת. פיתחו את Python, וכתבו לקו אשר מתקשר עם השרת אותו יצרתם, כפי שלמדנו בפרק תכנות Sockets. אל תנטקו את החיבור בסופו ועל תסגורו את אובייקט ה-socket, שכן אנו מנסים לשמור על החיבור פתוח.

להלן דוגמה לקווד צזה:

```
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 לסיום 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 Authority) <sup>37</sup>. 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 / תרגיל 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.

<sup>37</sup> 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 "ללא לאי-בוד" בדרך ולא להגיע ליעדן, או אולי להגיע בסדר הלא נכון (ח毕לה מס' 2 תגיע 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 תגיע. לשם כך, שכבת התעבורה צריכה לספק העברת אמינה של מידע מצד לצד.

עם זאת, לא תמיד רוצה שכבת התעבורה לספק העברת אמינה של המידע. לכן, מטרת זה היא אופציונאלית בלבד - ובחלק מהשימושים של פרוטוקולו' שכבת התעבורה אין הבטחה שהמידע יגיע ושיגיע בסדר הנכון. בהמשך הפרק נזכיר פרוטוקולים שונים של שכבת התעבורה, וכן בין מודיעע לעתים נעדיף להשתמש בפרוטוקול ש מבטיח 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 שמعلיה?

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 [התקנות ב-Sockets / תרגל 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()
```

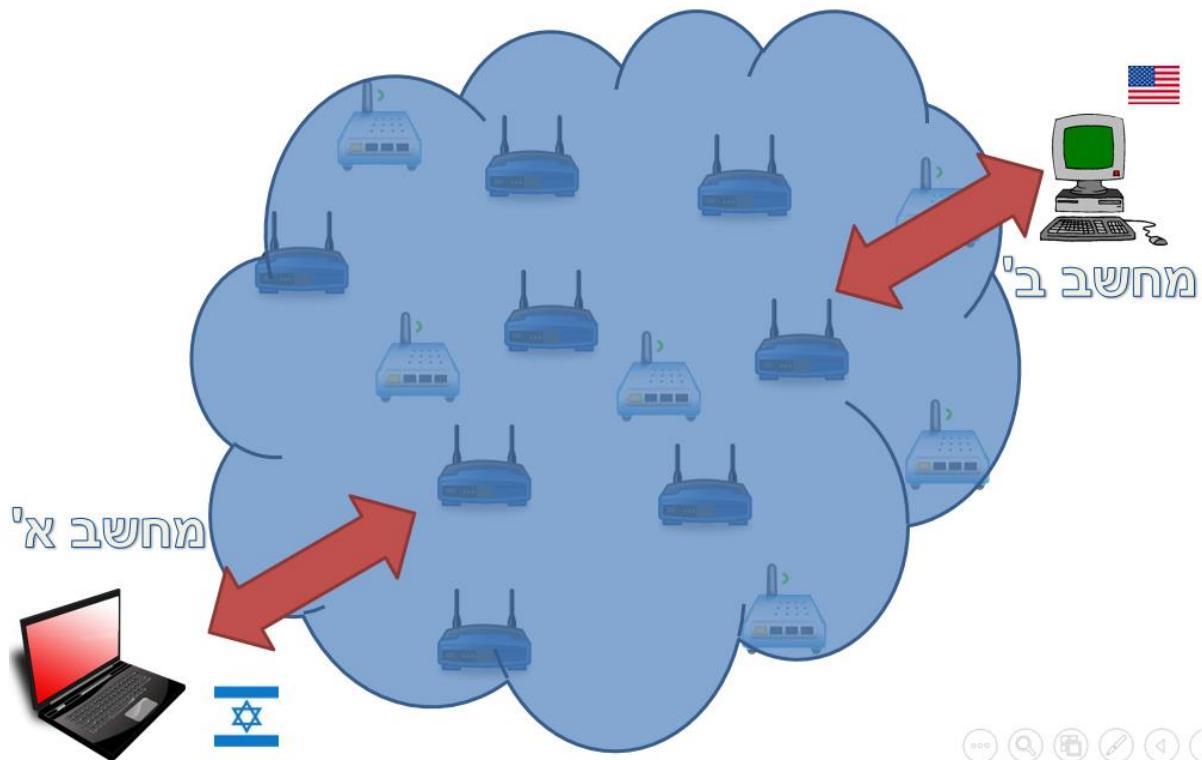
בתווך מתוכנים, שלחנו הודעות מהתוכנה שלנו, לתוכנה שנמצאת על הרשת. במקרה זה, שלחנו את המידע 'Omer' מהתוכנה שלנו, אל תוכנה שנמצאת בשרת המרוחק בכתביות 1.2.3.4 (הכתובת לדוגמה עbor שם

הדומיין networks.cyber.org.il ומאזינה לפורט 8820. ה-Socket דאג לכל שאר התהליך, ולכן שההודעה באמת תגיע 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 ששכבת הרשת אינה מודעת לפורטים. אך, 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 מבוסס קישור, יש ראיית "להקדים" 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 קשר עם התוכנה המרוחקת, ועתה כל חבילה שנשלח תהיה חלק מאותו קשר.

**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. כמובן, ישנו מידע רב שנשלח בראש בנוסף על המידע שרצינו להעביר. אם נרצה את המידע "שלום לכם" באמצעות פרוטוקול מבוסס קישור, علينا להנسر את הקישור לפני שליחת הודעה, 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 המידע מבלתי 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azel מצלם, אתם יכולים להגיע ברגל. אם אתם גרים במרקם מסוים, יתכן עליהם להגיע אל תחנת האוטובוס, להמתין עד שיגיע האוטובוס ולנסוע באמצעותו אל בית הספר. כל זאת הנה תקורה של התהלייר - מטרתכם היא אמונה ללמידה בשיעור בבית הספר, אך עליהם לעבור תהליך על מנת לעשות זאת. במקרה זה, ניתן היה למשל להנמיך את התקורה אם היותם בוחרים לישון בבית הספר, ובכך הייתה נמנעת התקורה של התהלייר ההגעה. עם זאת, כפי שווידאי מובן להם, לעיתים עדיף לשלים את מחיר התקורה על מנת לקבל את היתרונות שהיא מציעה (שינוי בבית, או העברת אמונה של מידע מעל הרשות).

## **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 ברשת הדואר מבלי קשר למכתבים אחרים שנשלחים. ברוב המקרים, אם נשלח שני מכתבים מכתובת אחת לכתובת 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. כשאנו, בתור מפתחי שכבת האפליקציה, משתמשים ב-UDP ב כדי לשוחח בibile, אין הבטחה שהחbillה תגעה ליעדה. כמו כן, אין הבטחה שהחbillות תגעה בסדר הנכון. אף לכך, אין גם צורך בהרמה וסירה של קישור. באם מתכוון בשכבת האפליקציה רוצה לשוחח בibile מעל פרוטוקול UDP, הוא פשוט שולח את החbillה.

### **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 הנכון. אף לכך, נבחר להשתמש בהם במקרים רבים. לדוגמה, כאשר אנו מורים קובץ מהאינטרנט, הגיוני שנעשה זאת מעל TCP: לא נרצה לחלוק מהקובץ יהיה חסר, שכן אז נוכל לפתח אותו. כמו כן לא נרצה לחלקים מהקובץ הגיעו בסדר לא נכון, ואז הקובץ לא יהיה תקין.

עם זאת, לא תמיד נרצה להשתמש בפרוטוקול מבוסס קישור כגון TCP. כמו שכבר למדנו קודם, ל-TCP יש תקורתה גבוהה יחסית: יש צורך בהקמה וסירה של קישור, יש צורך לוודא שהמידע הגיע ליעד והגיע בסדר הנכון... 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, ושיאפשר בסדר הנכון. אחרת, יתכן ולא נוכל לפתח את הקובץ בכלל. במקרה זה נעדיף "לשלם" את המחיר של הרמת וסירת קישור, ווידוא הגעת המידע וכל ה-Overhead המשתמע משימוש ב-TCP - על מנת שהמידע יגיע באופן אמין.

### מקרה מבחן: פרוטוקול DNS

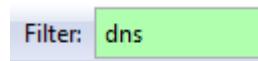
הזכירו בפרוטוקול DNS עליו למדנו בפרק [שכבת האפליקציה](#). מה דעתכם - האם בשימוש ב-S-DNS נעדיף להשתמש ב-UDP או ב-TCP?

התשובה במקרה זה היא UDP<sup>38</sup>. הסיבה לכך היא ש-DNS הוא פרוטוקול מסווג שאילתא-תשובה. הלוקוח שלוח לשרת שאלת (למשל: "מי זה [www.google.com](http://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?

נסה לאמת את ההנחה שלנו - האם באמצעות פרוטוקול DNS עובר מעל UDP? הריצו את [Wireshark](#), ופיתחו הסנהפה. השתמשו במסנן "dns":



כעת, פיתחו את ה-Command Line והשתמשו בכל **nslookup** כדי לשלוח שאלתא על הדומיין [www.google.com](http://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 והסתכלו על החביבה:

---

<sup>38</sup> 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: 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

```

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 לצפוף בהן, علينا לשלים את המחויר של שימוש בפרטוקול מובוסס קישור.

#### מקרה מבחן: Skype

מה דעתכם - האם בתוכנה לביצוע שירותי IP over Voice over IP כגון Skype נעדיף להשתמש ב-UDP או ב-TCP? במקרה זה בולט מאד הצורך ב מהירות - אנו רוצים שהקובל שלנו יגיע מצד לצד באופן כמו שהוא. גם כאן אין הפסד גדול באמ חילק מהחbillות הלווי לאיבוד בדרך. אך במקרה זה נעדיף להשתמש בפרטוקול UDP.

#### תרגיל 6.3 - מקרה מבחן: שרת HTTP



בפרק שכבת האפליקציה למדנו על פרוטוקול HTTP. חשוב בעצמכם - האם נעדיף במקרה של שרת HTTP להשתמש ב-UDP או ב-TCP?icut, וידאו את תשובה. השתמשו ב-Wireshark ובדפדף כדי לגלוש לשרת HTTP, ומצאו האם פרוטוקול שכבת התעבורה בו משתמש השירות הוא בהתאם לפרטוקול 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 להשתמש ב프וטוקול UDP ולא לשולח חבילות ישר מעל שכבת הרשת?

לשימוש בפרוטוקול לא אמין של שכבת התעborה (כדוגמת UDP) מעל שכבת רשת לא אמינה, יש שתי סיבות עיקריות. ראשית, השימוש בפורטים. כפי שהסבירנו קודם לכן 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 התעborה והשירותים שהואנות ל. בacr, שכבת התעborה "מעlimה" את שכבת הרשת משכבת האפליקציה, בין אם היא מספקת אמינות ובין אם לא.

## UDP - User Datagram Protocol

עכשו כשחכנו לעומק את מטרותיה של שכבת התעborה, 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, עליו לשים אותו במעטפה ולששל אותו לתיבה. אין לי צורך להגיד לך אל הנமען שהוא צפוי לקבל ממנו את הודעה, וכן אין הבטחה של רשות הדואר שהמכתב הגיע מהר, או שיגיע בכלל. יתכן והמכתב יאבך בדרך.

### תרגיל 6.4 מודרך - התבוננות בפרוטוקול UDP

הריצו את Wireshark. הסניףו עם המסן "kdpn". חכו עד שחבריות UDP תופענה על המסך. לחילופין, תוכלו לשולח שאלתת DNS, שנשלחת מעל UDP כפי שלמדנו קודם לכן.  
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 (מסומן 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 01 .....5.(.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....,w 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 היכן שלחצנו, וגם בציגה התchapתונה שמראה את הבטים שנשלחו (מסומן בירוק). בנוסף, הוא יכתוב לנו למטה את כמות הבטים שסימנו (מסומן 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 DNS.

השדה הבא הינו שדה האורך (Length).



### מה מציין שדה האורך ב-UDP?

אם 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 גלינו 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 'Domain Name System (query)' section. The packet details pane shows the DNS header and the query for 'www.google.com'. The bytes pane shows the raw hex and ASCII data of the DNS message. The status bar at the bottom indicates 'Domain Name Service (dns) 32 bytes'.

לאחר שנלכד על השורה של השורה של Domain Name System (DNS) (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 הגיע כל, או לרוץ של חבילות הגיעו ברכזם ללא נכון. אך בעיות ברשות יכולות גם לגרום לשגיאות בחבילה עצמה - כמו למשל שבחבילה תגיע עם תוכן שונה מהתוכן שנשלח במקור.

לדוגמא, 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 והחבילה השתנתה בדרך כלל תקלת כלשהי. כך למשל, יתכן והשרת יקבל את החבילה בצורה הבאה:

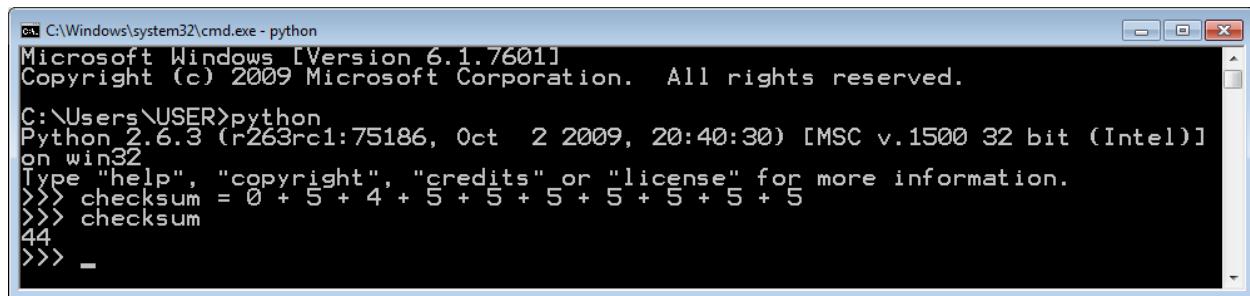


שיםו לב, הספרה הראשונה השתניתה, ועכשו היא כבר לא 0 אלא 6. במקרה זה, נרצה שהשרת ידע שאירועה שגיאה, ולא יתייחס לחבילה התקוליה. דרך אחת לעשות זאת, היא להשתמש ב-Checksum. הרעיון הוא כזה: נבצע פעולה כלשהי על המידע שהוא רצים לשלוח, ונשמר את התוצאה. בצד השני (במקרה זה, בצד השרת) החישוב יבוצע שוב, ויושווה לתוצאה שנשלחה. אם התוצאה שונה, הרוי שיש בעיה.

ນמשיך 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:

$$0 + 5 + 4 + 5 + 5 + 5 + 5 + 5 + 5 + 5$$

נוכל 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 רצה לשלוח (כלומר את מספר הטלפון), אלא גם את התוצאה של ה-Checksum. בדוגמה זו, תשלוח 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:\> 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 שהלכו שלח היה 44.<sup>39</sup> 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 הלקו התcoon 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.

אם אתם מעוניינים לראות דוגמה נוספת ל-Checksum, אתם מוזמנים לקרוא על סורת ביקורת במספר הזאות בישראל, בכתבות: <http://goo.gl/CvYtqt>. לכל אזרח בישראל יש מספר זהות בעל תשע ספרות. למעשה, שמונה הספרות השמאליות הן מספר הזאות עצמו, והספרה הימנית ביותר היא סורת הביקורת - תפקידה לוודא שאין שגיאה בכתיבתה של שמונה הספרות שלפניהם.

<sup>39</sup> אלא אם כן, הייתה שגיאה בשדה ה-Checksum עצמו. גם במקרה זה, החבילה צפוייה להיזרק.

אגב, אין חובה להשתמש ב-Checksum בפרוטוקול UDP. אם הלקוח לא מעוניין להשתמש ב-Checksum, ניתן לשלוח 0 בשדה של ה-Checksum.

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.

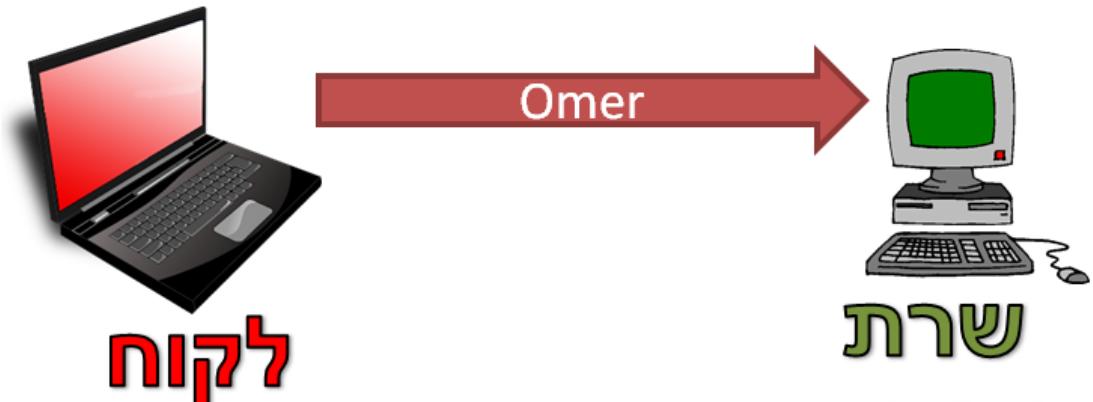
## UDP של Socket

עכשיו שלמדנו על פרוטוקול UDP, הגיע הזמן להשתמש בקוד ששולח הודעות UDP.



### תרגיל 6.5 מודרך - ליקוח UDP ראשון

cutut נכתב את ליקוח-h-UDP הראשון שלנו. הליקוח יהיה דומה מאוד לליקוח הראשון שכתבנו בפרק [תכנות ב-Sockets / תרגיל 2.1 מודרך - הליקוח הראשון שלו](#). לצורך התרגיל, ישנו שרת שאיתו נרצה לתקשר. השרת נמצא באינטרנט, והוא בעל שם הדומיין `networks.cyber.org.il`. לצורך ההסבר, 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.



## IP: 1.2.3.4

## Port: 8821

נתחל מילכטוב ל��וק פשוט באמצעות Python. ההתחלה זהה לחולוטין לקוד שכתבנו בפרק [תכנות ב-Sockets](#). הדבר הראשון שעשינו לעשות לשם כך הוא ליבא את המודול של תרגיל 2.1 מודרך - הלוקהו הראשון [של](#). הדבר הראשון שעשינו לעשות לשם כך הוא ליבא את המודול של `socket` לפייתון:

```
import socket
```

כעת, עליינו לייצור 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 הלוקהו שכתבנו בפרק [תכנות ב-Sockets](#) / [תרגיל 2.1 מודרך - הלוקהו הראשון של](#) לבין הלוקהו שאנו כתובים עכשווי. להזכירם, כאשר כתבנו את הלוקהו הקודם, השתמשנו בשורה הבאה:

```
my_socket = socket.socket()
```

מכיוון שלא סיפקנו פרמטרים ל-(`socket`), פייתון הניתן לנו משתמשים 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 לשלוח ('Omer') וכן את הtuple שמתאר את התוכנה המרוחקת ומכיל כתובת IP ומספר פורט. בשורה זו שלחנו את המחרוזת 'Omer' אל התוכנה שמזינה לפורט 8821 UDP בעלי 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 לקבל מידע מישות אחרת. לכן, `recvfrom` גם מאפשר לנו לדעת ממי קיבלנו את המידע שקיבلونו:

```
(data, remote_address) = my_socket.recvfrom(1024)
```

ונכל כמובן להדפיס את המידע שקיבلونו:

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

ולבסוף, "נסגור" את אובייקט `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)

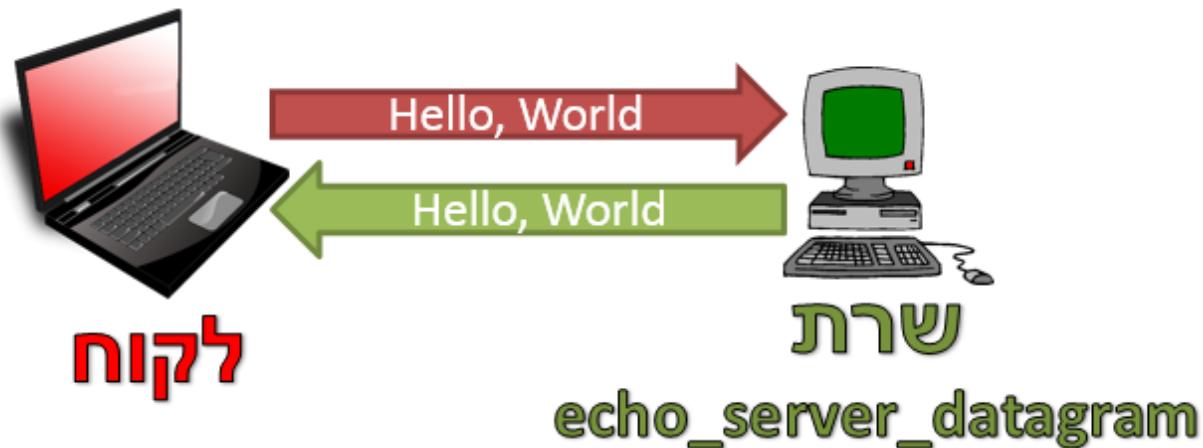
print 'The server sent: ' + data

my_socket.close()
```



### תרגיל 6.6 - ל��וח לשרת הדימ

כעת כתבו ל��וח לשרת הדימ, בדומה לכתבם בפרק [תכנות ב-Sockets](#) – [ל��וח לשרת הדימ](#). בתרגיל זה השרת כבר מושך עבורהם. להזכירם, שרת הדימ משכפל כל מידע שתשלחו לו, ושולח אותו אליכם בחזרה, כמו היד. כך למשל, אם כתבו אל השרת את המידע: "Hello, World" (שים לב – הכוונה היא למחוזת), הוא יענה: "Hello, World"



הורידו את השרת מהכתובת: [http://cyber.org.il/networks/c06/echo\\_server\\_datagram.pyc](http://cyber.org.il/networks/c06/echo_server_datagram.pyc). שמרו את הקובץ **למייקום 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 הלקו. עליים לחשב כמה זמן לוקח מזמן שליחתם את הודעה אל השרת, ועד שהתקבלה תשובה (רמז: השתמשו במודול **time** של Python). הדפיסו למסך את הזמן זהה.

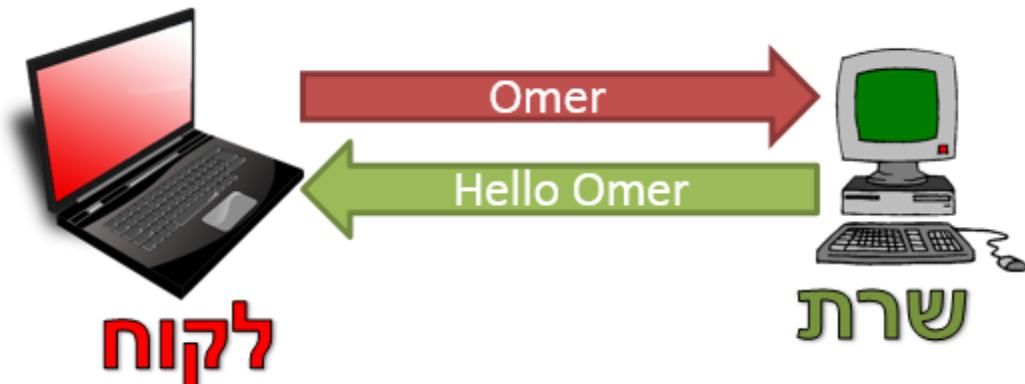
לאחר מכן, השתמשו בקוד שכתבתם בפרק [תכנות ב-Sockets](#) – [ל��וח לשרת הדימ](#), בו השתמשנו, כאמור, ב-**TCP**. הוסיףו גם ללקוח זה את יכולת למדוד זמן מהרגע שבו נשלחה הודעה אל השרת, בין התשובה.

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?

**שימו לב:** 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 ללקוח ששולח לשרת את שמו, לדוגמה: "Omer". כעת, נגרום לשרת לקבל את השם שהלקוח שלח, ולענות לו בהתאם. לדוגמה, השרת יענה במקרה זה: "Hello, Omer"



גם ב-UDP, הדרך לכתיבת שרת דומה מאוד לכתיבה של לקוח. גם הפעם, הדבר הראשון שעשינו לעשות הוא לייבא את המודול של **socket** לפייתון:

```
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**. המетодה 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 שמיימשנו 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 (אותו שמרנו במשתנה **client\_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 שבנייה 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)

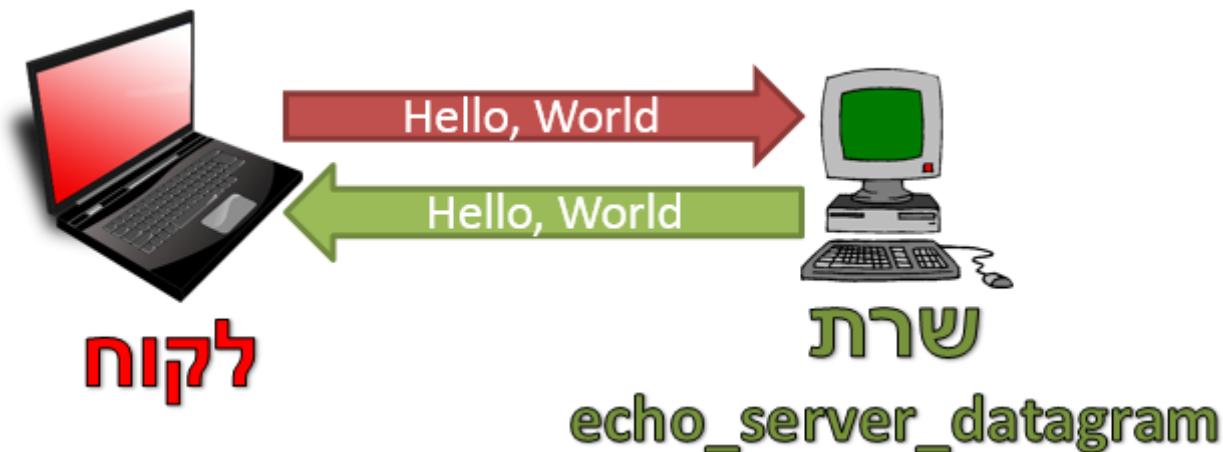
server_socket.close()
```



### תרגיל 6.9 - מימוש שרת הדימ

בתרגיל קודם, כתבתם לkokו שהתחבר לשרת מוקן. cut, באמצעות המידע שצברנו במהלך כתיבת השרת שביצענו קודם לכן, תממשו בעצמכם את השרת בו השתמשתם בתרגיל הקודם.

נזכיר כי השרת משכפל כל מידע ששלח 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 בעצמנו 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 שאליטה ב-DNS, علينا לציין כמה שאילותות 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()
```

```
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
>>>
```

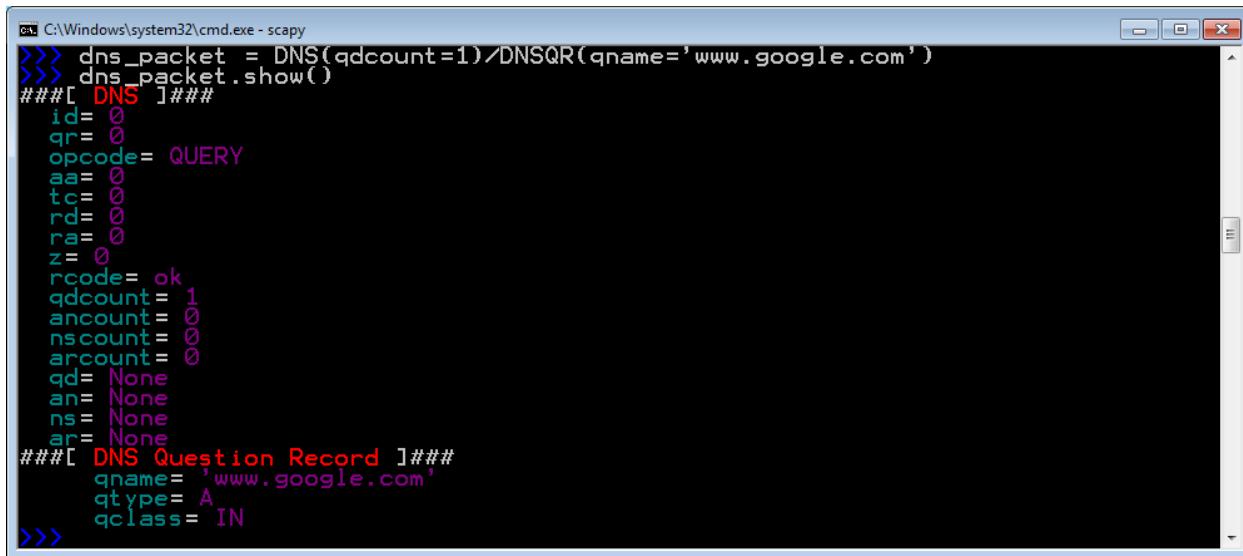
כעת علينا לבנות את השאליטה. ראשית נסתכל על הדרך שבה Scapy מציג השאילה:

```
>>> DNSQR().show()
```

```
C:\Windows\system32\cmd.exe - scapy
>>> DNSQR().show()
###[ DNS Question Record ]###
 qname=
 qtype= A
 qclass= IN
>>>
```

כפי שנitin לראות, Scapy מניח בעצמו שהשאליטה היא מסוג A, כלומר מיפוי של שם דומיין לכתובת IP. מכאן שעליינו לשנות רק את שם הדומיין, שהוא בשדה `qname`:

```
>>> dns_packet = DNS(qdcount=1)/DNSQR(qname='www.google.com')
>>> dns_packet.show()
```

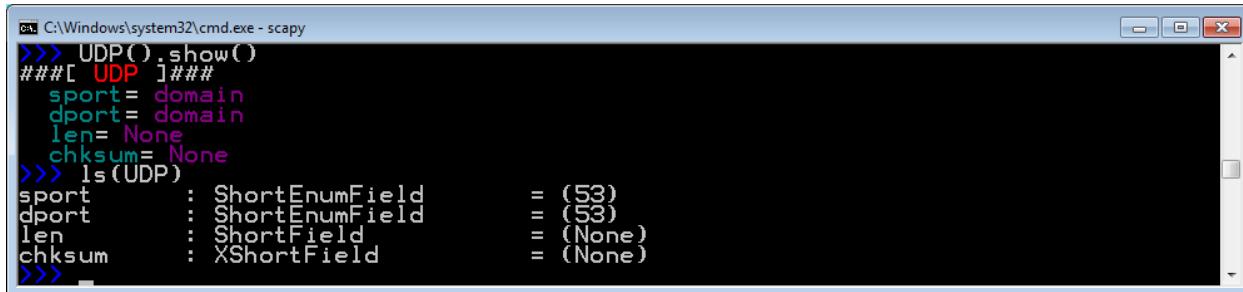


```
C:\Windows\system32\cmd.exe - scapy
>>> dns_packet = DNS(qdcount=1)/DNSQR(qname='www.google.com')
###[ DNS ]###
id= 0
qr= 0
opcode= QUERY
aa= 0
tc= 0
rd= 0
ra= 0
z= 0
rcode= ok
qdcount= 1
ancount= 0
nscount= 0
arcount= 0
qd= None
an= None
ns= None
ar= None
###[ DNS Question Record ]###
qname= 'www.google.com'
qtype= A
qclass= IN
>>>
```

cutet ברטוטנו יש חיבור שモרכיבת שכבת DNS בלבד. על מנת לשלוח אותה, נצטרק להרכיב גם את השכבות התחתיות. נתחיל מלהרכיב את שכבת ה-UDP. על מנת לעשות זאת, נבחר להשתמש ב-53 כפורט יעד (מכיוון שהוא הפורט המשויך DNS), ונבחר 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()
```

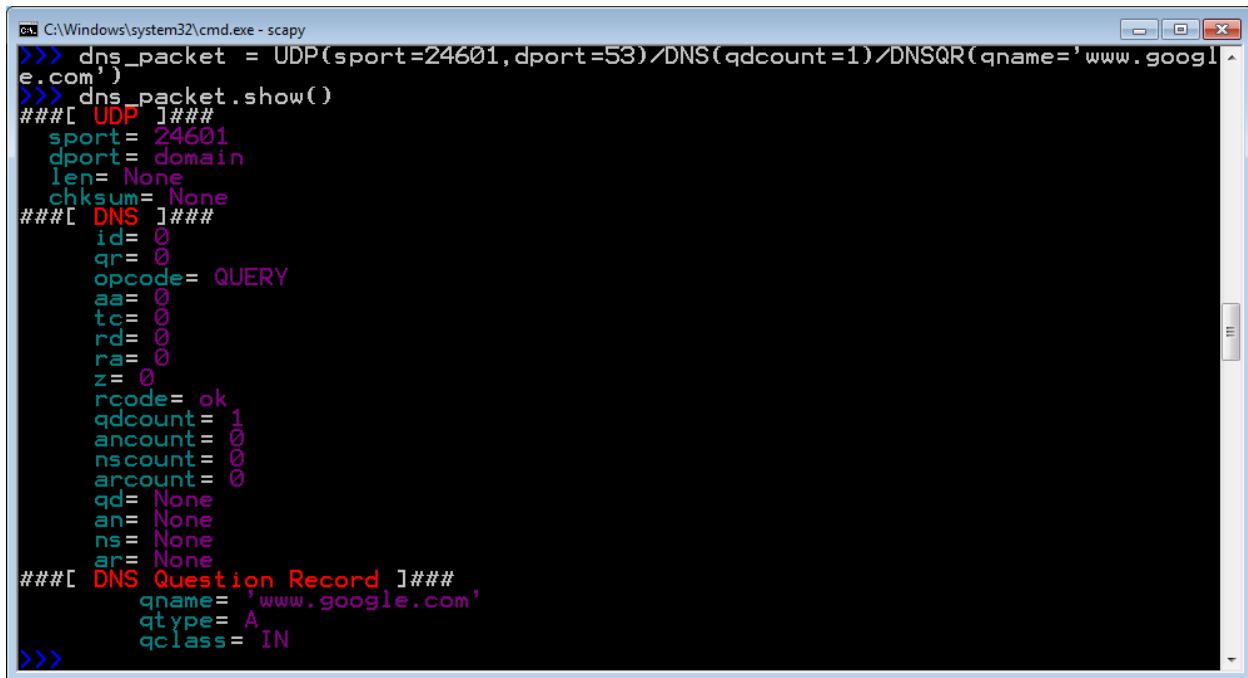
```
>>> ls(UDP)
```



```
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 ניזור 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
>>>

```

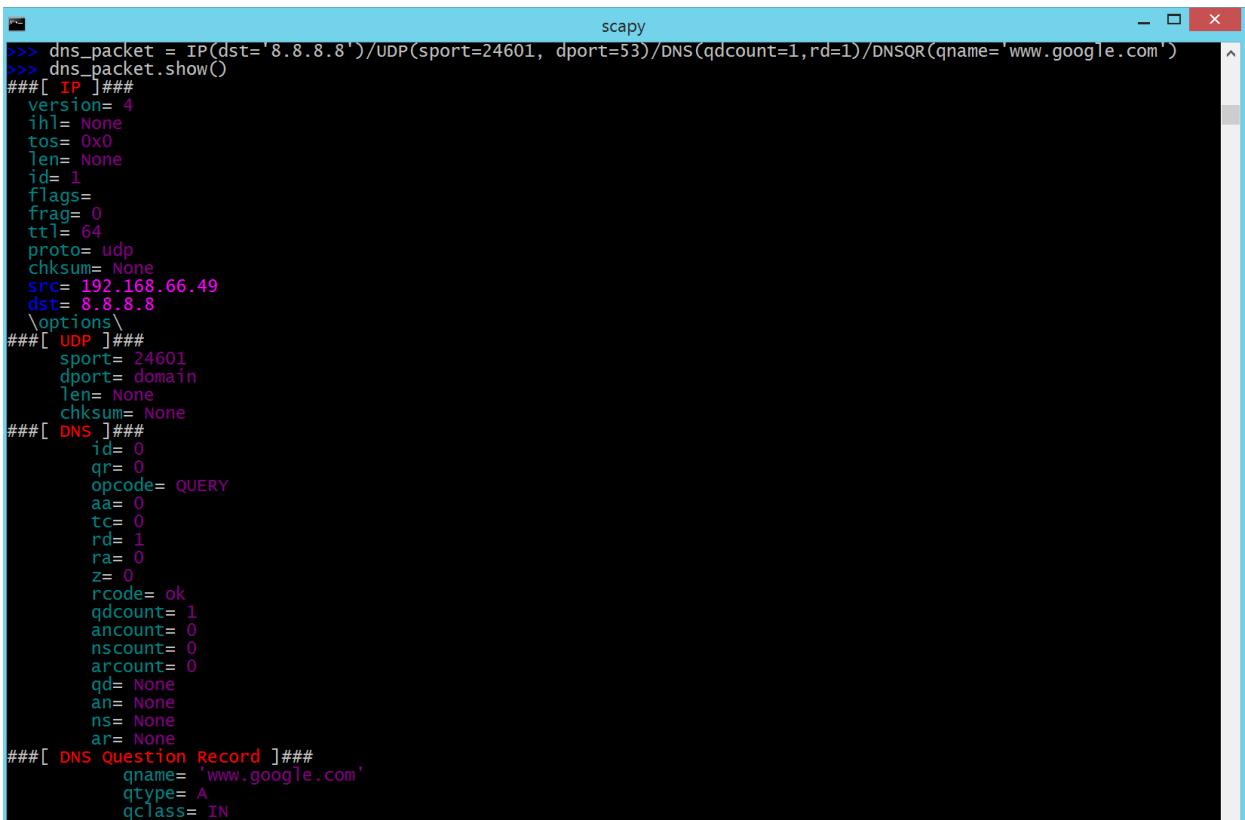
שים לב - לא הצנו אף ערך בשדות האורך (שנקרא על ידי Scapy בשם **len**) וה-Checksum (שנקרא על ידי Scapy בשם **checksum**). אל דאגה, ערכים אלו ימלאו באופן אוטומטי כאשר החbillה תישלח!

כעת علينا להפעיל לאייזו כתובת IP לשלוח את הchnbillה. לצורך התרגיל, נשלח לכתובת "8.8.8.8", שמשמשת שרת DNS באינטראנט. נבנה את הchnbillה 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 השאלת השואילת, כמו גם התשובה שהגיעה מהשרת:

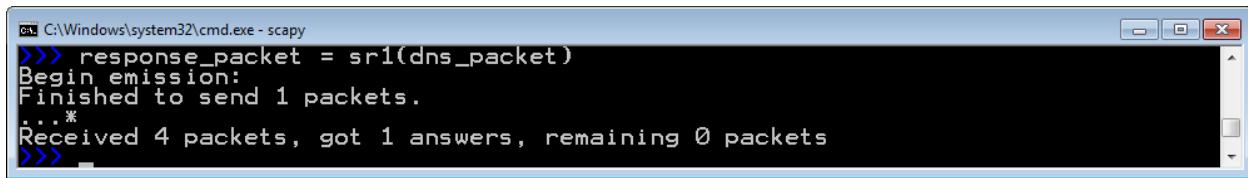
Filter: dns						Expression...	Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info			
9	1.39747000	192.168.14.51	8.8.8.8	DNS	74	Standard query 0x0000 A www.google.com			
10	1.46074700	8.8.8.8	192.168.14.51	DNS	330	Standard query response 0x0000 A 212.179.180.121 A 212.179.180.123			

## תרגיל 6.11 מודרך - קבלת תשובה לשאלת DNS באמצעות Scapy

از הצלחנו לשלוח שאלת DNS לשרת המרוחק, וגם ראיינו ב-Wireshark שההודעה נשלחה כמו שצריך ואף התקבלה תשובה. אך עכשו נרצה להצליח לקבל את התשובה באמצעות Scapy. ישנן מספר דרכים לעשות זאת, ובשלב זה נלמד דרך שהיא שימוש בפונקציה `sr` המשמשת לשילוח חבילה אחת וקבלת תשובה עליה.

השתמשו באוטה חבילת השאלת השואילת שיצרנו קודם לכן, ושילחו אותה. אך הפעם, במקום להשתמש ב-`send`, השתמשו בפונקציה `sr`, ושמרו את ערך החזרה שלה. פונקציה זו תשלח את החבילה, ואז תסניף את הרשות (כמו הפוקודה `sniiff`), ותשמר את התשובה לחבילה שנשלחה. עשו זאת כך:

```
>>> response_packet = sr1(dns_packet)
```



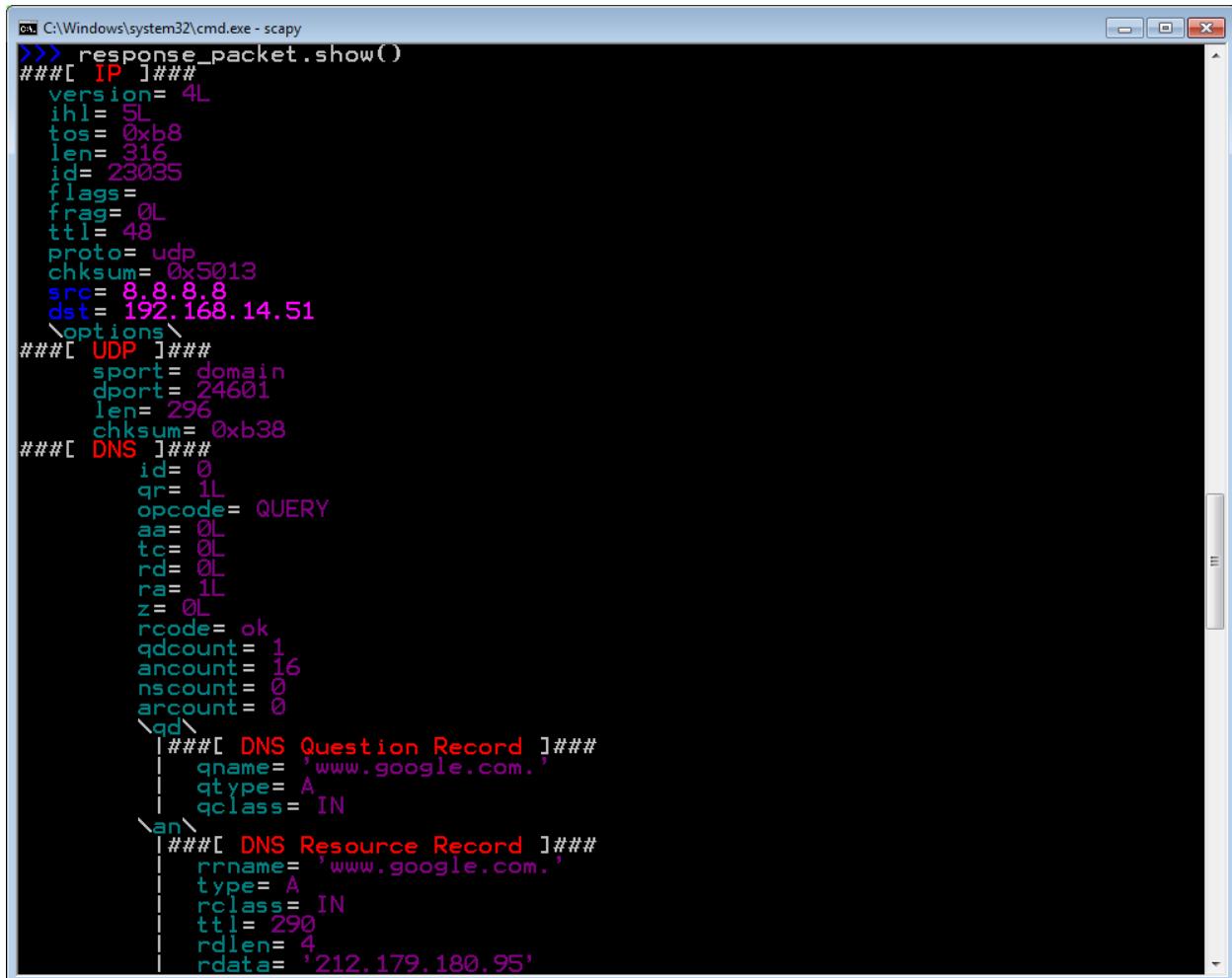
```
C:\Windows\system32\cmd.exe - scapy
>>> response_packet = sr1(dns_packet)
Begin emission:
Finished to send 1 packets.

Received 4 packets, got 1 answers, remaining 0 packets
>>>
```

שימוש לב לשורות התחתיונות. Scapy הציג, תור כד ריצה, שלוש נקודות (\*) ואז כוכבית (\*). כל נקודה צו היא פקטה ש-Scapy הסניף שלא הייתה קשורה לפקטה שלחנו (כלומר לא תשובה לשאלת ה-DNS שלחנו קודם לכן). הכוכבית 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
checksum= 0x5013
src= 8.8.8.8
dst= 192.168.14.51
\options\
###[ UDP ]###[ sport= domain
dport= 24601
len= 296
checksum= 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 יצרנו 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 - 212.179.180.95<sup>40</sup>). 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, מבלי 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, כהסימול הוא לפי קידוד ASCII (לקראת נוספת <http://en.wikipedia.org/wiki/ASCII>). לדוגמה, נאמר שיואב רוצה לשЛОח למאור את האות 'א'. לשם כך, עליו ראשית להבין מה הערך ה-ASCII שלה. בכך, לעומת זאת, הוא יכול להשתמש בפונקציה `ord` של פיתון:

```

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 גילה שערך ה-ASCII של האות 'א' הוא 97, הוא ישלח הודעה UDP ריקה לפורט 97 של מאור. במידה שיואב ירצה להעביר למאור את ההודעה "Hello", עליו יהיה לשЛОח הודעה ריקה לפורט 72 (הערך של

---

<sup>40</sup> במהלך העבודה על התרגיל, יתכן שתתקלו בשגיאה שנובעת מ-Bug בחרביה 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, ובה לתיקייה `scapy`.  
לדוגמה: `C:\Python26\Lib\site-packages\scapy`.

(2) פתחו לעריכה את הקובץ `py.pyc` שב-

(3) בשורה ה-66, החליפו את `"return socket.inet_ntoa(addr)"` ב-`"return 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 הסקרייפטים בהם ישתמשו יואב ומאור בצד העביר 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 בצד 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 בצד להסניף ולקבל את החבילות.

שיםו לב שעל מנת לבדוק את תרגיל זה, עלייכם להשתמש בשני מחשבים שונים – אחד ללקוח אחד לשרת<sup>41</sup>.

**בונוס:** כפי שלמדתם, פרוטוקול UDP אינם מבוסס קישור, ולכן יתכן שחלק מהמידע ששלחו מלקוח לשרת לא יגיע, או לחילופין 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 בצד העביר מידע, איננו יכולים פשטוט לשולח חבילה אל תוכנה מרוחקת. הראשית עליהם 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 רואית להגיד את המספר שלו, לשם צליל חיווג, ולהזכיר עד שירים את הטלפון ובכך יוקם בינינו קישור.

TCP תוכנן וועצב לרוץ מעל שכבת רשת שאינה אמינה. למשל, ההנחה הבסיסית היא שהשכבה השרת חבילות יכולות ללקת לאיבוד או להגיא שלא בסדר הנכון. בתור פרוטוקול מבוסס קישור, TCP מבטיח לשכנת האפליקציה שהמידע הגיע אל היעד בסדר הנכון.

---

<sup>41</sup> באופן תאורי, יכולנו לעשות זאת מעל `loopback device` – ככלומר מעל הכתובת "127.0.0.1", המוכרת לנו מתרגילים קודמים. עם זאת, עקב Bug של Scapy בשילחה וקבלת מסגרות מעל `loopback device`-bs-`Windows`, השתמש בשני מחשבים.



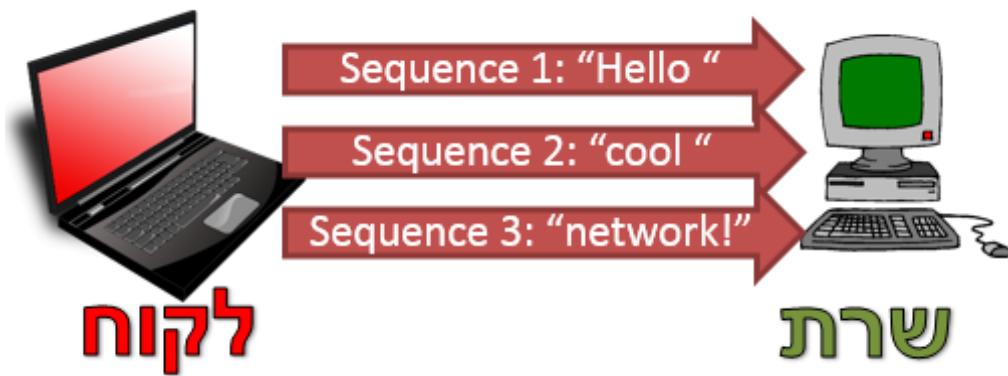
כיצד ניתן לוודא שהמידע מגיע אל היעד? כיצד ניתן לוודא שהוא מגיע בסדר הנכון?

על מנת לעשות זאת, TCP מנצל את העבודה שהוא פרוטוקול מבוסס קישור. מכיוון שכל החבילות (שנקראות בשכבת התעבורה בשם **סגמנטים**<sup>42</sup>) הן חלק מקשר, אנו יכולים לבצע דברים רבים.

ראשית, אנו יכולים לתת מספר סידורי לחבריות שלנו. נאמר שבשכבת האפליקציה רצינו לשלוח את המידע "Hello cool network!". בשכבת התעבורה, נאמר שהמידע חולק לחבריות בצורה הבאה:

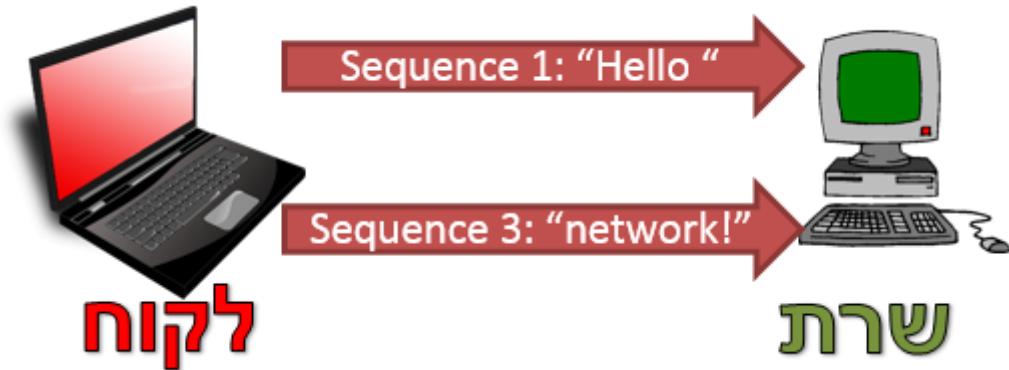
- חבילה מס' אחד - "Hello"
- חבילה מס' שני - "cool"
- חבילה מס' שלישי - "network!"

כעת נוכל לשלוח את החבילות כשלצדן יש מספר סידורי (Sequence Number):



כעת, נסתכל על צד השירות. נזכיר כי בראשת יתכן לחבריות מסוימות "נופלות" ולא מגיעות ליעדן. כך למשל, יתכן שחביבה מס' שני "נפלה" בדרך, והשירות רואה מהלך רק שתי חברות:

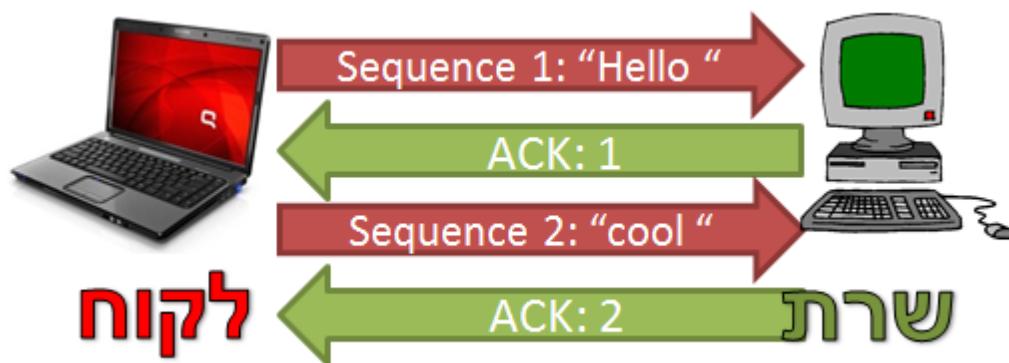
<sup>42</sup> גושי מידע בשכבת התעבורה נקראים "סגמנטים". עם זאת, כל סגמנט הוא למעשה גם חבילה של שכבה השלישית (שמכילה בתוכה את השכבה הרביעית, בהתאם למודל השכבות). על כן, ניתן לומר שכל סגמנט הוא גם פקטה (mono מה שיר לשכבת הרשת, שכבה השלישית) וניתן לקרוא לו כך.



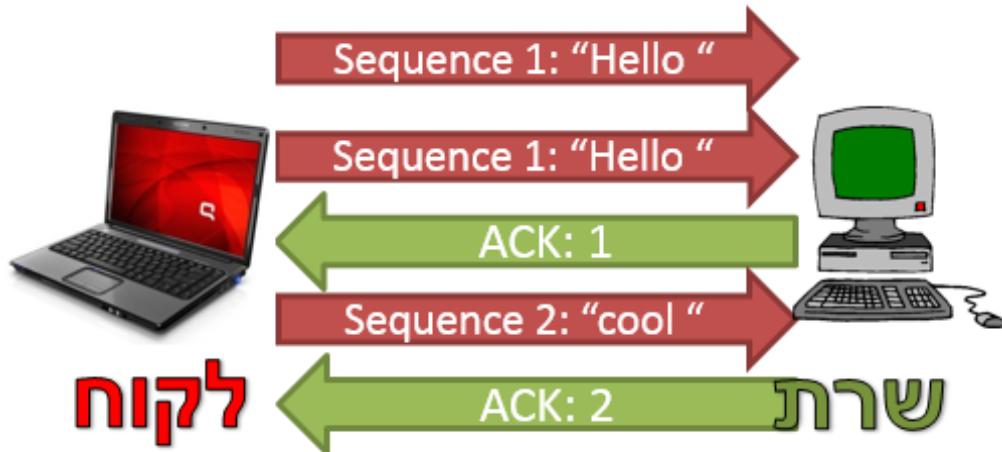
כעת, השרת יכול להבין ש.ssורה לו חביבה מספר שתים! הוא יכול לעשות זאת מכיוון שהוא יודע שהחיבור הנוכחי בין הלוקוט, הוא קיבל את חביבה מספר אחד וחביבה מספר שלוש, ולכן אמרור היה גם לקבל את חביבה מספר שתים.

ניתן להשתמש במספרי החביבות כך לודא ש.ssורה אכן הגיעה ליעדה. כך למשל, ניתן להחליט שעל כל חביבה שהגיעה, השרת ישולח אישור לлокוט. חביבה צזו נקראת בדרך כלל **ACK** (קיצור של **Acknowledgement**), ומשמעותה - "קיבלתי את החביבה שלך".

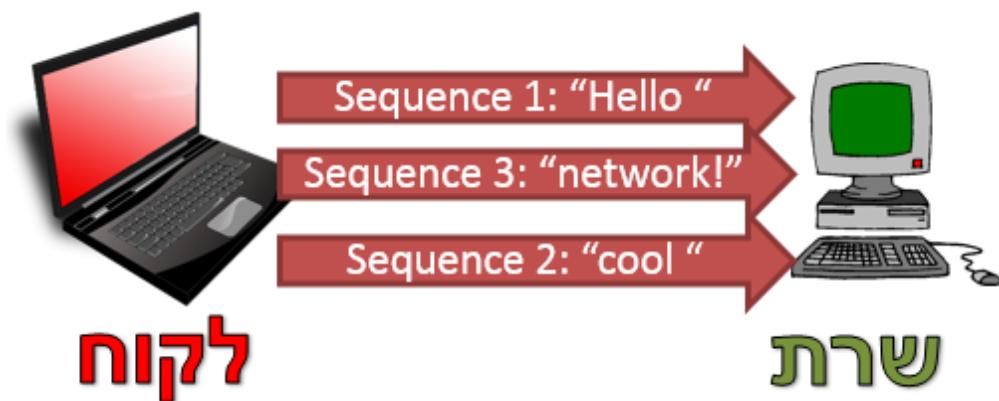
הлокוט יזכה ל接过 **ACK** על כל חביבה אותה הוא שולח לשרת:



בצד הלקוח, אם לא התקבלה חביבה **ACK** מהשרת לאחר זמן מסוים, נראה שהביבה שהוא שלח "נפלה בדרך". במקרה זה, הביבה תשלוח שוב:



כך הצלחנו להבטיח שהחbillות שלחנו באמת הגיעו ליעד!  
השימוש במספר סידורי לחבילה מאפשר לנו להתמודד עם בעיות נוספות. בגלל שהרשות לא אמינה, יתכן שהחbillות 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 שהחbillות 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 החbillות חלק מה קישור. בסיום הקישור, נרצה לסגור אותו.

 איך TCP משתמש ב-Sequence Numbers?

למדנו את חשיבותם של מספרים סידוריים ו-ACKים. עצת נסביר את השימוש של קונספטיים אלו ב프וטוקול 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 משלו. בכל חבילה שנשלח, 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-"a" השנייה היא מסטר 103, "o" הוא מסטר 104 והרוח 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 התחילה עם המספר הסידורי 106. המשמעות של כך היא שהבית הראשון שבה, כאמור התו "c", הוא בעל המספר הסידורי 106. ה-"o" שלו הוא בעל המספר 107, וכך הלאה.

שימוש לב שתקשרות TCP היא למעשה שני Stream'ים של מידע: רצף בתים לכל צד. התקשרות שבין הלקוח לשרת מבהווה רצף בתים בפני עצמה, וה-Sequence Number בכל מקטע מתיחס לרצף בין הלקוח לשרת בלבד, ולא לרצף שנשלח מהשרת אל הלקוח.



## תרגיל 6.14 מודר - צפייה ב-Squence Numbers של TCP

פיתחו את Wireshark והריצו הסנפה. השתמשו במסנן התצוגה "http", כפי שLEARנו בפרק **שכבות האפליקציה**. פיתחו דף-פן, וגילשו אל הכתובת: <http://www.ynet.co.il>. עצרו את הסנפה, וביחרו באחת מה宦יות ה-HTTP. לחצו על החבילה באמצעות המקש הימני של העכבר, וביחרו באפשרות "Follow TCP Stream":

The screenshot shows a Wireshark capture window with a list of network packets. A context menu is open over the 453 packet, which is highlighted in blue. The menu options 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 הח宦יות שהשרת שליהם, עדיף חבילה שאחראית נשלה מיד ועד חבילה מהשרת:

The screenshot shows a detailed view of a selected TCP segment from the previous list. The segment is highlighted in blue and has its sequence number (2958) and data payload (1360 bytes) highlighted with a red box. The detailed information pane at the bottom shows the following fields:  
 Sequence number: 2958 (relative sequence number)  
 [NEXT sequence number: 4518 (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)

**בדוק** ניתן לראות את ה-Sequence Number הנוכחי, קלומר מהו המספר של הבית הראשון בסegment זה, והוא בדוגמה שלנו. **בכחול** ניתן לראות את גודל המידע של הסegment הנוכחי - **1,360** בתים.

באמצעות שני נתוניים אלו, המספר הסידורי של הבית הנוכחי, וכמות הבטים שנשלחים בסוגמנט הנוכחי - נוכל לחשב את המספר הסידורי של הסוגמנט הבא! נעשה זאת כך:

$$\textcolor{red}{2,958} + \textcolor{blue}{1,360} = \textcolor{green}{4,318}$$

גם Wireshark מציין בפנינו שזה יהיה המספר הסידורי הבא (תחת הסעיף [Next sequence number]). נמשיך לבדוק זאת בעצמנו. נבחר את החבילת הבאה ונראה מה המספר הסידורי שלה:

```

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.

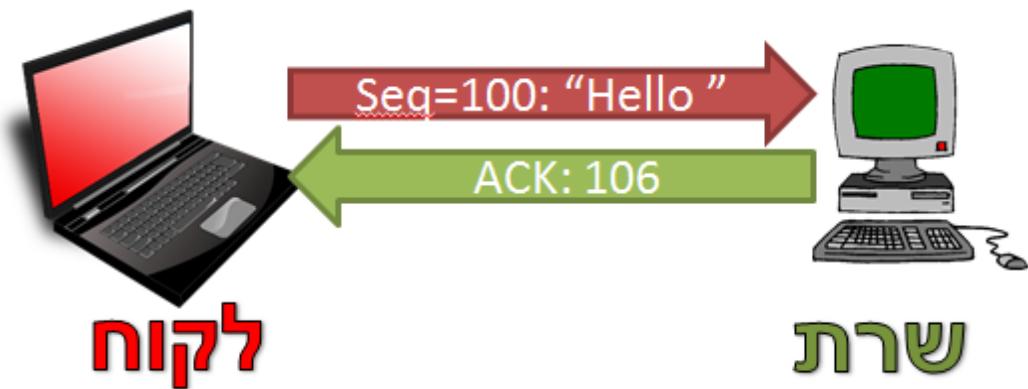
 נסו לעשות את החישוב הזה גם על חבילות נוספות.

## איך TCP משתמש ב-Acknowledgement Numbers?

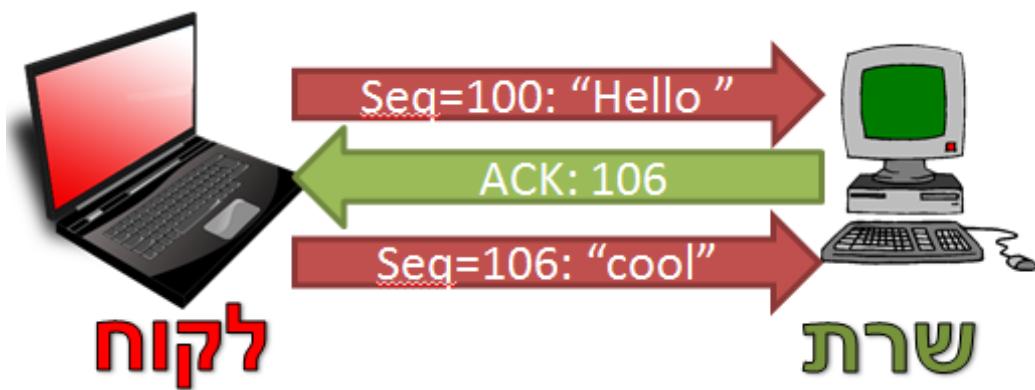
היות שהמספרים הסידוריים של TCP מתייחסים לבטים (bytes) ברכף המידע, כך גם מספרי ACK. מספר ה-ACK ב-TCP מציין את המספר הסידורי של הבית הבא שצפוי להתקבל. כך למשל, בדוגמה הקודמת שלנו:



ציינו שהבית הבא שאמור להשלוח מהלקוח יהיה בעל המספר הסידורי 106. אי' לך, ה-ACK אמור להכיל את הערך 106:

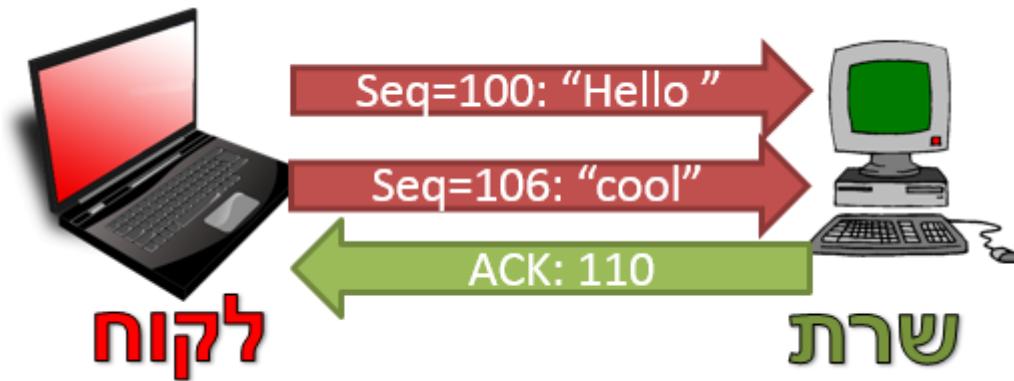


בצורה זו קל מאד לבצע מעקב אחרי התקשרות. מכיוון שה-ACK מכיל את המספר הסידורי הבא, הרי שזה יהיה המספר הסידורי שישלח בחבילת המידע הבאה. כך בדוגמה זו, רצף החבילות יראה כדלקמן:



בנוסף, כאשר נשלח ACK ב-TCP, הכוונה היא שכל המידע שהגיע עד לבית שמצויין ב-ACK הגיע נכון. כך לדוגמה, במקרה לעיל השרת יכול היה לא לשולח ACK עבור החבילה שככללה את המידע "Hello", אלא רק לאחר קבלת החבילה שככללה 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 שלוח אותו מחדש.



### תרגיל 6.15 מודרך - צפייה של TCP Acknowledgement Numbers

נסתכל שוב בחבילת האחורונה שליטה הסתכלנו באמצעות Wireshark

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: 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: 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: 331]  
 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. 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]

```

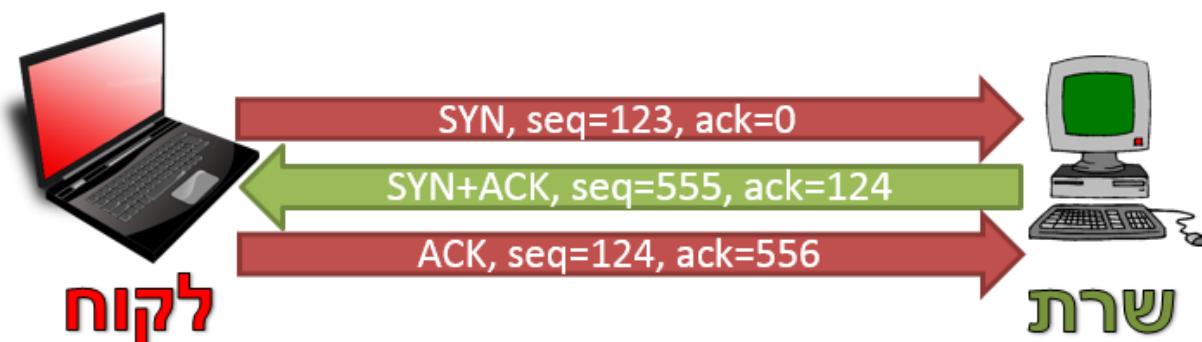
כמו שניתן לראות, אכן התקבל הערך הצפוי.



נסו עתה לחשב בעצמכם את ערכי ה-ACK שצפויים להתקבל עבור חבילות נוספות 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 להקים את הקישור בין הצדדים לפני שלב העברת המידע ביניהם. באמצעות הקמת הקישור, אנו מודיעים לצד השני שאנחנו מולו בתקשורת וublisherו להיות 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. ישותו שימוש בשדות Sequence Number וה-Acknowledgement Number של כל חבילה כדי להציג להרים את הקישור. כעת, נבין את התפקיד של כל חבילה ואת האופן בו מוחשבים הערכים בשדות אלו.

### חvíלה ראשונה - SYN

בשלב הראשון, הלוקו שולח לשרת חבילה שמטרתה להתחיל את הקמת הקישור. באופן זה, הלוקו מצין: "אני רוצה להקים קישור מולך". בכל חבילה של TCP יש כמה דגלים שניתן לצין, חלק מה-Header<sup>43</sup>. ב חבילה זו, הדגל SYN דלוק. משמעות הדגל SYN היא תחילת תקשורת. ה-Sequence Number של חבילה זו הינו ה-Initial Sequence Number ההתחלתי של הלוקו עבור הקישור זהה עם השרת, ונקרא בשם Sequence Number (Number) (ISN).



### יכיד נבחר ה-Initial Sequence Number?

ניתן היה להסכים שה-ISN, אותו מספר התחלתי 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. דמיין 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 חvíלה מהחיבור הקודם ליעדה, והשרת יחשוף אותה לחvíלה 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 החvíלה הראשונה כבוי, שהרי לא ניתן ACK על אף חבילה קודמת.

<sup>43</sup> במודון זה, דגל הינו בית שמצוין אפשרות מסוימת. הסבר על כל הדגלים קיימם ב[נספח א' - Header TCP](#).

בשלב זה, התקשרות נראה כך:

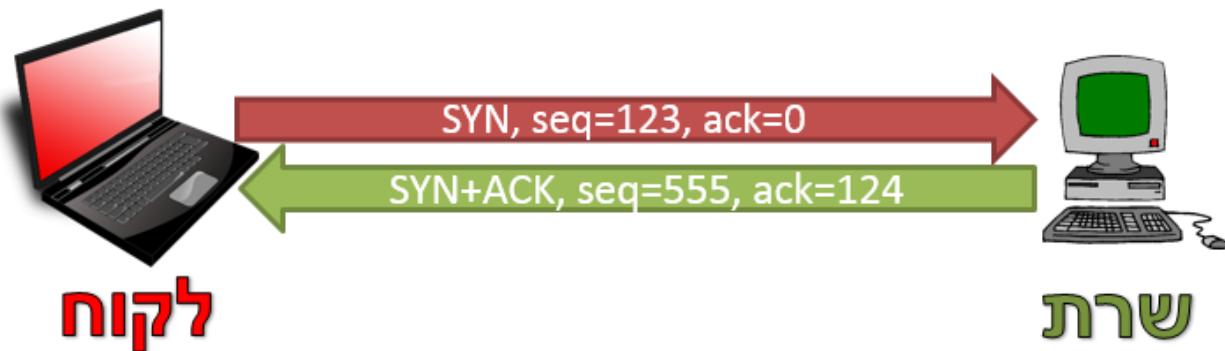


#### חbillah Shnina - SYN + ACK

בשלב השני, בהנחה שהשרת הסכים להקים את הקישור, הוא עונה בחbillah בה דלוקים שני הדגלים: SYN ו-ACK. הדגל SYN דלוק מכוון שזו חbillah שמודיעה על הקמה של קישור. הדגל ACK דלוק מכוון שהשרת מודיע ללקוח שהוא קיבל את החbillah הקודמת שהוא שלח, שהיא חbillah ה-SYN.

ה-Sequence של החbillah של השרת יהיה ה-*ISN* של התקשרות בין הלוקו. כאמור, יתאר את המספר הסידורי ההתחלתי של הבטים שנשלחו מהשרת אל הלוקו. נציג שbow שתקשורת TCP היא למעשה שני Stream של מידע: רצף בתים לכל צד. המספר הסידורי של התקשרות של הלוקו (שמהחיל ב-123) מצין את המספר הסידורי של הבטים בין הלוקו לשרת, והמספר הסידורי שהשרת ישלח בשלב זה יתאר את המספר ההתחלתי של הבטים בין הלוקו. גם השרת יגריל 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 החbillah שלו. כפי שהסבירנו קודם, ה-ACK מצין את המספר הסידורי של הבית הבא שצפוי להגיע. במקרה של חbillah SYN, החbillah נספרת בגודל של בית אחד (על אף שלא נשלח שום מידע). כאמור,ערך ה-ACK יהיה המספר הסידורי של החbillah שהשלח (בדוגמה שלנו, 123) ועוד 1 עבור ה-ACK. מכאן שערך ה-ACK יהיה 124. כך נראה התקשרות בשלב זה:



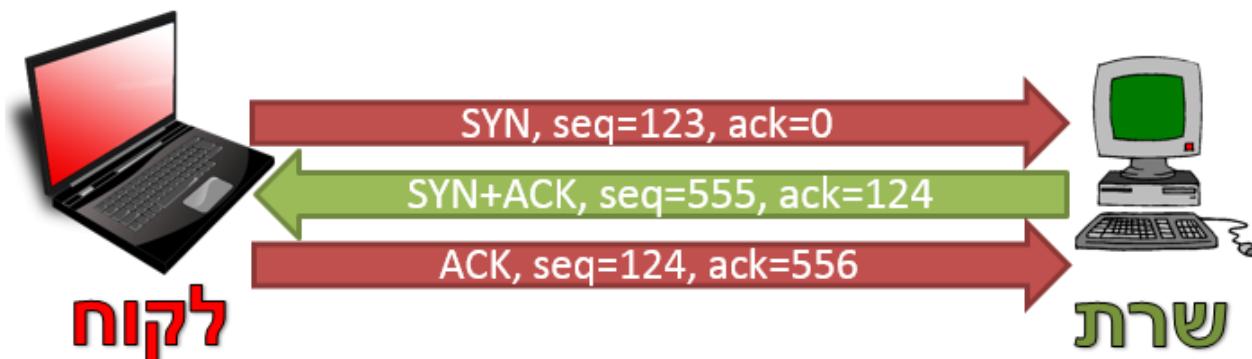
#### חבייה שלישית - ACK

על מנת שהקישור יוקם בהצלחה, על השרת לדעת שהחבייה הקודמת שהוא שלח, חבילת ה-SYN+ACK הגיעה אל הליקוּד בהצלחה. אם החבייה אכן הצלחה להגיע, גם הליקוּד וגם השרת יודעים שהקישור קם, הסכימו להתחילה אותו, וכן מסונכרנים על המספרים הסידוריים הראשונים (כלומר ה-NSN) אחד של השני.

על מנת לעשות זאת, הליקוּד שלוח חבילה כshedagel 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 מלהיקוּד לשרת בקשר הנוכחי.

כਮון שהליקוּד צריך גם לככל את המזהה הסידורי של הבית שהוא שלח, כמו בכל חבילה של TCP. הערך זהה הינו הערך שהוא ב-ACK של החבייה שהתקבל מהשרת, לחושב באמצעות לקיחת המספר הסידורי הראשוני (123) והוספת 1 עבור ה-NSN. מכאן שהמספר הסידורי הוא 124:



בשלב זה הוקם קשר קייזר בין הליקוּד לשרת, ועכשו ניתן לשלוח מעלי חבילות מיד!



### תרגיל 6.16 מודר - צפייה ב-Three Way Handshake של TCP

פתחו את Wireshark והריצו הסנפה. גילשו שוב אל האתר [www.bewan.com](http://www.bewan.com), בזמן שמשון הציגו שלכם הוא "http". השתמשו שוב ב-Wireshark 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: 6192  
 [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. ....3Q....  
 0020 1f 89 d5 ff 00 50 fa ba de bb 80 12 ..P.....  
 0030 20 00 40 65 00 00 02 04 03 b4 01 03 03 02 01 01 ..@. ....  
 0040 04 02 ..

כפי שניתן לראות בכתחום, הדגל הדלוק בחבילה הוא דגל SYN, שמצוין הקמת חיבור. ה-SYN כבוע 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 האמתי, נוכל ללחוץ על השדה זהה, וכעת Wireshark יציג לנו גם בשדה המידע של החבילה את הערך האמתי (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 . ....\*.. . . . 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 בכתחום, הדגלים הדולקים בחבילה הם אלו של SYN ו-ACK. באידט, 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, שמצוין כ-0). אם נלחץ על שדה זה, נראה שהוא שערך 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 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: 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]

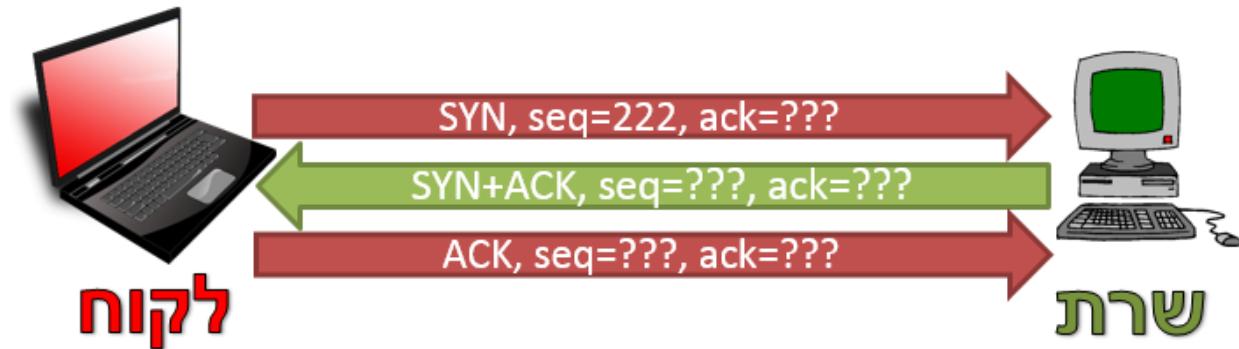
0000  00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00  . ....c.. *...E.
0010  00 28 21 c0 40 00 80 06 00 00 c0 a8 0e 33 51 da  .(!:@.... 3Q.
0020  1f 89 d5 f0 00 50 fa ba de bb cc e1 1e 8d 50 10  ....P.....P.
0030  41 14 40 59 00 00 A. @Y..
```

כפי שניתן לראות בכתחום, הדגל הדולק הוא אכן דגל ה-ACK, ואף לא דגל אחר. באידט ניתן לראות את המספר הסידורי של הבית הנוכחי. כפי שכבר ציינו, Wireshark מציג אותו באופן ייחודי כ-1, ולהזיהה עליו תראה לנו בבירוק 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 d8d 0xcce11e8c בbasis הקסיה-דצימלי, צפוי.



### תרגיל 6.17 - חישובי Three Way Handshake

בشرطוט שלפניכם מוצגת לחיצת יד משולשת, אך חלק מהערכים בה חסרים ובמקומם מופיעים שלושה סימני שאלות:



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. נתחיל מכך שנפתחה הסנפה ב-Wireshark. כעת, געלה את Scapy וניתן חבילת TCP: ניצור TCP חיבור Scapy.

```
>>> 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
chksum= 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 היעד הוא כרגע 0. פורט היעד הוא פורט 80, ו-Scapy מציג אותו עבורהנו כ-HTTP. עם זאת, נרצה ליצר בעצמנו את החבילה ולשלוט בפרמטרים האלו. ولكن, ניצור בעצמנו סגמנט TCP עם הדגל SYN, כשפורט היעד הוא פורט 80, וניתן בערך ה-Sequence Number 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, ولكن 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:

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, שהניח שאولي 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 ]###[<br>
version= 4L
ihl= 5L
tos= 0x0
len= 44
id= 30529
flags= DF
frag= 0L
ttl= 250
proto=tcp
checksum= 0x1b59
src= 173.194.113.147
dst= 192.168.14.51
\options \
###[ TCP ]###[<br>
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 ]###[<br>
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 ההזזה כדי 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 מצויים 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 וניתן 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 124601:

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			

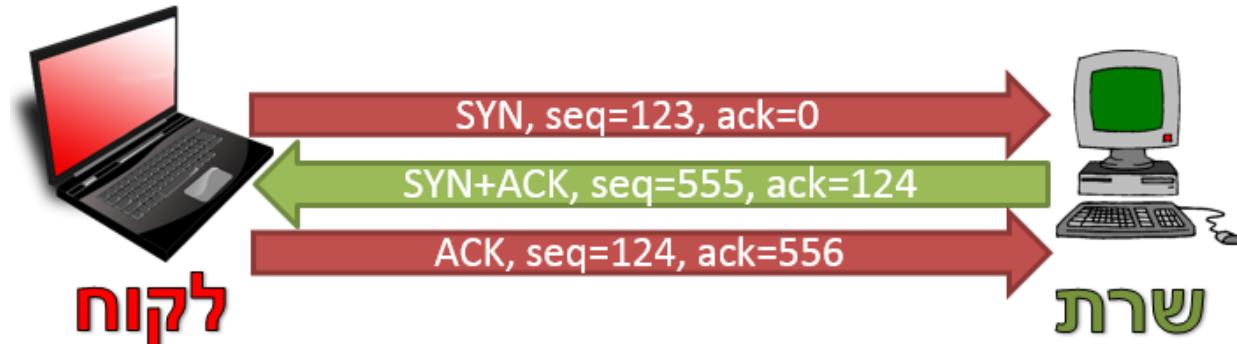
השרת של Google כלל לא נתן תשובה של SYN+ACK! במקרים מסוימים, שירותי מרוחקים יענו חבילת כshedgal-h-RST דולק, ומשמעותו שהשרות לא מוכן להרים את הקישור.

השתמשו בהתקנות זאת כדי לכתוב סקריפט אשר מקבל מהמשתמש כתובת IP, ומדפיס למסך איזה פורטים פתוחים במחשב המרוחק, בטווח ה포רטים 1024-2048. מכיוון שהסקריפט עתיד לשלוח תעבורת רבה, **אל תבדקו אותו על שירותי האינטרנט**, אלא רק על מחשבים נוספים בביביטם או בRICTם.

### 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 שמניע משגער 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, מושך באותו אופן שבו הוא בוצע במהלך ה-Three Way Handshake. כך לדוגמה, נאמר שהלכה והשתת הרימו קישור בצורה הבאה:



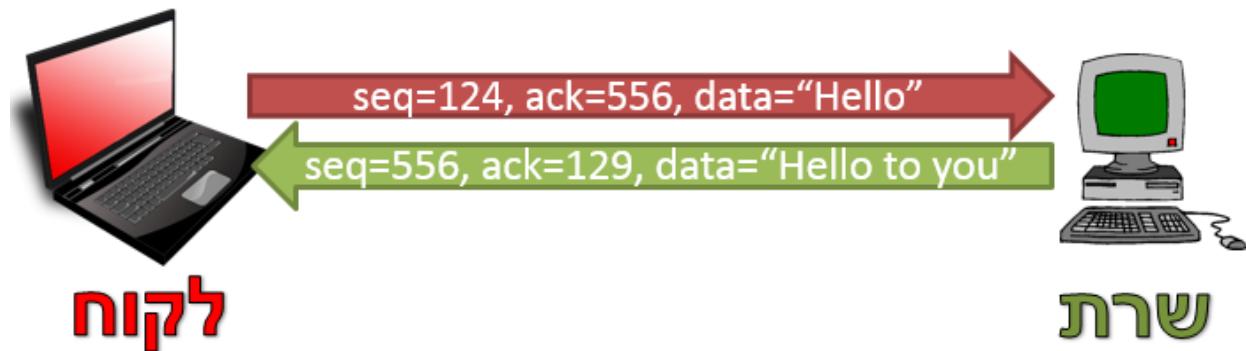
כעת, הלוקוט רוצה לשלוח הודעה נוספת, ובה המידע: "Hello". כיצד יראו המזהים? ובכן, שדה ה-Sequence Number יהיה 124, שכן זה הבית הבא שנשלח. בהודעה האחורונה שמוצגת בצייר לעיל, הלוקוט שלח ACK, אך הוא לא שלח כל מידע. שדה ה-ACK ישאר בעל הערך 556, שכן לא נשלח מידע חדש מהשרת:



כעת נאמר שהשרת 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 בחבילת האחורונה של הלקובות.

בשביל לחשב את הערך של שדה ה-ACK, علينا לקחת את ה-Sequence Number האחרון של הלקובות (והוא 124), ולהחבר אליו את אורך ההודעה שהוא שלח (האורך הוא 5, מכיוון שהוא האורך של המחרוזת "Hello"). אי-כך, ערך ה-ACK יהיה 129:



כעת הלקוב רוצה לשלוח לשרת ACK על החבילת שהתקבלת, אך להוסיף מסר - "?How are you".  
נaso לחשב בעצמכם: מה יהיה ערך ה-ACK-Sequene Number? מה יהיה ערך שדה ה-ACK?

ראשית נחשב את שדה ה-Sequence Number של ההודעה. הערך יהיה זהה לערך ה-ACK של החבילת הקודמת: קלומר 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 ערך 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 שלהם.

TCP אחראי גם לכך שהמידע יעבור מצד לצד בצורה יעילה עד כמה שניתן. לשם כך, למשל, TCP לא תמיד מחייב ל-ACK על מנת לשולח 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 מקטע<sup>44</sup>. כמו כן, TCP מנסה למנוע היוצרות של עומס על הרשת כולה. באם TCP מזיהה שיש עומס על הרשת, הוא פועל בכך לשולח פחות מידע ולאפשר לרשת "להתאושש".

ל-TCP יש גם שיפורים נוספים לאורק הזמן וקיימים בחלק מהIMPLEMENTATIONS, לא צריך לחכות לכך שלא יגיע ACK בצד לשולח חבילה מחדש. לעיתים, צד אחד של החיבור יכול להבין ששורה לו חבילה שלא הגיעה, ולהודיע על כך לשולח באמצעות חבילות מיוחדות בשם NAK.

על תפקידים אלו ועוד לא נרחב במסגרת ספר זה, אך אתם מוזמנים להרחיב אופקיכם ולקראן עליהם באינטרנט, ובחילק [עדדים להמשך](#) של פרק זה.

<sup>44</sup> גודל המידע שהמחשב מקבל מוקן לקבל נקרא גודל החלון. הוא דינامي, ונשלח בכל מקטע TCP בשדה Window Size. תהליך ניהול החלון הוא מורכב, ותוכלו להרחיב את הידע שלכם בסעיף [עדדים 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, זהור ושיר, להשיג מטרה דומה, אך בדרך אחרת.

את הפטון הוקדם 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 הלקווח יהיה כמו בתרגיל מודרך 6.18, אך בנוסף Three Way Handshake גם את צד השרת. מימוש Three Way Handshake גם את צד השרת:

- ישלח פקודות מסוג SYN לשרת, לפורט אקראי כלשהו ש"יבחר ע"י הלקווח (חשוב להמשך התרגיל). גם source port צריך להיות אקראי כפי שמקובל בתקשרות TCP.
- יצפה לקבלת SYN-ACK מהשרת. אם לא מתאפשרת פקטה זו, ישלח SYN בollowah כל זמן מוגדר.
- עם קבלת SYN-ACK, הלקווח ישלח את הודעת ACK שחותמת את התהילה. יש להקפיד על ערכי seq ו-ack נכונים, בהתאם לערכיהם 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 התהילה עליכם להיות מסוגלים למצוא את הפקודות שלוחתם הן בשרת והן בלקווח. 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
<hr/>						
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. במקרים אחרים, לא נוצר קשר אמיתי ברמת שכבת התעבורה וההודעות הבאות שיועברו בין השירות ללקווח 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 24601 Three Way Handshake) לפורט 24601.
- לשולח את המסר "c". ערך ה-Sequence Number יהיה 0. לשגור את הקישור.
- להרים קישור (באמצעות 1337 Three Way Handshake) לפורט 1337.
- לשולח את המסר "awesome". ערך ה-Sequence Number יהיה 20. לשגור את הקישור.
- להרים קישור (באמצעות 555 Three Way Handshake) לפורט 555.
- לשולח את המסר "connections are". ערך ה-Sequence Number יהיה 5. לשגור את הקישור.

שים לב: התקשרות איתה תיארנו לעיל אינה תקשורת TCP תקינה. לדוגמה, כאשר נשלח המסר "c", ערך Sequence Number אמור להיות מחושב בהתאם לערך h-ISN שהוגרל 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 לעיל, ובהתאם למספר החלקים שביקש המשתמש.

**דגשים**

- עליים למשתמש ב-Scapy כדי להסניף, לקבל ולשלוח את החבילות.
- את כתובות ה-IP של השירות אתכם יכולים בקוד שלכם באופן קבוע ולא לבקש אותה מהמשתמש.
- את הפורט עליים לקבוע באופן אקראי (בכל מקרה השירות לא יוכל להניח שידוע לו מספר הפורט).
- במידה שמספר החלקים שהמשתמש שלח גדול יותר מסהפר התווים שבהודעה, הציגו לו הודעה שגיאה.
- על מנת לבדוק את תרגיל זה, עליים למשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת<sup>45</sup>.

**חלק ב' - שירות שגורא מסר סודי**

כתבו סקריפט בשם `tcp_secret_message_receiver.py`. על הסקריפט לבצע את התהליך הבא:

- לחכות לפניות TCP (SEGMENT 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.

**דגשים**

- עליים למשתמש ב-Scapy כדי להסניף, לקבל ולשלוח את החבילות.
- בצדיה להדפיס את המסריהם שהתקבלו מהלקוחות, על השירות להבין מתי מסר אחד נגמר, ומסר אחר מתחילה. חשבו על דרך לעשות זאת. תוכלו לשנות את קוד הלוקוח לשם כך.
- בדקו את השירות באמצעות הלוקוח שתכתבם בחלק א'. וודאו כי אתם מצליחים לקרוא נכונה את המסריהם 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 הלוקוח והשירות שתכתבם בחלקם הקודמים של התרגיל, ויאפשר לשני לקוחות לתקשר זה עם זה כמו בצד'אט, באמצעות העברת מסרים סודיים.

---

<sup>45</sup> 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, וchlק מכך הסבירנו את המושג **פורט**. הבנו את מיקום השכבה במודל חמש השכבות, הכרנו את הכלים **netstat** ו**arp** תירגלו את השימוש בו. לאחר מכן הכרנו את המושגים **חיבור מבוסס קישור** ו**חיבור 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 UDP. לאחר מכן למדנו איך לכתוב לקווי ושרות משתמשים בפרוטוקול UDP באמצעות Sockets. כמו כן, למדנו לשЛОח 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 בכך לשומר על אמינותו. למדנו כיצד מרימים קישור ב-TCP באמצעות Three Way Handshake. צפינו ב-Wireshark כיצד נראות חבילות של TCP, בעת הקמת קישור ובכלל. בהמשך מישנו בעצמו Three Way Handshake באמצעות Scapy, והשתמשנו ביכולת זו בצד לגלות איזה שירותים פתוחים במחשב מרוחק. אז העמכונו בדרך בה נראה תקשורת לאחר שחיבור קם, והבנו כיצד מחושבים ערכיהם של שדות שונים בחבילות TCP. בסיום, הזכרנו בקצרה פקידי TCP עליהם לא העמכונו בפרק זה.

במהלך הפרק, הכרנו לעומק מספר בעיות אפשריות ברשות, ודריכים להתמודד עימן. להלן טבלה המסכםת את אתגרים ומנגנוני התמודדות אלו:

בעיה	מנגנון התמודדות	האם קיימ ב-TCP?	האם קיימ ב-UDP?
חביבה לא מגיעה ליעדה	שליחת acknowledgement על מידע שנשלח	כן	לא
חסוך לא נכון מתוך רצף בצד אחד	כלילת מספר סידורי עבור כל מידע שנשלח	כן	לא
באופן לא תקין	שימוש בChecksum	כן	כן

בפרק הבא, נמשיך ללמידה על מודל השכבות ונלמד להכיר את שכבת הרשות. נדבר על האתגרים הניצבים לפני שכבה זו, וכייד היא מתקשרת 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 מכמ 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 (מהדורה חמישית) מעת Wetherall, הפרק השישי מתיחס במלואו לשכבה התעבורה. באופן ספציפי, מומלץ לקרוא את החלקים:

- 6.5.4 - מספק מידע על ה-Header TCP, יכול להשלים את המידע השני [בנספח א' של פרק זה](#).
- 6.5.6 - מתאר על תהליך סיום תקשורת TCP, אשר לא סקרוño במהלך פרק זה.
- 6.5.8 - ניהול החולון 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.

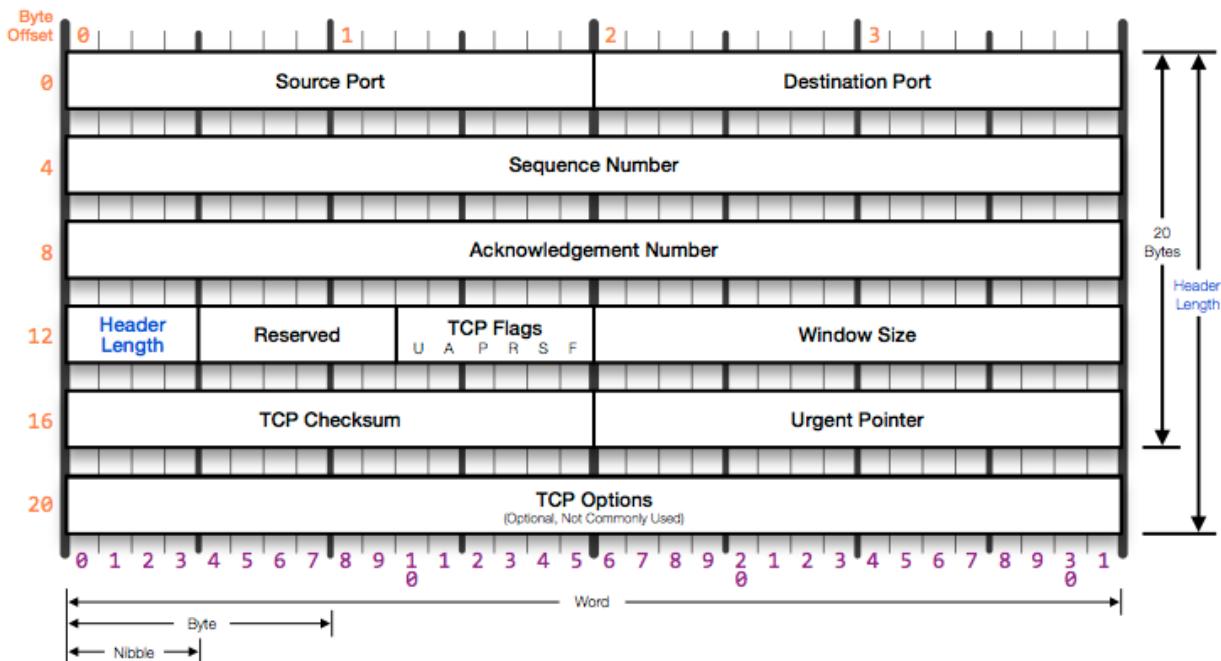
כמו כן, ניתן להרחיב את אופקיכם בפרק על TCP ו-UDP מתוך [The TCP/IP Guide](#), אותו ניתן למצוא בכתבota:

<http://goo.gl/GC69q4>

## נספח א' - TCP Header

נספח זה נועד כדי לתאר את כל השדות של ה-Header של TCP. לא חיוני להבין את כל השדות עד הסוף.  
מידע נוסף ניתן למצוא בכתובת: <http://goo.gl/CyVbvX>

## TCP Header

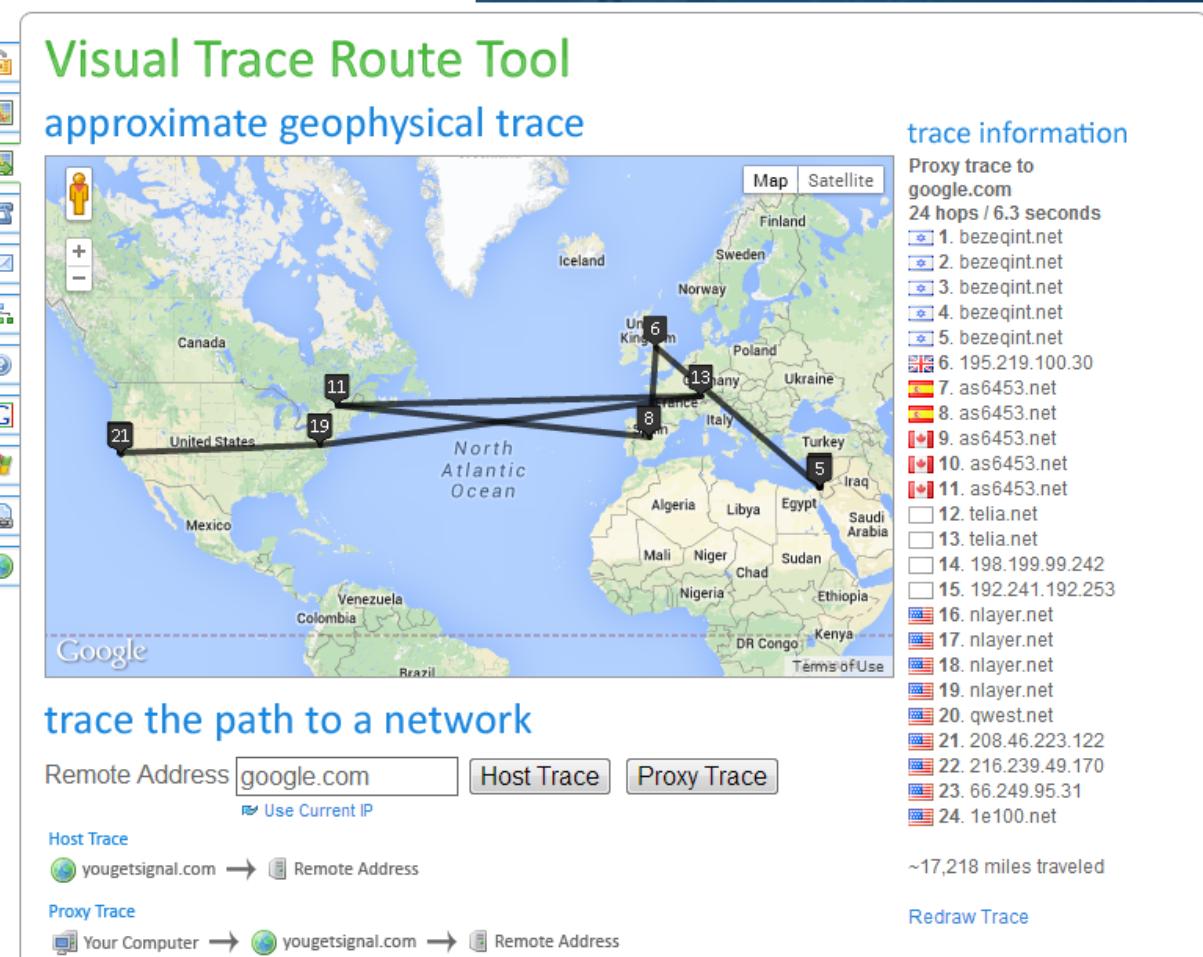


- Source Port - פורט המקור.
- Destination Port - פורט היעד.
- Sequence Number - מספר סידורי עוקב, המזהה את המקטע בתוך זרם המידע הכלול. כל Sequence Number מתאר בית אחד ברכף המידע.
- Acknowledgment Number - מתאר את הבית הבא שצפוי להתקבל.
- Header Length - מתאר את אורכו של החביליה, היוט שהוא עשוי לשנתנות מחייב להצלחה, 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 CAN יהיה 5 (מכיוון שמדובר ב-160 ביטים, שהם 5 פעמיים 32 ביטים). הערך 5 (שמתאר 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 וקשר 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 שופיע בשדה Acknowledgement Number.
  - P - דגל PSH - מבקש מערכת הפעלה להעביר את המידע ללא דיחוי. איןנו מרחיבים על נושא זה בספר זה.
  - S - דגל SYN - בקשה להקמת קישור.
  - F - דגל FIN - בקשה לסגירת קישור קיים.
- Window Size - גודל חלון השיליחה של TCP. מצין כמה מידע השולח מוכן לקבל ברגע זה.
- Checksum - מזודה 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 - שכבות 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 השירות מרוחק.

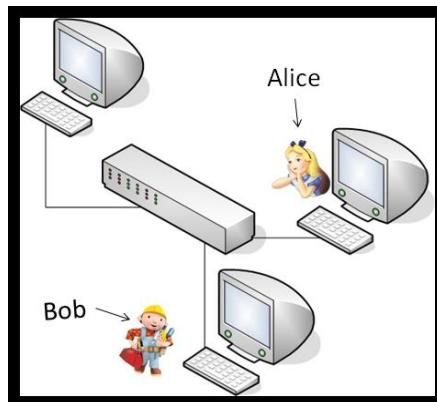
לשם כך, עליינו לענות על שאלות רבות ומרתקות - מהי תפקידה של שכבות הרשת? מהו פרוטוקול IP? מה היא כתובות IP? מה זה נתב? על כל שאלות אלו (ועוד) תוכלן לענות בעצמכם בתום פרק זה.

נתחיל מהשאלה הראונה:

## מה תפקידיה 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 מלחוש על רשותות ישנות, 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, שחוברו באמצעות כבילים וקופסאות קטנה.

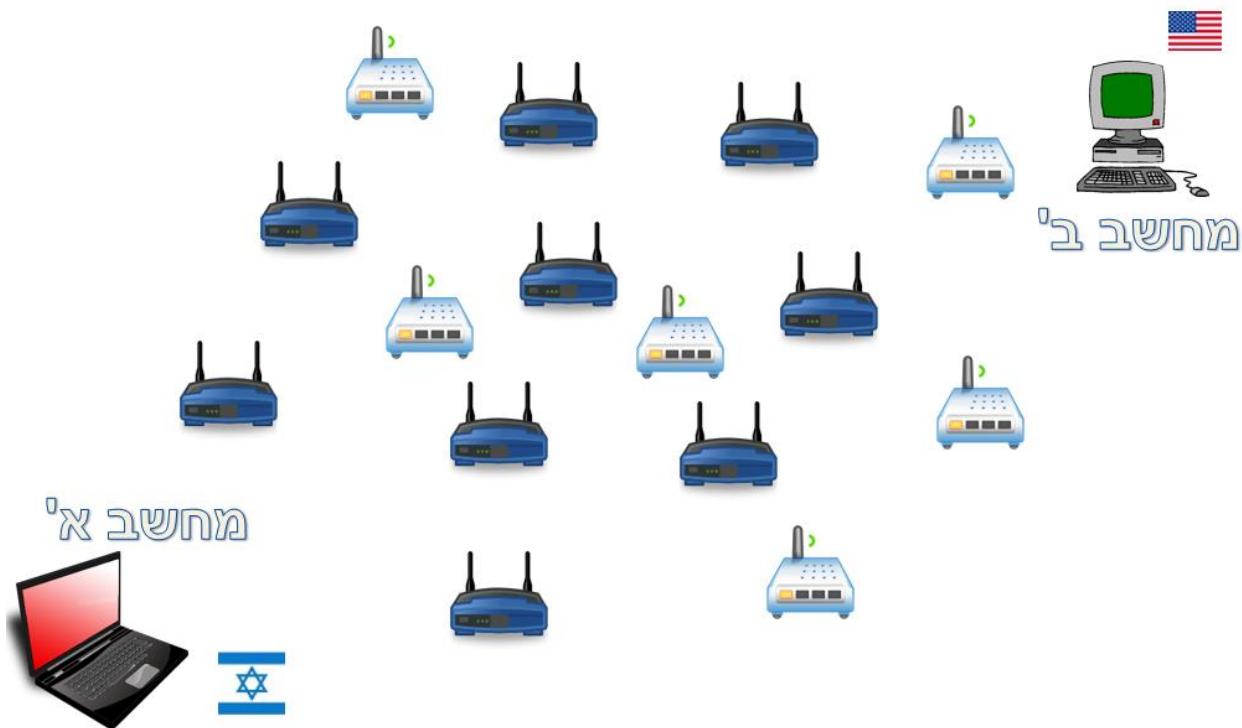


בפועל זהה, בו רשותות כוללות מספר מצומצם של מחשבים בלבד, קליחסית לגרום למיידע לעبور מצד לצד (בדוגמא שבצייר - מהמחשב של Bob אל המחשב של Alice). אך רשותות אלו אינן דומות כלל וכלל לרשותות שאנו מכירים היום! המחשב שלכם, שנמצא כאן בישראל, כמו גם הסמארטפון שלכם, מצויים באותו רשת כמו שרת Google שנמצא בארצות הברית, והוא רשת האינטרנט. על מנת להעביר מידע בין מחשבים<sup>46</sup> ברשת האינטרנט, علينا לעبور בדרך בהרבה רכיבים שונים.

---

<sup>46</sup> המושג "מחשבים" משומש כאן מטעני נוחות. למעשה, הדבר נכון夷 עבור כל ישות רשות - כמו נתבים או שרתים. שימוש לבשכאשר אנו מתיחסים ל"ישות" ברשות, אנו מתיחסים לקרים רשות מסוימים. כך למשל, מחשב עם שני קרטייסי רשות מייצג למעשה שתי "ישויות רשות" שונות.

הביתו בשרטוט הרשות הבא:



בشرطוט זה ניתן לראות את מחשב א', שנמצא בישראל, ומחשב ב', שנמצא בארצות הברית. עת, נניח והמשתמש במחשב א' רוצה לשלוח הודעה אל מחשב ב'.

שכבה התעבורת, עליה למדנו בפרק הקודם, מינהה כי אפשר להעביר חבילה מידע בודדת ממחשב א' למחשב ב'. שכבת הרשת היא האחראית לתהליך זה.



**מטרת שכבת הרשת היא להעביר חבילות מידע מיידן מישות<sup>47</sup> 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. כשהאנו מתיחסים ל"חבילות מידע" בשכבת הרשת, אנו קוראים להן בשם **Packets** **ובעברית - חבילות (או "פקטות")**. היות שרוב המידע שעובר באינטרנט מועבר באמצעות השכבה השלישי, אנו מכנים לרוב כל מידע כזה בשם "פקטה", כפי שראיתם לאורך הספר. בדוגמה לעיל, שכבת הרשת אחראית להעביר את חבילות המידע בין מחשב א' לבין מחשב ב'.

<sup>47</sup> למשל ממוחשב אחד לאחר או ממוחשב לשרת.

שימוש לב של נקודת קצה בפני עצמה מכירה את מבנה הרשת הכלול. למשל, 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 ומופתלת. בין השאר, שכבת הרשת עשויה להתמודד עם:

1. חומרות שונות - יתכן שבדרך בין ממחשב א' לממחשב ב' יהיו רכיבים שונים לחלוטין, כשאחד מהם הוא שרת עצם ואחד מהם הוא קופסא קטנה.
2. תקנים שונים - יתכן שהתקשרות בין ממחשב א' למעבור בלויין בחילל, לאחר מכן באמצעות כבל רשת סטנדרטי<sup>48</sup>, לאחר מכן באמצעות WiFi ובחזרה.

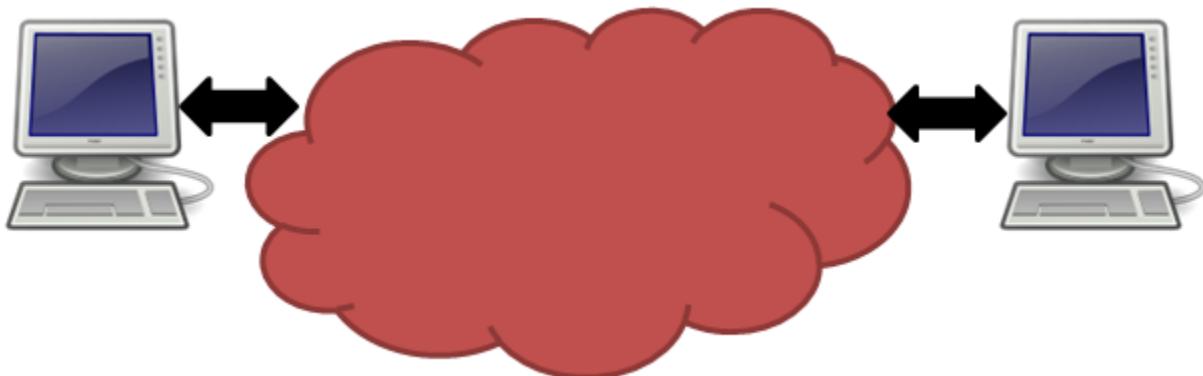
### מייקום שכבת הרשת במודל השכבות

שכבת הרשת היא השכבה השלישית במודל חמוץ השכבות.



### מה השירותים שsscבת הרשת מספקת לשכבה שמעליה?

sscבת הרשת מספקת לשכבת התעבורה, השכבה הרכיבית במודל השכבות, מודל של "ענן", שבו חבילות מידע מגיעה מצד אחד לצד שני. שכבת התעבורה אינה מודעת כלל למבנה הרשת המתוואר, ולמעשה מבחינה בה פשוט "רשת כלשהי" שמחברת בין ממחשב א' לממחשב ב'. "תמונה הרשת", מבחינה, נראה כך:



<sup>48</sup> הכוונה היא לכבל Ethernet, עליו נלמד בפרק שכבת הקו.

בצורה זו, שכבת הרשת מצלילה "להעלים" את הרשת מבחינה שכבת התעבורה מבקשת "שלחי לי חビלה מכאן לכאן" (למשל מחשב א' למחשב ב'), ו scavbet הרשת דואגת לכל התהילך ממש ואילך.



### מה השירותים ששכבת הרשת מקבלת מהשכבה שמתוחתיה?

שכבת הקו מספקת לשכבת הרשת ממשק להעברת מידע בין שתי יישויות מחוברות זו לזו באופן ישיר. באופן זה, שכבת הרשת לא צריכה לדאוג לסוגיות הקשורות לחיבור בין שתי תחנות. את scavbet הרשת לא מעוניין אם היישויות מחוברות בכבול, בלויין, או באמצעות יוני דואר. היא רק אחראית להבין מה המסלול האופטימלי. כמו ש-Waze רק אומרת לרכב באיזו דרך לעבור, ולא מסבירה לנוג שהוא צריך לתרדיק, ללחוץ על הגז או לאותה. בזה טיפול הנagger, או במקרה שלנו - scavbet הקו. על כל זאת, נלמד לעומק בפרק הבא.

## מסלולים ברשת



### תרגיל 7.1 מודרך - מי נמצא בדרך שלי?

עכשו נסו בעצמכם - כיצד תגלו מה הדרך 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). הנהם 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 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 שהחברה 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 נבין ממה נובעים שינויים אלו.

כעת, נסוו בעצמכם להשלים את הטבלה הבאה לפי הפלט של הפקודה **tracert** במחשביכם:

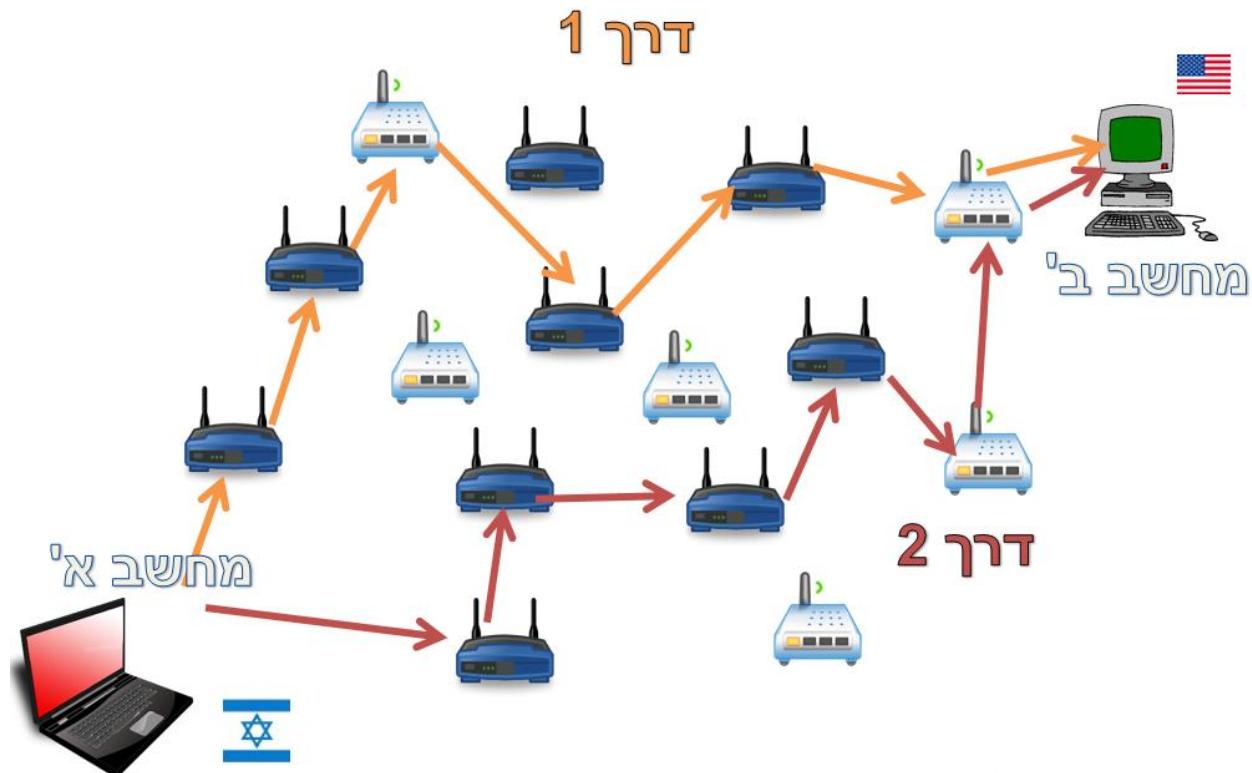
תחנות בדרך בין המחשב שלכם אל Facebook



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 מנקודת א' לנקודת ב'. הדבר דומה למציאת דרך נסעה ברכבת, למשל כמו שעשו האפליקציה Waze. על האפליקציה להבין מה הדרך שבה על הרכב (או במקרה שלנו - חבילת המידע) לעבור כדי להגיע מהמקורה אל היעד.

ນביט 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. כמובן שניתן לבחור בהרבה דרכים אחרות ואין מנעה לכך. על שכבות הרשת להחליט באיזה דרך להעביר כל חבילת 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 בציור, 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 להגיע 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 TEL אביב לירושלים חסומים כרגע, עדיף ש-Waze יגיד לנו להישאר בבית. כך גם צריכה שכבת הרשות לעשות: אם לא קיים כרגע אף חיבור בין מחשב א' למחשב ב', עליה להודיע למחשב א' שהיא 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.

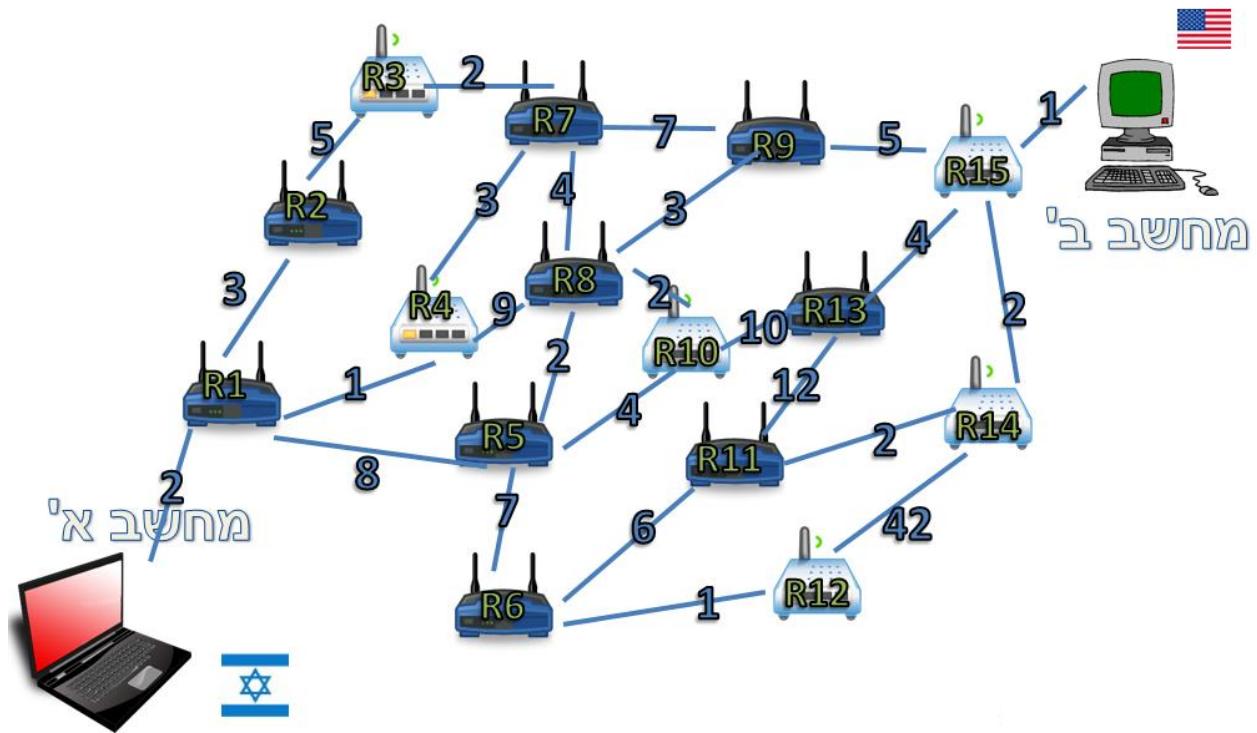
רביעית, באחריות שכבת הרשות להבין אילו דרכי אסورة. כפי שתיכון וכביש מסוים חסום כרגע, או שאנו אפשר לעبور בו בגלל סכנת מפולת, כך גם בעולם הרשות - ישנו נתיבים אשר לא ניתן לעبور בהם.



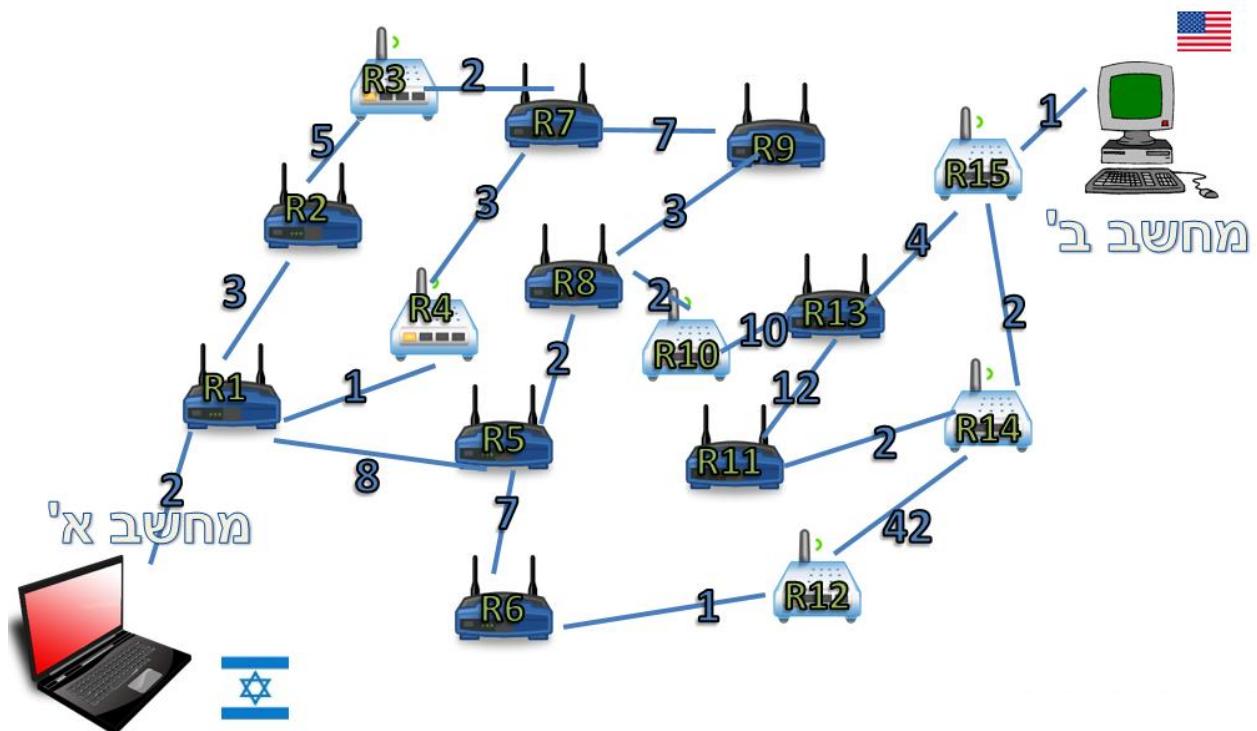
### **עשה זאת בעצמך - תכנן מסלול ניתוב**

לפניכם שובشرط 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 הקוו, סוג הקישור (קווי, אלחוטי) ועוד. כך למשל, שליחת חבילה ממוחשב א' אל R1 "עליה" 2, שליחת חבילה מ-R1 אל R2 "עליה" 3, וכן שליחת חבילה ממוחשב א' אל R2 דרך R1 "עליה" 5.

מצאו את הדרך ה"זולה" ביותר להגעה ממוחשב 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 החדש, ומיצאו שוב את הדרך ה"זולה" ביותר להגעה ממחשב א' למחשב ב':



האם הדרך השתניתה?

כך הבנו את אחת הסיבות שכבת הרשות משנה את החלטת הניתוב עבור כל פקטה (חבייה). בכל פעם גם מצב הרשות משתנה, ולכןו לנוהג בהתאם. בהקבלה לנסיעה ברכב, 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 מדוע צריך את שכבת הרשות, ולפחות 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, נכנסו ל-D (CMD) (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:2669: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. מה? כתובות ה-IP שלכם?



**מה זו כתובות IP?**

דיברנו במהלך הספר לא מעט על כתובות IP. כתע אנו יכולים לציין כי כתובות IP הן כתובות של שכבת הרשת - שכבה זו משתמשת בהן כדי לדעת מה הכתובת של היישויות השונות ברשת<sup>49</sup>. 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)<sup>50</sup>, ולכן תיראה צירוף של ארבעה מספרים, שכל אחד נع בין הערבים 0 ו-255. להלן דוגמא לכתובת IP:

192.168.2.5



**חשבו:** 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 הרשת<sup>51</sup>? למשל - האם היא שייכת לתלמיד משה או לתלמיד אהרון?

<sup>49</sup> 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.

<sup>50</sup> הדבר נכון כמובן רק עבור כתובות IP בגירסה 4 (IPv4), ולא עבור גירסאות אחרות (למשל IPv6). מטעני 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 שתחל 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

---

<sup>51</sup> להזכירכם, 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: 200.100.0.1, ומזהה הרשת שלו היה **200.100**. מכאן שמצוות הישות שלו ברשת הינו: **0.1**. נאמר ויש ברשת של מחשביל'ה ישות נוספת, למשל הדפסת של מחשביל'ה, הנקראת צפוי מדפסתלה. מזהה הישות 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 מצחה 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 הישות. במקרה ומזהה הרשת הוגדר כך, הרי שכל כתובת IP שמתחליה בערך 200 מייצגת כתובת באותה הרשת. כך למשל, הכתובת הבאה: **200.50.2.3**, נמצאת בכתובת של מחשביל'ה. זאת בגיןוד להגדירה הקודמת של כתובתו של מחשביל'ה, שבה כתובת היתה צריכה להתחילה ברכף הבתים 100.200 בכך 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 להסתכל לראשונה על חבילת IP.

בואו נעשה זאת ייחדי. פתחו את Wireshark והתחילו להסניף. השתמשו במסנן התצוגה (display filter) הבא: `."http.request"`

לאחר מכן, פתחו את הדפדפן האbove עליהם עלייכם, וגלשו אל הכתובת `www.google.com`. כתע, הפסיקו את ההסניפה. החילון שלכם אמור להראות בערך כך:

Realtek PCIe GBE Family Controller: \Device\NPF_{BA82DDC7-5015-4476-8CF7-12A4823707E4} [Wireshark 1.8.3]						
No.	Time	Source	Destination	Protocol	Length	Info
450	9.63187600	192.168.14.51	81.218.16.236	HTTP	70	GET / HTTP
464	9.81064400	192.168.14.51	173.194.113.146	HTTP	74	GET / HTTP

בריבוע **האDOI**, מוצג **display filter** בו השתמשTEM ב כדי לסקן בקשות HTTP, כפי שלמדנו בפרק [שכבות האפליקציה/תרגיל 4.1 מודרך - התנסות מעשית בתקשורת HTTP](#).  
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](http://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 שתתמשכו לקרוא את השורה הבאה.

ובכן, כתעת ברשוטנו כתובת ה-IP המלאה שלנו. עם זאת, כפי שלמדנו, הכתובת מחלוקת למזהה רשות ומזהה ישות (במקרה זה - מזהה המחשב שלנו בטוק הרשת). כיצד נדע מהם המזהאים?

לדוגמא, כיצד נדע האם המחשב בעל הכתובת 192.168.0.5 נמצאアイテנו באוותה הרשות? ככלומר - עליינו לדעת, מתוך כתובת ה-IP שلونו, מהו מזהה הרשות ומהו מזהה הישות. מכאן שעליינו להבין אילו מהabitיטים מגדרים את מזהה הרשות, ואילו מהabitיטים מגדרים את מזהה הישות.



### מהו מזהה הרשות שלי? מהו מזהה הישות?

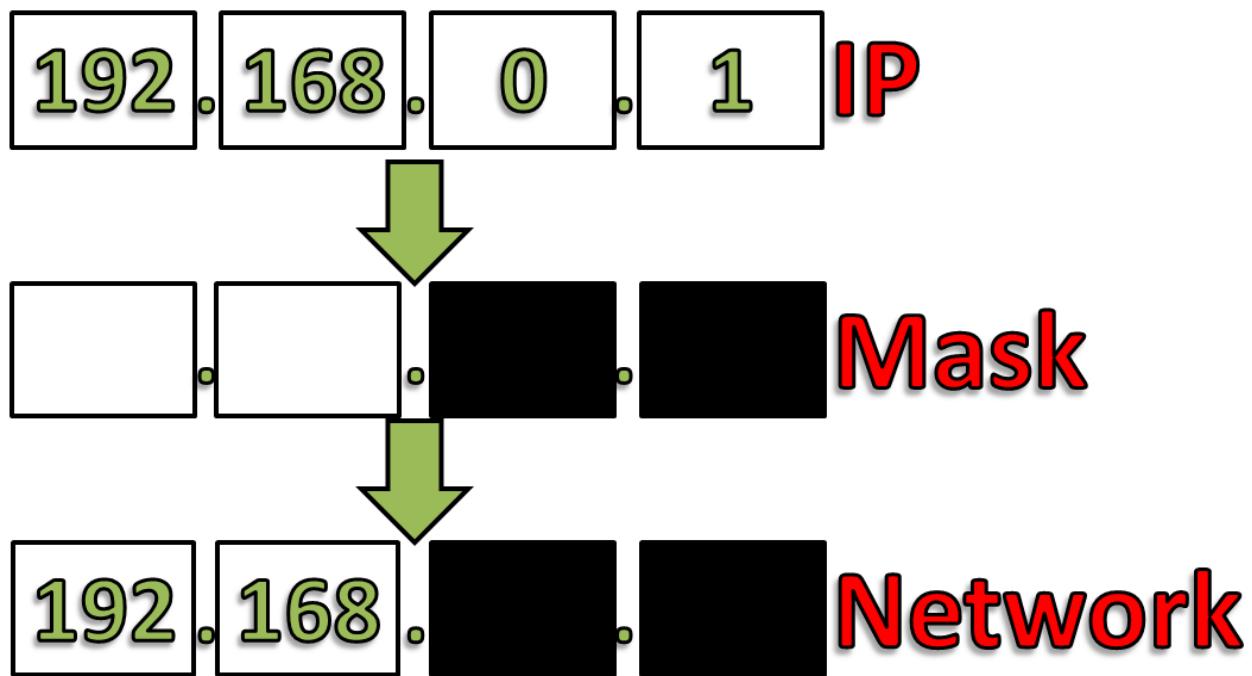
בכדי לענות על שאלה זו, עליינו ללמידה נוספת מונח חדש בשם **Subnet Mask** (טסיכת רשות). עבור כל כתובת IP, עליינו לדעת מהי ה-Subnet Mask שלה, על מנת לדעת מהו מזהה הרשות. ה-Subnet Mask מגדר כמה ביטים (bits) מתוך כתובת ה-IP מייצגים את מזהה הרשות.

נשתמש בדוגמא. הביטו בכתובת הבאה: 192.168.0.1. נאמר שטסיכת הרשות שלה מוגדרת כ-16 ביטים (או שני בתים<sup>52</sup>). מכאן ש-192.168.0.1 הינו מזהה הרשות, ו-0.1 הינו מזהה הישות. את מקורה זה ניתן להציג בדרכים שונות: 192.168.0.1/16 - הוספה "/"16" בסוף הכתובת מצינית שה-Subnet Mask מכיל 16 ביטים, ככלומר שהוא מזהה הרשות הירלאנטי.

דרך נוספת היא לציין שהכתובת היא 192.168.0.1 וה-Subnet Mask הינו 255.255.0.0 (16 הביטים הראשונים דולקים, ולכן הם מזהה הרשות. 16 הביטים הבאים כבויים, ולכן הם מזהה הישות).

<sup>52</sup> אם איןכם מרגישים 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.

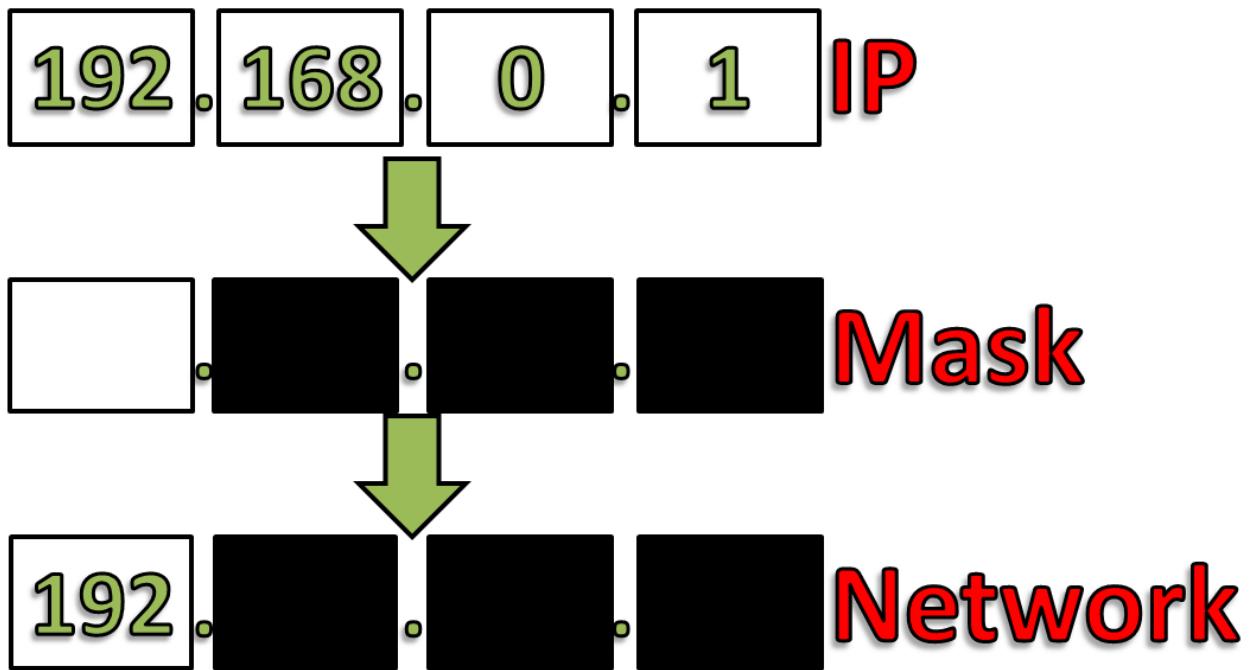
בדוגמה זו, המסכה "נראית" כהה:



כלומר, המסכה גורמת לנו להבין שרק 16 הביטים (שני הביטים הראשונים הם הרלבנטיים עבור מזזה הרשות, ובכך "מעליהם" את 16 הביטים (שני הביטים) הנוסתרים.

בואו נראה דוגמאות נוספות:

עבור הכתובת 192.168.0.1/8, המסכה נראהthus כהה:



כמובן, המסכה גורמת לנו להבין שרק 8 הביטים הראשונים (כלומר הבית הראשון) הם הרלוונטיים עבור מזהה הרשת, ובכך "מעלימה" את 24 הביטים (שלושת הבטים) הנדרטים.

**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 הרשת שונה (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 הרשות מוגדר באמצעות מספר ביטים (bytes) ולא בתים (bits), ולכן מסכת רשות 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 שהציגנו קודם:



**מהו מזהה הרשות שלי? מהו מזהה הישות?**

נסו לענות על כך בעצמכם.

נסתכל קצת בדוגמה שהציגנו 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 הרשות שלנו היא 255.255.255.0, לעומת שמאז הרשות מוגדר באמצעות 24 ביטים (או 3 בתים).

נחזיר לשאלת נוספת שהציגנו קודם: האם מחשב בעל כתובת 192.168.0.5 נמצא באותו הרשות?

התשובה היא - לא. זאת מכיוון שמאז הרשות שלנו מוגדר באמצעות 24 ביטים, והוא למעשה כתוב 192.168.14.5. הכתובת שהציגנו אינה מכילה את מזהה הרשות זה, ולכן המחשב בעל הכתובת זו אינו נמצא באותו הרשות 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 הרשות, וככתובת IP אחת שלא נמצא אתם באותה הרשות.



כעת הסניפו את הרשות שלכם ממש דקוט. הסתכלו בקובץ ההסנהה, ומצאו את כתובות ה-IP השונות שבו. אילו כתובות נמצאות ברשות שלכם? אילו כתובות לא?

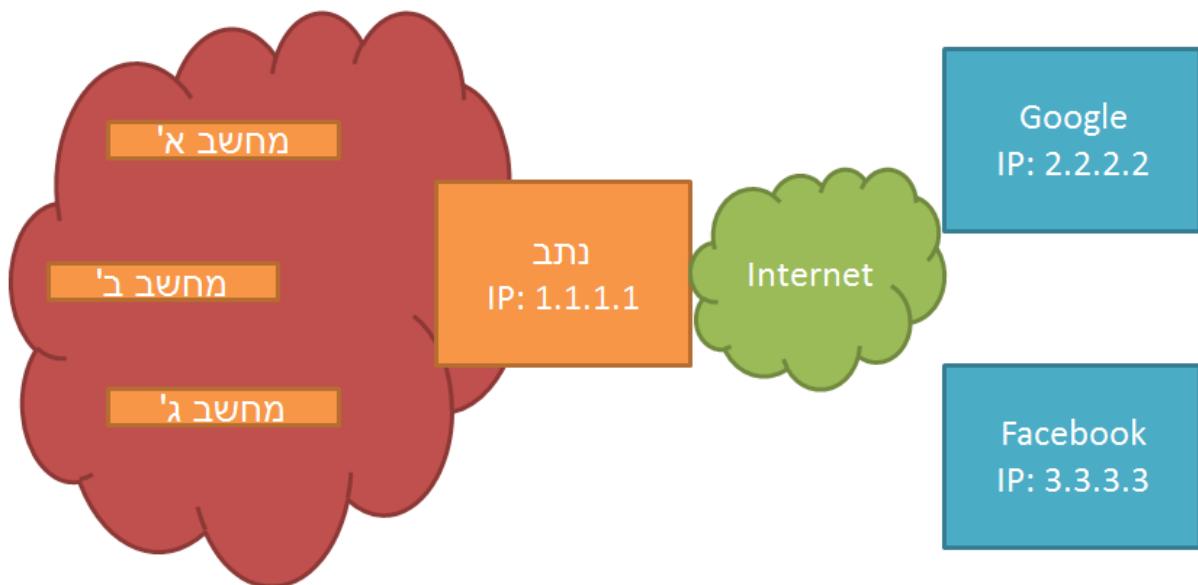
## כתובות IP מיוחדות

- ישן כמה כתובות IP ש모וגדרות כ"כתובות מיוחדות", ושווא להכיר אותן. כמו כן, נכיר רק חלק מהן:
- 255.255.255.255 - כתובות זו היא כתובות Broadcast. הכוונה היא שחבילה שנשלחת לכתובת זו מיועדת לכל הishiות ברשת. לדוגמה: ברשת בה יש ארבעה מחשבים: של דני, דינה, אורית ואורי, אם דני שולח חבילה לכתובת היעד "255.255.255.255", היא תגיע לדינה, אורית ואורי - כלומר לכל שאר המחשבים ברשת.
  - 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 שמהזה הרשת הינו בגודל 8 ביטים (או בית אחד), לכתובת 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 מיליארד<sup>53</sup> 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 ללקחות שלhn.<sup>54</sup>

.(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", ובה שלושה מחשבים. הרשות מחוברת, באמצעות הנטב בעל כתובת ה-IP 1.1.1.1, אל האינטרנט. בנוסף, ישנו השירותים של Facebook ו-Google אשר ירצו מחשבים ברשות לגשת. עד אשר החל השימוש ב-NAT, היה צריך לספק כתובת IP ייחודית לכל אחת מהשויות. כמו שמחשב א', מחשב ב' ומחשב ג', יזכו כל אחד לכתובת IP אמיתית ייחודית בעולם. דבר זה חיובי מהרבה בחינות, אך במקרה בה אין כבר כתובות IP לתת - הדבר לא נכון. אי לכך, נוצר הרעיון של NAT. לפי רעיון זה, כל השים 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 שמוגדרת לכתובת שכזו עתיד "לזרוק" אותה.

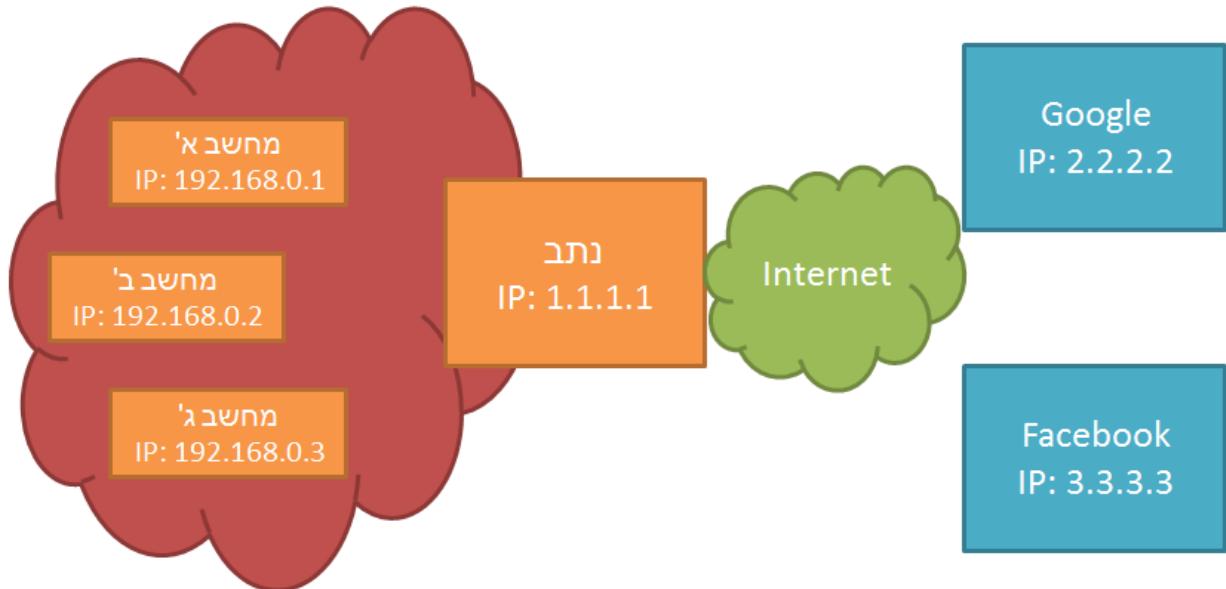
<sup>53</sup> מיליאון שבארבעה בתים (bytes) יש 32 ביטים (bits), שקל אחד מהם יכול להיות 0 או 1. אי לכך, מספר האפשרויות הינו  $2^{32}$ .

<sup>54</sup> לקריאה נוספת על תופעה זו - קראו בעמוד: [http://en.wikipedia.org/wiki/IPv4\\_address\\_exhaustion](http://en.wikipedia.org/wiki/IPv4_address_exhaustion)

לשם כך, הוגדרו שלושה טוויחים 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 RECEIPTS שונים ברחבי העולם?

באופן כללי, התהליך יעבוד כך:

בשלב הראשון, מחשב א' ישלח הודעה ממנו (כתובת המקור: 192.168.0.1) אל Google (כתובת היעד: 2.2.2.2). את החביליה הוא ישלח אל הנטב.

בשלב השני, הנטב <sup>55</sup> 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.

<sup>55</sup> מימוש ה-NAT לא חייב להתבצע אצל הנטב של הרשת. עם זאת, בפועל, ברוב המקרים הנטב הוא אכן זה שמשמש את ה-NAT.

בשלב השלישי, השרת של Google קיבל את החביליה. שימו לב: השרת 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 הרשת. בעת, Google יעד 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 מהשרת של Google. החביליה תכיל את כתובת המקור של השרת 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.

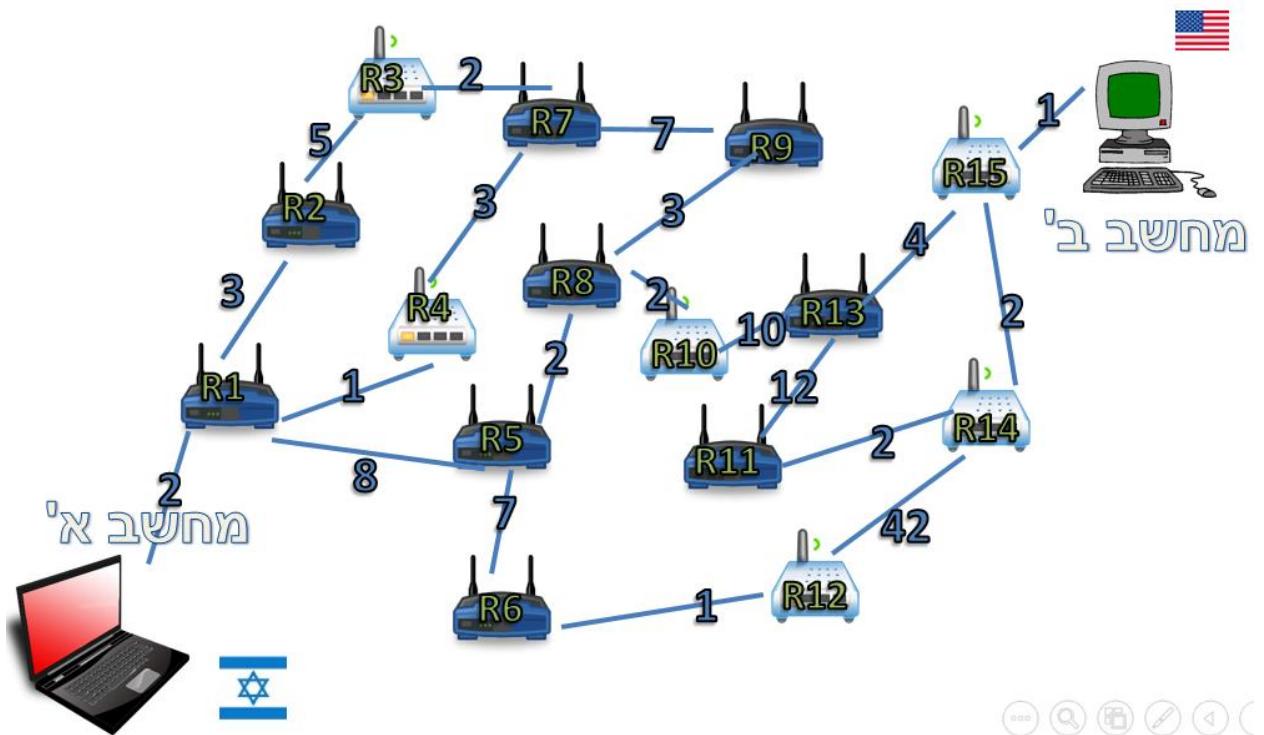
בצורה זו מצליחות ישות רשות האדומה שלנו לתקשר עם רכיבים מחוץ לאינטרנט, על אף שאין להם כתובת IP ייחודית. עם זאת, ישנה סוגיה לא פתורה - כיצד, בשלב הרביעי שהציגו, יידע האינטרנט שהחביליה שהגיעה מ-Google מיועדת למשה אל מחשב א' ולא למחשב ב'? כיצד הוא ידע לעשות זאת מידת שמחשב א' ומחשב ב' פנו שניהם, באותו הזמן 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](http://en.wikipedia.org/wiki/Network_address_translation).

## ניטוב

עכשו כשאנו יודעים כיצד בנויו 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, הרינו שתהיליך הניטוב הוא ההחלטה על הדרך בה הרכיב צריך לנסוע כדי להגיע מנקודת המוצא אל היעד. הרכיבים שמבצעים את רוב מלאכת הניטוב בעולם רשותות המחשבים, נקראים **נתבים**.

## נתב (Router)

**הנתב (Router)** הוא רכיב רשתī בשכבה שלישית. מטרתו היא לקשר בין מחשבים ורשתות ברמת ה-IP. נזכר בתרשיימי הרשות שהציגו קודם:



כל רכיב בדרך כן הימן למשה נתב, ומכאן גם נובע הייצוג שלו (R1) הוא למעשה קיצור עבור "Router 1". על הנתב לקבל כל חבילה, ולהחליט איך לנtab אותה הלאה בדרך הטובה ביותר.

צינו שנtab הוא רכיב של שכבת הרשת. מה המשמעות של קר? לכל נתב יש כתובת IP משלה<sup>56</sup>. מעבר לכך, הנתב "ambil" את שכבת הרשת - הוא יודע מה היא כתובת IP, מכיר את מבנה חבילת ה-IP, קורא את ה-Header (תחילית) של החבילה ומקבל 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.

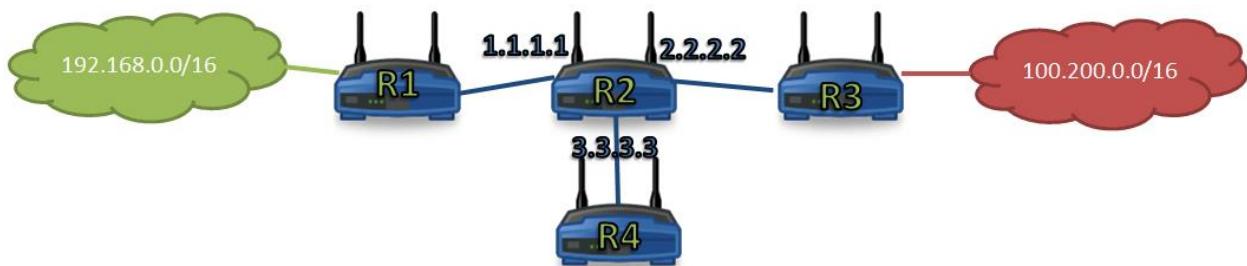
**יכן נתב מחליט לאן לנtab?**

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?

<sup>56</sup> ישנו גם נתבים בפרוטוקולים שאינם פרוטוקולי IP. עם זאת, מטעם הנוחות, נניח במהלך הספר שכל הנתבים הם נתבי IP.

על מנת להחליט כיצד לנתח את החבילה, לכל נטב יש **טבלת ניתוב (Routing Table)** משלו. טבלה זו מתחארת لأن יש להעביר כל חבילה שmagiuha אל הנטב. 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, הנטבים בדרכם צריכים לשנות את דעתם על הרשות ולהחליט על דרך חדשה לנטב.

נביט לדוגמה בתמונה הרשת הבאה (הערה: מדובר בתמונה רשת חלקית בלבד):



בתמונה הרשת זו ישנו ארבעה נטבים. הנטב R1 מחובר באופן ישיר לרשת בעל המזהה 192.168.0.0/16. הנטב R3 מחובר באופן ישיר לרשת בעל המזהה 100.200.0.0/16. הנטב R2 מחובר באופן ישיר לנטבים R1 ו-R3, וגם לנטב נוסף בשם R4.

שים לב שנטב 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 ברשת שלו ושל R1 הינה: 1.1.1.1
- כתובות ה-IP של R2 ברשת שלו ושל R3 הינה: 2.2.2.2
- כתובות ה-IP של R2 ברשת שלו ושל R4 הינה: 3.3.3.3

כל אחד מהנטבים עשוי להיות מחובר גם לנטבים נוספים, אך נועלם מכך לצורך החשב.

אל הנטב R2 מגיעה חבילה, כשתובעת היעד שלה היא: 100.200.5.8. כיצד ידע הנטב R2 אל מי להעביר את החבילה? אם ל-R1, ל-R3 או ל-R4?

על מנת לענות על שאלה זו, נסתכל בטבלת ניתוב של הנטב 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 הרשות 100.200.0.0 עם המסכה 255.255.0.0. כתה הוא שואל את עצמו:  
**"אם הכתובת 100.200.5.8 שיכת לרשת זה?"**

התשובה היא, כפי שלמדנו, כן. הרי מזאה הרשות הינו 100.200.100, 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 הממשק<sup>57</sup> R3, כולם אל R3. בטורו, 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 שיכת לרשת זה?"**

התשובה היא לא, זאת מכיוון שמזאה הרשות הינו 100.200.192.168.5.8 אינה עונה על מזאה זה. כתה, הנטב ימשיך לרשותה הבאה, רשומה מספר שניים, אשר מתארת את הרשות 192.168.0.0 עם המסכה 255.255.0.0. הנטב שוב שואל:  
**"אם הכתובת 192.168.5.8 שיכת לרשת זה?"**

התשובה היא כן, שכן מזאה הרשות הינו 192.168.5.8, 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, 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 לרשותת הניתוב الأخيرة שיש לו, כולם לרשותה מספר שלוש, המתארת את הרשות 100.200.0.0/16. בשלב זה אנו כבר מבינים שהכתובת לא נמצאת ברשות המתווארת ברשימתה זו, ולכן הנטב

---

<sup>57</sup> 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 מהרשת, ולכן הנטב יעבור אל הרשומה الأخيرة בטבלה.  
כעת, הנטב שואל את עצמו - "אם הכתובת 5.5.5.5 היא חלק מהרשת 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 טבלת ניתוב. 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 שנייתן לראות שני ממשקים (Interfaces) למחשב זה:

- הממשק בעל הכתובת "192.168.14.51". זוכרים שמצאנו את כתובת זו בתחילת הפרק? זהו כרטיס הרשת שלנו.
- הממשק בעל הכתובת "127.0.0.1". זוכרים שדיברנו על הכתובת זו קודם קודם תחת [סעיף כתובות מוחדרת](#)? חבילות שנשלחות לממשק זה לא באמת יגיעו לכרטיס הרשת, אלא "ישארו במחשב".

שימוש לב לרשותה הראשונה בטבלה: אותה רשותה שעלייה תתאמת 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 הניתוב, 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 הניתוב 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.

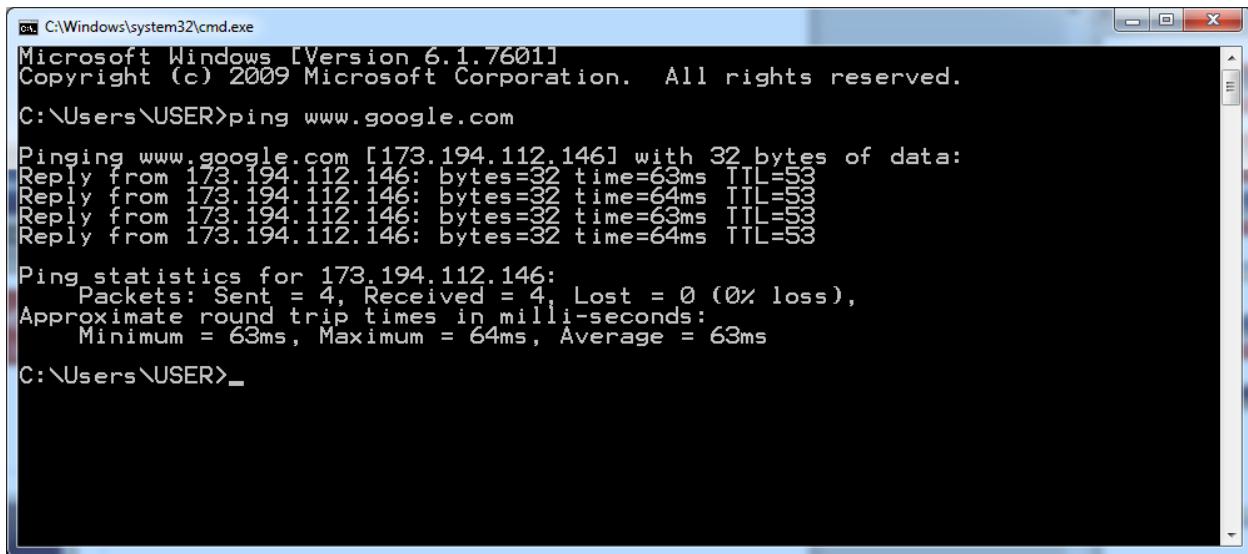
הכלי המוכר ביותר המשמש בפרוטוקול ICMP הוא הכלי **ping**, אשר פגשנו מספר פעמים לאורך הספר. כלי זה נועד בכדי להבין האם ישות מסוימת "למעלה" - כלומר דילוקת, עובדת ומגיבת לשילוח הודעות אליה. לחופין, הוא יכול לוודא שיש תקשורת מהמחשב שלנו אנו מרים את הפקודה אל הרשת אליה הוא מחובר.



### 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", שלוחה באופן בירור מחדל ארבע הודעות ומספר לנו כמה מהן הגיעו ליעדנו ונענו, ובאיזה מהירות.



מה, לא הסנפתם כשהרצתם קודם לכן את **ping**? ובכן, אתם אכן עדין חדשים בעולם הרשתות. פתחו עתה את Wireshark, הסניפו והריצזו שוב את פקודת **ping** שהרצתם קודם. אל תשחחו להשתמש ב-filter בכך לسان את הפקחות הללו רלבנטיות:

Filter: icmp							Expression...	Clear	Apply	Save
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]

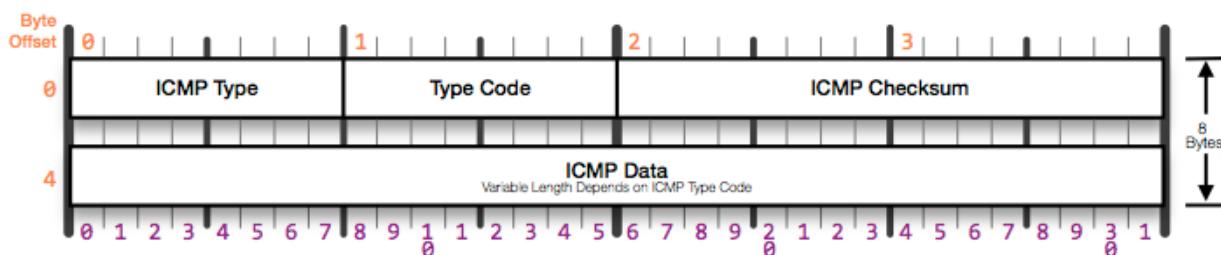
0000 00 0c c3 a5 16 63 d4 be d9 d6 0c 2a 08 00 45 00 . . . . c . . . . . E .
0010 00 3c 01 b4 00 00 80 01 4a dd c0 a8 0e 33 ad c2 . < . . . J . . . 3 . .
0020 71 92 08 00 4c d1 00 01 00 8a 61 62 63 64 65 66 q . . L . . . abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcde fg hi

```

נשים לב למודל השכבות:

- בשכבה השנייה, ניתן לראות את השימוש ב프וטוקול Ethernet<sup>58</sup>. דבר זה מלמד כי כרטיס הרשות ממנו נשלחה החבילה הוא כרטיס מסוג Ethernet<sup>59</sup>.
- בשכבה השלישית, י箇ו שימוש בפרוטוקול IP. כתובות המקור היא הכתובת של המחשב שלוח את ה- ping, וכתובות היעד היא הכתובת של Google.
- לאחר מכן, בהתאם לשכבה השלישית, י箇ו פרוטוקול ICMP.

cut נבחן את מבנה ה-Header של חבילת ICMP:



ה-Header של חבילת ICMP הוא לעולם בגודל קבוע של 8 בתים:

- בית 0<sup>60</sup> - Type: כולל את סוג החבילה. ישנו סוגים שונים של חבילות ICMP. בקרוב נראה דוגמה.
- בית 1 - Code: זה למעשה תת-סוג של Type 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.<sup>61</sup>.

<sup>58</sup> על שכבה השנייה בכלל, ופרוטוקול Ethernet בפרט, נרחיב בפרק הבא.

<sup>59</sup> יתקן שבמחשב שלכם תראו שכבה שנייה אחרת, בהתאם לכרטיס הרשות שלכם. למשל, יתקן ותראו כי מדובר בשכבה WiFi, באם אתם מחוברים באמצעות כרטיס רשת WiFi.

<sup>60</sup> שימו לב לכך שהבית הראשון הוא תמיד בית באינדקס 0. הבית השני הוא בית באינדקס 1, וכך הלאה.

<sup>61</sup> להזכירם, למדנו על [Checksum](#) בפרק [Checksum מה זה](#).

- בתיים 7-4 - כל השאר: ערך זהה יהיה שונה בהתאם לסוג החבילה, שהוגדר ביד' הบทים **Type**-**Code**-**Checksum**.

נחזיר לחבילה שלחנו ל-Google וنبיט 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: 173.194.125.121
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  wahcdefg hi

```

- **ב-**Type**** - אנו רואים את הבית הראשון, הלווא הוא **Type**, סוג החבילה. מדובר בסוג 8, כמו ש-
- **Wireshark** מועל בטובו להגיד לנו - מדובר ב-**Echo Request**. זאת ועוד, Wireshark מגיל לעשנות ואף מתאר בסוגרים כי זו חבילה שקשורה ל-ping. השימוש ב-ping כה נפוץ, עד שabitיות מסווג 8 (שהוגדרו במקור כ-"Echo Request") נקראות לעיתים "Ping Request".
- **ב-**Code**** - אנו רואים את הבית השני, **Code**. עבור **Type** מסווג 8 (חbillת **Echo Request**), שדה ה-**Code** תמיד מכיל את הערך 0.
- **ב-**Checksum**** - הบทים השלישי והרביעי, המציגנים את ה-**Checksum**. זהו אותו חישוב שמתבצע במחשב ששולח את ההודעה, כמו גם במחשב שמקבל את ההודעה. אם התוצאה היא אותה התוצאה - אין שגיאה בחבילה.
- **ב-**CRC**** - אלו מזחים ייחודיים להודעת Echo. הלוקוט, אשר שולח את בקשת ה-ping, יכול להשתמש במזחים אלו כדי לזרוח את חbillת הבקשה שלו כשהוא שולח מספרabitיות. כמובן, הוא יכול לציין כאן את הערך "1", וב-**Chbillת הבקשה** הבהא את הערך "2". כשהשרת יענה, הוא יgive לכל החbillה עם המספר שלו. מבולבלים? אל דאגה, המשיכו לעקוב אחר הדוגמה.
- **ב-**Length**** - ישים ה"מידע" של Chbillת **Echo Request**. בשימוש ב-Windows, כפי שתם יכולים לראות, נשלח פשוט כל ה-ABC האנגלאי עד האות 'w', ואז שוב מהאות 'a' ועד 'z'.



**שאלה 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	.....*	....c..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							wabcde	fg 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** - ישרו ה"מידע" של חבילת Echo Reply. השירות (במקרה שלנו, Google) חייב להעתיק את אותו המידע שניית בחלוקת 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 כיצד שימוש נוסף בכלים Wireshark, מאפשר 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 כדי<sup>62</sup>.  
לאחר שניסיתם 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, כשמייה יש ICMP. עשה זאת כך:

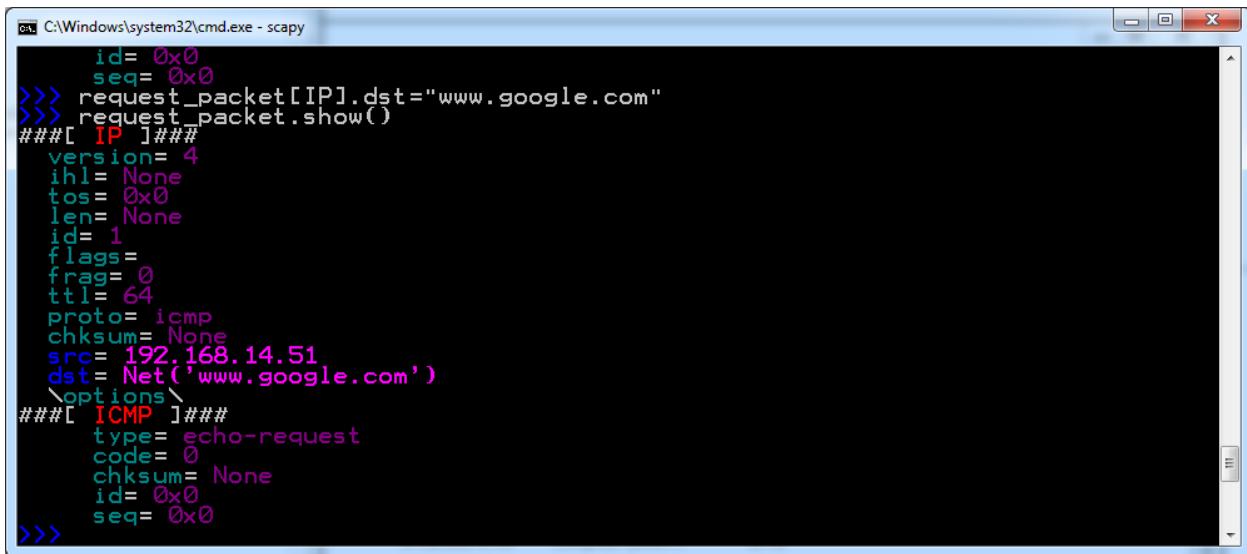
```
>>> request_packet = IP()/ICMP()
```

כעת, נביט בחבילה שנוצרה:

```
C:\Windows\system32\cmd.exe - scapy
>>> 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
chksum= 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 הכתובת של "www.google.com". נסו לשנות זאת, ובדקו שהחbillah אכן השתנתה. ניתן לעשות זאת כך:

<sup>62</sup> כפי ששמתתם לב לאורך הספר, הסניפה היא פעולה אשר אנו מבצעים באופן שוטף וועזרת לנו במספר תחומים - תוך כדי כתיבת קוד, במהלך הרצת כלי קיימ (כגון Ping), ובמקרים נוספים. זכרו להשתמש בכללי עצמותי 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 לדוגמה:

```
>>> request_packet[IP].dst="173.194.113.178"
```

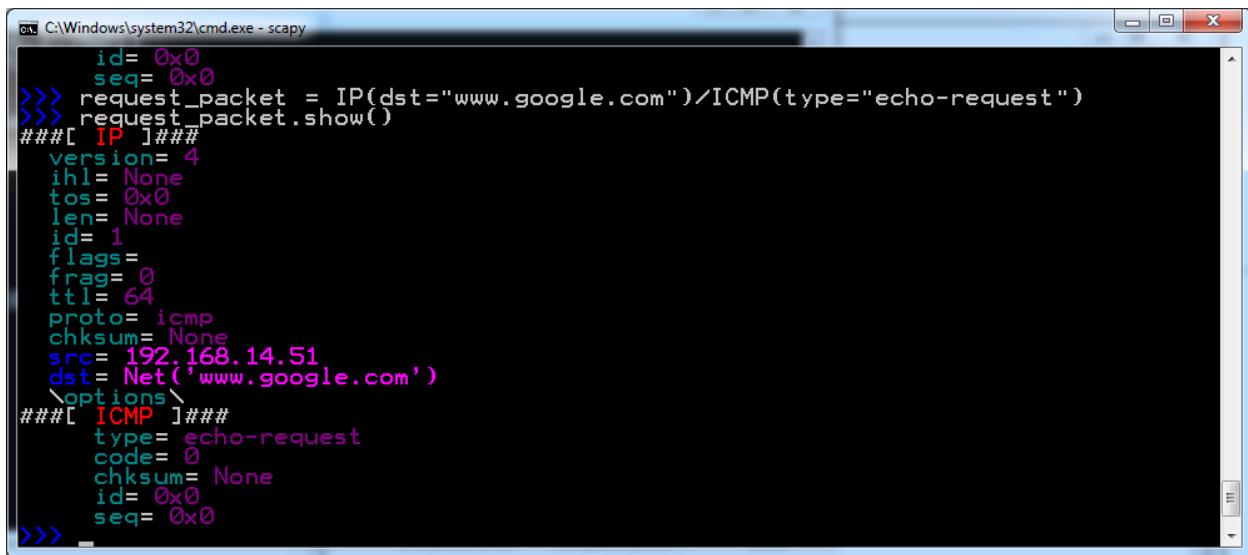
כעת יש לנו חבילת IP, שכותבת המקור שלה היא המחשב שלנו, וכותבת היעד שלה היא "www.google.com" החביבה כוללת גם Header של ICMP. אם נבחן את השדות, נראה כי Type מכיל "echo-request". מכאן ש- Scapy מוחש שם אנו מבקשים ליצור חבילה ICMP, אנו ככל הנראה רוצים חבילה מסווג Ping. מהר לנו!

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 ולוזדה שאכן נוצרת חבילה מסווג Echo Request, 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
chksum= 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 כתובת מקור, וב-0 בתור שדה ה-**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, שakan 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	00	.....c..	....*..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	00	00	00	00	00	00	q.....	...

cutת הסתכלו בחבילת התשובה. ציינו קודם לכך שהשרת (במקרה זה "www.google.com") צריך להשתמש באופןם מזהים שנשלחו בבקשתו. במקרה שלנו, התשובה צריכה לכלול את הערך 0 בכל השדות: Identifier (BE) ,Sequence Number(LE) ,Sequence Number (BE) ,Identifier (LE)

הצלחנו לראות את בקשת התשובה! אך עשוינו 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.
>>>

```

cutת, נשתמש בפונקציה שמאפשרת גם לקבל תשובה: **sr1**. את הפונקציה החזינו לראשונה בפרק [שכבות](#) [העברית/תרגיל 6.11 מודרך - קבלת תשובה לשאלות DNS באמצעות Scapy](#). נשתמש בפונקציה זו לצורך הבאה:

```
>>> response_packet = sr1(request_packet)
```

cutת Scapy ישלח את החביליה, ויככה לתשובה. כאשר התשובה תחזור, היא תשמור במשתנה **response\_packet**. בדקו זאת על ידי סוטכלות בתשובה:

עתה הגיעו העת לבחון הנחות שהיו לנו קודם לכן. טענו שהשתתף יזכה כל מידע שנשלח לו בחבילת התשובה. נסו לשלוח אל השתתף חבילת Echo Request, עם המזג "Cyber Bagrut is cool!". אם תצליחו, אתם צפויים לקבל את המידע הזה מהשתתף.

```
>>> request_packet = IP(dst="www.google.com")/ICMP(type="echo-request")/"Cyber Bagrut is cool!"
```

www.wireshark.org - www.wireshark.org "www.google.com" w/wireshark Wireshark e hanno

Filter: icmp

No.	Time	Source	Destination	Protocol	Length	Info
122	64.5103410	192.168.14.51	173.194.113.16	ICMP	63	Echo (ping)
123	64.5732090	173.194.113.16	192.168.14.51	ICMP	63	Echo (ping)

Frame 123: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface br0  
 Ethernet II, Src: Bewan\_a5:16:63 (00:0c:c3:a5:16:63), Dst: Dell\_d6:0c:2a (00:0c:2a:d6:0c:2a)  
 Internet Protocol Version 4, Src: 173.194.113.16 (173.194.113.16), Dst: 192.168.14.51  
 Internet Control Message Protocol  
 Type: 0 (Echo (ping) reply)  
 Code: 0  
 Checksum: 0x4153 [correct]  
 Identifier (BE): 0 (0x0000)  
 Identifier (LE): 0 (0x0000)  
 Sequence number (BE): 0 (0x0000)  
 Sequence number (LE): 0 (0x0000)  
[Response To: 122]  
 [Response Time: 62.868 ms]  
 Data (21 bytes)  
 Data: 43796265722042616772757420697320636f6f6c21  
 [Length: 21]

0000	d4	be	d9	d6	0c	2a	00	0c	c3	a5	16	63	08	00	45	b8	.....*	..C..E.
0010	00	31	ae	05	00	00	37	01	e7	60	ad	c2	71	10	c0	a8	.1....7.	...q...
0020	0e	33	00	00	41	53	00	00	00	00	43	79	62	65	72	20	.3..AS..	Cyber
0030	42	61	67	72	75	74	20	69	73	20	63	6f	6f	6c	21		Bagrut	i s cool!

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 סקירהפיטים 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 ומビינים איזו תשובה שייכת לאיזו שאלה.
3. כתבו סקירהפיט אשר ניתן להריץ משורט הפקודה. על הסקירהפיט לקבל כפרמטר כתובת של שרת, לשЛОח אליו 4 חבילות Echo Request, ולכתוב למסך כמה תשנות 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 `recv`.



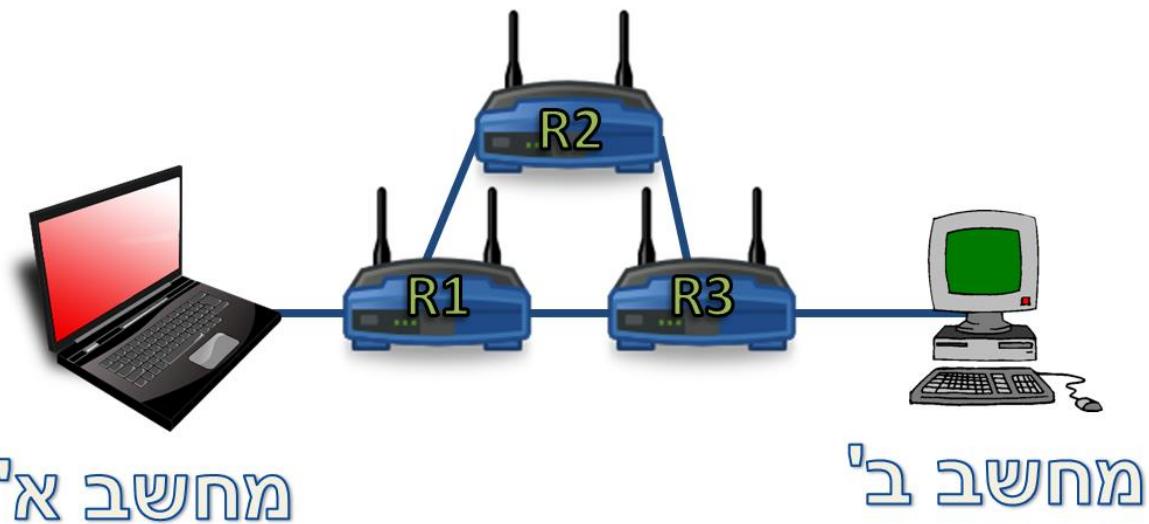
### איך Traceroute עובד?

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 כל'i כה מגניב בעצמכם. אתם יכולים להירגע ולשtotות כו' מים בטרם תמשיכו.

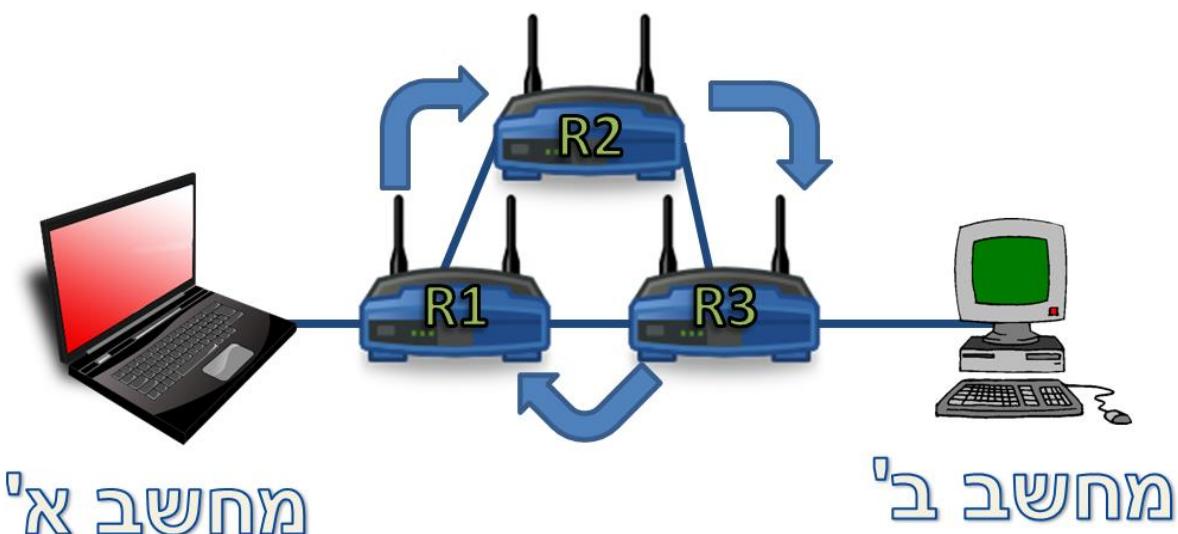
ראשית, נלמד על אחד השדות ב-Header של IP, שנקרא **Time To Live**, או בקיצור - **TTL**. השדה הוא באורך בית אחד, כלומר שהערך שנייתן בו יכול להיות בין 0 ל-255. על מנת להבין את השדה, נתחיל מלהסביר את הצורך בו.

הסתכלו בתמונה הרשות הבאה:



נאמר שמחשב א' שלוח חבילה למחשב ב'. החבילה הגיעו לראשית 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 הגיעו 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 שווידאי הבחנתם, החבילה "לכודה" הגיעו בין הנטבים R2, R1 ו-R3, ותמשך להיות מועברת בתוך ה"lolaha" שנוצרה:



כעת חשבו - איזה נזק נגרם כתוצאה מכך שהחbillה הזו נשארת "תקועה" ולא הגיע אל יעדה?

כמובן, מחשב ב' לא יקבל את הchnila שיעודה לו. אך מעבר לכך, הchnila זו יוצרת עומס על הרשות. שלושת הנתבים - R1, R2 ו-R3, ממשיכים לعباد אותה בכל פעם מחדש. הם מקדישים לה זמן ומשאבים שהיו יכולים להיות מוקדשים לחבילות אחרות. כך גם הקישוריהם בהם הchnila מועברת (הקשר בין R1 ל-R2, הקשר בין R2 ל-R3 והקשר בין R3 ל-R1), עמוסים יותר כיון שהchnila 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 הchnila יכולה להיות מועברת הלאה. נשתמש בדוגמה. נאמר שערק ה-TTL הראשון של הchnila אותה: TTL: שלח מחשב א', הוא 4. את ערך TTL זה קבע מחשב א'. נבחן את מסלולו של הchnila, וכן את ערך השדה TTL: • הchnila 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 בחנילה.

- הchnila 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 הchnila. הוא בוחן את ערך ה-TTL, והוא שהוא 2. במידה שהוא עבר אותו אל מחשב ב' - תהליך השילחה הסטטי, הchnila הגיעה ליעדה והכל בסדר. אך מכיוון שלא כך המצב, הנתב מוריד את ערך ה-TTL, ועביר את הchnila לנtab R1, כשער ה- $2 - 1 = 1$ .
- כעת הנתב R1 מסתכל על הchnila. 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 הchnila הלאה - והוא משミニ את הchnila!

כך למעשה מנגנון ה-TTL מנע מהchnila להישאר "תקועה" לנצח. עם זאת, במקרה זה, מן הראוי להודיע למחשב א' שהchnila שלו לא הגיעו ליעדה. לשם כך, נוצרה הודעת ICMP בשם Time-to-live exceeded. כאשר הנתב R1 מבין שעליו להשמי את הchnila היה שערך ה-TTL שלה נמוך מדי, הוא ישלח אל מחשב א' הודעה מסוג Time-to-live exceeded, כדי לידע אותו על כך.

שימוש לב לשימוש שנוכל לעשות בשדה זה: נניח כי אין בעיות ניתוב וחנילה יכולה להגיע מחשב ב', כשהיא עוברת בדרך הנתב R1, לאחר מכן בנתב 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 הchnila. אם נשלח מחשב א' למחשב ב' חבית IP עם ערך 2 = TTL, היא צפוייה להגיע לנtab R2 שיגיד לנו שהוא לא יכול להעביר את הchnila 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 לעשות 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 ה-TTL ישנו הערך 1. על מנת להימנע לימוש של הכלי **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
>>> tracert_packet = IP ttl=1, dst="www.google.com")/ICMP()
>>> send(tracert_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  ....*... .c..E.
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..... .E.....
0030 00 00 01 01 cd 11 c0 a8 0e 33 ad c2 70 31 08 00  ..... .3..p1...
0040 f7 ff 00 00 00 00 00 00 ......


```

נבחן בדgesים הבאים:

- באדום** - מסומנת כתובות השולח של החבילה. זהו למשה הנטב שקיבל את חבילת-h-  
.Time-to-live exceeded exceeded
- שאנו שולחנו, הבין שהוא לא יכול להעביר אותה הלאה, ושלח את חבילת-h-  
במילים אחרות, זורי הכתובת של הנטב הראשון בין לBIN "www.google.com"
- בכחול** - אלו השדות המאפיינים חבילה מסווג ICMP, כאשר שדה ה-h-  
Code קיביל את הערך 0. ניתן לראות שתחת שדה Type יש את הערך 11, ותחת השדה  
יש את הערך 0.
- בכתום** - זהו למשה העתקה של חבילה המקורי, כלומר חבילה שאנו שולחנו אל  
Header בערך 192.168.14.1 העתיק אותה ושלח לנו אותה לאחר ה-h-  
.ICMP

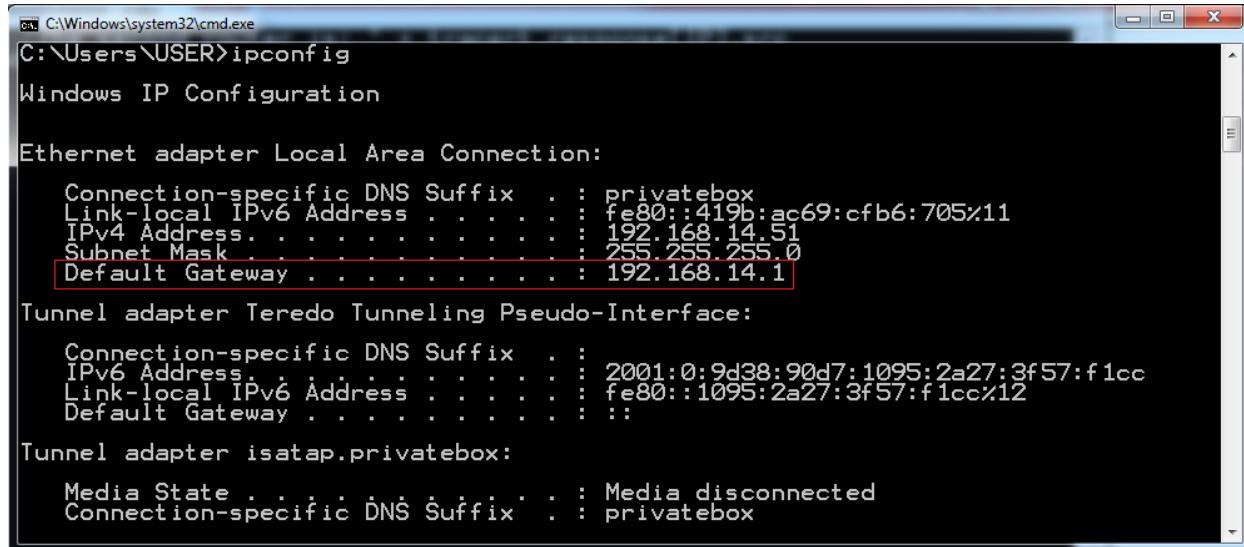
ובכן, הצלחנו למצוא את הכתובת של הנטב הראשון! עם זאת, לא עשינו זאת באופן תכונתי. נסו לשפר את הקוד  
שלכם כך שימצא את הכתובת של הנטב הראשון באופן תכונתי.  
ניתן לעשות זאת כך:

```

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** שלו.  
בכדי לוודא זאת, נוכל להריץ את פקודה **ipconfig** אוטה הכרנו קודם לכן:



```
C:\Windows\system32\cmd.exe
C:\Users\USER>ipconfig
Windows IP Configuration

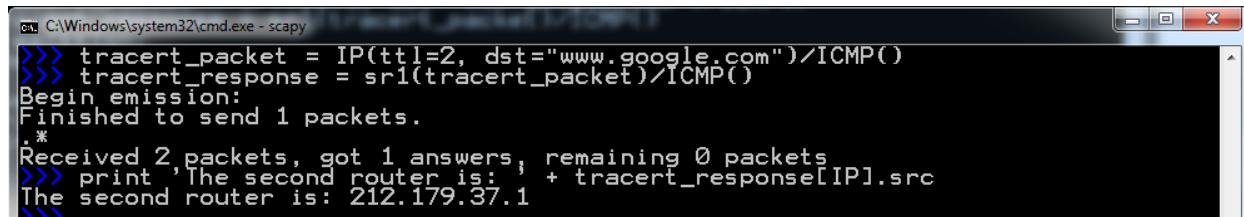
Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix . : privatebox
  Link-local IPv6 Address . . . . . fe80::419b:ac69:cfb6%11
  IPv4 Address . . . . . 192.168.14.51
  Subnet Mask . . . . . 255.255.255.0
  Default Gateway . . . . . 192.168.14.1

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
>>>
```

הסנה גם בהסנה:

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)

הסנה כוללת ארבע פקודות:

1. פקעת ה-**Echo request** שלחנו אל "www.google.com", כאשר שדה ה-TTL מכיל את הערך 1.
2. תשובה מהתוב הראשון בדרך מהמחשב שלנו אל "www.google.com".
3. פקעת ה-**Echo request** שלחנו אל "www.google.com", כאשר שדה ה-TTL מכיל את הערך 2.
4. תשובה מהתוב השני בדרך מהמחשב שלנו אל "www.google.com".

מcean שرك הסתכלות בהסנפה יכולה להראות לנו את רשימת הנטבים בין לBIN "www.google.com". בעת, נשתמש בכל tracert כדי למצוא את שני הנטבים הראשונים בדרך בין לBIN "www.google.com", ונבדוק כי התשובה זהה<sup>63</sup>:

```
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 בaczmo. מעבר לכך, היא כוללת מידע נוסף לא מודדנו. אם נביט בהסנפה, נראה שהחבילות ש-tracert שלוח דומות מאוד לחבילות שלחנו בaczmo.



### תרגיל 7.8 - עכשו תורכם - Traceroute

- כתבו סקריפט אשר מדפיס את כל התחנות בין לBIN "www.google.com". על הסקריפט להדפיס את כתובות ה-IP של התחנה בלבד.

**dagshim:**

- כארה 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 שלחחתם לו (כלומר הזמן שלוקח מהרגע שלחחתם 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

---

<sup>63</sup> כפי שכבר למדנו, שכבת הרשת יכולה לבחור לשנות את הנитוב עבור כל חבילה וחביבה, ולכן יתכן והතשובה 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 על מנת לבצע הרבה דברים אחרים, ישן דרכים נוספים, אלגנטיות יותר מלאו שモוצגות 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 בס-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. מונח שחרר הרבה במהלך הפרק הוא "כתובת IP" - אותה כתובה לוגית בשכבת הרשת המשמשת כל ישות ברשת כדי להזדהות. אך שאלת גודלה עדין נותרה ללא תשובה:

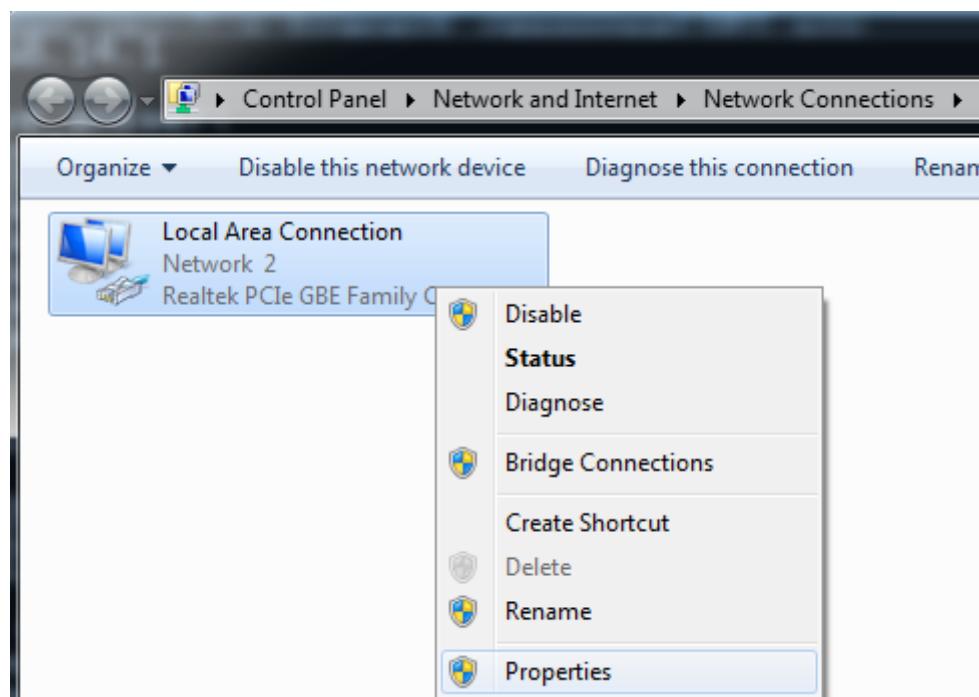


איך רכיב מקבל כתובת IP?

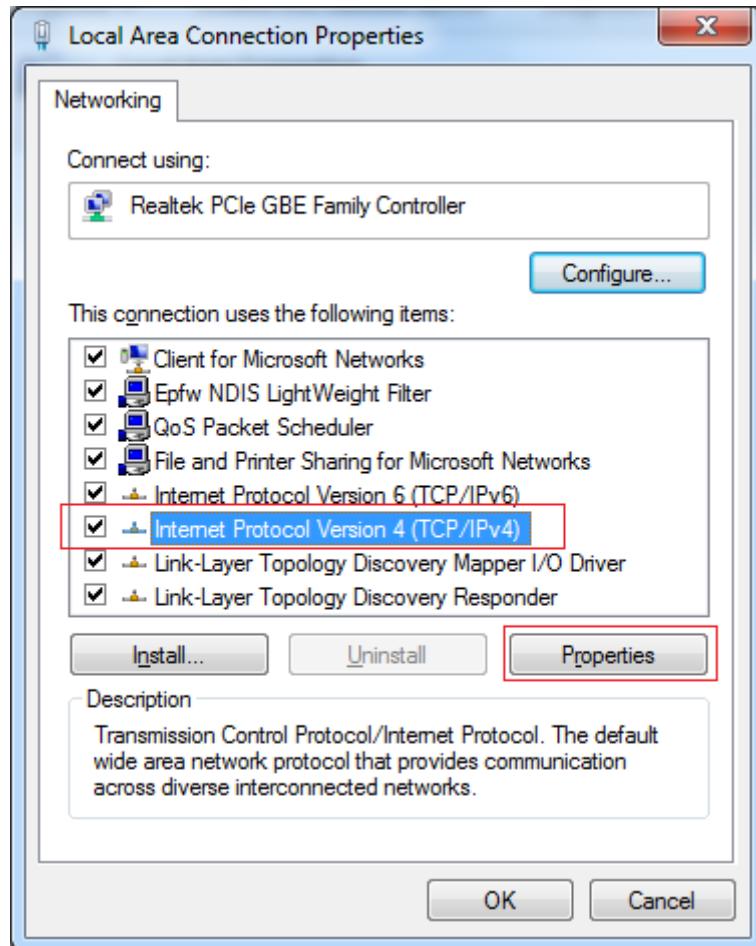
לצורך הדוגמא, כיצד המחשב שלו ידוע מה כתובת ה-IP שלו?

ישן מספר דרכים לקבל כתובת IP, ולאណון בכלל. הפשטה שבהן נקראת **הקצת סטטיטית** של כתובות IP. על מנת לבצע כתובת IP בזרה 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**.

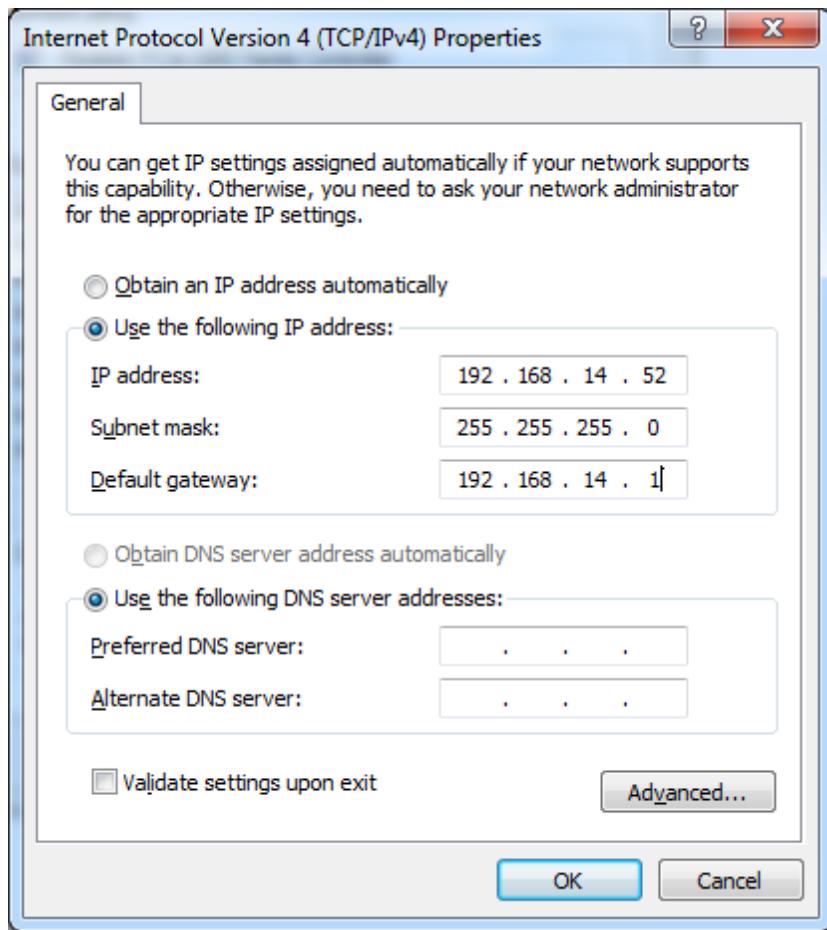
לאחר מכן, לחזו על החיבור שלכם באמצעות הממשק הימני של העכבר, ובחרו **במאפיינים** או **Properties**:



מתקן החלון 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 Description. לדוגמה:



על מנת שאפשרות זו תעבור,عليكم לבחור בכתובת שלא קיימת כבר ברשת. כמו כן, שימו לב להשתמש באותו הגדרות Default Gateway ו- Subnet Mask שהיו לכם קודם<sup>64</sup>.

דרך נוספת לקבל כתובת IP היא הדריך 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 ישיראה את הבקשה יוכל לענות אליה, בעוד ישויות אחרות יתעלמו מבקשת זו.

<sup>64</sup> 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 - תהילך שינוי הכתובת מסובך 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**.

בהצעה, השירות כותב למחשב איזו כתובת IP הוא יכול לקבל, וכן פרטיים נוספים - כגון משך הזמן אותו הוא מבטיח להקצתו 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.

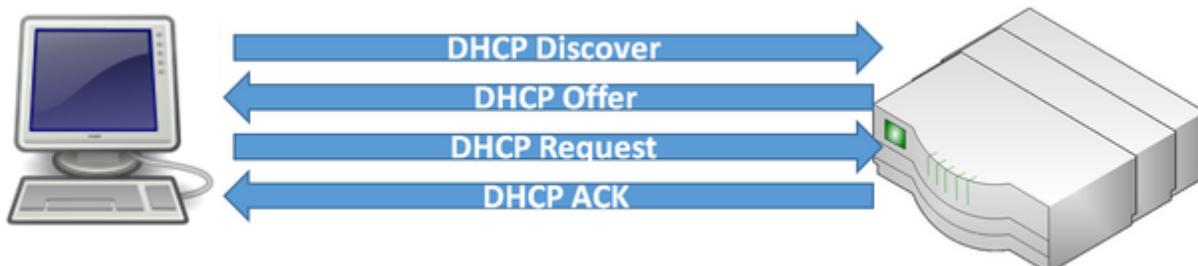
המחשב קיבל שתי הצעות שונות לכטובות IP. עליו לבחור באחת מהן, ואז לשלו בקשה לקבל באממת את הכתובת הזאת. הודעה זו נקראת **DHCP Request**.

בחבילת הבקשה, הליקוח מציין מי השירות DHCP ממנה 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 בשרת הספציפי הזה, ולא ישמרו עבורי את הכתובת. לדוגמה, אם המחשב קיבל הצעה משרת A והצעה נוספת משרת B, DHCP\_A, והוא בחר בהצעה של השירות A, כאשר הוא שולח את הודעת ה-**DHCP Request** שלו לכטום, גם השירות B יראה אותה ויבין שמחשוב 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 לליקוח. החל מעכשיו, הליקוח יכול להשתמש בכטובות IP-הו שניתנה לו באמצעות השירות DHCP.

לסיכום, התהליך נראה כך:



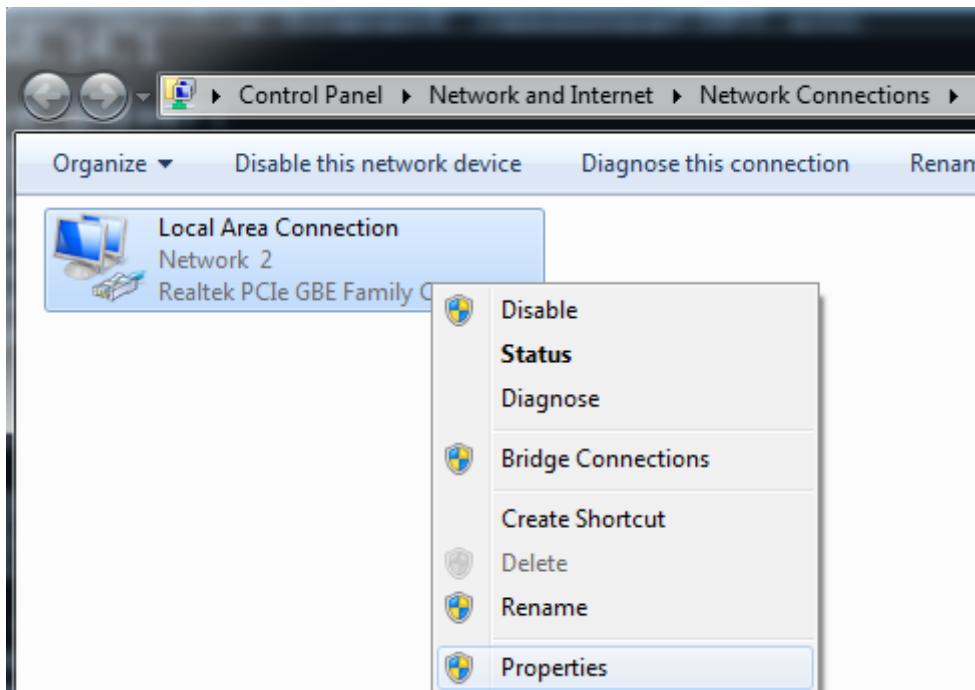
לשימוש ב-DHCP יתרונות רבים. בין השאר, הוא מאפשר לחסוך בכטובות IP בכך שהוא מקצה כתובת לליקוחות רק כשם זקנים להן, ולא כל הזמן. בנוסף, באמצעות DHCP ניתן לתת פרטיה קונפיגורציה נוספים לליקוחות מלבד לכטובות IP (למשל - מי השירות DNS שימוש את הליקוח). על כן לא נרchia במספר זה.



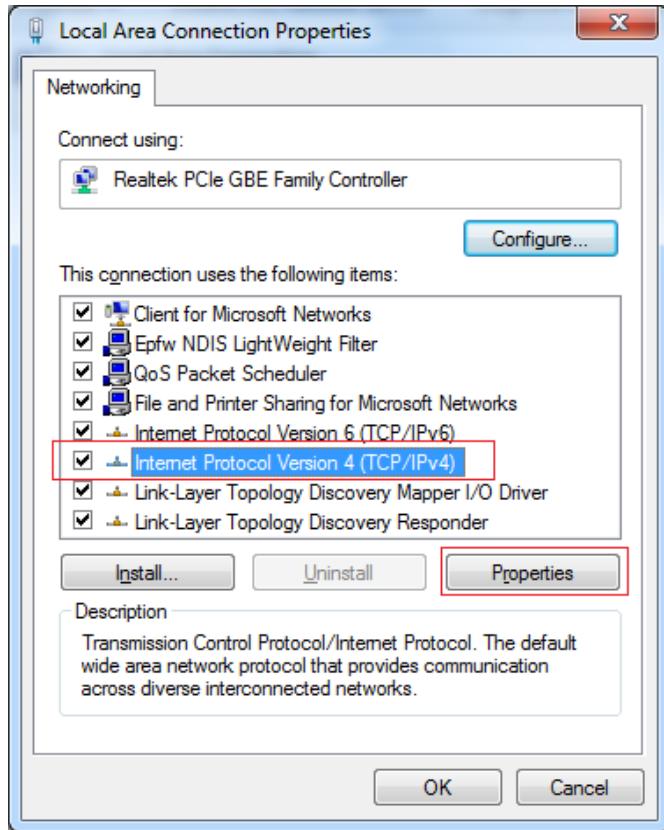
## תרגיל 7.9 מודרך - קבלת IP באמצעות DHCP

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 WinKey+R, והקשת ncpa.cpl.

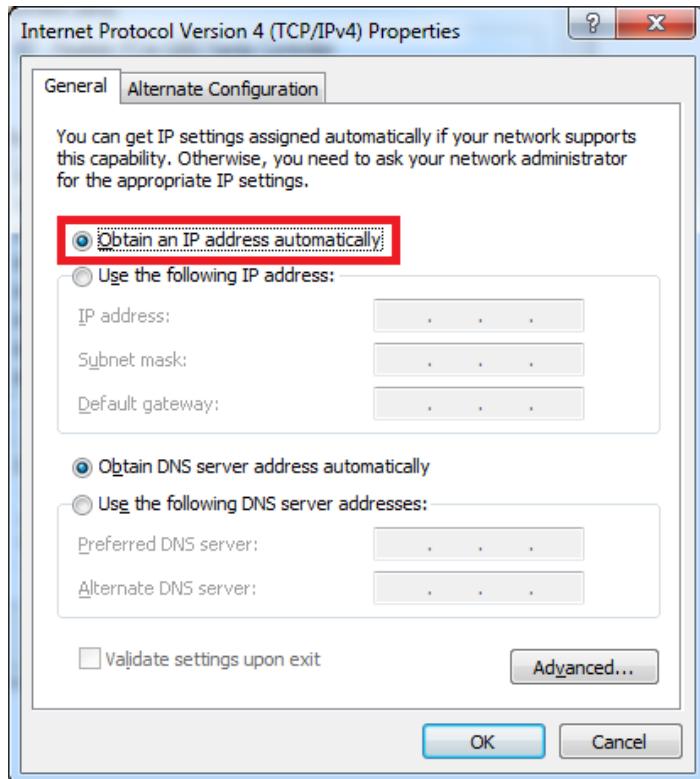
לאחר מכן, לחזו על החיבור שלכם באמצעות המקש הימני של העכבר, ובחרו במאפיינים (Properties).



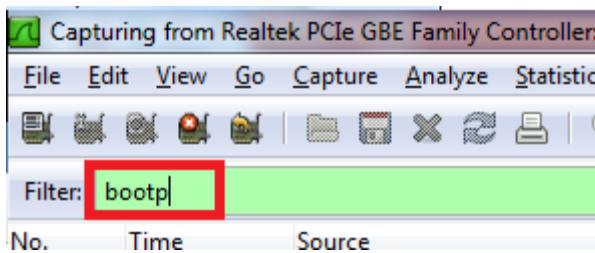
מתוך החלון שנפתח, בחרו ב-Internet Protocol Version 4 (TCP/IPv4), ולחצו שוב עיל מאפיינים (Properties).



:Obtain an IP address automatically



כעת הריצו את Wireshark, והשתמשו במסנן התצוגה (display filter) הבא: `bootp`<sup>65</sup>



כעת הכנסו ל-Command Line, והריצו את הפקודה הבאה

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

C:\Users\USER>ipconfig /release

Windows IP Configuration

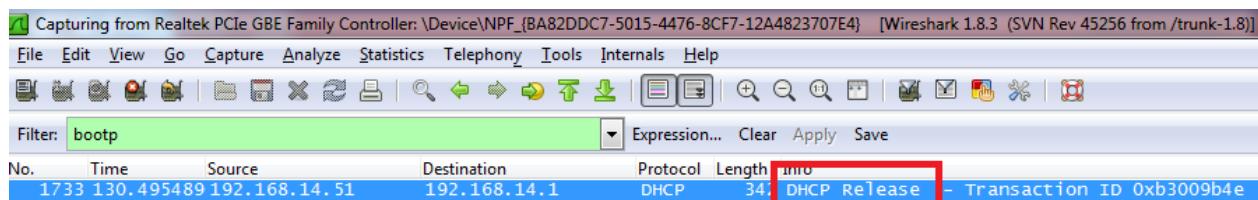
Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix . .
  Link-local IPv6 Address . . . . : fe80::419b:ac69:cfb6:705%11
  Default Gateway . . . . . .

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 למחשבם כתובת IP. אם נסתכל בהසנה, נגלה שלמעה המחשב שלח הודעה לשרת ה-DHCP שמצין כי הוא מבקש להתנתק ממנו:



לא נטעב על הودעה זו. כעת, נסו לקבל כתובת IP חדשה. לשם כך, הריצו את הפקודה: `ipconfig /renew`

---

<sup>65</sup> DHCP הינו למעשה הרחבה של פרוטוקול יישן יותר בשם BOOTP, ו-Wireshark לא מכיר את מסנן התצוגה "dhcp". על ההבדלים בין DHCP ל-BOOTP לא נתעכט בספר זה, אך אתם מוזמנים לחפש על כך באינטרנט.

```
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 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 (d4:be:d9:d6:0c:2a) Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0) 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: 00000000000000000000

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 <sup>66</sup>Ethernet<sup>66</sup>. 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 שלנו עדין אין כתובת IP, הוא משתמש בכתובת הזו. 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 "תלו 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 בשם המלא - (Bootstrap Protocol).

לא נתעכט על כלל השדות של פרוטוקול DHCP, אך נשים לב לנוקודה 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 שהייתה לו קודם לכן, וمبקש לקבל אותה שוב.

---

<sup>66</sup> 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)
+ Bootstrap 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: (252) Route
```

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 שכובות ה-IP המוצעת מוקצת עבורו ולא עבר אף אחד אחר. כלומר, לאחר מעבר הזמן הזה (בדוגמה שלנו - יום אחד), אם הלקוח לא ביצע חידוש של ה-IP, אסור לו להמשיך ולהשתמש בכובות ה-IP הזו, שכן יתכן והוא מוקצת לשימוש אחרה.

**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 עליו להכיר את ה-Subnet Mask שלו. כתוב לנו מבינים שהמחשב יודע אותה באמצעות הודעה DHCP.

לאחר מכן נשלחת הודעה DHCP Request, המצינית בפני שרת DHCP זהה (וגם בפנים שרתים נוספים), אם יש כאלה), שהלקוח בחר בו ומעוניין להשתמש בהצעה שלו. היה ששהודה זו חוזרת באופן כמעט מלא על הودעות קודמות, לא נתעכבר אליה.

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 אליה.

בתום תהליך זה, המחשב שלנו ידע מה כתובת ה-IP שלו ומה ה-Subnet Mask הrelsיבנטית. 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**, הпрוטוקול 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 מתחברת 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 מtbody החלטות הניתוב - ככלומר איך הנטבים בונים את טבלאות הניתוב שלהם. לא הרחכנו על שיטות להתמודד עם עומסים ברשת, ולא נגענו בברחת איכות (Quality Of Service). לא סיירנו 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 David J. - Andrew S. Tanenbaum (מהדורה חמישית) מאת Wetherall, 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 James F. Kurose (מהדורה ששית) מאת Kurose, הפרק הרבעי מוקדש כולו לשכבה הרשת. באופן ספציפי, מומלץ לקרוא את החלקים:

- 4.3 - התהיליך שקורה בנתב.
- 4.5 - ניתובים ואלגוריתמי ניתוב.
- 4.6 - על אלגוריתמי הניתוב בשימוש באינטרנט.
- 4.7 - ניתוב Multicast ו-Broadcast.

כמו כן, ניתן להרחיב את אופקיכם בפרק על IP מתוך The TCP/IP Guide, אותו ניתן למצוא בכתובת: <http://goo.gl/igqBmk>.

### תרגיל 7.10 - פרגמנטציה של IP (אתגר)

בתרגיל זה תמשכו תקשורת מעל פרגמנטציה של IP (על פרגמנטציה תוכלו לקרוא [בנוסף לפרק זה](#)). השתמשו במודול **socket** של פייתון כדי לכתוב שרת פשוט המאזין על פורט 55555 לחבילות UDP נכנסות. על הסкриיפט להדפיס למסך כל חבילה מידע שהוא מקבל.

עת כתבו סкриיפט המקבל מהמשתמש מסר שעליו לשולח (כלומר, מחרוזת - למשל: "hello world", ומספר פרוגמנטים. עלייכם לשולח את המסר שהложен שלח אל שרת ה-UDP שכתבתם קודם לכן, ולחلك אותו באמצעות

프로그램נטציה של C] למספר הrogramנטים שהליך ביקש. כך למשל, אם הליך ביקש לשלוח את המסר " hello world " בשלשה חלקים, תוכלו לשלוח אותו כך:

- חלק ראשון - "hel"
- חלק שני - "llo worl"
- חלק שלישי - "d"

כמובן שתוכלו לשנות את מספר התווים שנשלחים בכל חלק.

וודאו כי השירות מצליח להציג נכון הודעה שלחתם לו. כמו כן, הסניף באמצעות Wireshark וודאו שגם כוונתם של שולחים מסטר נקבעה נכון של פרוגרנטים. שימו לב שעל מנת לבדוק את תרגיל זה, עליכם להשתמש בשני מחשבים שונים – אחד ללקוח ואחד לשרת<sup>67</sup>.

### dagshim

- יש לבנות ולשלוח את החבילות באמצעות Scapy.
- ה-Header של UDP צריך להופיע רק פעם אחת, אין צורך לחזור עליו בכל פרוגרנט מחדש.

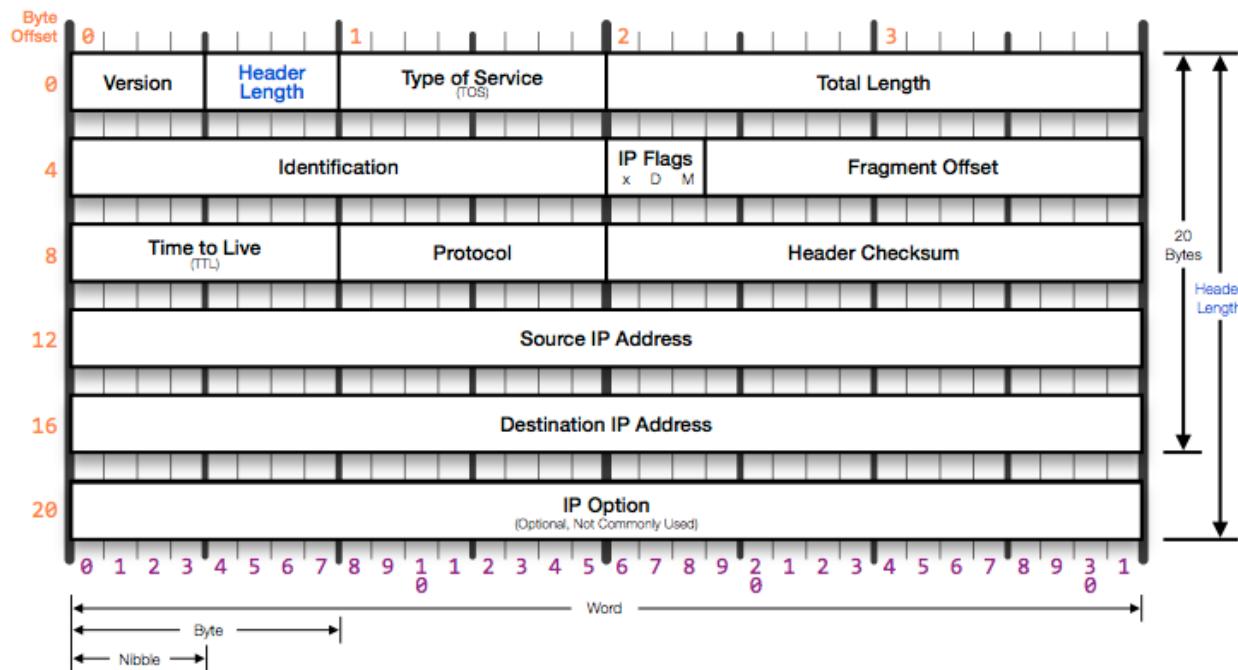
---

<sup>67</sup> באופן תאורי, יכולנו לעשות זאת מעל loopback device – ככלומר 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 נועד 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](http://en.wikipedia.org/wiki/IPv4_header#Header)

### IP Header



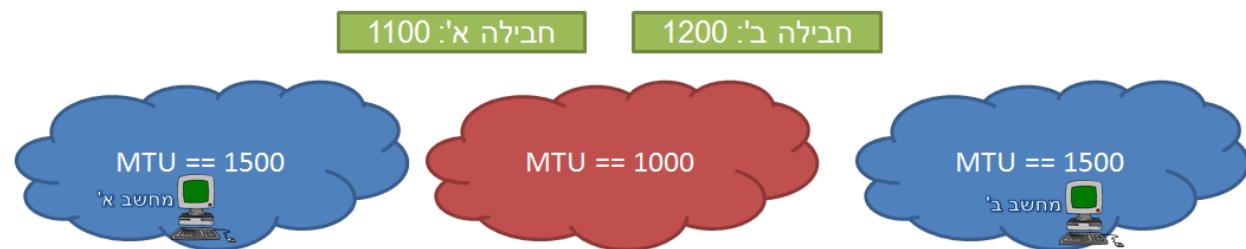
- שדה זה מתאר את גירסת ה-IP. לדוגמה, עבור פקטה של IPv4, השדה יכיל את הערך 4. •  
עבור פקטה של IPv6, השדה יכיל את הערך 6.
- מתאר את אורך ה-Header של החבילה, להיות שהוא עשוי לשינוי בהתאם לחבילה •  
לחבילה, מכיוון שישנם שדות של IP Options אוטם נקבע בהמשך. הערך של השדה מתאר את הגודל ביחידות מידת של 32 ביטים. כך למשל, עבור חבילה בגודל 20 בתים, הערך כאן יהיה 5 (מכיוון שמדובר ב-160 ביטים, שהם 5 פעמים 32 ביטים). הערך 5 (שנמדד 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 Total Length • 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.

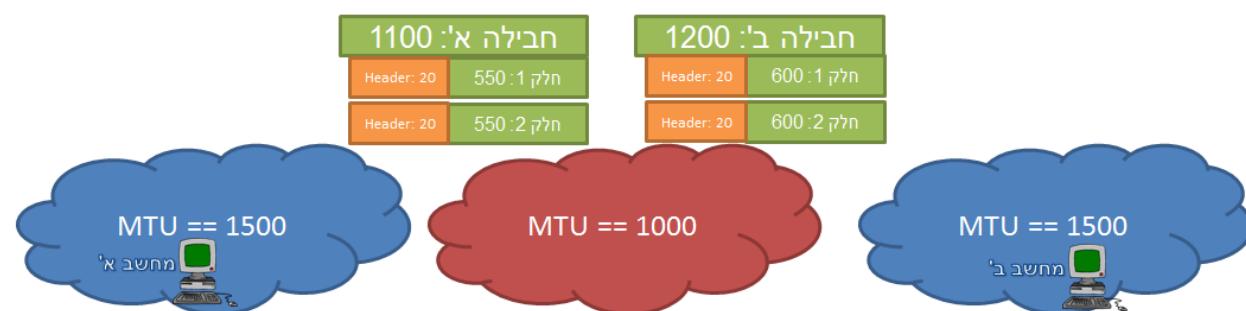
נביט בתמונה הרשות הבאה:



כאן מחשב א' נמצא ברשות 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 MTU שלו הוא 1,500 bytes. עם זאת, על מנת להגיע למחשב ב', החבילות צריכות לעבור ברשות MTU של 1,000 bytes בלבד (הרשת האדומה). מכיוון שלא חבילה א', ולא חבילה ב' יכולות לעבור בה. אם חבילה שכוונה להגיע לרשת האדומה, היא צפופה להזירק.

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:



כאן חבילה ב' התחלקה לשני חלקים, שכל אחד מהם בגודל של 600 bytes, 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 א<sup>68</sup> יכול לשלוח את כלן מבלי להיתקל בבעיית ה-UTM. כאשר החלקים יגיעו למחשב ב' בראשת הכהולה, יהיה עליו להבין איזה חלקים היו שייכים לאיזו חבילה, לחבר אותם מחדש ולקבל את החבילות שמחשב א' ניסה לשלוח אליו במקור.

ניתן לקרוא מידע נוסף על פרגמנטציה בכתובות: [http://en.wikipedia.org/wiki/IP\\_fragmentation](http://en.wikipedia.org/wiki/IP_fragmentation)

כעת נמשיך להסביר על השדות השונים של IP Header:

- Identification - שדה זה משומש במקורה של פרגמנטציה. 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 כך, 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 פרגמנטציה לחבילה שנשלחת עם הדגל הזה דולק<sup>69</sup>.
  - בדוגמה שלעיל, במידה שחבילה א' נשלחה עם הדגל DF דולק, אף נתב בדרך אסור לפצל אותה ולבצע פרגמנטציה. אם כך, החבילה לא תוכל להשליח למחשב ב', ותשלח על כך הודעה שגיאיה.
  - (MF) More Fragments - דגל זה משומש במקורה של פרגמנטציה. 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 מתאר מהו הפרוטוקול שנמצא מעל שכבת ה-IP. לדוגמא, הערך "6" מצביע שהשכבה שמעל לשכבה ה-IP היא שכבת TCP. הערך "17" מצביע 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 ישوت כלשהי בראשת,

<sup>68</sup> בפועל, בחלק גדול מהמקרים, יהיה זה אחד הנתבים בדרך שיבצע את הפרגמנטציה ולא מחשב הקצה.

<sup>69</sup> 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 זה ?Checksum](#).

- 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, ונitin להפיך מהן ל开玩笑 וליצור גירסה טובה יותר של הפרוטוקול. לשם כך, עלתה בשנת 1995 הצעה 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. כתובה של IPv6 הייתה, כאמור, באורך של ארבעה בתים (bytes), מהם 32 בתים (bits), ומכך  $2^{32}$  האפשרויות השונות לכתובות של IPv6. ב-IPv6 הוחלט שכל כתובה תהיה באורך של 16 בתים (bytes), כלומר 128 בתים (bits). אי לכך, ישן  $2^{128}$  אפשרויות לכתובות IPv6. זהו מספר עצום של כתובות שלא אמרו להיגמר גם כאשר לכל מקרה, מצטמם ומידח תהיה כתובה 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**.

כתבות מוצגות באמצעות שמוֹנָה "קבוצות" של ארבע ספרות הקסה-דצימליות, כאשר כל "קבוצה" מייצגת למעשה שני בתים (bytes). ה"קבוצות" מופרדות באמצעות התוו נקודות (:). להלן דוגמה של כתובה IPv6:

2340:0000:0000:0000:0000:0000:0001

מכיוון שכתובה שכזו 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 עיל גם כך:

2340:0:0:A:0:0:0:1

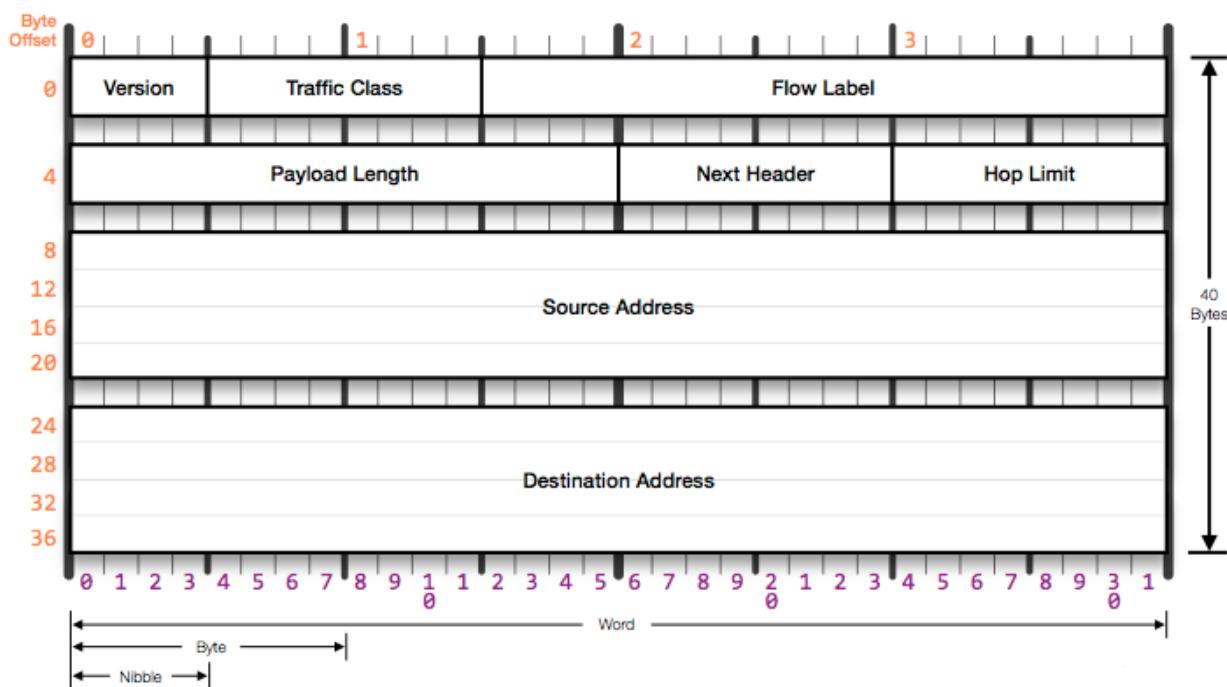
זאת ועוד, קבוצות כאלה רקי את הספרה אפס, שמשמעותו ברצף זו אחר זו, יכולות להיות מוצגות באמצעות שני תווים נקודות (:). עם זאת, ניתן להשתמש בשני תווים נקודות (:): רק פעם אחת בכתבאות. כך למשל, את הכתובת לעיל ניתן להציג גם בצורה הבאה:

2340::A:0:0:0:1

## IPv6 Header

כך נראה ה-Header של חבילת IPv6:

## IPv6 Header



השדות 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 מסוימו זרם מידע. על אף-Sh-IP הינו פרוטוקול שלא מבוסס קישור, יש כאן דרך לשערר חבילה בודדת לקשר מלא. כך למשל, ניתן לתת את אותו ערך בשדה Flow Label לכל החבילות הקשורות לשיחת VoIP מסוימת, או לתקשורת בין דפדן לבין אתר. במידה שהחביבה לא מקושרת לאף "flow", בשדה זה יהיה הערך 0. הרעיון הוא שכל הנתבים בדרך יטפלו בכל החבילות הקשורות לאותו ה-"flow" באותה דרך. כך, 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 - מתאר איזה Header מגיע אחרי ה-IPv6 Header. כך למשל, ערך של 6 מזזה שה-Header הבא הוא של TCP, בעוד הערך 17 מזזה שה-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.

הבדל משמעותי 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 שכוללתChecksum (כגון Ethernet), ומעלה נמצאת שכבה רבייעית שגם כוללתChecksum (כגון TCP או UDP, עליהם נלמד בפרק הבא). אי לכך, ה-Header checksum מחושב בכל כרטיס רשות בדרך, והן על ידי מכשירי הקצה. מכאן שאין צורך לחשב את ה-Header checksum גם בכל נתב ונתב. פועלות חישוב ה-Checksum הינה יקרה ומעמיסה על הנתב. אי לכך, נתבים העובדים עם IPv6 יכולים להשיק את זמןם בניתוב של פקודות, ולא בחישוב של checksum.

ל-IPv6 יתרונות רבים נוספים על IPv4, ביניהם תמיימה נוחה בכתובות Multicast, אפשרות לישיות תחת עצמן כתובות IP מלבני צורך ב-DHCP (תהליך הנקרא SLAAC) ועוד. על אלו לא נרחיב בנספח זה, אך אתם מוזמנים להרחב אופקיכם.

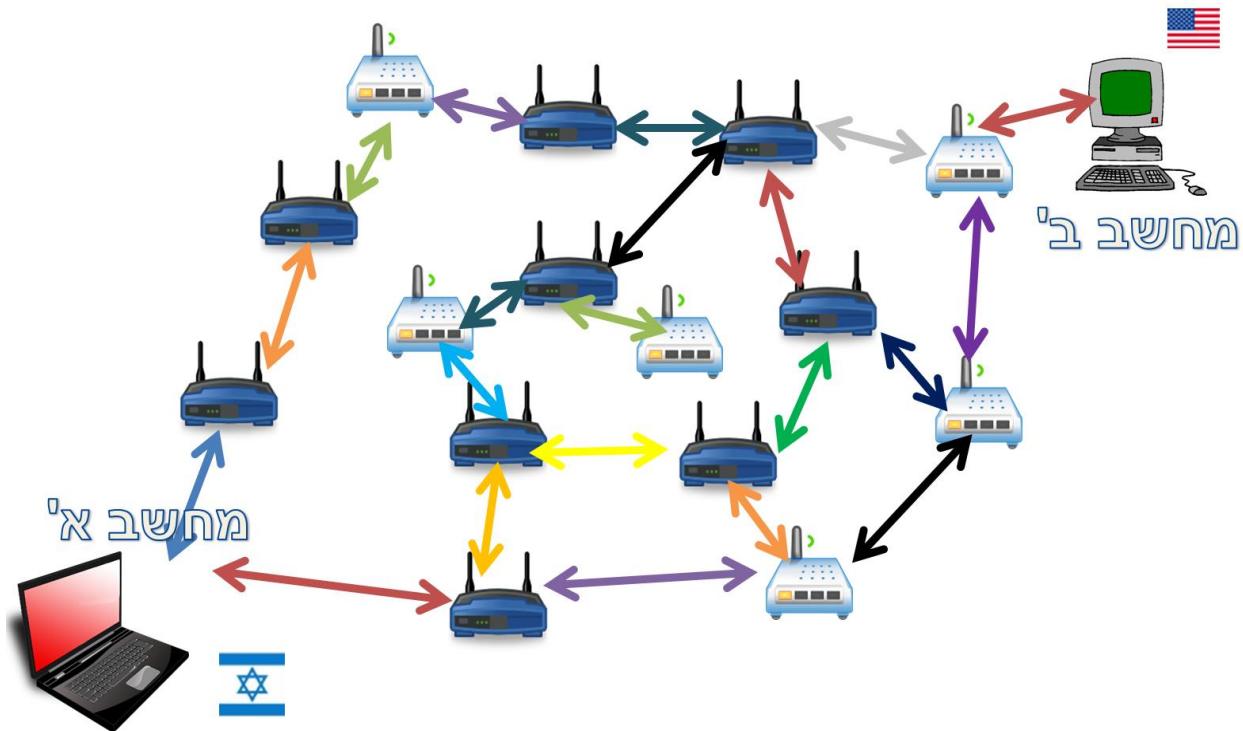
## פרק 8 - שכבת הקן

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 שניינו 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 לאורח הספר:



בפרק הקודם ראיינו שכבת 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 ישותות הקשורות זו לזו באופן ישיר. באior לעיל, כל חץ צבעוני מייצג תקשורת בשכבה השנייה בין שתי ישויות - 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 ישויות 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 ישיר היא שמיידע יכול לעבור בין הישויות מבלתי לעבר בישות אחרת בדרך. חיבור ישיר יכול להיות לצורךות שונות. תcan ומדובר בחיבור קווי - משתמשים בכבל פיזי בצד אחד לחבר ישוט אחת נוספת. למשל, נאמר שמחשב א' מחובר לאחד הנתבים הקרובים אליו בכבל. חיבור אחר יכול להיות חיבור אלחוטי - לדוגמה, תcan והנתב של מחשב א' מחובר לנtab הבא באמצעות WiFi. תcan אףיו 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 שabitות המידע יצליחו לעבר מישות לשוט בקרה אמינה.

השכבה צריכה להתמודד עם תקלות שיכולות להיות בkn, עליה נפרט בהמשך.



### **שכבת הkn מספקת לשכבת הרשות אפשרות 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 בכבל, בלויין, WiFi, או באמצעות יוני דואר. היא רק אחראית להבין מה המסלול האופטימלי. כמו Sh-Waze רק אומרת לרכיב באיזו דרך לעבר, ולא מסבירה לנו הגשה הוא צריך לרדוף, ללחוץ על הגז, לאוותת ולעוצר ברמזור או להולך רגל. בזה יטפל הנאג, או במקרה שלנו - שכבת הkn.



### **אייפה ממומשת שכבת הkn?**

השימוש של שכבת הkn "נמצא" בכל ישוט ברשות - וספציפית, בכרטיס הרשות של הישוט. כך למשל כרטיס Ethernet ימשח את פרוטוקול Ethernet, וכרטיס WiFi ימשח את פרוטוקול WiFi. כך כרטיסי 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 כך<sup>70</sup>:



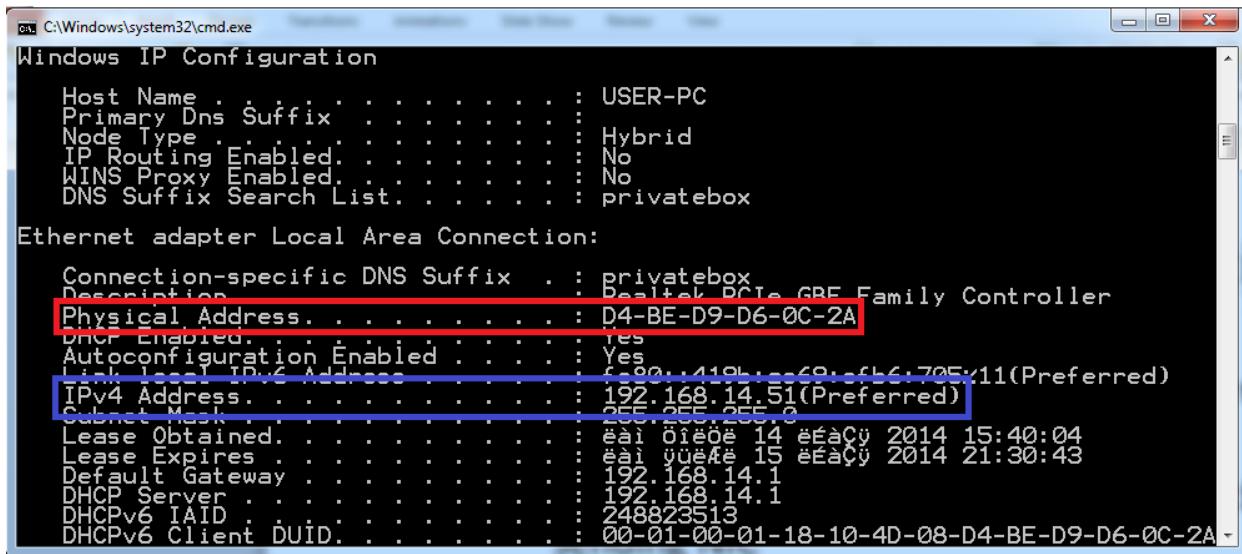
**מהי הכתובת שלי בשכבה הקו?**

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 מהמתקנים 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 אף תחנה אחרת בדרך. באם ישות מסוימת רוצה לפנות אל ישות אחרת, היא צריכה לפנות אל הכתובת שלה. כתובות בשכבה השנייה נקראות **כתובות MAC (באנגלית - MAC Addresses)**.

היכנסו שוב ל-[Command Line](#), והקישו את הפקודה הבאה: **ipconfig /all**. שימוש לב שהשתמשנו בפקודת **ipconfig /all**, שכן אחרת הפקודה לא מציגה את כתובות השכבה השנייה.

<sup>70</sup> בפרק [השכבה הפיזית / הרשת המשדרית](#), נלמד יותר על כבל זה.



ניתן לראות שהמידע מוצג עבורי הכרטיס עם כתובת ה-IP שמצאנו בפרק הקודם, 192.168.14.51, כפי שמסומן **בכחול**. מעת לעלה יותר, יש שדה בשם **Physical Address**, כתובת ה-**MAC**, המסומנת **באדום**. שימושו לבשcomo שהפקודה מציגה לנו, אכן מדברת **בכתובת פיזית** - כתובת זו "צרכובה" על כרטיס הרשות עצמו, ולא אמורה להשתנות<sup>71</sup>. 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 **yufy**, 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
```

<sup>71</sup> חלק מהשימושים של השכבה השנייה יש אפשרות להשתמש בכתובת שאינה צרכובה על הכרטיס.

```
a5\x a5\x a5\x a5\x a5\x a5' |>>>
>>> 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' /> IP version=4L
ihl=5L tos=0x0 len=41 id=703 flags=DF frag=0L ttl=128 proto=tcp cksum=0x0 src='1
92.168.14.51' dst='173.194.78.125' options=[]
|> TCP sport=53442 dport=5222 seq=222
8948902L ack=3568354024L dataofs=5L reserved=0L flags=A window=16325 cksum=0xcb
36 urgptr=0 options=[]
>>> -
```

כפי שניתן לראות, בשכבה ה-Ethernet נמצאות ראות כתובות的目的 (destination address) שמוינעה בשם "dst", וכתובת המקור של המסרה (source address) שמוינעה בשם "src".

כפי שראנו בפרקים הקודמים, ניתן להשתמש בפורמט **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 המסגרת תואם את כתובת ה-MAC שלנו. לשם כך, נגדיר קודם את כתובת ה-MAC אותה מצאנו קודם לכך באמצעות **ipconfig**, בצורה הבאה:

```
>>> MY_MAC = 'd4:be:d9:d6:0c:2a'
```

שימוש לב Ci הכתוב ש滥ם יהיה שונה מהכתוב שמוינעה בדוגמה. כמו כן, על אף ש-**ipconfig** הציג את הכתובת כאשר כל בית (byte) מופרד באמצעות התו מקף ('-'), אנו משתמשים בפורמט של Scapy בו כל בית מופרד באמצעות התו נקודתיים (':'). 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 המסגרת היא הכתובת של כרטיס הרשות שלנו. אם לא, היא מחזירה False. לשם הבהרה, ניתן להרשות 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 שיעדו 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 בראצוננו להציג את כתובות MAC של היעדים הפונוט אלינו. לשם כך - נשתמש בפרמטר **prn** של הפונקציה **sniff**, אשר גם אותו הכרנו בפרקים 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 לאוთה היעדות (עליה למד בהמשך). במצבו הנוכחי, הסקריפט שכתבנו לא כל כך מועיל.



## תרגיל 8.2 - מציאת כתובות MAC שפונגו אל כרטיס הרשת שלו

שפרו את הסקריפט שכתבנו עד כה. גירמו לכך שהסקריפט "יזכור" האם הוא הדףיס כתובת מסוימת למסך, ואם כן - לא ידפיס אותה שוב. אפשרו לסקריפט לזרז במשר חמש דקוט (رمز) - קראו על פרמטרים נוספים לפונקציה `(sniff)`.

על מנת לגרום לסקריפט להדפיס כתובות נוספות, נסו להוסיף אליו מישיות נוספת ברשות שלכם, למשל מחשב אחר שקיים ברשת. תוכלו להיעזר לשם כך בפקודה `ping` עליה למדנו בפרק שכבת הרשת/ איך Ping עובד?.



### איך בנויה כתובות Ethernet?

עד כה בפרק הספקנו לראות מספר כתובות MAC של Ethernet. כפי שווידאי הבחנתם, כתובות Ethernet בנויות משישה בתים (bytes). ניתן לייצג את הכתובות בדרכים 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, כתובות Ethernet מחלוקת לשני חלקים:

- **מזהה יצרן (Vendor ID)** - מזהה מי החברה שייצרה את כרטיס הרשת.
- **מזהה ישות (Host ID)** - מזהה של כרטיס הרשת הספציפי.

שלושת הבתים העליונים (הראשונים) שייכים למזהה היצרן, 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 היצרן, 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 יצרן, שכן מזהה היצרן שלהם (המסומן ב**אדום**) - זהה.

מזהה היצרנים ידועים, ولكن ניתן לדעת בקלות לאיזה יצran שייכת כתובת מסויימת. לדוגמה, נכנס לאתר [http://www.coffer.com/mac\\_find](http://www.coffer.com/mac_find) ונצין לתוךו את אחת משתי כתובות ה-MAC לעיל, שמתחלות בmazeה היצran D4:BE:D9. האתר יספר לנו שמדובר בכתובת השיכת ליצרנית Dell. השימוש לבשילצראניות שונות עשוי להיות יותר מזהה יצran אחד. כך למשל, גם mazeה היצran E0:DB:55 שייך לדell, גם mazeה DB:BA:A4, וגם רבים נוספים.



### תרגיל 8.3 - מציאת היצרנית של כרטיס הרשת של'

באמצעות האתר שהוצג לעיל, כמו גם פקודה **ipconfig**, מצאו מי היצרנית של כרטיס הרשת שלכם.



### תרגיל 8.4 - מציאת יצרניות של כרטיסי רשת מתוך הסנפה

הורידו את קובץ הסנפה Layer2\_1.pcap מהכתובת: [http://cyber.org.il/networks/c07/Layer2\\_1.pcap](http://cyber.org.il/networks/c07/Layer2_1.pcap) והעזרו בהסנפה כדי למצוא את שתי כתובות ה-MAC שנמצאות בה, ולאחר מכן גלשו לאתר שהוצג לעיל וממצויאו את היצרנים של כתובות ה-MAC המופיעות בקובץ. הוסיפו 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 משמעותה שליחת המסגרת לכל הישויות שנמצאות איתנו ברשת<sup>72</sup>.

<sup>72</sup> כתובת Broadcast שייכת 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://cyber.org.il/networks/c07/Layer2\\_Broadcast.pcap](http://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 כדי לסנן רק את המסגרות שנשלחו לכתובות Broadcast בرمת-h-Ethernet?



### תרגיל 8.6 - כתובות Ethernet

בתרגיל זה תכתבו בפייתון סקריפט אשר מבקש מהמשתמש כתובת MAC ומדפיס עליה מידע.

1. קבלו מהמשתמש כתובת 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 כתובת 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 לסייע למכרז את שני הצדדים על כך שמתחליה מסגרת חדשה. שמו 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 זה<sup>73</sup>.
- 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 - מנגנון Checksum לגילוי שגיאות. על משמעותו של Checksum למדנו במסגרת פרק שכבת התעבורות / מה זה Checksum? בפרוטוקול Ethernet, אורך ה-CRC הוא 32 ביטים (bits), שהם ארבעה בתים (bytes). גם שדה זה לא נראה ב-Wireshark, שכן הוועדא 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](http://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

```

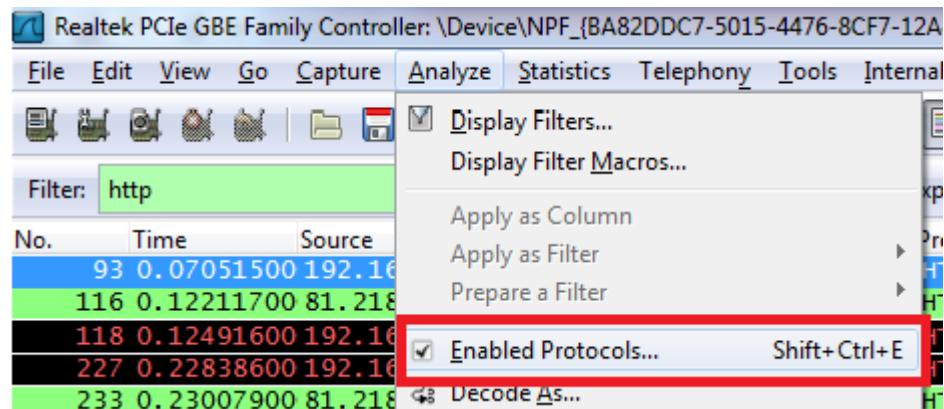
<sup>73</sup> במקרים מסוימים, שדה זה גם יכול להיעיד על האורך הכולל של המסגרת. עם זאת, במספר זה ניתן מאפשרות זו. אתם מוזמנים לקרוא עליה באינטרנט.

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?. מוגרת 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: לכהר, 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

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]

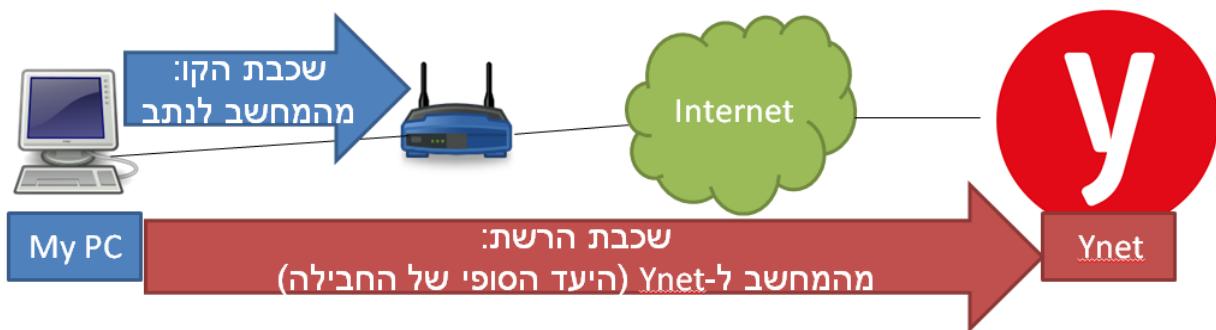
0000	00 0c c3 a5 16 63 d4 be	d9 d6 0c 2a 08 00 45 00	....c... *..E.
0010	04 40 00 bc 40 00 80 06	b5 9e c0 a8 0e 33 51 da	.@...@... .3Q.
0020	1f a8 d5 8f 00 50 af 14	cb 2c 70 52 08 4e 50 18	....P... ,PR.NP.
0030	41 14 44 90 00 00 34 31	31 2d 34 31 32 2d 34 31	A.D. .41.1-412-41
0040	33 2e 33 36 38 40 2d 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 1347@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 627350@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 2d 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 **Net**? התשובה היא **לא!** היזכרו בפרק שכבת הרשות, בו דיברנו על כך שבדרך כלל ישנו רכיבים רבים שמקשרים בין ישויות קצה בראשת. בהנחה ואינכם מחוברים באופן ישיר אל **Net** באמצעות כבל (או דרך אחרת), אתם עוברים בדרך בשיטת נוספת. הישות הקורובה ביותר אליוים היא **הנתב שלכם**, וכותבת ה-MAC זו היא הכתובת שלהם. כמו כן, שימוש לב-**Wireshark** יודע להגיד לנו שהיצרנית של הנתב הינה **Bewan**.
- שימוש לב:** לנtab יש יותר מכתובת MAC אחת, שכן יש לו יותר מקרים רשות אחד. כתובות ה-MAC שמוצגת בהסכמה היא הכתובת של כרטיס הרשות של הנתב המחבר אל הרשות הביתית שלכם, ולא של כרטיס הרשות של הנתב שמחובר אל האינטרנט.
- כתובות המקור - כתובות זו צפוייה להיות הכתובת של כרטיס הרשות שלכם. כדי שניתן לראות בדוגמה לעיל, הכתובת **a2:d4:be:d9:0c:2a** זהה לכתובת שמצאנו באמצעות הפקודה **ipconfig /all**. כמו כן, Wireshark יודע להגיד לנו שהיצרנית של כרטיס הרשות הינה **Dell**.
- סוג - הערך 0x800 מצביע על כך שהמsegret היא מסוג IP. כך כרטיס הרשות ידע להפנות את כל מה שנמצא בשדה **Data** אל הישות שמטפלת בחבילות IP (במקרה שלנו - מערכת הפעלה).
- שדה **Data** יכול את כל השכבות שנמצאות "מעל" ל-Ethernet, החל משכבה ה-IP, דרך שכבת TCP ועד שכבת HTTP.
- שימוש לב שודות **Checksum** ו-**Preamble** אינם מופיעים בהסכמה, כפי שציינו קודם לכן.



**שימוש לב** - כאן רואים באופן יפה את ההבדל בין השכבה השנייה לשכבה השלישייה. בחבילת ה-GET שמוצגת 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 [שכנת הרשות / ניתוב](#), בשלב הבא החבילה תועבר מהנטב הקרוב למחשב שלנו אל הנטב הבא בדרך. בשלב זה, הנטב הקרוב למחשב שלנו ייצור את המסגרת בשכנת ה-Ethernet כך שכנתובת המקור של המסגרת תהיה הכתובת של כרטיס הרשות שלו שמייחדת לאינטרנט, וכתובות היעד תהיה הכתובת של הנטב הבא. כך, שכבת הקו מתארת נכון את ה-Hop הנוכחי: מעבר בין הנטב הקרוב עליו אל הנטב הבא אחריו. עם זאת, בשכנת הרשות עדיין ישמרו הנטונים על נקודות הקצה של החבילה - המעברת מחשב שלנו ואלי Ynet.





### תרגיל 8.8 - התבוננות בתשובה HTTP

cutת הסתכלו בתשובה ה-HTTP שחרזה בתגובה לבקשת GET. ענו על השאלות הבאות:

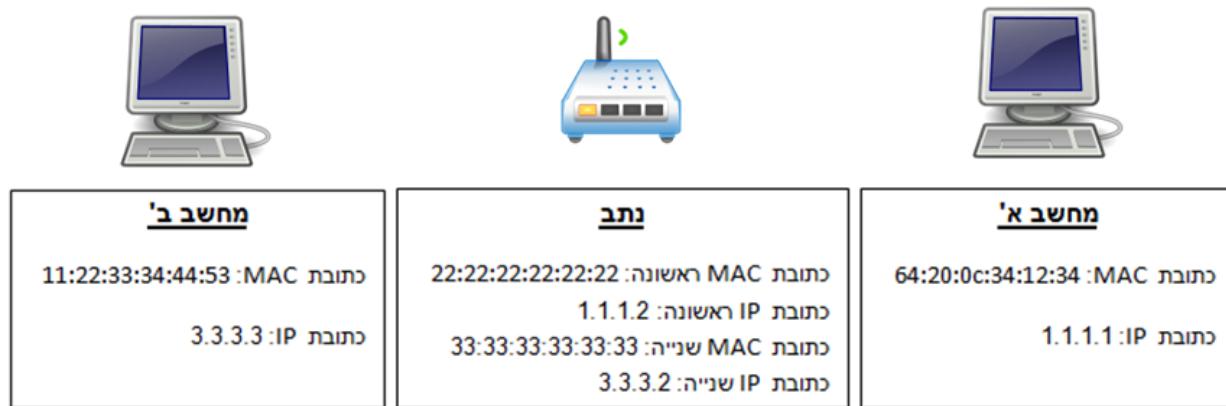
1. למי שייכת כתובת היעד של המסגרת? כיצד תוכלו לוודא זאת?
2. למי שייכת כתובת המקור של המסגרת? כיצד תוכלו לוודא זאת?
3. מהו סוג המסגרת?

בסיום התרגיל, אל תשכחו לחזור לתרפיט Enabled Protocols ב-Analyze->Enabled Protocols Wireshark ולהחזיר את הסימון לפוטוקול IPv4.



### תרגיל 8.9 - כיצד תיראה החבילה שלי?

עימם בתרשימים הרשות הבא:

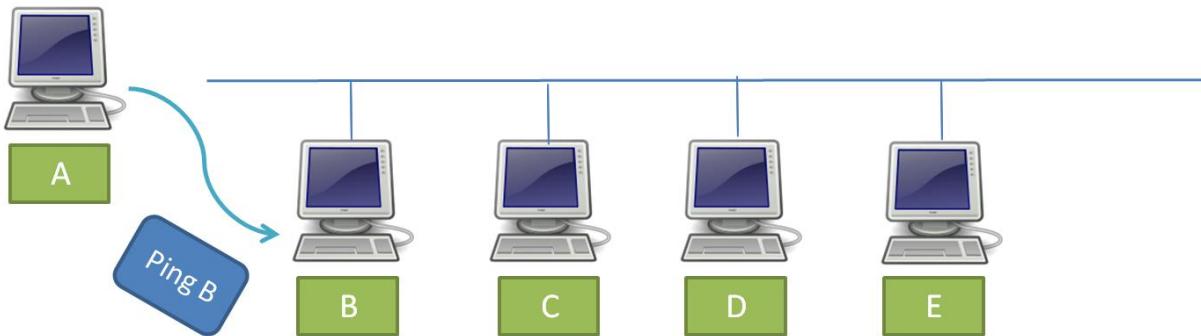


מחשב א' רוצה לשЛОח פקטה למחשב ב'. מחשב א' מחובר למחשב ב' דרך הנטב.  
השלימו את השדות של הפקטה אותה ישלח מחשב א':

1.1.1.1	כתובת IP מקור
c:34:12:3464:20:0	כתובת MAC מקור
	כתובת IP יעד
	כתובת MAC יעד

## פרוטוקול ARP - Address Resolution Protocol

עד כה תיארנו את פרוטוקול Ethernet וכי怎 הוא עובד. אך עדין, משחו חסר. הבינו בשרטוט הרשות הבא:



כל המחשבים כאן נמצאים על תווך משותף. כלומר - מסגרת הנשלחת לכתובת Broadcast תגיע אל כולם, ואין צורך בישות נוספת (כגון נתב) כדי להעביר הודעות ממחשב אחד למחשב אחר. במקרה לנו, המחשב שנראה A רוצה לשולח הודעה ping (כמו Echo-Request, כמו שלמדנו בפרק שכבת הרשת) אל המחשב B. המחשב A יודע את כתובת ה-IP של מחשב B, למשל, באמצעות שימוש ב프וטוקול DNS. אך דבר זה אינו מספיק - על מחשב A לדעת גם את כתובת ה-MAC של כרטיס הרשות של מחשב B!

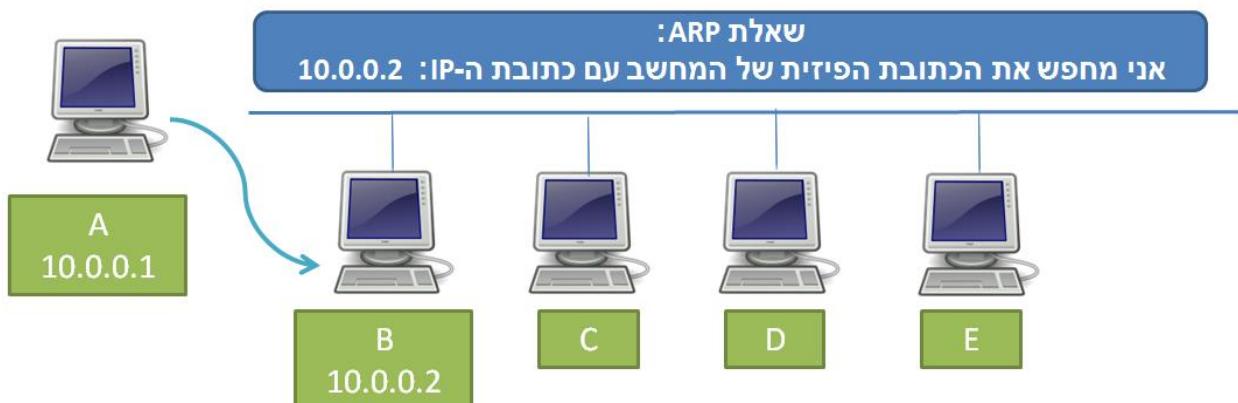


**מדוע הדבר כך? מדוע לא מספיקה כתובת ה-IP?**

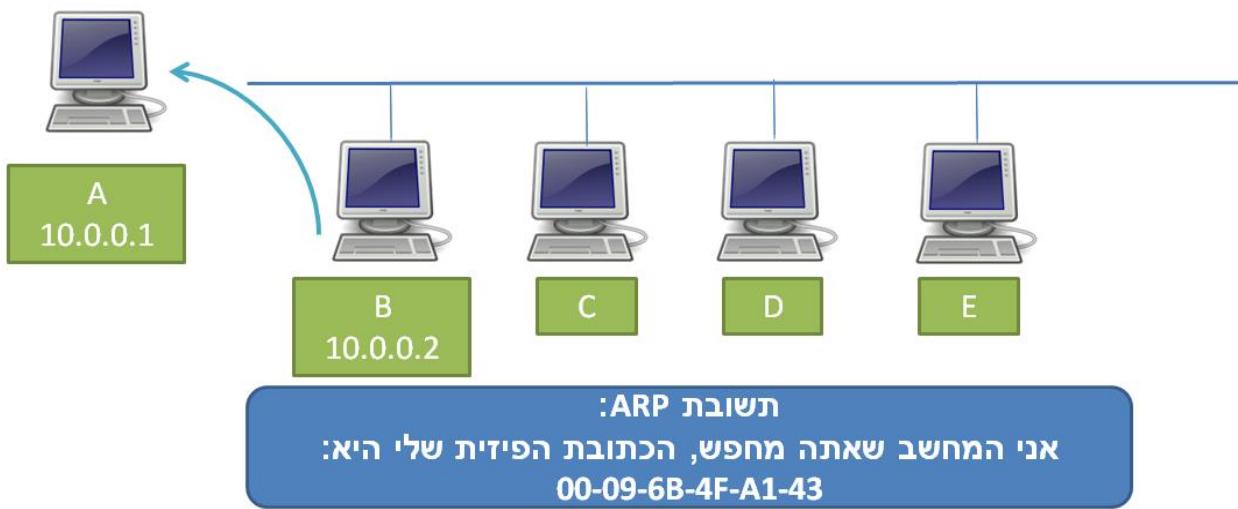
ראשית, היזכרו במודל השכבות. חבילת Ping A שתשלח ממחשב A צפופה להיות בניה שכבת ה-IP בשכבה השנייה, מעליה שכבת IP ולבסוף שכבת ICMP. על מנת לבנות את מסגרת ה-IP, על המחשב A לדעת מה כתובת היעד של המסגרת, כלומר מה כתובת של כרטיס הרשות של B. שנית, על כרטיס הרשות של המחשב B להבין שהמסגרת מיועדת אליו. את זאת הוא עשו באמצעות הסתכלות בשדה כתובת היעד של המסגרת. **כרטיס הרשות אינו מבין כתובת IP**, הוא כרטיס מסווג Ethernet ומכיר כתובת Ethernet בלבד. על כן, הkartis 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 כתובות לוגיות 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 (כמו - לכתובת Broadcast) שמשמעותה: **מי יש את הכתובת הפיזית של המחשב עם כתובת ה-IP של B?** נאמר שכותבת 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 ה-MAC של כרטיס הרשת של מחשב B. במקרה, יש ברשותו את כל המידע שהוא צריך כדי לשלוח את חבילת Ping:

- כתובת ה-IP שלו עצמו (מחשב A) - שכן הוא יודע מה כתובתה שלו, למשל באמצעות DHCP.
- כתובת ה-MAC שלו עצמו (מחשב A) - שכן הוא יודע מה כתובתה שלו, שהרי היא צרובה על הכרטיס.
- כתובת ה-IP של מחשב B - שהוא גיליה, למשל, באמצעות DNS.
- כתובת ה-MAC של מחשב B - שהוא גיליה 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 סטטית (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



פתחו את Wireshark והריצו הסנהפה. כמו כן, מיהקם מה-ARP Cache שלכם את הרשומה שקשורה לנット שלכם (ה-default gateway). להזכירם, כדי לאלוות את כתובת ה-IP של הנット, 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 לגשת אל הנット. תוכלו להשתמש במסנן התצוגה "arp" כדי לסנן את החבילות הרלוונטיות:

Filter: arp      Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
6	4.12820200	dell_d6:0c:2a	Broadcast	ARP	42	who has 192.168.14.1? Tell 192.168.14.51
7	4.12899300	Bewan_a5:16:63	Dell_d6:0c:2a	ARP	60	192.168.14.1 is at 00:0c:c3:a5:16:63

כעת נביס 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 שלנו.
- **בירוק** - סוג המסגרת. מדובר במסגרת מסוג ARP.

כעת נביט בשדות של שכבה ה-ARP :

- **בכתום** - נתונים המתארים כי המיפוי הוא מכותבת IP לכתובת MAC של השכבה Ethernet. שדות אלו נוחצים כיוון שרARP יכול למפות גם כתובתות לוגיות אחרות לכתובות פיזיות אחרות.
- **בתכלת** - קוד חבילת ה-ARP. הקוד הינו 1, והוא מציין 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

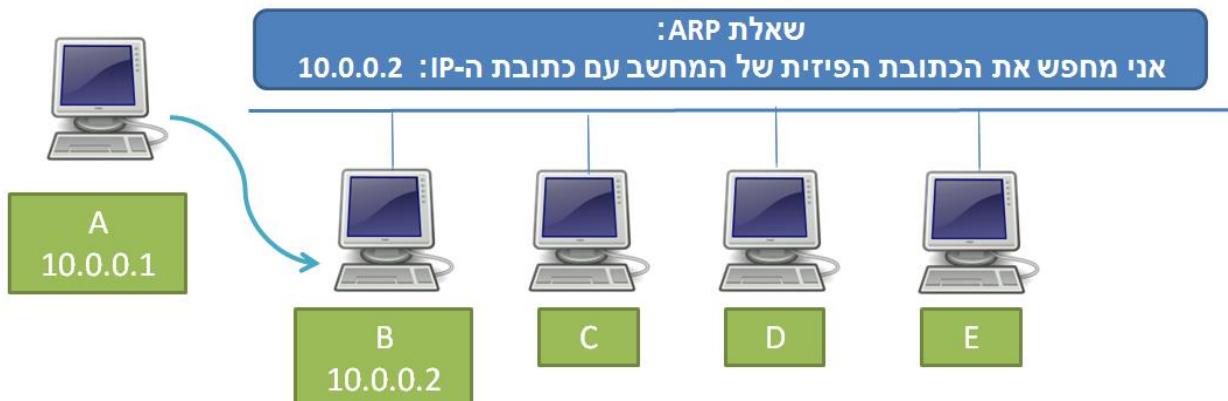
כעת, מצאו את מסגרת התשובה. ענו על השאלות הבאות:

1. בשכבה ה-Ethernet, מהי כתובת המקור של המסגרת? מי שלח אותה?
2. בשכבה ה-Ethernet, מהי כתובת היעד של המסגרת? האם היא נשלחת אל כולם (Broadcast) או רק לשיטת מסוימת?
3. מהו הערך של שדה ה-opcode בתשובה?
4. איפה במסגרת 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 ARP הינה הכתובת הפיזית שמופיעה ב-ARP Cache?



### למי נשלחת שאלת ה-ARP?

היות שמסגרת ARP הינה בשכבה שנייה בלבד, ולא בשכבה שלישית, היא לא מועברת הלאה על ידי נתבים. מכאן שבדיקה ה悲אה:



אם מחשב A מעוניין לשלוח שאלת 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 הרשת, כמונו שלמדנו - כתובת IP היא כתובת לוגית שלא מלמדת בוודאות על מקום פיזי של כרטיס הרשת. 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 הרשת היא: 255.0.0.0. כפי שלמדנו בפרק שכבת הרשת/מהו מזהה הרשת של? מהו מזהה הישות?, משמעותה של מסכת רשת זו היא שהבית הראשוני שייר למשהה הרשת, ועל-כן המחשבים נמצאים באותו-Subnet. אי לכך, במקרה זה, המחשב A אכן שלוח שאלת 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, כתובת ה-IP שלו הינה, לדוגמה, 3.3.3.3? במקרה זה, מכיוון שהכתובת 3.3.3.3 אינה באותו-Subnet של מחשב A, החבילה צריכה להישלח אל ה-**Default Gateway** של מחשב A, אותו ניתן למצוא בחלוקת היוצאות מהרשת, כפי שלמדנו בפרק שכבת הרשת/מהי טבלת הניתוב של?. לכן, במידה המיועדת לטפל בחבילות היוצאות מהרשת, כתובת ה-IP של Default Gateway עשויה להיות כתובת ה-IP של ה-Subnet.

**Default Gateway**

- לשיכום, כאשר מחשב מעוניין לשולח חבילה אל כתובת IP כלשהי, מתבצעת בדיקה האם כתובת ה-IP של היעד הינה באותו-Subnet של המחשב השולח:
- אם כתובת היעד נמצאת באותו-Subnet של המחשב השולח, הרি שנitin לשולח שאלת ARP עבור כתובת ה-IP של היעד, ואז לשולח את החבילה ישירות אל כתובת ה-MAC המוחזרת בתשובה.
- אם כתובת היעד לא נמצאת באותו-Subnet של המחשב השולח, הרי שלא ניתן לשולח שאלת ARP עבור כתובת ה-IP של היעד. במקרה זה, נשלחת שאלת ARP לגילוי כתובת ה-MAC של ה-Default Gateway, ואז 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 השלישי 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= 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
>>>
  
```

The screenshot shows a Windows command prompt window running Scapy. The user has defined a packet object 'my\_packet' using the syntax 'IP(dst="www.google.com")/ICMP()'. The 'show()' method is then called on this object to display the packet structure. The output shows the IP header with destination 'www.google.com' and the ICMP header with type set to 'echo-request'. The Scapy interface shows the packet structure with its various fields and their values.

כעת נוכל לשולח את הפקטה:

```
>>> send(my_packet)
```

Sent 1 packets.

עכשו, כשאנו יודעים יותר על רמת ה-קוו, העובדה ש-Scapy הצליח לשלוח את החבילה אמורה לגרום לנו להרמת גבה. כיצד Scapy עשה זאת? איך הוא הצליח לשלוח פקטה בשכבה שלישית בלבד?

ازCIDOU 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 לשלוח את הפקטה (כשכתובות המקור היא הכתובת של כרטיס הרשת שלנו, כתובות היעד היא של הנטב, והסוג הוא IP).

אר מה אם היינו רוצים לשלוח את הפקטה דווקא מכרטיס רשת מסוים? מה אם היינו רוצים שהיא תשלוח מעל WiFi ולא מעל Ethernet? או מה אם היינו רוצים לשנות את השדות של שכבת-h Ethernet באופן ספציפי?

לשם כך, Scapy מציע לנו לשלוח 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**. שליחת 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 - שליחה עם שליטה בשדות ה-Ethernet



השתמשו בכתובת ה-IP של ה-Default Gateway שלכם שמצאתם קודם לכן. השתמשו בפקודה **ping** כדי לשלוח 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.

אם הנטב ענה לשאלתך שלכם? מדוע?

## רכיבי רשת בשכבה הקרו ובסכבה הפיזית

از למדנו רבות על שכבת הקרו, ובבמו 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 (בנהנעה שמדובר בתווך קווי ולא אלחוטי).

### Hub (רכזת)

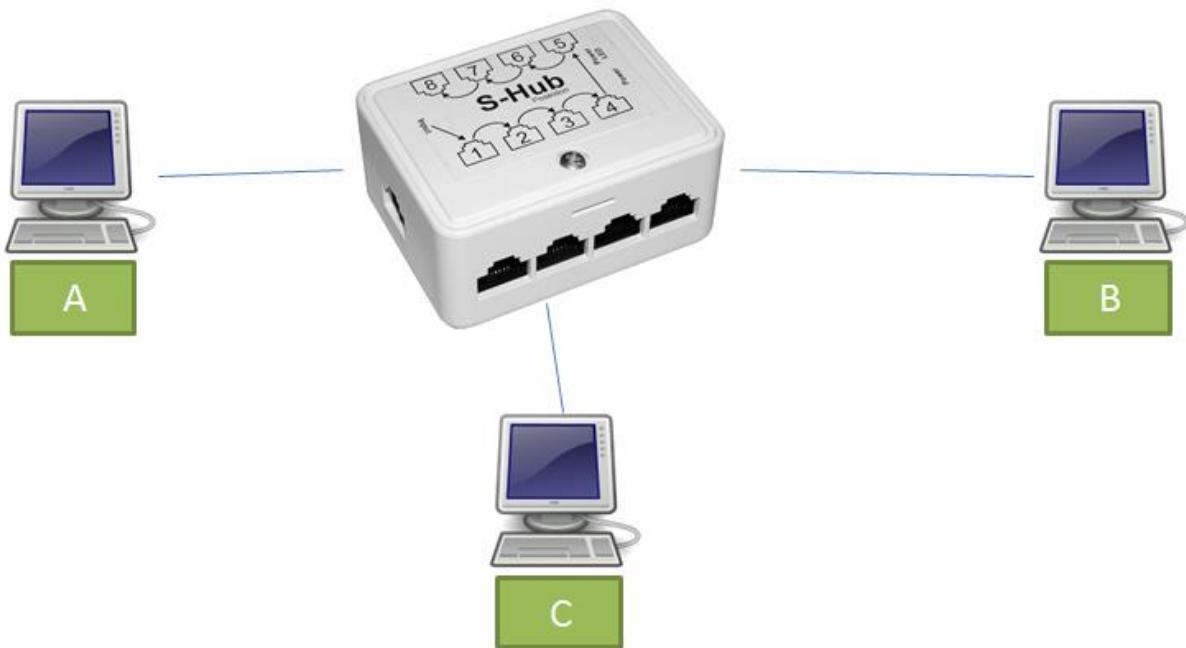
Hub (ובעברית - רczת) הינו למעשה רכיב של שכבה הפיזית, כלומר השכבה הראשונה במודול השכבות. הוא הוא "קופסה" מתוחברים מספר כבלי רשת. הוא לא יודע איך נראות מסגרות Ethernet, לא מבין מה זה כתובת MAC ולא יודע לחשב checksum. הוא רק לחבר כמה יישויות זו לזו.



כל "כניסה" ב-Hub אליה ניתן לחבר כבל רשת נקראת **פורט** (באנגלית - **Port**)<sup>74</sup>. כאשר מחשב שמחובר ל-Hub שולח מסגרת, ה-Hub מעתיק את המסגרת ושולח אותה לכל הפורטים שלו, מלבד לזה שמננו המסגרת נשלחה. כך למשל, בדוגמה הבאה:

---

<sup>74</sup> 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 - כתובות 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.

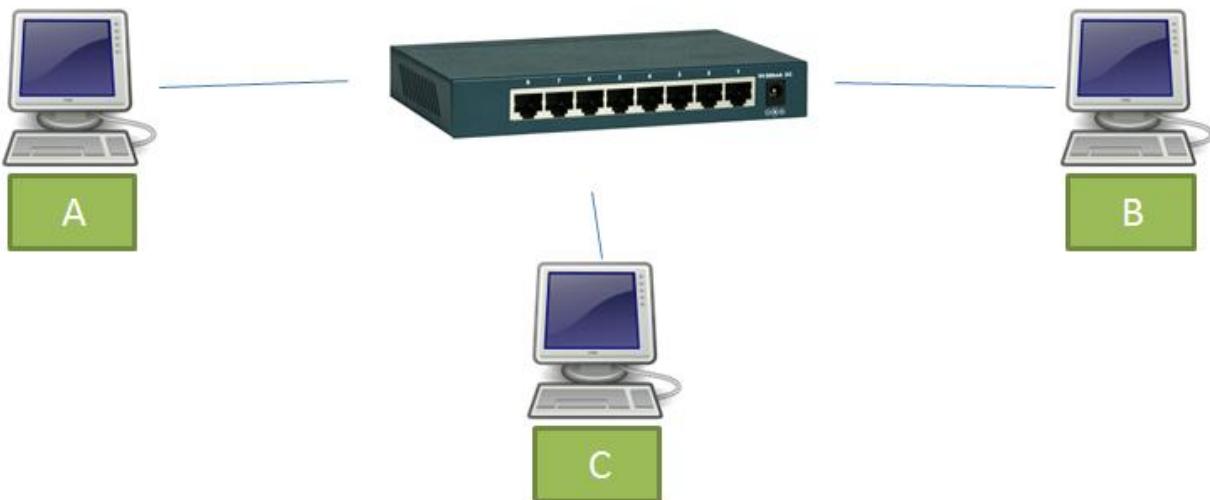
השימוש ב-Hub מאפשר אמנון לחבר מספר מחשבים זה זהה, אך יש בו בעיות רבות. ראשית, העובדה שכל המסגרות מגיעות לכל המחשבים עשויה לפגוע בפרטיות של המשתמש, שכן כרטיס רשות שלא אמור לראות את המסגרת מקבל אותה. שנית, העובדה שהמסגרות מגיעות תמיד לכל הישויות מעמיסה בצורה משמעותית על הרשות. היא מעמיסה הן על החיבורם (שבהם יכולה להשליח מסגרת אחת בלבד בכל פעם), והן על כרטיסי הרשות של כל ישות שצריכים 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 עליו, 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 checksum 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. באופןו אופן, אם המחשב B שלח מסגרת למחשב C, המסגרת תגיע אליו בלבד ולא תגיע אל המחשב A (וכМОון לא תוחזר אל המחשב B).

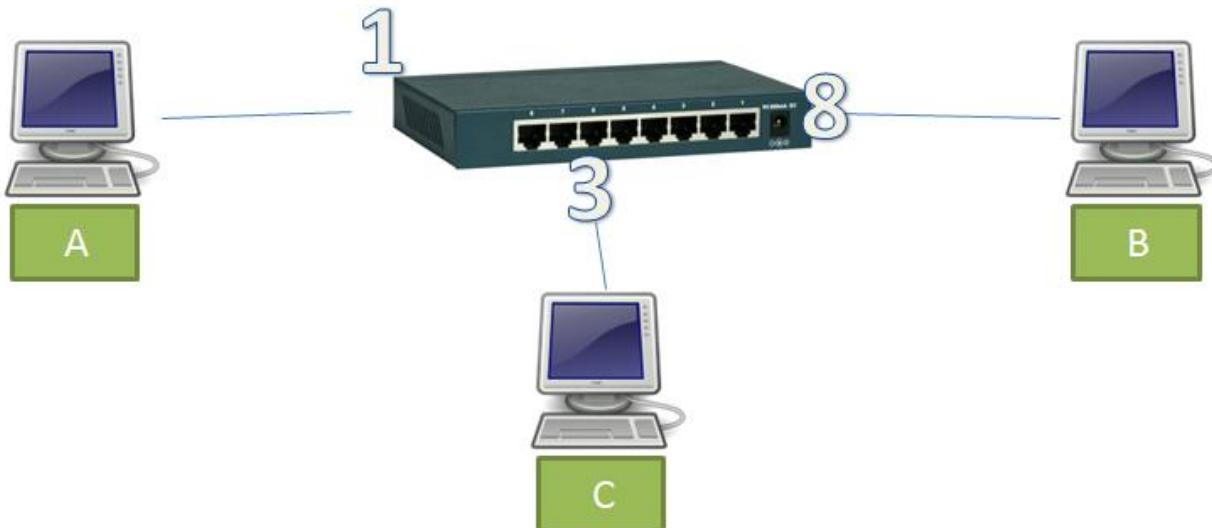
אי לכך, ל-Switch יש יתרונות רבים על פני ה-Hub. היה שכל מסגרת מגיעה רק אל היעד שלה, אין פגיעה בפרטיות המשתמשים. בנוסף, הוא חוסך את העומס הרוב שהוא נוצר לו הינו משתמשים ב-Hub. כמו כן, Switch מסייע במניעת התנגשויות, עליו נלמד בהמשך.



**كيفية عمل Switch؟**

הבנו שה-Switch יודע להעביר כל מסגרת אל הפורט המזוהה אליה בלבד. אך 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 הקרייה, 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. איבר לך, 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. איבר לך, 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 להויסף את מידע זה בטבלה:

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 רכיב 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



הביטו 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 שאלה מסתמכת על השאלות הקודומות (בשאלה מס' 3 ניתן להניח שהמסגרת משאלה 2 כבר נשלחה).

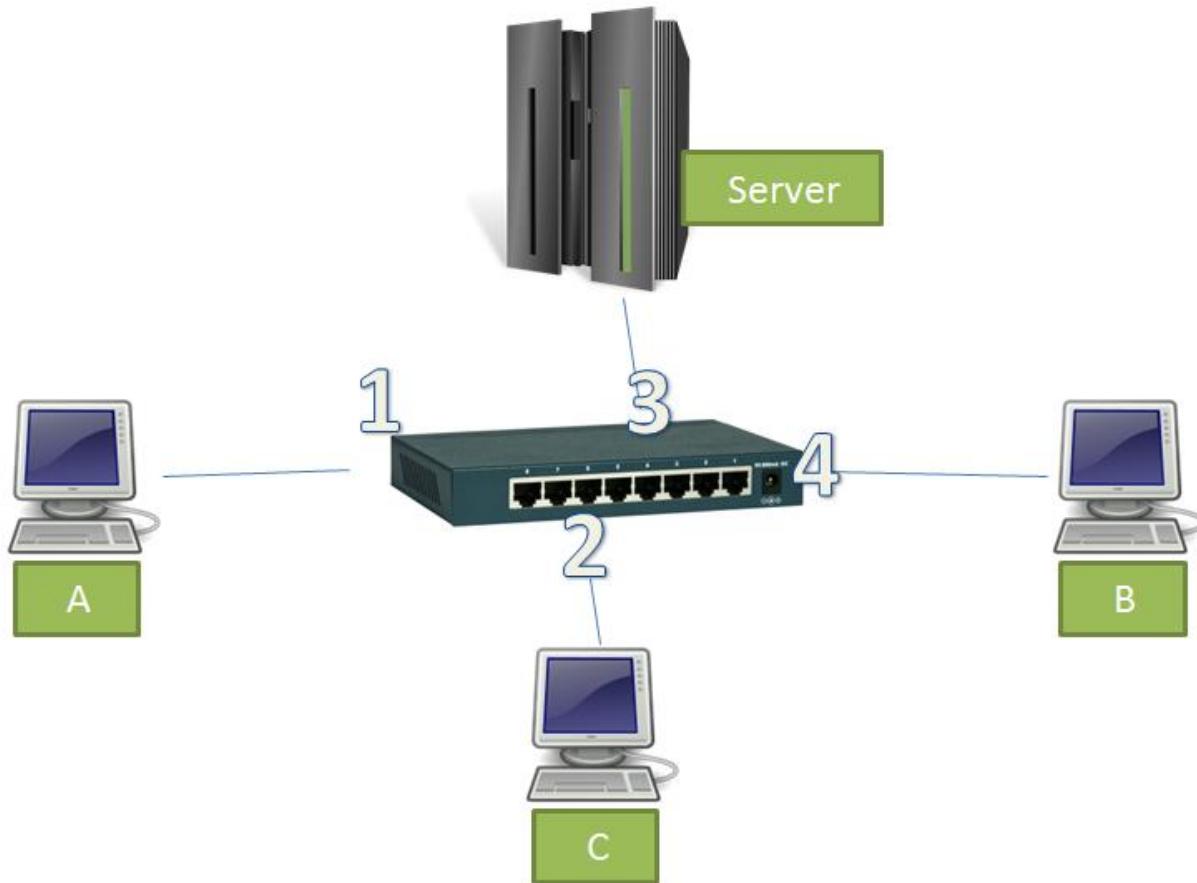
ענו על השאלות הבאות:

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 שהוצג בתרגיל הקודם, אך הפעם - הישיות השונות מחוברות באמצעות **Switch**, ולא באמצעות **Hub**:



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 שהמ桑גרת משאלת 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<sup>2</sup> - סיכום

בפרק זה הכרנו את השכבה השנייה במודל חמש השכבות, שכבת ה<sup>2</sup>. בתחילת הפרק למדנו על תפקידה של שכבה וכייד היא משלבת במודל השכבות. לאחר מכן, הכרנו את **פרוטוקול Ethernet**. למדנו על כתובות **MAC**, איך הן בנויות, וכייד 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<sup>2</sup> 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 ה<sub>0</sub>, הכרנו פרוטוקול לדוגמה של שכבה זו וראינו רכיבי רשות המאפשרים לחבר יישויות שונות. עם זאת, לא הتمודדנו עדין עם בעיה שעל שכבה ה<sub>0</sub> לטפל בה - בעיית ההתנגשויות.

כדי להסביר את הבעיה, נדמיין מקרה המוכר לנו שאינו 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 יישויות ידרשו יחד, תיווצר התנגשות (**Collision**), והמידע לא יגיע בצורה מסודרת.

### ערוץ משותף - הגדרה

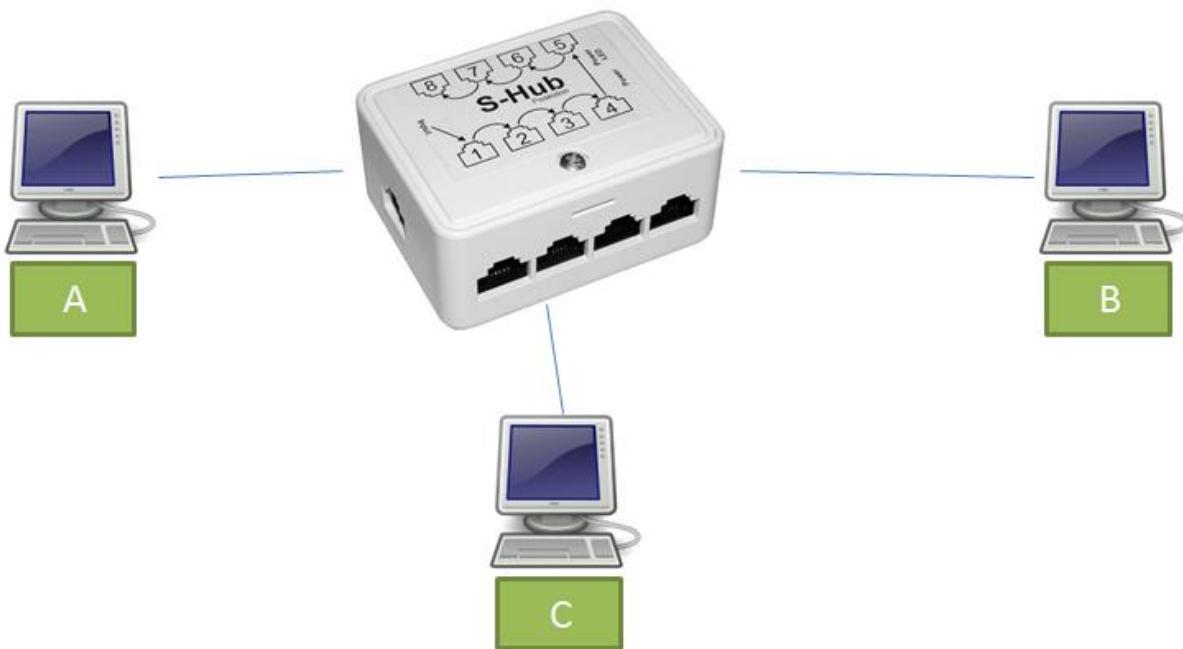
ערוץ משותף הוא קו תקשורת המחבר בין מספר יישויות ומאפשר להן להחליף מידע. המידע בערוץ מופץ ב-*Broadcast* - כלומר, כאשר ישות משדרת, המידע מגיע לכל היישויות. בנוסף, כל ישות יכולה להאזין לעזרץ בזמן שהיא משדרת - וכך לגלות האם המידע שהוא שידרה הועבר בהצלחה.

דוגמה לרשות המחברת בערוץ משותף היא רשות בה היישויות השונות מחוברות ב-*bHub*. כפי שכבר קודם לכן, במידה שברשת ישנים מספר מחשבים מחוברים ב-*bHub*, כל מסגרת שתשלוח תגיע לכל המחשבים.



### מהי התנגשות?

כפי שציינו קודם לכן, כאשר שתי ישויות (או יותר) משדרות בערוץ המשותף בו זמנית, 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 B<sup>75</sup>. במקרה זה תהיה התנגשות, והמידע שיגיע למחשב C יהיה משובש - הוא לא יהיה זהה המידע 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 ממשוערי, ויש לשדר מחדש כל מסגרת שתשלוח - אין סיבה להשתמש עוד בערוץ המשותף ולשלוח עוד מסגרות. ברור למד' כי זויה תופעה שלילית, שכן אנו מבזבזים זמן בו לא נשלח שום מידע על הערוץ.

<sup>75</sup> ישנם Hubים חכמים שיודעים להמנע מקרים כאלה ובכך למנוע התנגשיות, אך בפרק זה נתעלם מקרים אלו.



## מה תפקידה של שכבת הקו בNetworking להתגשיות?

על שכבת הקו להגיעה לנצלות מירבית של הקו - כלומר, לצמצם את הזמן המתאים עד כמה שניתן.

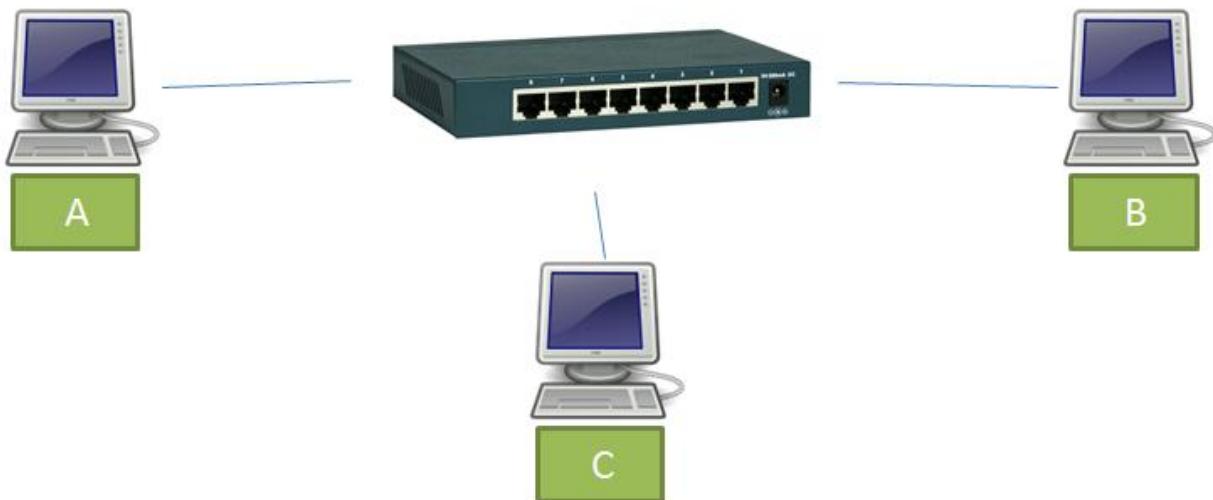
### מניעת התגשיות

ישנן דרכים רבות למנוע התגשיות, והשיטות התקדמות עם הזמן. אחד הפרוטוקולים הראשונים שניסו להתמודד עם סוגיה זו נקרא ALOHA. לפי פרוטוקול זה, כאשר ישוט מסויימת 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 קולט מהערוץ זהה למידע שאותה היא שידרה. באם המידע זהה - הכל בסדר. אך אם המידע שונה - הייתה התגששות.

במקרה של התגששות, 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 להתגששות, הייתה מחכה אותו זמן כמוותה, והייתה נוצרת התגששות נוספת. חשבו למשל על חמישה אנשים הנמצאים בחדר חסר לחילוטין. עליהם לנסות ולדבר, אך לא להתחיל לדבר יחד. אם שני אנשים (למשל: נגה ואופיר) מתחלים לדבר באותו הזמן ממש, מתרחשת התגששות. באם לאחר ההתגששות, נגה ואופיר יჩכו חמש דקות בדיק, ויתחילו לדבר מחדש - תיווצר שוב התגששות. לכן, כל אחד מהם ימתין זמן רבડומלי. למשל, נגה תמתין דקה ואז תנסה לדבר, בעוד אופיר ימתין שלוש דקות לפני שהיא תשמע את קולו. כך, השניים צפויים להצליח להעביר את המסר שלהם מבלי שתיווצר התגששות.

זהו דוגמה אחת בלבד לניסיון להתמודד עם התגשיות על ערוץ משותף. פרק זה לא נסקור דרכים נוספות, אך קוראים סקרנים מוזמנים להרחב את הידע שלהם [בסעיף צעדים להמשך של פרק זה](#).

במקום להתמודד עם התגשיות כשהן מתרחשות, ניתן גם למנוע מראש מההתגשיות. 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 לשולח את המסגרת רק כאשר הערוץ 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, בלי ניהול מורכב.

הפתרונות של שימוש ב-Switch אפשרי במקרים מסוימים (כמו רשתות Ethernet), אך לא תמיד. למשל, ברשות WiFi בה החיבור עובר באוויר, כל המסגרות מגיעות לכל הישויות שנמצאות באותו הקיליטה. במקרים כאלו, יש להשתמש בפתרונות אחרים.

## שכבה הקו - צעדים להמשך

על אף שלמדנו רבות על שכבה הקו, נותרו נושאים 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 שאלנו 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 הקו, מוזמנים לבצע את הצעדים הבאים:

### קריאה נוספת

בספר המצוין Computer Networks (מהדורה חמישית) מאת Andrew S. Tanenbaum ו- Wetherall, הפרקים השלישי והרביעי מתיחסים במלואם לשכבה הקו. 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, הפרק החמישי מוקדש כולו לשכבה הקו. כמו כן, הפרק השישי 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 להכניס את כרטיס הרשות ל"מצב פרוץ", מה שיגרום לכך שנראה בהסנפה את כל המסגרות שראהו כרטיס הרשות, גם כאלה שלא מיועדות אליו. בעת אנו מסוגלים להבין את משמעות ה-Promiscuous Mode בצורה טובה יותר. במצב זה, נראה בהסנפה גם מסגרות שהגיעו אל כרטיס הרשות שלנו מבלי שכותבת ה-MAC בשדה כתובת היעד של המסגרת תהיה הכתובת של כרטיס הרשות שלנו, או כתובת של קבוצה בה הוא חבר<sup>76</sup> (למשל מסגרת המיועדת ל-Broadcast, כלומר לכל הישויות ברשת). תcan שמסגרות כאלה יגיעו אל כרטיס הרשות שלנו, למשל, מכיוון שברשת שלנו המחשבים מחוברים באמצעות Hub. תcan גם שנראה מסגרות כאלה מכיווןשה-Switch עדין לא למד להכיר את הכתובות השונות.

האם נוכל לזהות מרוחק מחשבים ברשות שכרגע מסניפים? המסמך הבא: <http://goo.gl/2LZwgP> מຕאר שיטה לזיהוי מרוחק של כרטיסי רשות במצב Promiscuous Mode. מכיוון שבדרך כלל מי מסניף אכן משתמש באפשרות זו, נוכל להשתמש בשיטה המתוארת כדי לזהות מחשבים מסניפים ברשות.

קראו את המאמר, והבינו את השיטה המוצעת לזיהוי כרטיסי רשות שנמצאים ב-Mode Promiscuous. לאחר מכן, כתבו סקריפט באמצעות Scapy שմMESS את השיטה הזו, ומדפיס את כתובת ה-MAC של כל כרטיס רשות שהוא כרטיס שנמצא במצב Promiscuous Mode. הריצו את הסקריפט ברשות שלכם מבלי שאף מחשב מסניף, וודאו כי הסקריפט לא מתריע על כך שיש מחשבים מסניפים. לאחר מכן, הפעילו הסנפה באחד המחשבים ברשות והריצו את הסקריפט בשנית. וודאו כי הפעם הסקריפט מתריע על המחשב המסניף.

---

<sup>76</sup> על המשמעות של כתובות Ethernet של קבוצות, תוכלן 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 הishiות שMRIOT 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 מחייב מסווג IP) - הוא מעלה את המידע למי שמתפל בחבילות IP, למשל מערכת הפעלה). 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) מסוים. 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. נמיר את הבית הזה לפורמט הבינארי (ולומר, בתצוגת בייטים) שלו<sup>77</sup>:

00000010

כעת, علينا להסתכל על הבית התיכון ביותר של הכתובת (מסומן באדום):

00000010

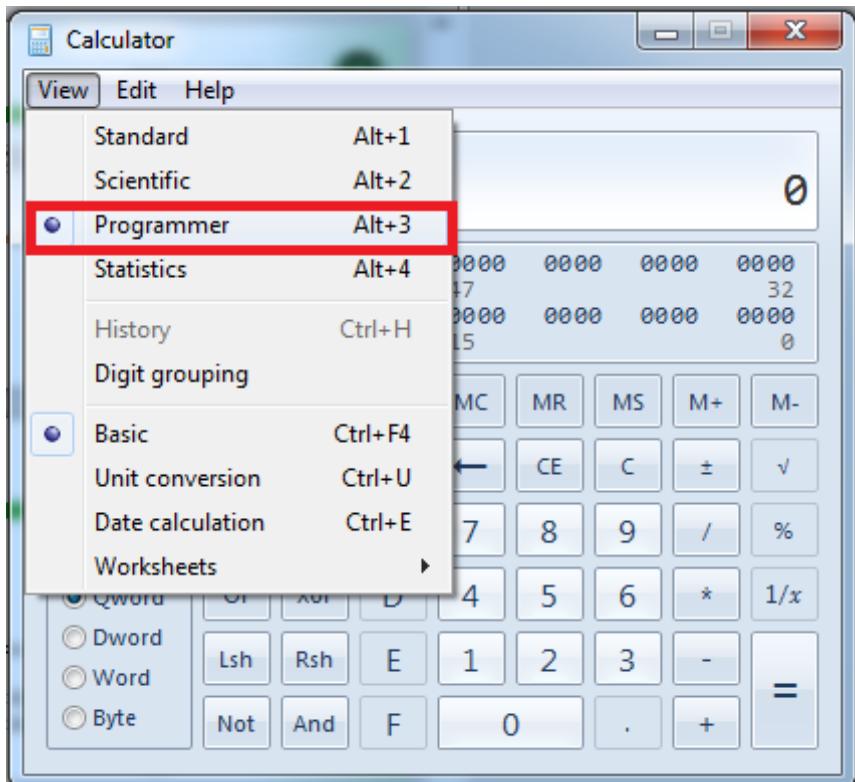
מכיון שבית זה קבוע (הערך שלו הוא 0), מדובר בכתובת Unicast.

נסתכל על כתובת כרטיס הרשות שלנו שהציגנו קודם לכן:

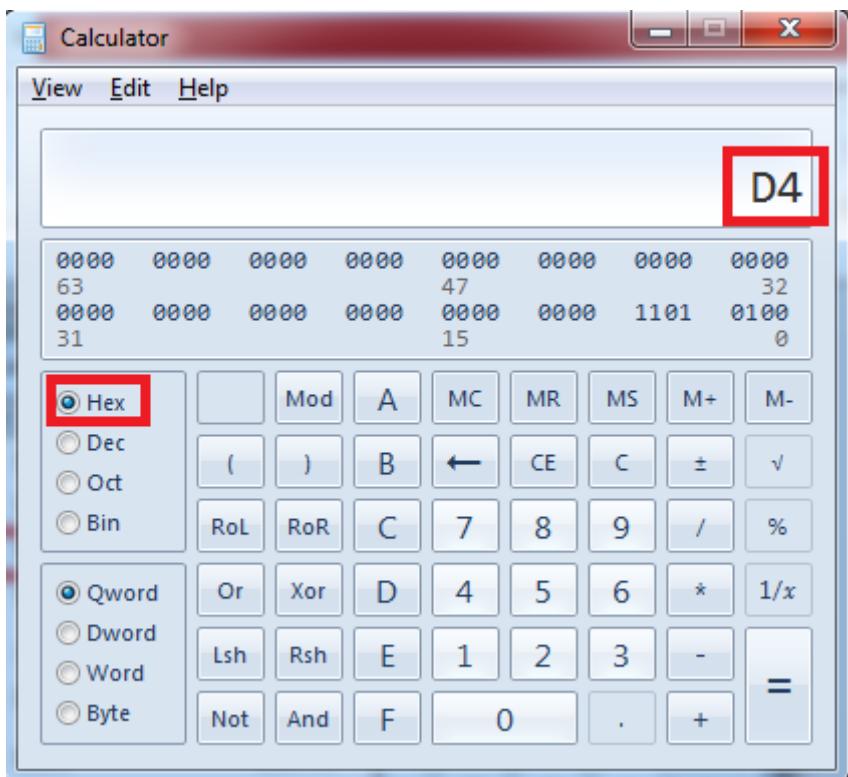
D4:BE:D9:D6:0C:2A

על מנת להבין האם כתובת זו היא Unicast או Multicast, נסתכל בbijt העליון ביותר, שהוא הבית 4. כעת, על מנת להמיר אותו לפורמט בינארי, ניעזר במחשבון של Windows. היכנסו למחשבון, ובתפריט בחרו באפשרות View->Programmer

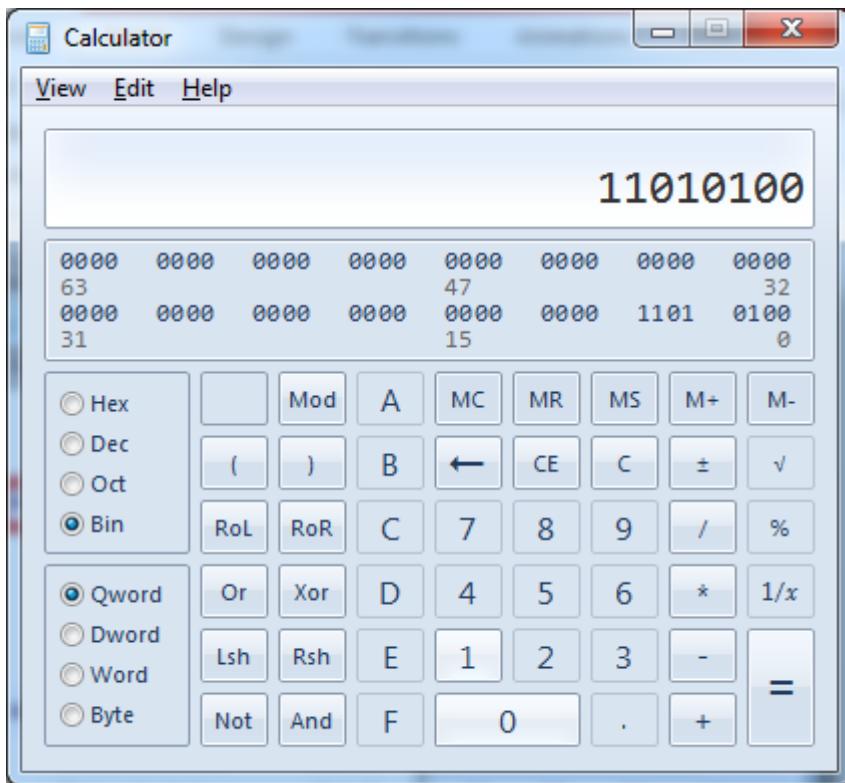
<sup>77</sup> אם איינכם מרגשים עדים בטוחים במונחים "bijits" ו-"bijits" או בהמרו בין הפורמטים השונים - אל תדאגו, הביטחון נרכש עם הזמן. עם זאת, קראו לאט וודאו כי אתם מבינים את הכוונה בדוגמאות שניתנות לפניכם.



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 בbite ה

- 03
. נבצע המרה לבסיס בינארי:  
**00000001**

הבית התיכון (מסומן באדום) דולק - ולכן מדובר בכתובת של קבוצה, כלומר כתובת FF:FF:FF:FF:FF:FF.

כתובת Multicast נוספת הינה הכתובת FF:FF:FF:FF:FF:FF. כתובת זו היא כתובת Broadcast - כלומר הקבוצה אליה שייכות כל הishiות בראשת. שליחת מסגרת עם כתובת היעד FF:FF:FF:FF:FF:FF משמעותה שליחת המסגרת לכל הishiות שנמצאות איתנו בראשת.



### תרגיל 8.16 - זיהוי כתובות Multicast

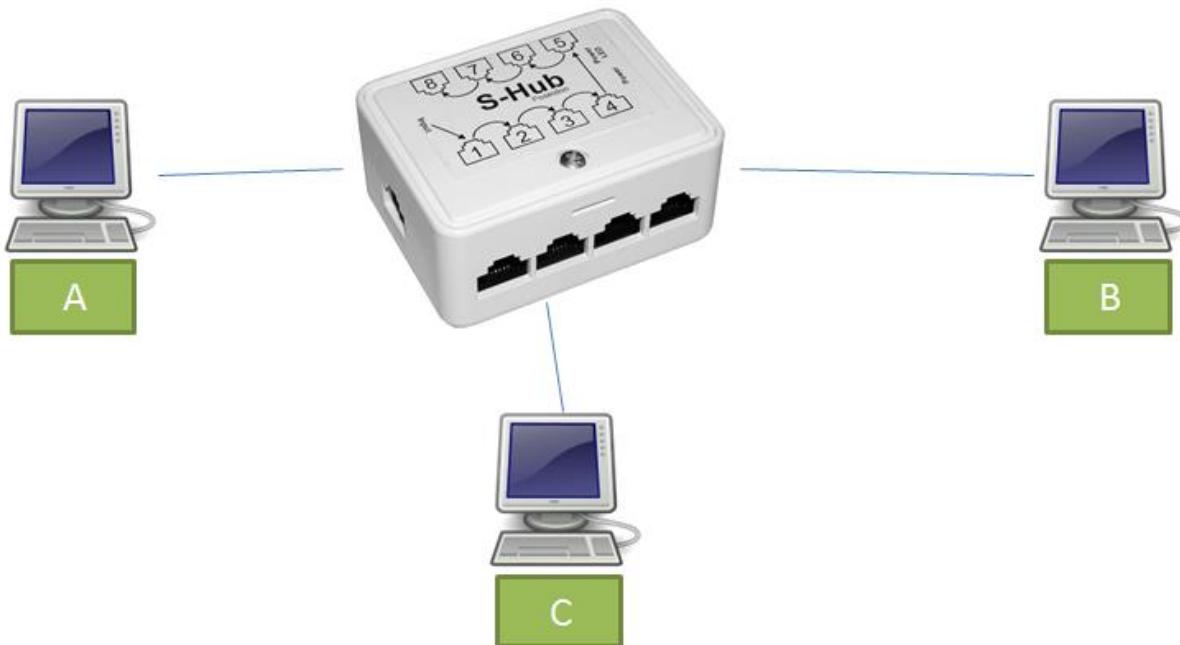
השתמשו בסקריפט שכתבתם בתרגיל 8.6 - כתובות Ethernet, אשר מבקש מהמשתמש כתובת MAC ומדפים עליה מידע. ערכו את הסקריפט כך שידפיס גם האם הכתובת היא **Unicast** או **Multicast**.

## פרק 9 - רכיבי רשת

ב פרקים הקודמים הכרנו מספר רכיבי רשת. פרק זה נועד כדי לעשות סדר ברכיבים עליהם למדנו.

### Hub (רכזת)

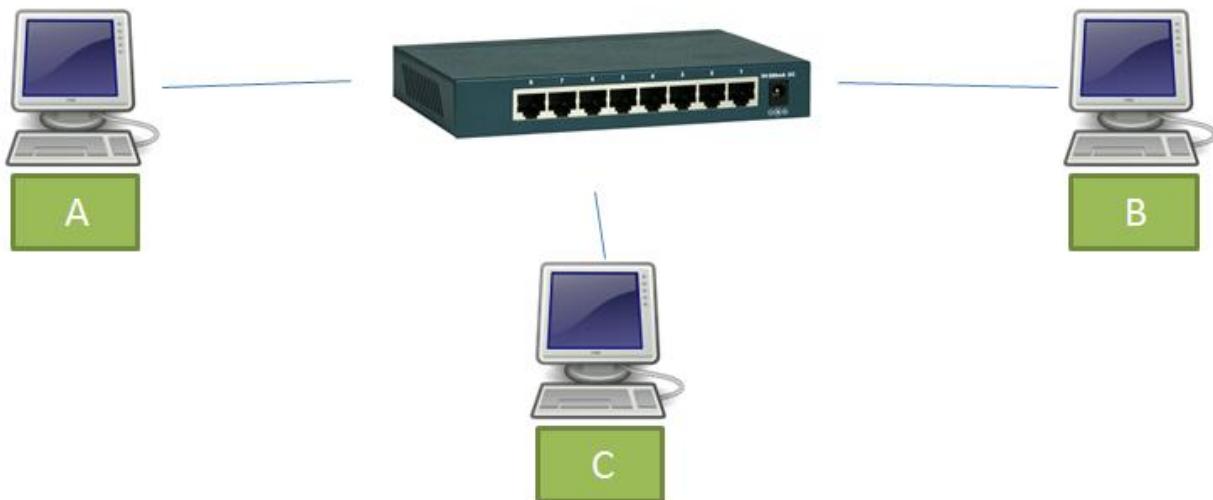
ה-Hub הינו רכיב של השכבה הפיזית, השכבה הראשונה. הוא נועד כדי לחבר כמה ישויות רשת ייחד. ה-Hub אינו מכיר כתובות Ethernet או IP, מבחינתו הוא רק מעביר זרם חשמלי מפורט אחד אל פורטים אחרים. כאשר מחשב שמחובר ל-Hub שלוח מסגרת, ה-Hub מעתיק את המסגרת ושולח אותה לכל הפורטים שלו, מלבד זהה שמננו המסגרת נשלה. כך למשל, 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)

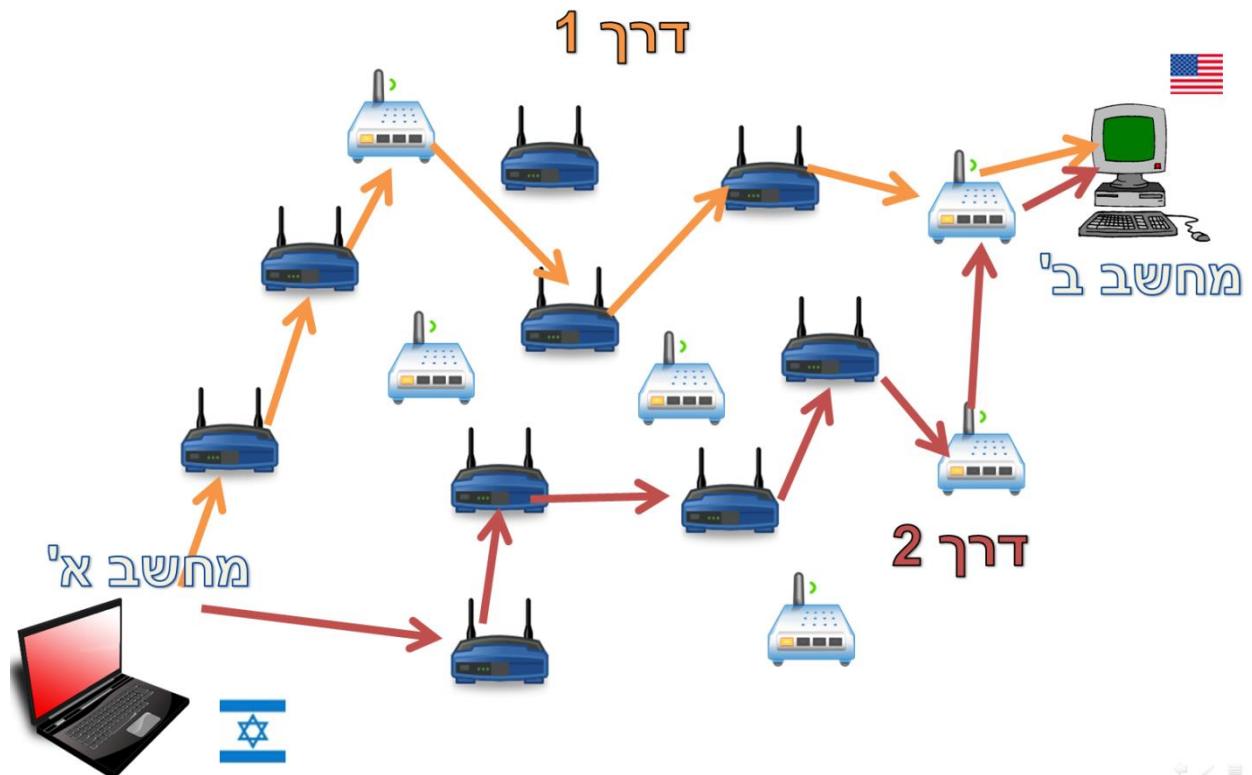
ה-Switch הינו רכיב של שכבת הקו, השכבה השנייה. אי לכך, ה-Switch מכיר כתובות MAC, מבין את המבנה של מסגרות בשכבה שנייה (למשל מסגרות Ethernet), ויודע לחשב checksum. לאחר שה-Switch למד את הרשת, הוא מעביר מסגרת מה포רט בה הוא קיבל אותה אל ה포רט הרלוונטי בלבד. בדוגמה הבאה:



המחשבים B 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 הנטבים באמצעות טבלאות ניתוב דינמיות. הטבלאות מתעדכנות באמצעות הודעות שהנתבים 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 מיועדת אליו, באמצעות טבלה המפה 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. לרבות 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 גאים 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 דיאדיסית, הרי בסך 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 הזאת.



**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 וטלויין.
- הטלפון הסלולארי.
- הדיבורית האלחוטית באותו.
- מכשיר ה-GPS.
- אפילו מנורות הרחוב, שבאופן מסוומי נדלקות בתזמון מדויק על פי השעה בה שוקעת וזרחתה המשמש.

כל אחת מהטכנולוגיות הללו מסתמכת על שיטה אחרת שתפקידה להגדר כיצד להעביר מידע בית או אוסף ביטים על גבי תווים שונים: באוויר, בכלי מתקנת ובכלי זכוכית.



**טוור (Medium)** - ברשותה תקשורת, טוור 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 תווים:

- 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 בחיננו 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 יבשתיות של חברות היי-טק גודלה.

## עמודי הטוור של התקשרות

כפי שציינו קודם, הטוור הינו החומר המוחשי בו אנו עושים שימוש על מנת להעביר מידע (למשל, כבל נוחות או האוויר). כל טוור מכתיב אופן שונה להעברת ביטים, ו מגבלות שונות על קצב העברת הביטים והמרקך אליו ניתן להעברים.

### העברת מידע באוויר

האם אי פעם שאלתם את עצמכם את השאלה הבאה:



אי 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 החשמונאים. אבותינו השתמשו באש ובעשן על

מנת לאוות ולהעביר הודעות שונות בין גבעות מרוחקות. זה אמן נשמע מוזר, אבל השימוש באש ובעשן הינו 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 להעביר אור למרחקים, ובאמצעות קידוד מוסכם מראש העבירו הודעות. דוגמה נפוצה מאוד של קידוד פשוט שמאפשר העברת הודעות באמצעות איות או היא קוד מורס.



**קידוד (Encoding)** או **Coding** - תהיליך בו מידע מתורגם לאוות מוסכמים (למשל: אור או חושך). כל שיטת שימוש של השכבה הפיזית מגדרה אופן בו מתרגם את הספרות 0 ו-1 לסימנים מוסכמים על גבי התווך.



קוד מorus שהזכרנו קודם לכן, מהו זה בעצם משיטות הקידוד הותיקות בעולמנו, ושימושם נרחבים ומגוונים נעשו בו בעבר ובווהה. קוד מorus מגדר לכל אות באלף-בית האנגליקן המורכב משני סימנים: קו ונקודה. הטבלה הבאה מראה כיצד מתרגמים כל אות באלף-בית לרצף של קויים 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 בינארי. אבל אם נעמיק ונחשוב על קידוד מorus והאופן בו משתמשים בו, נבין שיש לנו מאפיין אחד חסר.



לו היitem חיים במאה ה-19, וברשותכם פנו המאפשר لكم להعبر איתיות או רוחקים, כיצד היitem מבדילים בין מקטע או רוחק נקודה לבין מקטע או רוחק?

המאפיין החסר הוא הגדרה של זמן, או יותר נכון תזמון. ההבדל בין קו לנקודה הוא האורך, או משך הזמן של הבハוב האור. אפשר לומר שהמאפיין הוא מעט "סמי", מאחר שכל אדם יבחן בהפרש המשכים בין הבהוב 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 בשניה אחת). כמו כן, הגדרה מדויקת של משכי הקו והנקודה מאפשרת להפריד בין אותיות ובין מילים בקלות רבה יותר. הסתכלו שוב על הטללה וחשבו: כיצד נוכל להבדיל בין האות Q, לבין רצף האותיות MA?<sup>78</sup>



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 הבhbובים באורךים משתנים, בצורה דומה מאד לקידוד MOrus, אשר הטלויזיה קולעת ומפענחת. במקום לקודד אותיות, אפשר לקודד כפתורים כגון "העלה ווליום", "הורד ערוץ" ו-"כבה".

### גלים נושא מיידע

פריצת הדרכ הבא בניצול האויר כתווך תקשורת התרחשה כמעט מאות שנים לאחר שהומצא קו Morus, בסוף המאה ה-19, כאשר חוקרים הצליחו להعبر קו Morus באמצעות גלים אלקטромגנטיים למרחק של שישה קילומטרים. כדי להבין קצת מהם גלים אלקטромגנטיים, עלינו להבין ראשית מהו גל.

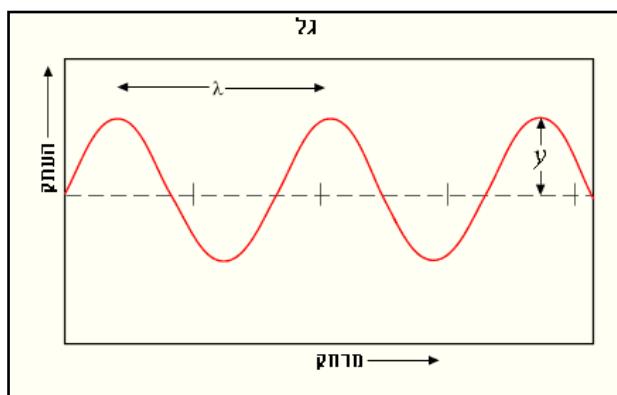


**גל (Wave)** - התפשטות (או התקדמות) של הפרעה מחזוריית בתווך למרחב. גל יכול לנوع בחומר (כמו גלים במים), אך גם באוויר (כמו גל קול) ואף בvakuum (כמו גלים אלקטромגנטיים, שיכולים לנوع בvakuum ובתווכים רבים אחרים).

<sup>78</sup> אם תרצו להרחב את היכרותכם עם קו Morus והשימושים הרבים לו, תוכלם לקרוא עליון כאן: [http://en.wikipedia.org/wiki/Morse\\_code](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 נוכל לדמיין שאם היו מסתכלים על גובה המים מהצד, היינו רואים גל שמציר פונקציית סינוס. לכל גל יש מספר מאפיינים מאוד חשובים שמתארים אותו. מאפיינים אלו ניתן לראות הסרטוט:



- כל גל הוא תופעה מחזורית (כמו פונקציית סינוס).
- **אורך הגל (wave length)** ( $\lambda$ ) הוא המרחק שהגל עובר כשהוא מבצע מחזור אחד. אורך זה מסומן בשרטוט ב- $\lambda$  (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" המksamלי 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 כgal קול), שינוי 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 שמאגים מהמשמש ושהואנו רואים הוא gal האלקטרומגנטי שהתדר שלו נמצא בתחום השעון רואה (אורך הgal הרלבנטי נע בין 400 ל-800 ננומטר<sup>79</sup> בקירות). עוצמת האור שווה לאמפליטודה של הgal האלקטרומגנטי, שמעידה על חזק התנועה בשדות

<sup>79</sup> ננומטר אחד שווה ל- $10^{-9}$  מטר.

החשמליים והמגנטיים. הgalים האלקטרומגנטיים הם תופעה מיוחדת מאוד בטבע, ותוכלו לגרום עוד עליהם בזקיפידה.<sup>80</sup>



בחזרה להעברת קוד מORS 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 מהיות יעיל (זכרו, היום כל טלפון נייד מקודד 100 mega 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 ומיצרים galים. אם תעצרו לשניה ואז תמשיכו, האם galים בבריכה יעצרו גם הם? מובן שלא! לא רק שיש להם קצב התקדמות מסוימת, הם גם מוחזרים לדפנות 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 שהתפתחו כבר בראשית השימוש בגלים אלקטרומגנטיים: הראשונה נשענת על שינוי האmplיטודה, והשנייה נשענת על שינוי התדר. אם זה לא נשמע לכם מוכך, חישבו שוב על תחנות הרדיו האהובות علينا - רובן 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.

<sup>80</sup> [קרינה\\_אלקטרומגנטית](http://he.wikipedia.org/wiki/אלקטרומגנטיות)

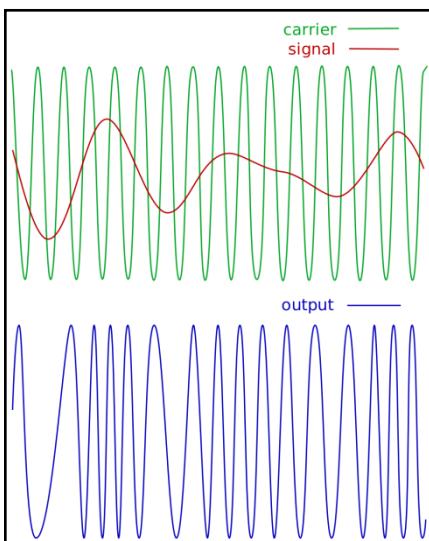
כעת נוכל להבין מה הן שתי שיטות האפנון שהתפתחו עם ראשית השימוש בಗלים אלקטرومגנטיים:

- 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, מושנים את התדר של הגל הנושא על פי האמפליטודה של גל המידע.

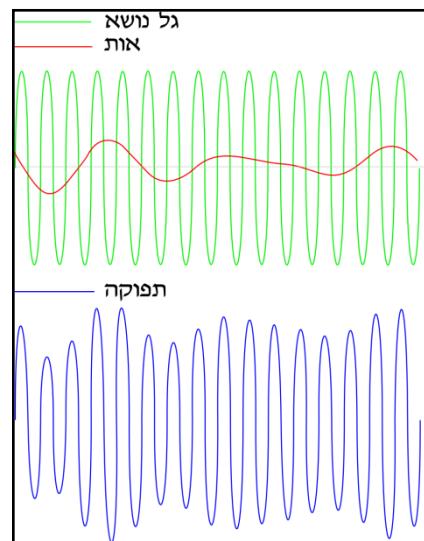


הביטו בשני התרשימים הבאים. בירוק מסומן הגל הנושא, גל בתדר גבוה וקבוע. באדום מסומן גל המידע ("אות"). בכחול, תוצאות האפנון של גל המידע בגל הנושא.izia תרשימים מראה אפנון AM ואיזה FM?

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. תרשימים ב' מראה Frequency Modulation, حيث שההתדר של הגל הכחול משתנה, בעוד האמפליטודה שלו נשארת קבועה.

השימוש בಗלים אלקטромגנטיים לצורכי העברת מידע הפתחה מאוד מאז סוף המאה ה-19, בה היה ניתן להעביר מספר בודד של נתונים בשניה. היום ניתן להעביר מאות מיליוני נתונים בשניה (100Mbps) וכל זאת על-ידי מכשיר סלולרי קטן. לפני שנבין את שאר ההתקנתיות, 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 אמרם לנו שקידוד המידע נעשה באמצעות קוד מורס.



העברת מידע באמצעות זרמים חשמליים על חוטי נחושת הניח את היסוד לתעשייה ענקית של תקשורת, שבאוף משעשע התפתחה תחת ענף הדואר. רשות הדואר בעולם פרשו חוטי נחושת בין ערים ומדינות, ואף על קרקעיה 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 כבלי נחושת. בסוף המאה ה-19 נמצא הטלפון, וסחף אחריו גל פיתוחים טכנולוגיים שנגעו אליום בהמשך הפרק.

### הרשות בבית

הבית הממוצע בישראל הוא בית מודרני מאוד מבחינה תקשורתית. זה לא מפתיע, בהתחשב בכך ש办好shi ההכרח לתקשר עם הקרובים והאהובים علينا. 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:

תווון	אמצעי	תווון	אמצעי
כבלי נחושת	טלפון	אוויר	רדיו
כבלי נחושת	פקס	כבלי נחושת	טליזיה בכבלים
כבלי נחושת	מודם ADSL או כבלים	אוויר	טליזיה בלויין
אוויר	נתב אלחוטי	אוויר	טלפון סלולרי

## קו הטלפון הביתי, או למה צריך פילטר למודם ADSL?

אחרי שלמדנו קצת על סוג תווים שונים, ועל גלים אלקטרומגנטיים, הגיע הזמן שנתמקד באמצעות התקשורת על מנת להבין איך מושגנו מושג. הטלפון משתמש בכבלי נחושת וזרמים חשמליים בשיבול להעברת קול אנושי. אם זה לא מספיק, כשתאינטרנט הגיע הביתה באמצעות שנות ה-90, הוא הגיע גם על גבי אותו כבל נחושת שנשא את קו הטלפון. אם אתם לא המומינים ברגע זה, אתם צריכים להיות(!). אין כבל<sup>81</sup> נחושת בעובי של מילימטר בודד יכול להעביר לא רק קול אנושי, אלא גם מוסיקה וסרטים באיכות HD? נחקרו את השאלה hvordan דרך הפילטר הקטן שנמצא בשקע הטלפון של משתמשי ADSL. על אף שפירוש המילה פילטר הוא מסנן, הפילטר מפעיל את השקע לשני שיקעים: אחד עבור הטלפון הרגיל, ושני עבור מודם ADSL.



פילטר למודם ADSL



air חתיכת פלסטיק מפעילה כבל נחושת אחד לשניים? ואם היא מפעילה, למה קוראים לה פילטר (מסנן)? מה יקרה אם נחבר טלפון ללא פילטר לשקע אחד, ומודם ADSL לשקע שני?

התשובה פשוטה לשאלת האחורנה היא - כשהנחבר טלפון ומודם ללא פילטר, ברגע הראשון לא יקרה כלום. אך ברגע שנרים את הטלפון לתחילת שיחה, ניצור התנגדויות והפרעות בין שני ערוצי המידע שעוברים על כבל הנחושת הדקיק: ערוץ הקול, וערוץ ה-data. כדי להבין מדוע זה קורה, נדרש קודם כל להבין קצת בראשת הטלפוניה הביתית.

### air עובד טלפון?

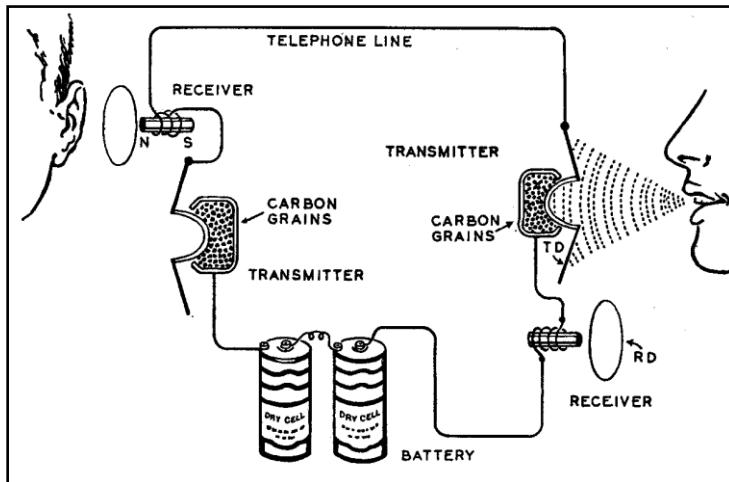
טלפון הוא דוגמה מצינית לפרק שלנו, מאחר שהוא אפליקציה לשיחה אנושית שמשתמשת בשירות שכבה הפיזית, ללא אף שכבה מתווכת. מכשיר הטלפון ממיר גל קול לזרם חשמלי ולהיפך, באמצעות מיקרופון<sup>82</sup> ורמקול. התהליך כולל את השלבים הבאים:

- 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 טוں הדיבור ותוכנו.

<sup>81</sup> לדיקנים שבינינו, כבלי הנחושת תמיד באים בזוגות. קצת כמו חיבור '+' ו '-'.

<sup>82</sup> הידעת: מיקרופון בטלפון עשוי מגראגי פחם: [http://en.wikipedia.org/wiki/Carbon\\_microphone](http://en.wikipedia.org/wiki/Carbon_microphone)

- המיקרופון בשופורת הטלפון מייצר זרם חשמלי שעוצמתו ותדירותו תואמות את עצמתו ותדירותו של גל הקול.<sup>83</sup>
- הזרם החשמלי מועבר לטלפון בצד השני, דרך מעגל חשמלי פשוט.
- בטלפון השני, עוצמת הזרם מתורגמת באמצעות רמקול<sup>84</sup> 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, מלופפים<sup>85</sup> 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 בדברים, כמו גם סנכרון הדיבור ("הלו?", "היא!", "באי!").
- הטלפון הינו דו כיווני - שני הדברים יכולים לדבר בו בזמן.

<sup>83</sup> 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.

<sup>84</sup> רמקול הוא בעצם מגנט עטוף בסיליקון ומוחובר למשטח.

<sup>85</sup> 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 תקשורת<sup>86</sup>.

המודם הוא הצעד הראשון בתהליך מהפכת המידע שמתחללת בשני העשורים האחרונים. עד המודם, אמצעי התקשרות הקיימים 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, בקצב ובמהירות גבוההים, פתחה מגוון רחב של אפשרויות להעברת מידע מכל סוג, כל עוד אפשר לקודד אותו בביטים. נראה שכל פיסת מידע שתוכלן לעלה אפשר לקודד בביטים, אבל עניינינו כעת אינו בקידוד אלא באופן בו הביטים עוברים מודם אחד אל מודם אחר.

המודמים הראשונים הקיימים ביטים בקצב מאד איטי על גבי כבל 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 מבצע 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-20KHz<sup>87</sup>.

המודם מנצל תוכנה זו של כבלי הזוגות, ומשתמש בתדרי זרם חשמלי כדי לקודד ביטים. המודם הראשון קודד 0 ו-1 על פי הطבלה הבאה:

ביט	תדר צד א'	תדר צד ב'
0	1070Hz	2070Hz
1	1270Hz	2270Hz

מאחר שהמודם לא מדבר עם עצמו, וכי לא ניתן תקשורת דו כיוונית, המודם הצד השני קודד 0 ו-1 באמצעות תדרים שונים.

<sup>86</sup> 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.<sup>87</sup> 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 איטי מאוד למחשב), הייתם שומעים<sup>88</sup> רחש קבוע.



איך מודמים התקדמו ל מהירות של  $C-aksoM$  100 המקבילות בביטחון כו? זה שיפור של פי מיליון!

השיפור בקצב התקשרות נועז במספר דרכי לניצול עיל יוטר 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 שידור הביטים, דהיינו קצב ההחלפה בין תדרים. במקום 300 חילופים בשניה לא- $aksoM$ , אפשר לבצע 600 חילופי תדר בשניה, ולהגדיל את קצב השידור ל-600bps.
- ביטול ההד הוא שיטה נוספת לניצול עיל יוטר 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. תcona זו של קו הטלפון מפריעה למודם לאחר 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 ה"הד"), שני הצדדים יכולים להשתמש באותו תדרים, ובכך מגדילים את סך כל כמות התדרים שאפשר להשתמש בהם.
- סינון רעשיות ותיקון שגיאות - רעים בקו מייצרים שגיאות תקשורת (שגיאה היא כאשר צד אחד שולח בית 0, והצד השני מקבל בית 1, או להיפך). 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 ויעבוד אותות מתקדם.

<sup>88</sup> [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 (Hz 50 עד 20kHz) למרחוקים 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 מ-20kHz, שמאפשרים קצב העברת מידע גבוהים מאוד, וזאת מוביל לסייע מחסורן האורך המגביל של כבלי הזוגות. כך התפתח מודם ה-ADSL, שהגעתו לכל רחבי המדינה נעשתה בהדרגתיות עקב תחיליך התקנת המרכזיות<sup>89</sup> 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, ומקודם אותו באמצעות תדרים קבועיםüber 0 וüber 1. לכן המודם הוא מכשיר דיגיטלי.

בקידוד **אנלוגי**, טווח ערכי המספרים שאפשר להעבירה הוא רציף. שעון אנלוגי הוא שעון אשר מודד את הזמן באמצעות קפיץ מתוח שמשחרר בזמן מודוד 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 להעבירה הוא בדיד וסוף. שעון דיגיטלי מtabsoס על קויסטול שמשחרר זרם בפרק זמן קבועים ומדודים. השעון הדיגיטלי סופר כפולות של הזרמים הבודדים, ולכן מקדם את השעה בפרק זמן קבועים ובודדים (לדוגמה נאנו שנייה).

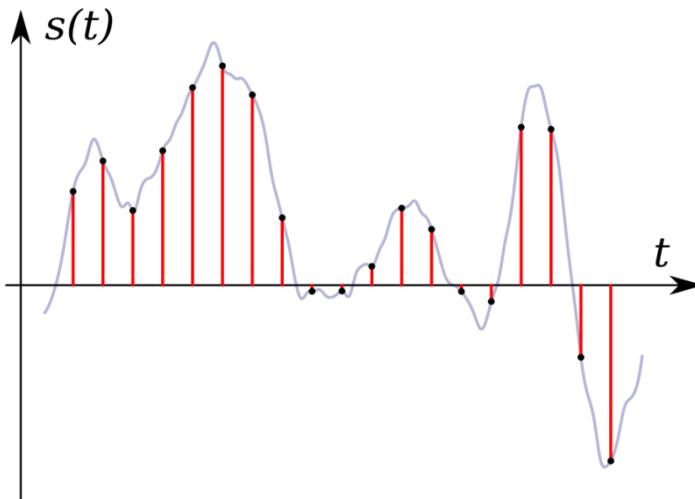
דוגמא שמחישה היטב את ההבדל בין אנלוגי לדיגיטלי היא ההבדל בין תקליטים לדיסקים.

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. המהלך בטופון עברת מהחריתות בצורה גל הקול שבתקליט, ומשחררת את הצליל המקורי. אפשר לדמיין שהתמונה

---

<sup>89</sup> מרכזיות אלו נקראות מרכזיות DSLAM - [http://en.wikipedia.org/wiki/Digital\\_subscriber\\_line\\_access\\_multiplexer](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 בדיקן אותו דבר<sup>90</sup>.



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 שנמדדה. אפשר לראות תהליך זה בתרשימים לעיל, לפי הנקודות השחורות שהן הדגימות, והקוויים 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 אט הגל בנקודות יותר צפופות), יהיה יותר קל לשחזר את הגל המקורי.



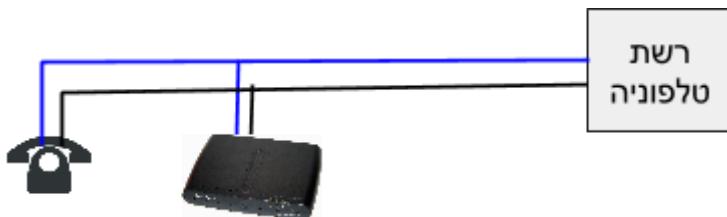
הסיבה שקובץ קול (או ידאו) בקידוד 192kbps הוא באיכות טובה יותר מאשר קובץ בקידוד 128kbps, נעוצה ביכולת להעיבר יותר נקודות של גל הקול, ובכך לחדר את הרכבת גל הקול המקורי בעט ניגון הקובץ.



מה קורה כשמחברים טלפון לשקע אחד ומודם ADSL לשקע נוסף?

<sup>90</sup> ניתן לקרוא עוד על תקליטים ופטפונים בקישור: [http://en.wikipedia.org/wiki/Gramophone\\_record](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 המכים מוחברים לאותו מעגל, הזרמים החשמליים שהם מייצרים מגיעים לשנייהם, ולכן יש סיכון  
שהם יפריעו אחד לשני.



עכשו אפשר להבין למה צריך את הפילטר בשקע הטלפון ומה בדיק הוא מסכן!

תפקידו של הפילטר הוא להפריד ולסנן תדרים.

- הטלפון והמודם מוחברים 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 מקבל 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 המכים אחד לשני. כל עוד יש הפרדה מלאה בתדרים, אין בעיה. אם מוחברים 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 ביןימ

עד כה למדנו על עמודי הטווח ההיסטוריים של השכבה הפיזית:

- תקשורת מבוססת אור (מדורות, מօר במערכות 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 (אורן 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 פועלם. 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 כבל קווקסיאלי (coax)	טליזיה
AIN DECT	אנלוגי דיגיטלי	כבל זוגות גלים אלקטרומגנטיים	טלפון
V.34	דיגיטלי	כבל זוגות	פקס
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 לרוב רגש יותר (סודות עסקיים, כספים, וכו') מהמידע ברשות הביתית.

כדי להבין מעט מפתרונות התקשרות המסחריים ננסה ללמוד על חיבורו הרשות במשרדים בגודל שונה.

## משרד קטן

באו נדמיין שאנו מנהלים חברת לפיתוח יחס ציבור באינטרנט. החברה חמשה עשר עובדים שימושיים את מרבית הזמן בפעולות תקשורתית ברשות חברותiot. החברה מספקת לכל עובד שולחן כתיבה, מחשב שולחני, מסך גדול ומקלדת. כמו כן, בחברה יש שרת קבצים המ אחסן תיקיות 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 נמצאים במשרד אחד מהווים מקרה ד' פשוט עבור רשות מקומית מבוססת switch. נזכר sh-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.

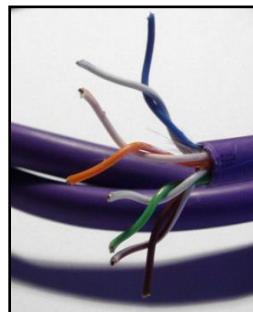
השכבה הפיזית שמתוחת לפרטוקול Ethernet היא כבל הרשות הסטנדרטי. תשמדו לדעת שכבל זה נקרא כבל CAT 5, החיבור בקצה שלו נקרא RJ-45, והתקן שמנגדיר את השימוש בכבל נקרא Base-T10. הרבה מושגים לככל כמה פשוט – כתע נגידיר אותם בצורה מסוימת:



**כבל CAT 5** – כבל שמאגד בתוכו 4 כבלי זוגות, כל זוגצבע שונה. אפשר לראות את כבלי הזוגות המלופפים סביב עצמן בתמונה הבאה. אם לדיביך, ישנים מספר סוגים כבליים כאלה: CAT 3, CAT5, CAT5e, CAT6, CAT6. מבחו צולם נראה אותו הדבר, אך מבפנים הם נבדלים באיכות בידוד ההפרעות החשמליות. איכות

הbidod משפיעה על קצב העברת הביטים. כשאתם הולכים לחנות לקנות כבל רשת, ברוב המקרים תצטרכו כבל

<sup>91</sup>CAT5.



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-45<sup>92</sup>. עבור כבלי הרשת Ethernet, הטקן הוא RJ-45<sup>93</sup>, אך ישנו טקנים דומים גם עבור כבלי אחרים כגון כבל הטלפון (RJ-11).



חיבור RJ-45

בדרכ נדבר על הטקן 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 אונ או כיווני אך לא סימולטני. דוגמא למערכת Half Duplex היא ווקי-טוקי (מכשיר קשר אלחוטי), בו רק צד אחד יכול לדבר בזמן שהצד השני מקשיב. כשהשני הצדדים מנסים לדבר, אף אחד לא שומע את השני. מערכת שהיא Full Duplex מאפשרת לשני צדדים לתקשר אחד עם השני בו אונ או סימולטני, כלומר שני הצדדים יכולים לדבר באותו הזמן. הטלפון הוא דוגמא למערכת Full Duplex, לאחר שהיא מאפשרת לשני הצדדים לדבר בו זמינות וגם לשמוע אחד את השני.

<sup>91</sup> ניתן לקרוא עוד על כבלי הרשת בקישור: [http://en.wikipedia.org/wiki/Category\\_5\\_cable](http://en.wikipedia.org/wiki/Category_5_cable)

<sup>92</sup> ניתן לקרוא עוד על טקן RJ RJ בקישור: [http://en.wikipedia.org/wiki/Registered\\_jack](http://en.wikipedia.org/wiki/Registered_jack)

<sup>93</sup> יש לציין שחבר זה נקרא גם חיבור P8C8, ובמקומות בהם הוא נקרא, הכוונה היא לאווטו סוג לחבר כמו RJ-45.



**תקן Base-T10** - תקן זה מגדיר כיצד משתמשים בcabלי CAT5 וחיבור 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 (תדרים<sup>94</sup> וקידוד<sup>95</sup>), כיצד קובעים את קצב התקשרות, והאם התקשרות היא Full Duplex או Half Duplex. תקן Base-T10 הוא רק אחד משפחחת תקנים הנקראים Twisted Pair Ethernet Over Twisted Pair (הינו המונח האנגלית 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		

טבלה<sup>96</sup> המפרטת את השימוש של כל תת-cabל בתחום cabl CAT5

(TX - Transmit, RX - Receive)



תקן שמשמעותם פעם את המושג "cabl מוצלב", כאשר ניסיתם לחבר שני מחשבים ישירות אחד לשני עם cabl רשת. בכרטיסי רשת ישנים יותר, לו היו מחברים שני מחשבים עם cabl רשת רגיל, הם לא היו מצליחים לתקשר.



התבוננו בטבלה פירוט תתי הcabלים cabl CAT5. האם תוכל לחשב על סיבת מודיעע חיבור ישיר של שני מחשבים לא יယובוד?

המושג "cabl מוצלב" מרמז על התשובה. שימו לב שהcabל הירוק מסומן כcabl "שידור" (Transmit - TX), והcabל הכתום מסומן "קליטה" (Receive - RX). אם נחבר שני מחשבים לכабל אחד, הם ינסו לשדר אותות על אותו cabl, ובcabל הקליטה לא יעברו אף מידע. כאשר מחברים cabl בין מחשב ל-Switch, Switch קורא מידע מהcabל הירוק וכותב מידע אל הcabל הכתום. מכאן אפשר לנחש שגם cabl מוצלב פשוט מצליב בין cabl הירוק לבין cabl הכתום:

<sup>94</sup> ניתן להרחיב על התדרים בהם נעשה שימושocabli רשת: <http://en.wikipedia.org/wiki/Baseband>

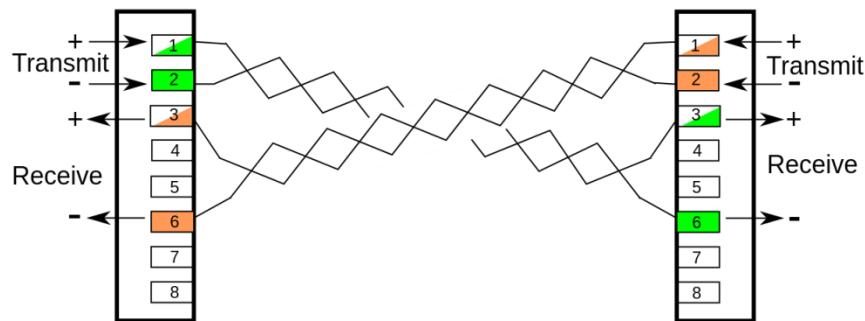
<sup>95</sup> כדי להבין קידוד ביטיםocabli רשת, קראו עוד על קוד מנציגטור: [http://en.wikipedia.org/wiki/Manchester\\_code](http://en.wikipedia.org/wiki/Manchester_code)

<sup>96</sup> התרשים המקורי: [http://en.wikipedia.org/wiki/Ethernet\\_physical\\_layer](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 השילוחה והקבלה, ולכן כמעט אין יותר צורך בכבלים מוצלבים.



**cabל רשת מוצלב (Ethernet Crossover Cable)** - הינו cabל רשות בו כבלי הזוגות של השילוחה וקבלת הוצלבו, דבר המאפשר לחבר שני מחשבים ישירות אחד לשני, ללא Switch ביניהם. ניתן לראות את ההצלה בין הcabלים בשרטוט שולחן:



## משרד גדול

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 גדלה ומתרפרפת על פניו מספר בניינים? או במקרה יותר מורכב, החיבור שלנו היא בעצם מפעל שפירושו על שטח די גדול, ובתוכו נדרש להעביר תקשורת מקצת אחד לקצת השני? חברת פרטיט, שאינה חברת תקשורת כמו בזק, או הווט, אינה יכולה להרשות לעצמה למשוך cabלי תקשורת למרחקים ארוכים מאחר ומדובר בתהיליך יקר וממושך. במקרים כאלה, נדרש פתרון אחר שמחזר אותנו לתקשורת האלחותית.

בחצי הראשון של פרק זה פגשנו בתקשורת אלחותית בשלט הרחוק. [בנוסף א' של פרק זה](#), אתם מוזמנים להרחב על הרשת האלחותית הביתה. טכנולוגיות אלו של תקשורת אלחותית סובלות מרוחק תקשורת מוגבל ומקצב יחסית נמוך. כדי לחבר שני בניינים בהם שירותי או מאות מחשבים, נדרש פתרון תקשורת שייעבור מרחק של מאות מטרים ועד קילומטרים בודדים, ויעביר קצבים גבויים של מידע (כאלו שעומדים בקצבית רשת Gigabit Ethernet). תקשורת מיקרוגל היא הפתרון שוננה לצרכים האלו.



**תקשורת מיקרוגל (Microwave Transmission)** - העברת מידע באמצעות גלים אלקטרומגנטיים בטווח אורך גל שנitinן למדود בסנטימטרים. כפי שלמדנו קודם, אורך הגל ותדרותו קשורים ביחס הפוך, ולכן ניתן להסיק שגם מיקרוגל הם גלים בטווח התדרים בין 1GHz ל-30GHz.



למי שאינו מבין מוסיף בפיזיקה או הנדסת חשמל, גלי מיקרוגל יכולים להישמע כמו טווח תדרים מאוד שרירותי:

למה דוקא גלים בתדר בין 1-30GHz 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 (שצורך, הוא גם גל אלקטромגנטי), פנס ממוקד מאריך למשך גדול יותר מאשר נורה "עגולה". באותה 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 הקולטת<sup>97</sup>.

לסיכום, לגלי מיקרוגל יש תכונות מיוחדות המסייעות בהעברת מידע בקצבים גבוהים ולמרחוקים ארוכים (כל עוד הם "ישראלים" ולא מחסומים). שיטת אפנון הביתים בגלי מיקרוגל דומה לשיטה שלמדנו בראשית הפרק, המתבססת על שינוי תדר.



בתמונה לעיל מצולמת אנטנת "טופ", המשמשת לתקשורת מיקרוגל. אפשר למצוא אותה לרוב על עמודי אנטנות סלולריות, ולא בגל שהיא מתחברת עם מכשירים סלולריים, אלא בגל שהיא מאפשרת תקשורת בין אנטנות לבין מרכזי התקשורת של חברות הסלולר.

---

<sup>97</sup> אם תחשבו לרגע, גלי רדיו וגלים עוביים מרוחקים ארוכים בהרבה, עוקפים הרם וגבועות וככנים לבית דרך הקירות. אין זה כך בתדרי המיקרוגל.

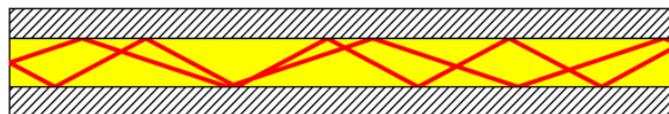
## ספק תקשורת

ראינו שמשרדי קטן עד בינוי יכול להסთפק באוסף Switchים ו-Routerים, ושהבראה יותר גדולה יכולה להיעזר בתקשורת מיקרגול על מנת לחבר משרדים או אתרים מרוחקים. עם זאת, התעלמנו עד כה חלק גדול מאוד בשרשרת התקשרות שבה אנו משתמשים כל יום: ספק התקשרות. ספק התקשרות כוללים גם את ספקי התשתיות (בזק, הוט, חברות הסלולר) וגם את ספקי השירותים (ISP'ים<sup>98</sup> לMINIHEM). חברות אלו מעבירות כמויות בלתי נתפסות של נתונים בכל שנה, ולמרחוקים בין לאומיים ארוכים מאוד. הפתרון 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, גועץ בכך שהוא "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:

- נדרשת כמות מעטה מאוד של אנרגיה בשכיל להעביר אוור דרך הסיב: הבוהב קל של נורת LED לצד אחד מאפשר להעביר את האור למרחוק של עשרות קילומטרים (לשם השוואה, דמיינו כמה רוחק אתם יכולים להאיר עם פנס-LED חזק ביותר שלכם).
- מידע לא נכנס ולא יוצא מהסיב - אין הפרעות בתקשורת ולכן ניתן להגיע לקצבים גבוהים מאוד של תקשורת. לשם השוואה, קצבי המידע אליהם ניתן להגיע באמצעות סיבים אופטיים מגעים לש-1TB/s, למרחוק של מעל 100 ק"מ. בקצב זה אפשר להעביר 12.5 Terabit/min של TB-1 בשניה!

אפנון בית על גבי גל אוור בתוך סיב אופטי נעשה בצורה פשוטה, שדומה יותר להבהוב - אוור חזק מייצג 1, אוור חלש מייצג 0. כמו כן, ישן מספר דרכי יצירתיות להגברת קצב שידור הביטים. דרך אחת היא כMOVן להגברת את קצב ההבהוב, 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 LEDs בצבעים שונים ובמספר גלי אוור (שריגשים לצבע יחיד), נוכל להכפיל את קצב שידור המידע. דרך שלישית היא פשוט להעביר

---

.ISP - Internet Service Provider<sup>98</sup>

מספר סיבים אופטיים בלבד: אם כבר מושכים לקבל למרחוק, אפשר לקבץ עשרה סיבים אופטיים בלבד ולחזור אותם בפעם אחת.

שאלה נוספת שיכולה לעלות לגבי סיבים אופטיים למרחוקים ארוכים היא כיצד מושכים סיב למרחוק גדול מ-100 ק"מ? התשובה לכך פשוטה, והוא דומה כמעט לכל אמצעי התקשרות: מסרים. **מסר (Relay)** הוא רכיב שמקבל את תקשורת, מגביר אותה ומשדר אותה להלאה. מסר אוור מקבל את האותות מסיב אופטי אחד, ומיצר אותם מחדש בסיב האופטי השני. במקרה שתהיתם, מניחים מסרים גם על קרקעית הים, בשביל למשור כבלים תת-ימיים.

לסיכום, סיבים אופטיים הם תווך יעיל ביותר להעברת מידע, והם אחראים להעבירה את מרבית תקשורת הנתונים בעולם. אמנם השתמשנו בספק תקשורת צרכנים 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 הדריכים השונות בהן אפשר להעיר את יחידת האינפורמציה הבסיסית: הבית.

למדנו על תקשורת קוית (טלגרף, טלפון, סיבים אופטיים) וגם על תקשורת אלחוטית (שלט רחוק ותקשורת מיקרוגל), ועל האופן בו סוג תקשורת השונים משתמשים 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**. לתוווכים שונים ושיטות קידוד שונות מוכתב **קצב** אחר. הקצב של ערוץ תקשורת נקבע על פי הנקודות המקסימלית של ביטים שנייתן להעביר על גבי הערז בשניה אחת. אך לא רק הקצב משתנה משיטת תקשורת אחת לשניה, אלא גם  **מרחק השידור**: כמה רחוק אפשר להעביר את הביטים. מאפיין נוסף של ערוצי תקשורת הוא אופן **הסנכרון**: האם שני הצדדים המתקשרים יכולים לשדר בו זמינות, ואם כן איך דואגים שביטים יועברו בשני היכוונים 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. בעזרתו המושגים והדוגמאות שuberנו עלייה בפרק זה, קיימים בידיכם הכלים להבין קצת יותר טוב את אמצעי התקשורת הסובבים את חיינו.

במהלך הפרק, נתחנו מספר דוגמאות מהחיים. התחלנו במבט אל העבר בו נסדו שיטות שונות להעברת מידע, ולאחר מכן התבוננו באמצעות תקשורת 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 כ망שך 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 - עידן<sup>99</sup>), שידורי הלוויין, שלטי טלזיה, דיבוריות Bluetooth ועוד. בכל רשת אלחוטית, ישנו מספר מאפיינים שמספרדים אותה מהשarra:

- טווח התדרים בו הרשת יכולה לשדר.
- עצמת האות המשודר וכיוונו למרחב.
- קצב התקשרות.
- שיטת הסyncrown והתזמון בין רכיבי הרשת.

הרשת האלחוטית הביתית, בשמה הנפוץ WiFi, מتبוססת על פרוטוקול 802.11 אשר גדר ע"י מכון IEEE<sup>100</sup>. פרוטוקול זה מכיל תתי-פרוטוקולים רבים: a, 802.11b, 802.11g, 802.11n, 802.11ac ועוד. תתי ה프וטוקולים שונים זה מזה במאפייניהם הנ"ל, אך כולם משמשים לאותה המטרה – חיבור אלחוטי בין מספר מחשבים שהמרחיק ביניהם לא עולה על כ-15 מטרים.

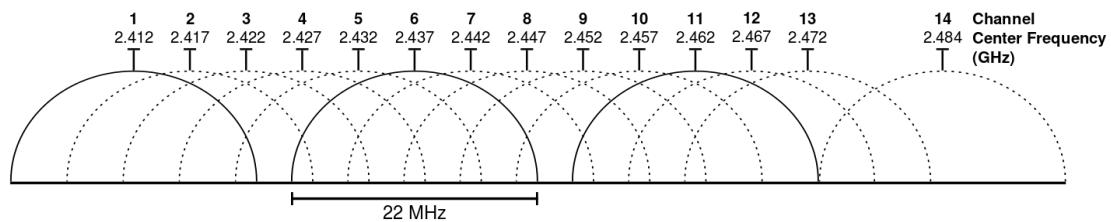
הרשותות האלחוטיות משדרות בתדר 2.4GHz (תתי-פרוטוקול b ו-g) או 5GHz (תתי-פרוטוקול a). תדר של 5GHz מאפשר קצב שידור גובה יותר, בעוד, בעוד תדר 2.4GHz מאפשר מרחק שידור גובה יותר<sup>101</sup>.

רשת WiFi אינה משתמשת בתדר המדויק, אלא בתדר קרוב לתדר שציין לעיל. התדר המדויק 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 בערך אחד אשר מכיל טווח תדרים של 22MHz. לדוגמה, רשת אלחוטית g, 802.11bg, בערך מס' 6, משדרת בתדרים 2.4GHz עד 2.426GHz. כמו כן, בין הערוצים השונים יש חיפפה, ולכן רשותות אלחוטיות המשדרות בערכאים קרובים עלולות להפריע אחת לשניה. התרשים הבא מראה את הערוצים של רשת WiFi בתדר 2.4GHz. ניתן לראות בתרשימים שלכל ערך יש תדר מרכזי, וקשת המיצגת את טווח התדרים בהם הערך עשויה שימוש.

<sup>99</sup> עידן+: שידורי טלזיה אלחוטיים דיגיטליים. ניתן להרחיב בקישור הבא: <http://goo.gl/xtwlca>.

<sup>100</sup> מכון IEEE הוא מכון בין לאומי שתפקידו לכתוב ולתזקק תקנים המאפשרים לייצר מוצרים שיעבדו אחד עם השני בתיאום. פגשנו גם בפרק שכבת הקו את ארגון IEEE, אשר הגדרת את תקן Ethernet, סמסטרו 802.3. ניתן להרחיב: [http://en.wikipedia.org/wiki/Institute\\_of\\_Electrical\\_and\\_Electronics\\_Engineers](http://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers).

<sup>101</sup> כאשר התדר נמוך יותר, הגל מבצע פעוטות מחזורים בשניה, וכך אורך הגל גדול יותר. ככל שאורך הגל גדול יותר, כך טווח 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 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?  
ניסינו 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 Scapy ו-Wireshark, והכרנו רכיבי רשת שונים. עכשו, מוצדים 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 לאפשר באינטרנט? לשם כך נשאל הרבה שאלות. נסו לחשב על התשובות, ובדקו האם אתם מצליחים לספר **את הסיפור בעצמכם**.

הסיפור שלנו מתחילה עם המחשב שלנו:



בתור התחלה, המחשב שלנו יצרך לדעת כל מיני פרטים. הוא צריך לדעת מה כתובת-h-IP שלו, כדי שיוכן לשלווה אחר כך חבילות נוספות. הוא צריך לדעת מה מס' כתת הרשות שלו, כדי לדעת איזה מחשבים נמצאים איתו ב-Subnet ואילו לא.

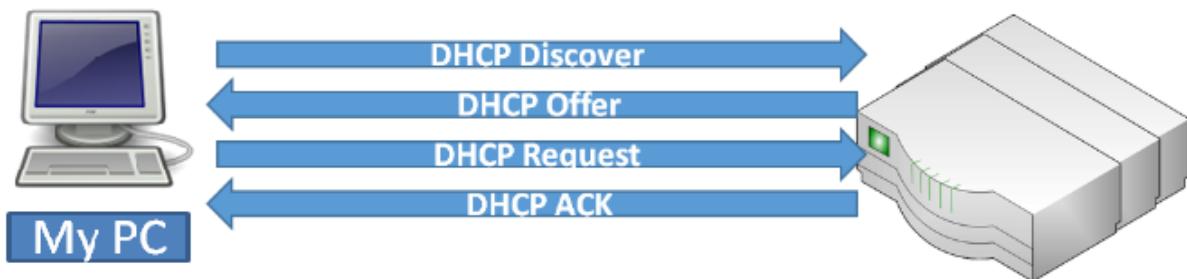


בשלב זהה, המחשב יודע רק את **כתובת ה-MAC** של כרטיס הרשות שלו. הוא יודע את הכתובת, כיון שגם צרובה באופן פיזי על כרטיס הרשות.



## איך המחשב שלנו מושג את כתובת ה-IP ושאר פרטי הרשת שלו?

כפי שמדובר בפרק [שכבה הרשת](#)/[DHCP](#), ישן מספר דרכים לקבל את פרטי הרשת. הדרך הנפוצה כיום היא באמצעות **פרוטוקול DHCP**. באמצעות פרוטוקול זה, כרטיס הרשת שלנו שולח הודעה *discover* DHCP Discover. הודעה נשלחת ב-*Broadcast*, כלומר שכל הישויות ברשת יקבלו אותה. שירות DHCP רואה את הבקשה, ומחזיר *DHCP Offer*, הודעה בה הוא כולל את פרטי הרשת המוצעים לכרטיס הרשת שלנו: כתובת ה-IP שלו, שירות ה-DNS הרלוונטי ועוד. מכיוון שברשת שלנו יש רק שירות DHCP אחד, לא תהיה הצעה נוספת, והמחשב שלנו ישלח הודעה *DHCP Request* לשירות DHCP שהוא מבקש לקבל את ההצעה. לבסוף, השירות DHCP ישלח הודעה ACK, שאחריה יוכל המחשב שלנו להתחילה להשתמש בכתובת ה-IP שנימנתה לו.



**מזל טוב! קיבלנו כתובת IP!**

לצורך הדוגמה, נאמר שקיבliśmy את הכתובת 255.255.0.0<sup>102</sup>, כמשמעות הרשת 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 שלו. נאמר שהנטב קיבל את הכתובת: 1.1.1.1, ומשמעות הרשת היא .255.255.0.0

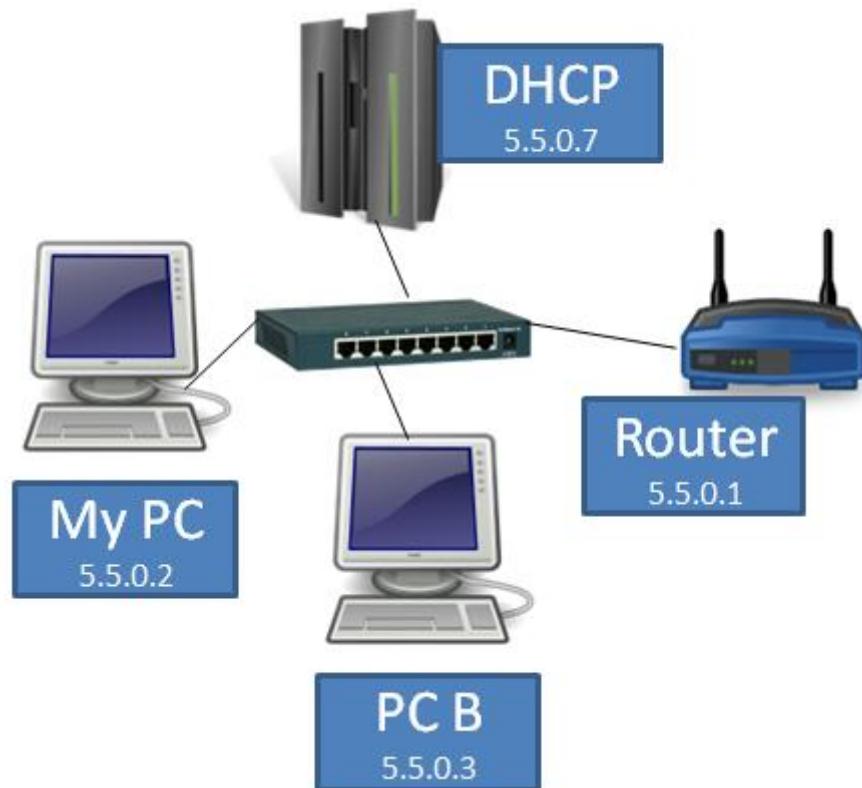


## איך ההודעה הגיע אל השירות DHCP?

על מנת שההודעה תצליח להגיע אל השירות DHCP, על הלקוח להיות איתו באותו ה-*Broadcast Domain*. כלומר, ההודעה צריכה להגיע מבלתי לעבור אף נתב בדרך. זה הזמן להזכיר שבעצם, כפי שמדובר בפרק [שכבה ה-2](#)/[רכיבי רשת שכבה ה-2](#), המחשב שלנו מחובר אל **Switch** (מtag), אליו מחוברים גם מחשבים נוספים ברשת (למשל המחשב בשם "PC B"), אותו השירות DHCP<sup>103</sup> מחובר - **הנטב (Router)** שלו.

<sup>102</sup> כפי שמדובר בפרק [שכבה הרשת](#)/[נספח ב'](#) - [כתובות פרטיות NAT](#), הכתובת תהיה לעיתים תכופות כתובת IP פרטית. על מקרה זה נרchie בנספח א' [של פרק זה - תקשורת מאחור NAT](#).

<sup>103</sup> במקרים מסוימים פרטיטים, הנטב הביתי הוא בדרך כלל גם השירות DHCP של הרשת. לצורך פשוטות הסביר, נניח במקרה זה שימוש השירות DHCP נפרד.



מה השלב הבא?

כעת, המחשב שלנו צריך לגלות מה היא כתובת ה-IP של [www.facebook.com](http://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](http://Facebook).

מהו שרת ה-DNS שלנו? כיצד המחשב יודע זאת?

שרת ה-DNS שלנו הוא שרת ה-DNS של ספקית האינטרנט שלנו.<sup>104</sup> המחשב שלנו יודע זאת כיון 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 שכותובת ה-IP של שרת ה-DNS הינה 2.2.2.2.

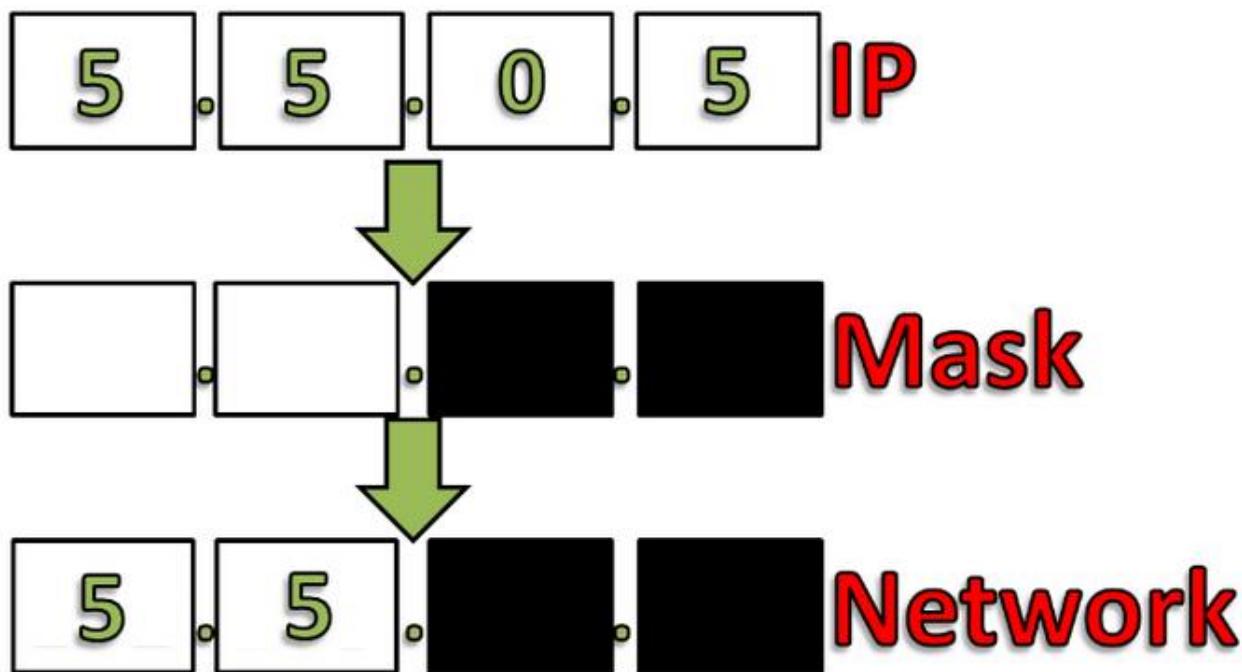
<sup>104</sup> במקרים מסוימים, הנטב ישתמש כשרת ה-DNS. כלומר, המחשב ישלח שאלות DNS אל הנטב, שבתו问他 ישאל את השירות של הספקית.



## כיצד המחשב שלנו יודע לפנות אל שרת ה-DNS?

unless כשהמחשב יודע שעליו לפנות אל שרת ה-DNS ולשלוח אליו שאלת DNS, איך הוא יוכל לעשות זאת? האם הוא יפנה אל שרת ה-DNS *ישירות?* אם לא, אל מי הוא יעביר את החבילה?

בשלב זה, כפי שלמדנו בפרק [שכבת ה-הoco](#) [למי נשלחת שאלת ה-ARP?](#), המחשב בודק האם כתובות ה-IP של שרת ה-DNS נמצאת אליו באותו ה-Subnet. כזכור, כתובת ה-IP של המחשב שלנו הינה: 5.5.0.2, ומסכת הרשת היא: 0.255.255.0.0. כפי שלמדנו בפרק [שכבת הרשת/ מהו מזזה הרשת שלי? מהו מזזה היחסות?](#), משמעותה של מסכת רשת זו היא שני הביטים הראשונים שייכים למזזה הרשת:



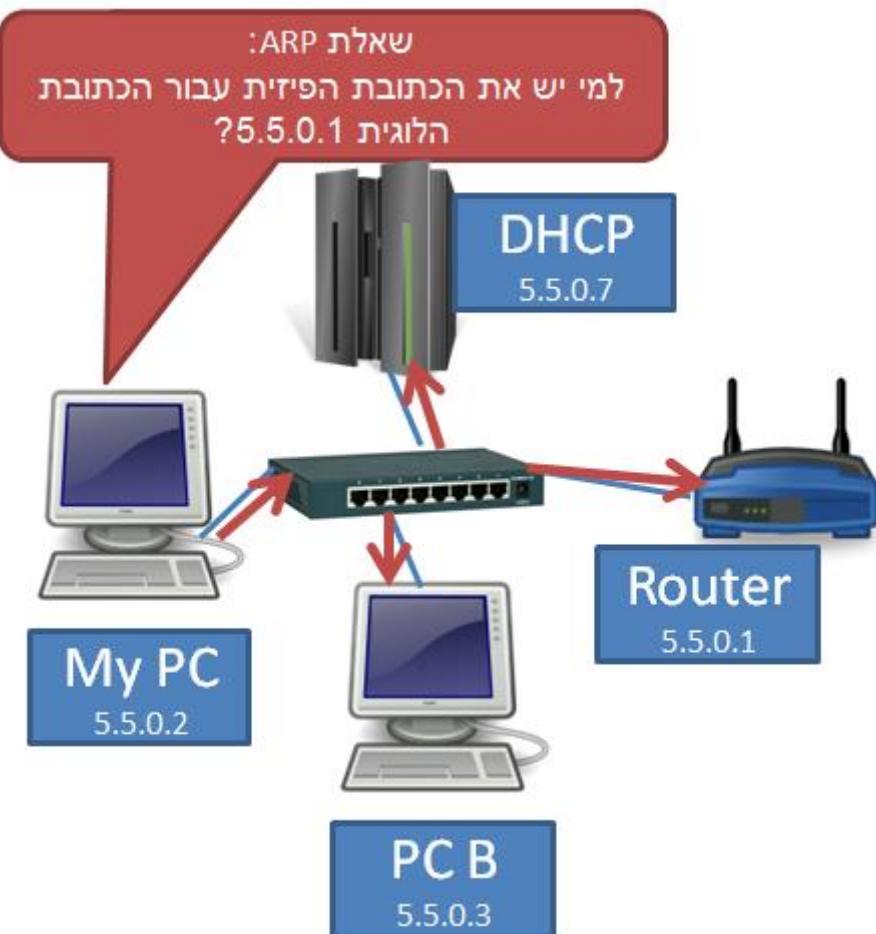
כתובת ה-IP של שרת ה-DNS, אותה מצאנו קודם, היא 2.2.2.2. מכיוון שני הביטים הראשונים של כתובות זו הם 2.2 ולא 5.5, הרי ששרת ה-DNS לא נמצא באותו ה-Subnet של המחשב.

היות שהכתובת 2.2.2.2 לא נמצאת באותו ה-Subnet כמו זו של המחשב, מערכת הפעלה מבינה שעיליה לפנות אל ה-**Default Gateway**, אותו הנטב שיאפשר למידע לצאת מהרשת המקומית אל האינטרנט. גם את הכתובת של נתב זה מצאנו קודם לכן, בתהליך ה-DHCP.

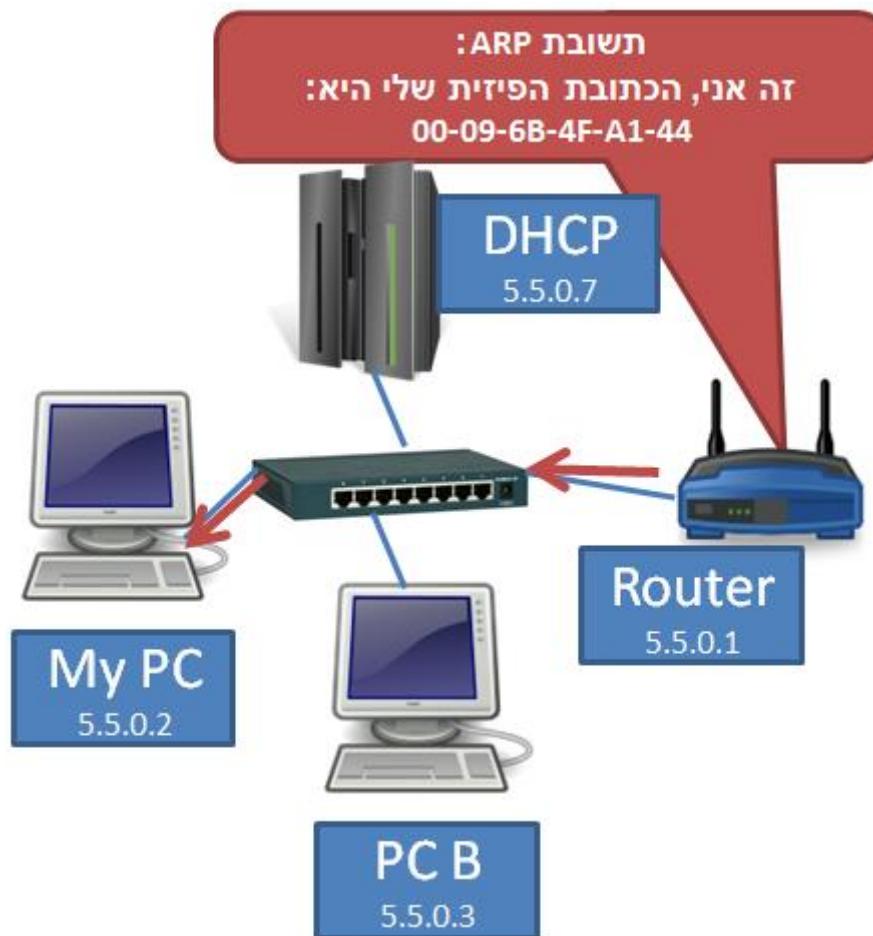
## איך נצלח לתקשר עם הנטב?



הבנו שאנו צריכים לשולח את חבילה DNS אל שרת DNS, 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, וכרטיסים 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 עברו 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 שנראית 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 אר וرك אל המחשב שלנו. כיצד הוא עשה זאת?

כפי שלמדנו בפרק [שכבות הקיו/ כיצד Switch פועל?](#), ל-Switch יש טבלה אוטה 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 שהיא מיועדת לכתובת MAC של המחשב שלנו, ומכאן שהיא מיועדת לפורט הפיזי שלו. כך ידע ה-Switch למתג את ARP Reply אך ורק אל המחשב שלנו.

כעת נוכל להמשיך עם החבילה שברצוננו לשלוח אל שרת DNS.



### מהן 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 הტבלה בעצמכם.

## כתובות MAC

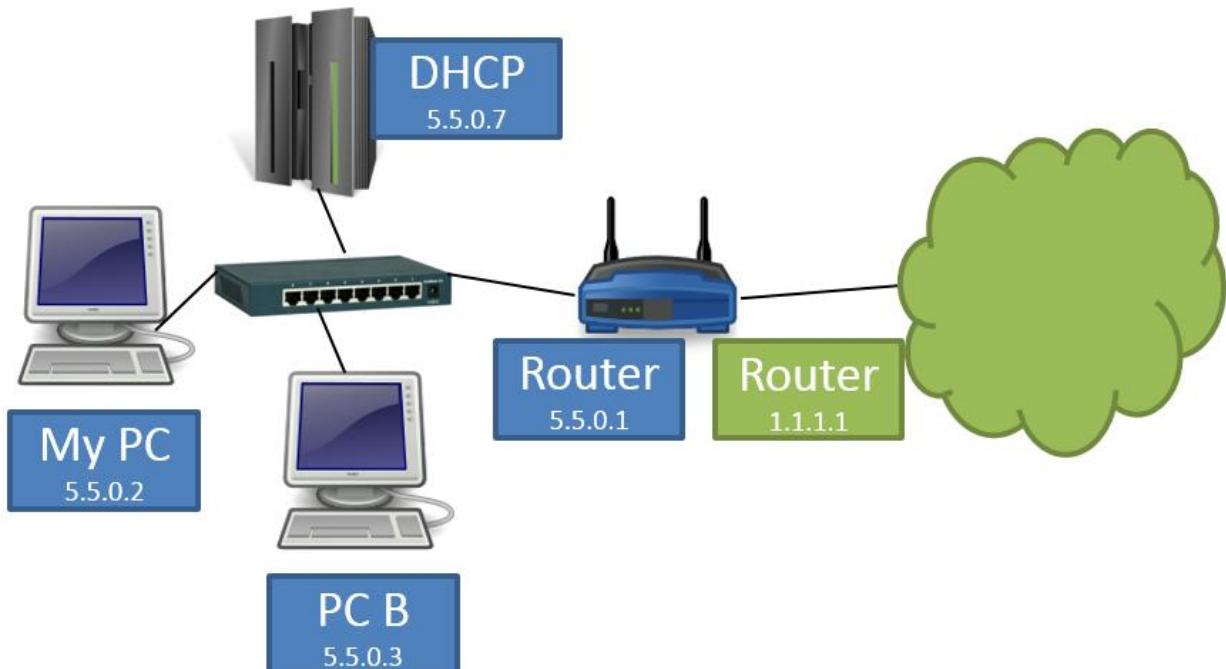
הכתובת הפיזית של המקור שייכת לכרטיס הרשת של המחשב שלנו - הרי הוא זה ששולח את החבילה. המחשב יודע את הכתובת זו שכן היא שייכת לכרטיס הרשת שלו עצמו.

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 כתובות MAC של הנטב משוייך לפורט הפיזי שמחובר אל ה-Switch ברשת שלנו, ולא לפורט פיזי אחר. לכל פорт פיזי של הנטב יש כתובות 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 מסוימת. 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 בשדה 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.

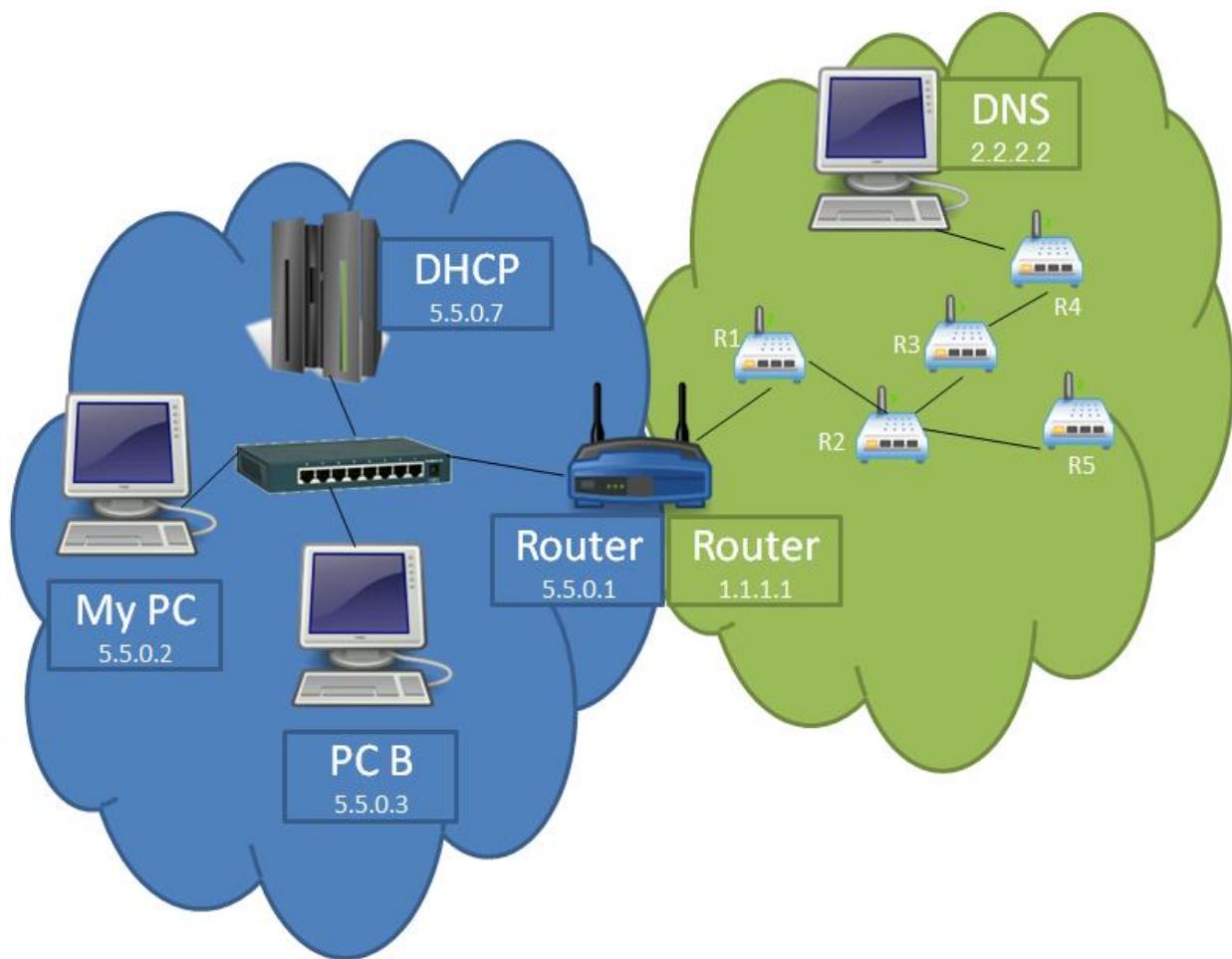
כתובת ה-IP של היעד תהיה הכתובת של שרת DNS. זאת מכיוון שבשכבה הרשת, כתובות היעד מצביעה על היעד הסופי - אל מי החבילה אמורה להגיע בסופו של דבר. את הכתובת של שרת DNS גילינו קודם לכן באמצעות תהליך ה-DHCP.  
הטבלה המלała תיראה כך:

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-IP 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 נטבים נוספים השייכים לספקית האינטרנט:



לצורך הבהרה, הרשת המקומית (LAN) שלנו סומנה בכחול, והרשות של הספקית סומנה בירוק. שימו לב שהנתב שלנו נמצא בשתי הרשומות, 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 MACיעד
	כתובת IP מקור
	כתובת IPיעד

## כתובות ה-MAC

כתובת MAC של המקור תהיה בעצם הכתובת של הנטב שלנו. זאת מכיוון שהוא זה ששולח את החבילות - כולם כרטיס הרשת שלו הוא זה שעביר את החבילות הללו. שימוש לב Ci כתובות MAC שייכת לפורט הפיזי של הנטב שנמצא ברשות של הספקית (הרשות היורקה באיר 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 הרשת הביתית שלנו (ולא הפורט הפיזי שמחובר אל הנטב R2). הכתובת תהיה של הנטב R1 מכיוון שהתחנה הבאה של החבילות היא הנטב R2, וצורך השכבה השנייה אחראית לתקשורת בין תחנות הקשורות זו לזו. נמלה 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 החבילות. 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

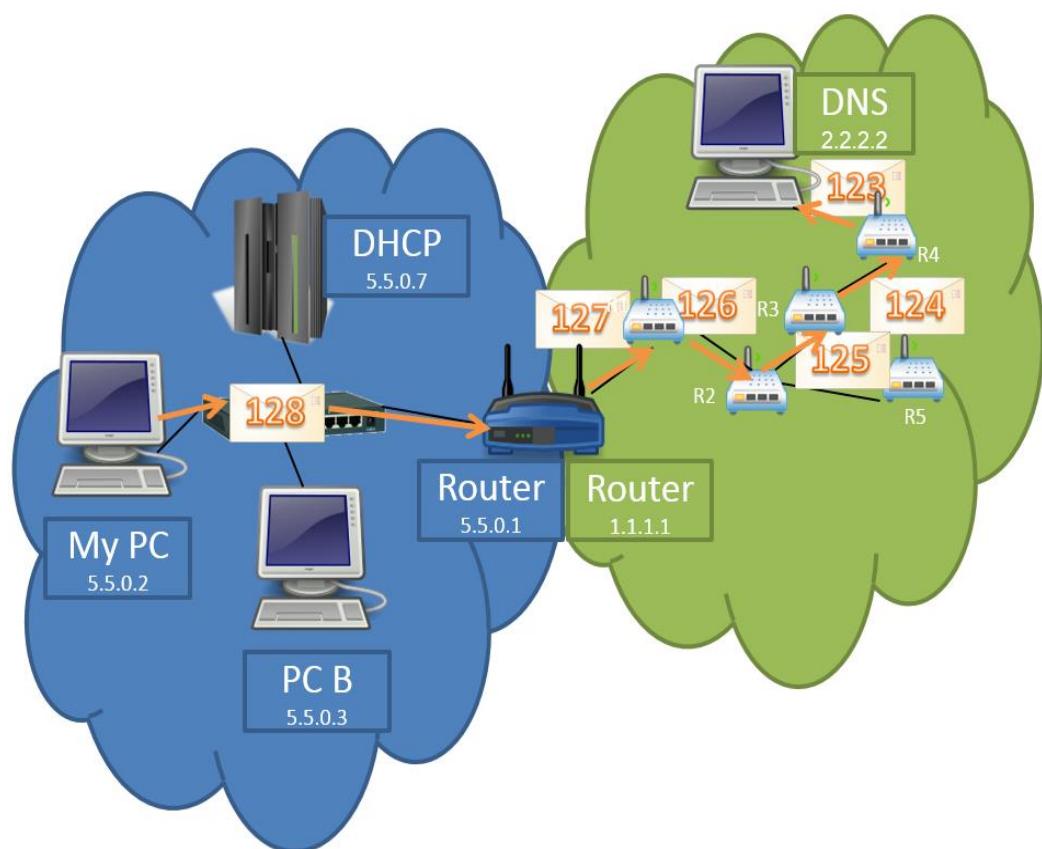
למעשה, כתובות ה-IP נותרו ללא שינוי! כך ניתן לראות שבעוד כתובות ה-MAC משתנות בכל Hop בדרך ומציניות מה השלב הנוכחי שהחbillה עוברת, כתובות ה-IP נשארות קבועות לאורך המשלוח ומציניות ממי החbillה הגיעו במקור ומה יעדה הסופי.

## 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 ללחשב את ה-**Checksum** ולודודא 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 ללחשב 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 (מוצג בתור המספר של החbillה):



כפי שניתן לראות, החבילה מגיעה אל שרת DNS כשער TTL הוא 123, כיוון שהוא חמשה נתבים בדרך. שימושו לב שהחbillה הגיע 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 למיניהם.

כמו כן, הנטב צריך **לנתב את הפקטה**, כלומר להחליט מה המסלול שעלייה לעבור. כפי שלמדנו בפרק [שכבת הרשת/יתוב](#), הנטב מבין لأن עליו להעביר את החbillה באמצעות טבלאות ניתוב שנשמרות 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 שאלתת (Query). כפי שלמדנו, שאלות ותשובות DNS מורכבות מרשות (Resource Records (RR)). השאלתת 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 השאלתת, גם התשובה תכלול את הערך 1337 בשדה ה-ID Transaction. כמו כן, חבילת התשובה תכלול את השאלתת המקורית ששלח המחשב שלנו, והן רשומות תשובה אחת או יותר.



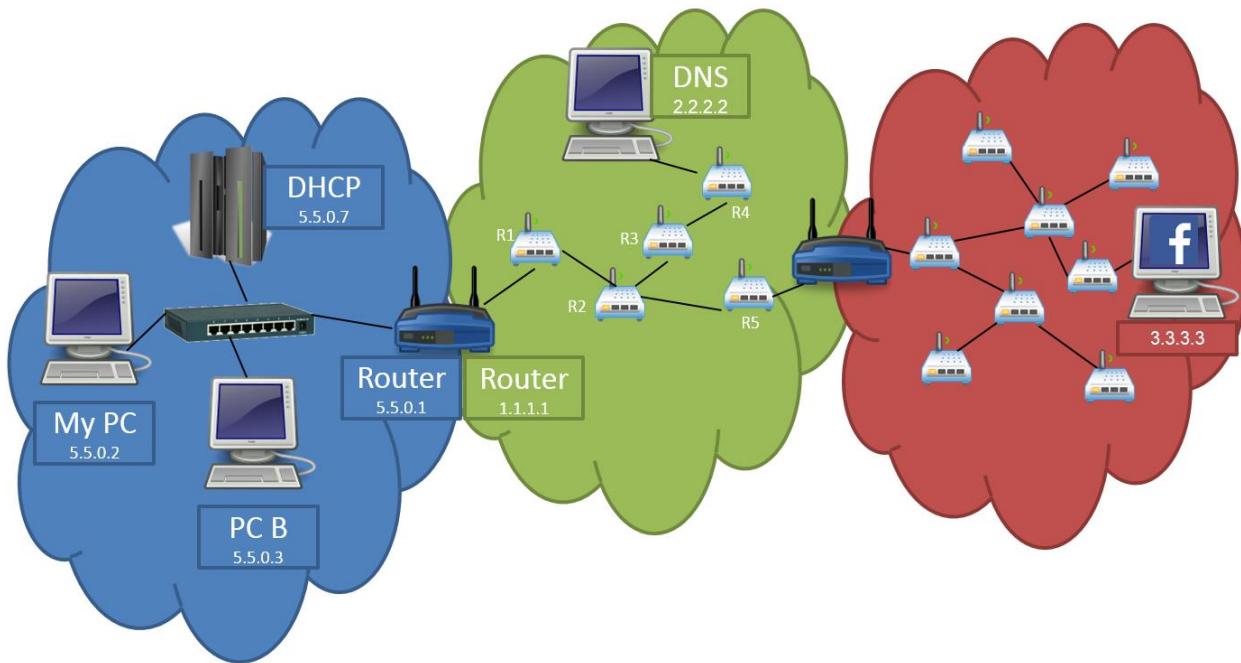
## מה השלב הבא?

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:

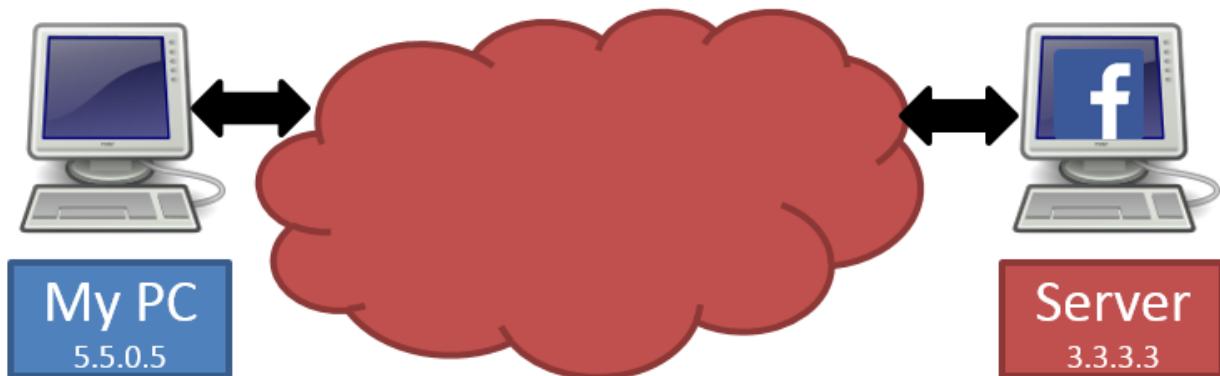


אל שלב בו אנו רואים את הרשת המקומית, הרשת של הספקיות וכן האינטרנט:



שימוש לב Ci באיר האינטראנט אמנים נראה כמו ספקית יחידה, אך הוא מכיל הרבה ספקיות. על מנת לפחות את האיר, שלא כולל בתוכו אזכורים 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 שכבות התעבורה / מיקום שכבות התעבורה במודל השכבות, שכבת הרשת מספקת לשכבות התעבורה מודל של "ענן", ובכך מבטלת את הצורך שלה להכיר את מבנה הרשת. אי לכך, נוכל להציג את התמונה גם בצורה שבה שכבות התעבורה "רואה" אותה, כך:



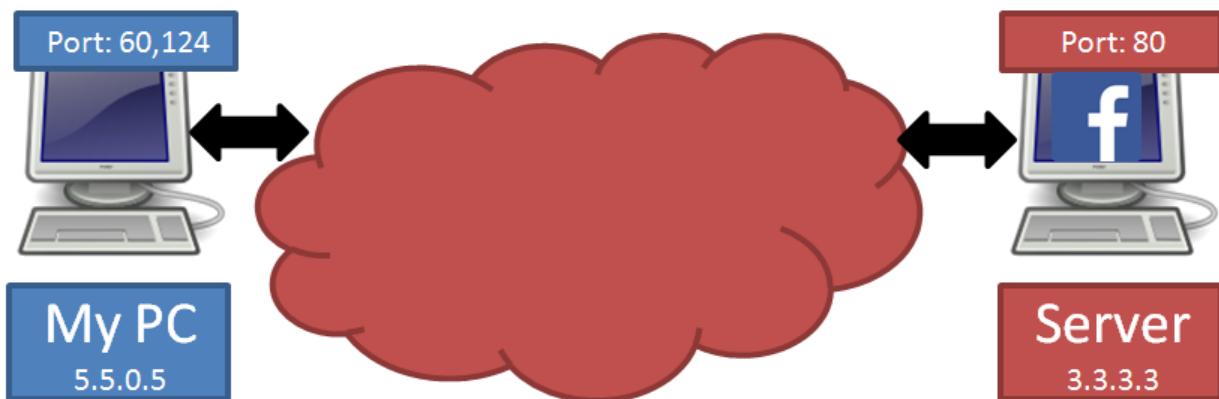
## בailo פורטימ תtabצע התקשורת?



כפי שלמדנו בפרק [שכבת 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 מתבצע הפנימה, ככלומר הפורט המאזין בשרת של Facebook, יהיה כל 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 מתבצע הפנימה, ככלומר הפורט במחשב שלנו, יהיה מספר רנדומלי שתגrial מערכת הפעלה. עם זאת, מספר זה לא יהיה רנדומלי לחוטין, מכיוון שישנם מספרי פורטים השמורים לאפליקציות מסוימות. אי לכך, רוב מערכות הפעלה מגרילות מספר פорт בטווח שבין 49,152 וביין 65,535. נאמר שהפורט שהוגרל הינו 60,124. מכאן שהתקשרות תיראה כך:



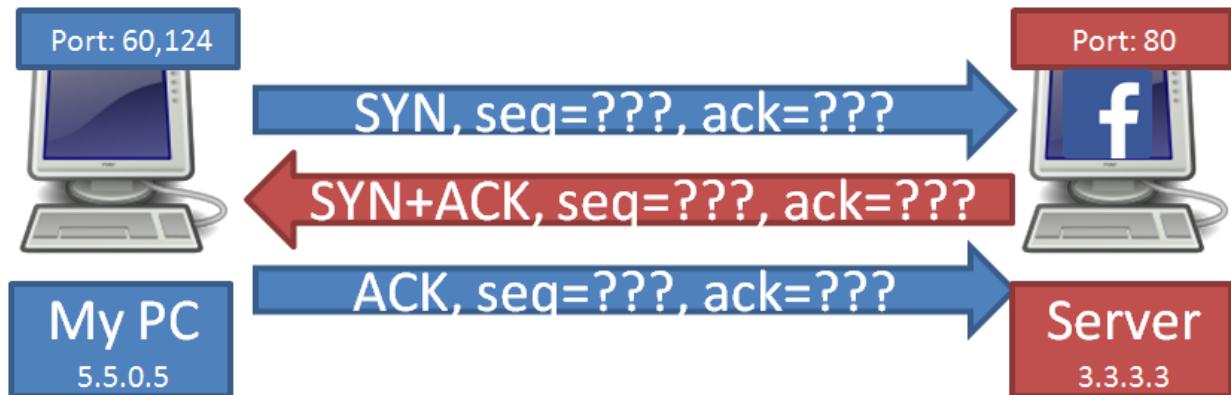
## כיצד נראהית הרמת הקישור?



על מנת להרים קישור TCP בין המחשב שלנו לבין השרת המרוחק, נשתמש ב-**Three Way Handshake**, כפי שלמדנו בפרק [שכבת התעבורה / הרמת קישור ב-TCP](#). היות ששכבת התעבורה אינה מודעת למבנה הרשת, נסתמך על מודל ה"ען" שמספקת שכבת הרשות ונתיחה לתקשורת城际. היא מתבצעת באופן ישיר בין המחשב שלנו לבין השרת של Facebook.



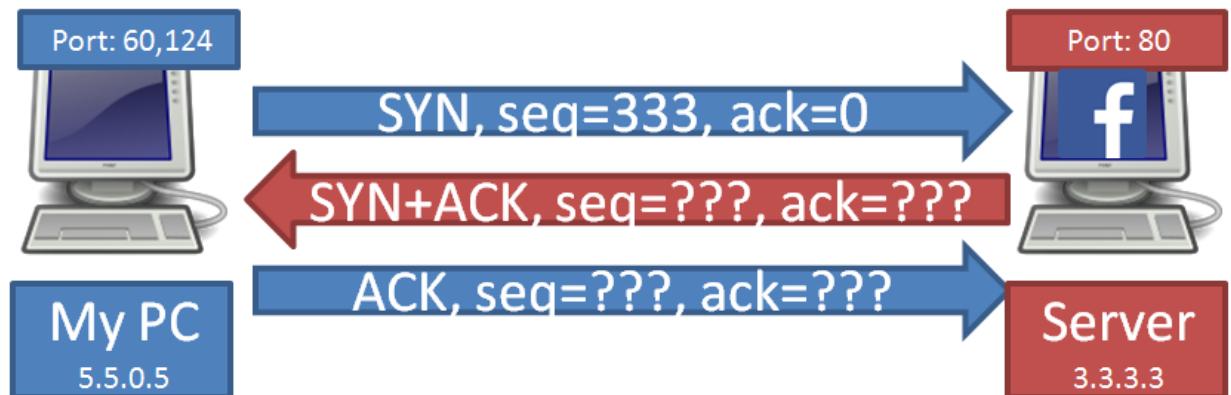
נסו להשלים בעצמכם את הערךם שבתרשים הבא, במקום סימני השאלה:



נתחיל מהחיבור הראשונה, היא חיבור SYN. זו החיבור המציין את תחילת הקישור, ולכן הדגל SYN בה דלוק. הדגל ACK כבוי, מכיוון שגם החיבור הראשונה בקישור, ולא מתבצע אישור על קבלה של מידע קודם.

ערך ה-Sequence Number של חיבור זו הינו ערך ה-Sequence Number ההתחלתי של הקישור, ועל כן הוא יהיה רנדומלי, כפי שמדובר בפרק שכבת התעבורה/[חיבור ראשונה - SYN](#). לצורך הדוגמה, נאמר שהערך שהוגרל הוא 333.

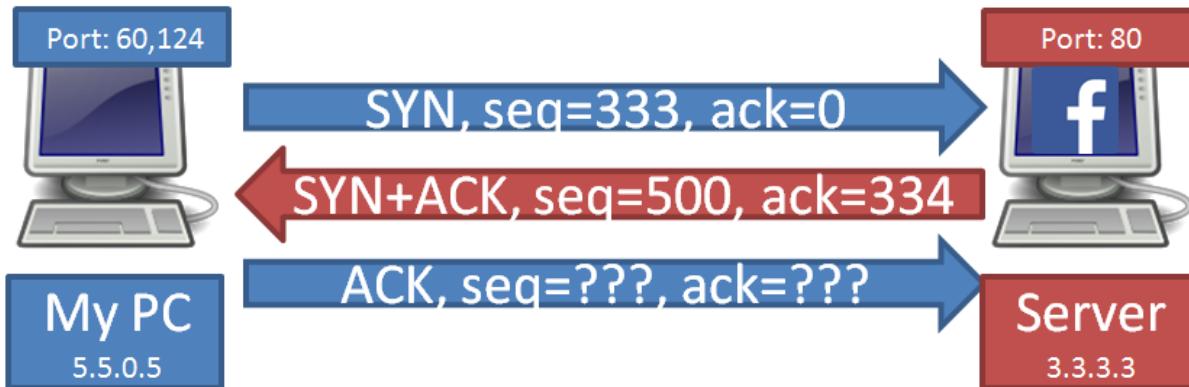
ערך ה-Acknowledgement Number של חיבור זו יהיה 0, זאת מכיוון שדגל ACK כבוי. נוכל למלא את הערךים הבאים בשרטוט:



cut עבור לחיבור השנייה. בחיבור זו, דגל SYN דלוק היה שמדובר בחיבור הראשונה מצד של הקישור שבין השרת למחשב שלנו. דגל ACK דלוק שכן יש לתת אישור על הגעת חיבור SYN מהלkers.

שדה ה-Sequence Number של חיבור 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 SYN התקבלה. כפי שלמדנו בפרק [שכבות התעבורה/AIC TCP משתמש בס-Numbers Acknowledgement](#), ערך שדה זה מחושב באמצעות ערך Sequence Number של החbillת שהתקבלה (333), בנוסף לאורך המידע המועבר בה. היות שהחbillת SYN, גודל המידע שמחושב עבורה הוא גודל של בית (byte) אחד. אי לכר, הערך יהיה .334, כולם 333+1.



נותרנו עם החbillת השלישית והאחרונה לתהילך הרמת הקישור. בחbillת זו, דגל SYN כבוי, מכיוון שהוא המעיד על יצירת הקישור. דגל ACK דלוק, שכן יש לאשר את קבלת החbillת הקודמת שנשלחה מהשרת.

מה יהיה ערך ה-Acknowledgement 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, כולם 501.



## איך נראה בקשה 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 בקשה GET. מכיוון שהפניה מתבצעת אל העמוד הראשי של Facebook, מוביל לבקשת אפוא שMASER שמייצג את המחשב שנמצא בכתובת "/". [זכור מפרק שכבות האפליקציה / מבנה פרטלי של בקשה HTTP](#), אחרי המילה GET תופיע המחרוזת "HTTP" והגירסה של הפרוטוקול, למשל: 1.0. לאחר מכן יופיעו ה-HTTP Headers, עליהם לא נעמיק בפרק זה.



## איך נראה תשובה HTTP?

בהנחה ו-Facebook מוכן להחזיר את העמוד הראשי שלו ללא עיכובים נוספים, הוא יענה בתשובה HTTP מסווג (OK). כפי שראינו בפרק [שכבות האפליקציה / מבנה פרטלי של תשובה HTTP](#), התשובה מתחילה במחרוזת "HTTP" והגירסה של הפרוטוקול. מיד אחר כך, יופיע הקוד של התגובה (200) ואז הפירוש הטקסטואלי של הקוד (OK). לאחר מכן, יופיעו כל ה-Headers הרלוונטיים, ולבסוף - המידע עצמו.



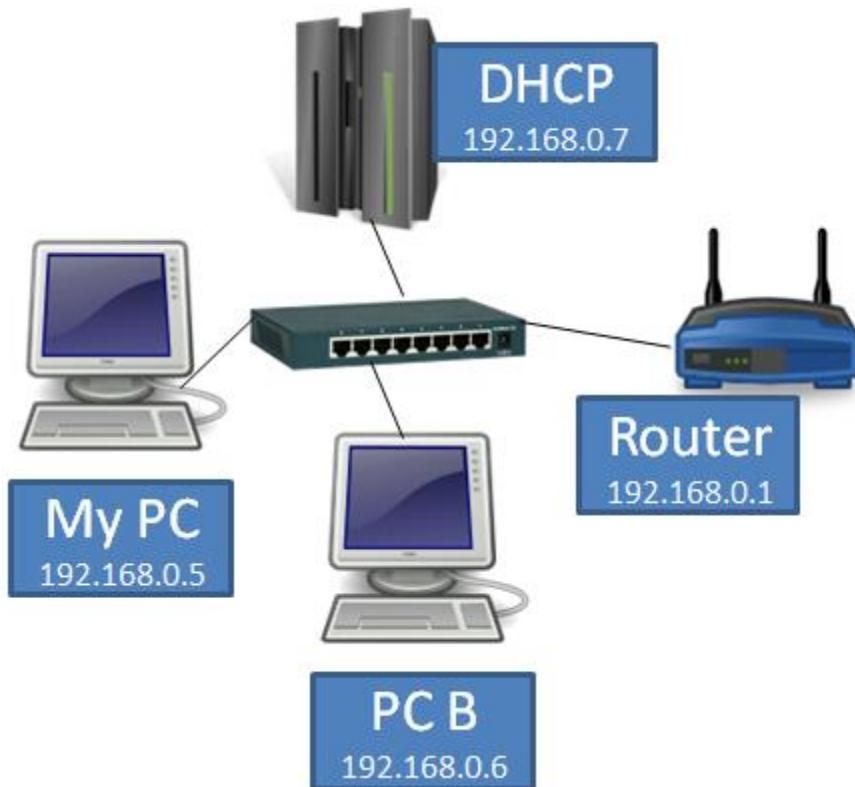
בתשובה זו הגיענו לסוף התהילה, והדפדן שלנו יוכל סוף סוף לראות את העמוד הראשי של Facebook.



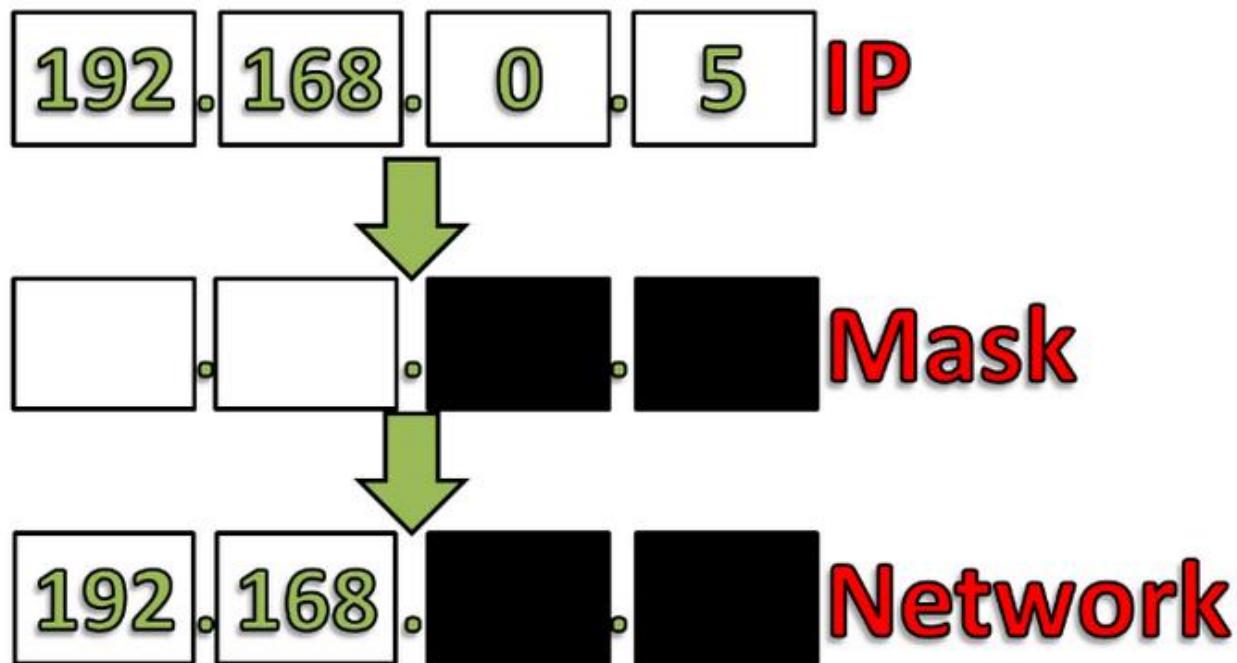
## מה קורה כאשר המחשב שלנו נמצא מאחורי NAT?

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 [שכבת 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 [שכבת הרשת](#), הכתובת שהמחשב שלנו מקבל משרת DHCP שלו צפופה להיות **כתובת IP פרטית**, וזאת בכך שהיא לחסוך בכתובות IP בעולם. במקרה שלנו, נאמר שקיבלנו את הכתובת: 192.168.0.5. מס' כתוב הרשת היא: 255.255.0.0. בנוסף לכך, ניתן לקבל **כתובת IP חיונית**, הניתב שלנו עבר תהיליך דומה. כאשר הנטב מתקשך עם שרת ה-DHCP של ספקית האינטרנט, הוא מקבל ממנו כתובת IP שאינה פרטית. נאמר שהנתב קיבל את הכתובת: 1.1.1.1, ומס' כתוב הרשת היא: 255.255.0.0. תמונה הรสת עשויה להיראות כך:

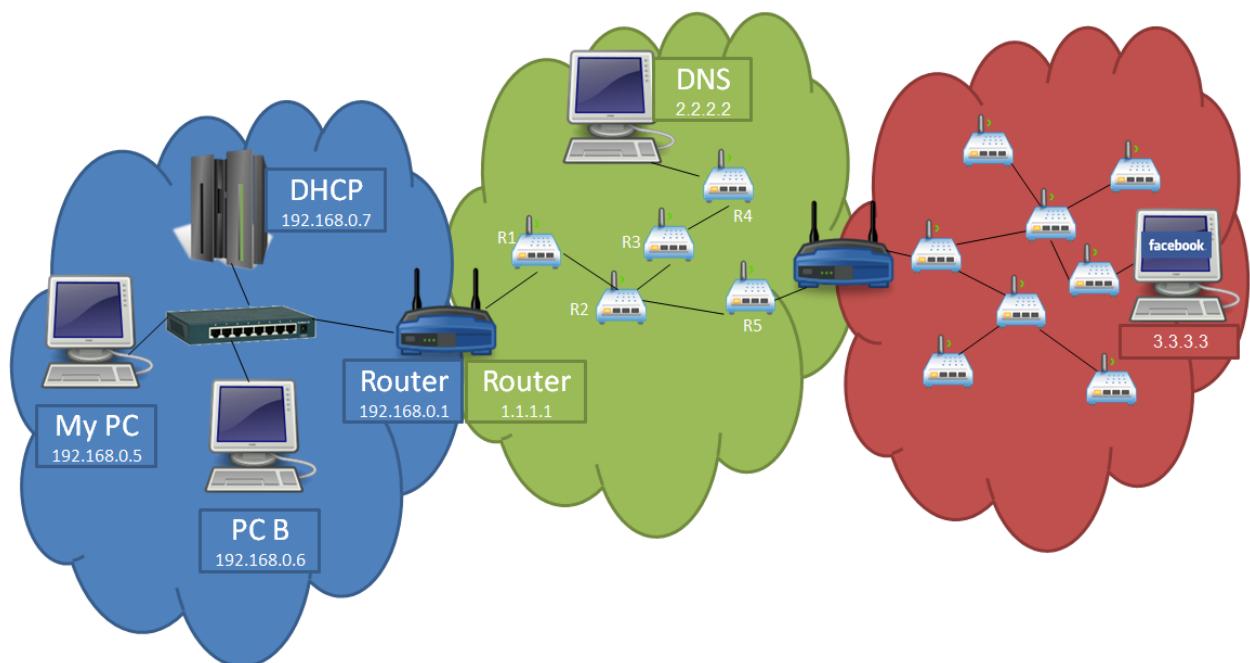


גם במקרה זה, כאשר המחשב ירצה לגשת אל שירות DNS, עליו לבדוק האם השרת נמצא באותה הרשת כשלו. לשם כך, המחשב יבצע בדיקה על מסכת הרשת שלו:



כתובת-ה-IP של שירות DNS, אותה מצאנו קודם, היא 2.2.2.2. מכיוון שני הביטים הראשונים של כתובות זו הם 2.2 ולא 192.168, הרי השירות DNS לא נמצא באותו ה-Subnet של המחשב.

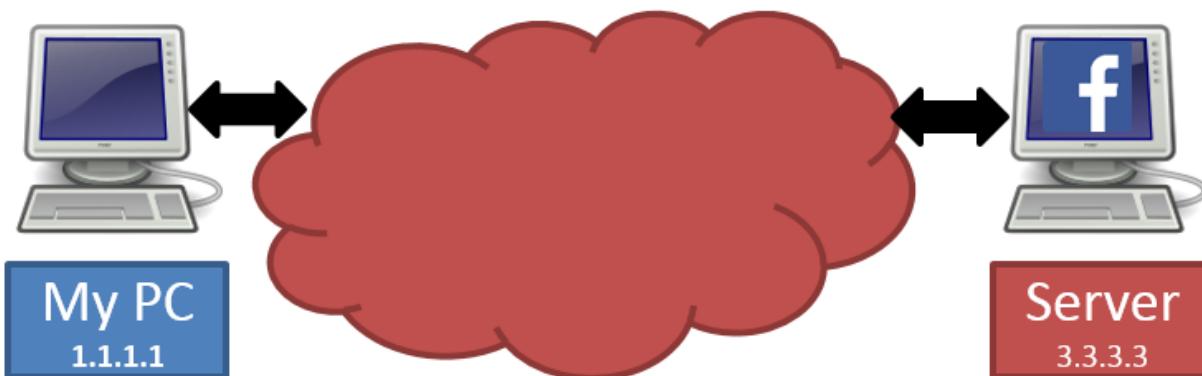
המשך התהילה דומה מאוד לתהילה ללא שימוש ב-NAT, ולכן לא נורחיב עליו כאן. עם זאת, עליים להבין את ההבדל הנובע מהשימוש בכתובות פרטיות. תמונה הרשות שלנו נראה כך:



שיםו לב שלנתב שלנו יש שתי כתובות IP - אחת "פנימית", שהיא הכתובת הפרטית 192.168.0.1 המשמשת אותו ברשת הביתית שלנו, והשנייה "חיצונית", שהיא הכתובת 1.1.1.1 המשמשת אותו אל מול הספקית בפרט והאינטרנט בכלל.

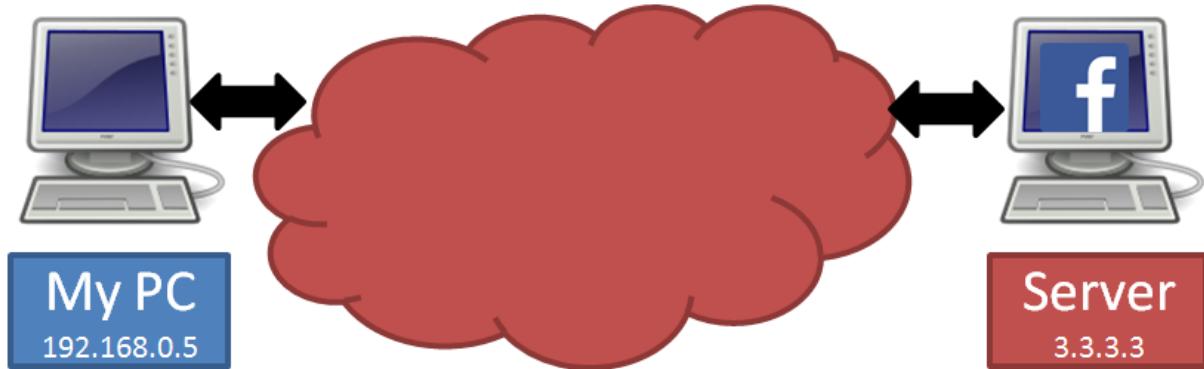
כאשר המחשב שלנו רוצה לתקשר עם Facebook, הוא לא יכול לעשות זאת ישירות. תקשורת שכזו אינה אפשרית, מכיוון שאם Facebook ינסה לענות אל המחשב שלנו, הוא לא יוכל לשולח אליו חבילת - הרי שהכתובת 192.168.0.5 הינה כתובת פרטית, והנתבים שבדרך לא יכולים לנתר אליה חבילות.

כאן נכנס לפעולה ה-NAT. כאשר המחשב שלנו ישלח את החבילת, הוא ישלח אותה אל השרת של Facebook באופן רגיל. בעת, כאשר הנתר יקבל את החבילת, הוא יחליף את כתובת המקור של החבילת לכתובת שלו, כאשר החבילת מגיעה אל Facebook, היא תיראה כאילו היא הגיעה מהכתובת 1.1.1.1. נראה זאת בשרטוט הבא (שモצג מנקודת המבט של השרת של Facebook):



הxBBבולה מגיעה בשלב זה אל השרת של Facebook, שלא מודע כלל לכך שהיא נשלחה במקור מהכתובת 192.168.0.5. מבחינתו, החබילה הגעה 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 החබילה, הוא מחליף את כתובת היעד של החබילה, מהכתובת 1.1.1.1 אל הכתובת 192.168.0.5. לאחר החלפה זו, הוא מעביר אותה אל המחשב שלנו.

מכיוון שהנתב מחליף את הכתובות בטרם הוא מעביר את החබילה, המחשב שלנו כלל לא צריך להיות מודע לתהליך NAT, והוא "שׁקוף" עבורי. למעשה, עברו המחשב שלנו, התהילך נראה כאילו לא היה NAT בכלל. נראה זאת בשרטוט הבא (ש모צג מנקודת המבט של המחשב שלנו):



בדרכו זו, ה-NAT מאפשר חסcoon בכתובות IP מוביל לגורם לשינוי הצד של לקוח הקצה - המחשב שלנו במקרה זהה.

איןנו מתעכבים בספר זה על דרכי שונות למשתמש NAT. עם זאת, אתם מוזמנים להרחב את הידע שלכם בנושא בעמוד: [http://en.wikipedia.org/wiki/Network\\_address\\_translation](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 מאשרינו בפרק הראשון.

התחלנו מהמחשב הבודד שלנו, והצלחנו 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 לעשות זאת, הסתכלנו על **מסכת הרשות (Subnet Mask)** שלנו והבנו שהשרת ה-DNS לא נמצא איתנו באותה הרשת. על כן, פנינו אל **טבלת הניתוב** והבנו שעליינו להעביר את החבילה אל ה-**Default Gateway** שלנו.

על מנת לפנות אל האינטרנט, היינו צריכים לגלוות את הכתובת הפיזית שלו, ולשם כך השתמשנו בפרוטוקול **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 שבו היו בחבילה שנשלחת אל שרת ה-DNS, וראינו איך הן משתנות לאורך המסלול. כמו כן, הזכרנו כיצד **נתב מטפל** בחבילה שגיעה אליו.

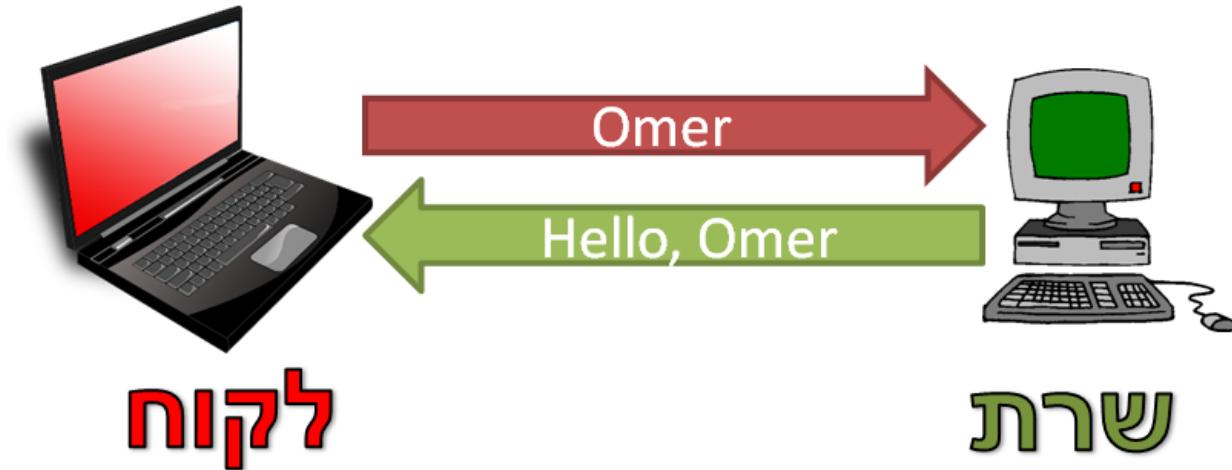
כשהבנו את הדרך שעשו החבילה באינטרנט, חזרנו על דרך הפעולה של פרוטוקול **DNS**, באמצעותו מצאנו את כתובת ה-IP של Facebook. בשלב זה, היינו צריכים להקים  **קישור TCP** עם Facebook, ולכן הזכרנו כיצד נבחרים מספרי הפורטטים בהם נעשה שימוש. הקמנו את הקישור באמצעות **Three Way Handshake**, וחזרנו על השדות **Acknowledgement Number**- **Sequence Number** של Facebook מעל  **קישור TCP** שהקמנו באמצעות פרוטוקול **HTTP**, שלחנו בקשה אל השרת וקיבלו את התשובה.

באופן זה נגענו בנושאים רבים במהלך הפרק, ועבדנו 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 מעמיקה בקונספטים רבים עליהם הרחכנו במהלך הספר. כמעט ולא נכנסנו לשדות ספציפיים של פרוטוקולים, והחסכנו התייחסות לנושאים חשובים בכל שכבה ולמבנה החבילות. אם תרצו, תוכלו לחזור אל הפרקים הרלבנטיים ולרענן את זכרונכם.

דריכנו עדין לא הסתימה. 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)

בפרק [תכונות ב-Sockets](#) למדנו מהו Socket, וכייזד נכתב ללקוח ושרת. כתבנו מספר לקוחות ומספר שרתים, וכך שרת שמקבל מהלקוח את שמו ומחזיר לו תשובה בהתאם ([פרק תכונות ב-Sockets/תרגיל 2.3 מודרך - השרת הראשון שלו](#)):



זכור, על מנת לטפל בבקשת לקוח נוסף, נאלצנו לסגור את החיבור עם הלקוח הראשון. כלומר, במקרה של לקוח נוסף לשרת, השרת יהיה צריך לענות לבקשת הלקוח הראשון, ולאחר מכן לסיים את החיבור אליו. פועל זה אמנם הגיוני עבור שירות שמציע שירות כה פשוט כמו תשובה בהתאם לשם הלקוח, אך הוא לא הגיוני עבור שירות שנועד לספק שירותים למספר לקוחות במקביל. חישבו למשל, עד כמה קשה היה להשתמש בפייסבוק במקרה שהשרת היה מסוגל לתת שירות רק ללקוח אחד בכל רגע נתון.



מדוע זו בעיה?

ובכן, מדוע אנו משקיעים פרק שלם כדי לדון בסוגיה זו? האם הפתרון לא כולל רק להוסיף לוולה ללקוח?

הסוגיה הבסיסית הראשונה נובעת מה הצורך ליצור חיבור מול לקוח חדש. להזכירם, כאשר רצינו לקבל חיבור מלקוח חדש בעת שמימשנו שירות שמתפל בבקשת אחת בכל פעם, השתמשנו במתודה `accept` בצורה הבאה:

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

כפי שציינו בפרק תכונות ב-Sockets, המתודה **accept** הינה blocking - כלומר, הקוד "יקפא" ולא ימשיך לרצוץ עד אשר יתקבל בשרת חיבור חדש. מכאן שלאחר שקיבלנו חיבור מלוקה ראשון, علينا להחליט בין שתי אפשרויות:

- לטפל בבקשתו שmagiuot מהלוקה הראשון.
- לאפשר לлокוח חדש להתחבר.

הסיבה לכך נועצה בעובדה, שעל מנת לאפשר לлокוח חדש להתחבר, علينا לקרוא שוב למетодה **accept**, אשר עוצרת את ריצת התוכנית עד אשר יתחבר לлокוח חדש.

סוגיה בעייתייה נוספת קשורה לקריאת מידע מלוקות קיימים. כאשר רצינו לקרוא מידע מלוקוח בעת שמיימשו שירות 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 הלוקוח לא ישלח אלינו דבר?שוב, התוכנית תיתקע ולא יוכל לטפל בלוקות נוספים. אי לכך, כאשר נקרא ל-**recv**, לא יוכל לאפשר לлокוח חדש להתחבר לשרת, או לחילופין - לקבל מידע מלוקוח אחר המעניין לכתוב אליו.

## הפתרון - **select**

אחד הדרכים<sup>105</sup> 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 כרגע מרשיימה זו.

כל אובייקט **socket** יכול להיכנס לתוך אחת או יותר מהרשימות הללו.

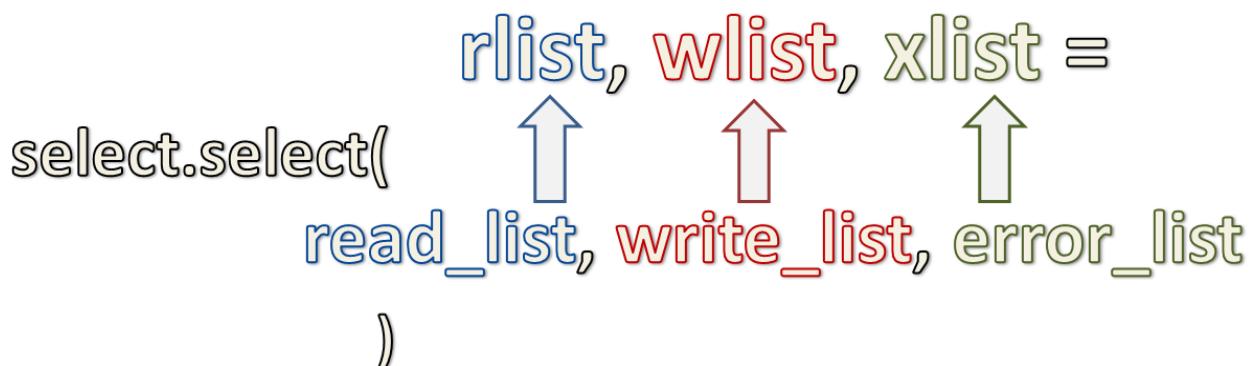
לאחר ש-**select** מסיים את הריצה שלה, היא מחזירה שלוש רשימות:

- רשימת Sockets מהם ניתן כרגע לקרוא (באמצעות **recv**).
- רשימת Sockets אליהם ניתן כרגע לשלוח (באמצעות **send**).
- רשימת Sockets שהזדקנו שגיאה כלשהי.

---

<sup>105</sup> ישנן דרכים נוספות, כגון שימוש ב-**Threads**, אוטן לא נסקור בספר זה.

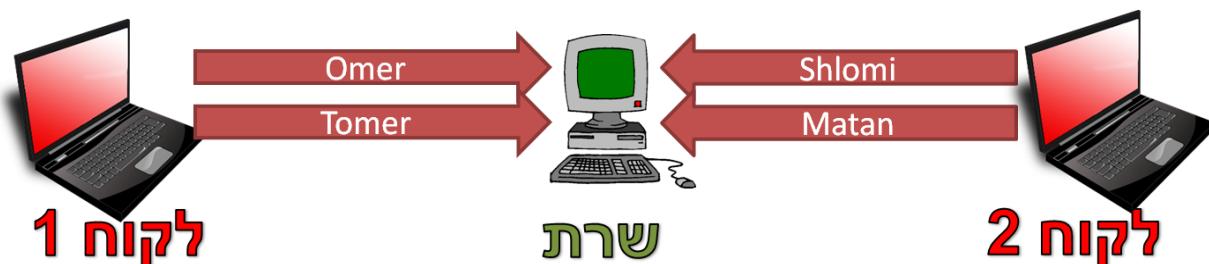
כל אחת מהרשימות הלו צוללת Sockets מתוך הרשימה התואמת שהעבירהנו:



כך למשל, באם העבכנו ברשימת-h Sockets מהם אولي נרצה לקרוא (בشرطוט לעיל: `read_list`) את האובייקטים `client_2_socket` ו-`client_1_socket`, יתכן שהfonקציה `select` תחזיר ברשימת-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 - השרת מרובה המשתתפים הראשון שלו

בתרגיל זה נמשח שרת שמקבל חיבור מלוקו, קורא שם כלשהו מהלוקו, ומדפיס שם זה למסך. עם זאת, בנגדו לשרת שמיימנו בפרק תכנות-h Sockets, השרת יוכל לטפל במספר לקוחות במקביל. מעבר לכך, כל לקוח יוכל לשלוח בכל פעם שם אחר. השרת ידפיס למסך כל שם שהוא מקבל:



שימוש לב שהטיפול במקרה זה צריך להיות מקבילי - כמובן, השרת יוכל להשאיר את החיבור עם הלוקו הראשון פתוח בעודו מספק שירות לקוחות השני.

על מנת לעשות זאת, נתחילה בדרך דומה לזה שעשינו עד כה - ניצור אובייקט מסוג **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:blkooch אחד בכל פעם. כעת נתחל בשלבים שיהיו 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))

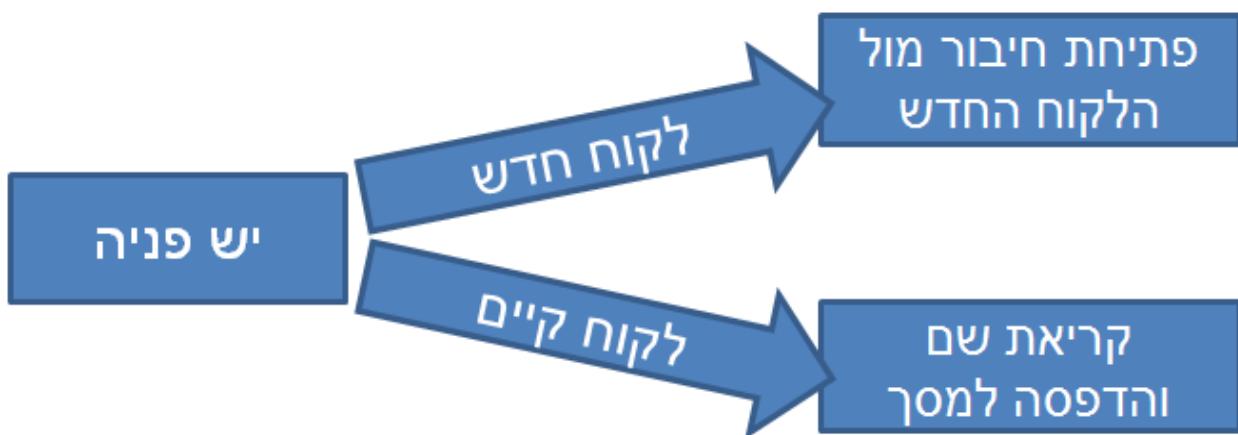
server_socket.listen(5)

open_client_sockets = []
```

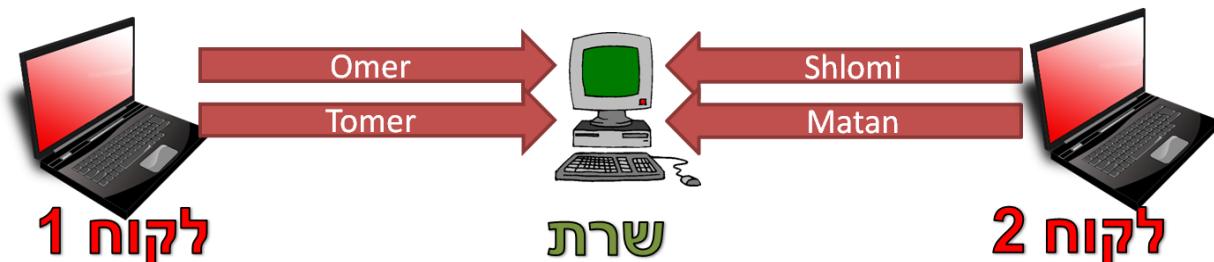
בשלב זה, علينا לטפל בפניות המגיעות אל השרת. לשם כך, ניצור לוולה שככל פעם:

- 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 לוולה, על הסקריפט לבצע את הלוגיקה הבאה עבור כל פניה:



זכור, כל לקוח יכול לכתוב שם שונה בכל פעם, מבל' לסגור את החיבור:



הלוואה המתוארת לעיל תרוץ באופן תמידי, ולכן נכתבו:

`while True:`

בתוך לולאת `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 = []
```

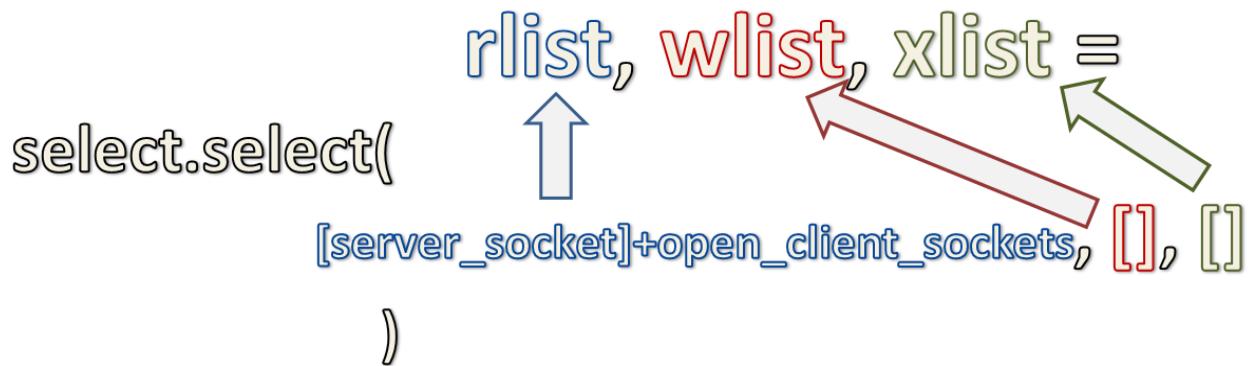
`while True:`

```
rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, [], [] )
```

זהו למעשה שורת המפתח, ולכן מתעכבר בכך להבין אותה. כאמור, הפונקציה `select` מחזירה שלוש רשימות:

- רשימת Sockets מהם ניתן כרגע לקרוא - תשמר למשתנה `rlist`.
- רשימת Sockets אליהם ניתן כרגע 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 כרגע לשולח, מתוך רשימת Sockets שניתנו לפונקציה `select` בתור הfrmtr השני. להיות שהfrmtr השני והשלישי הם רשיימה ריקה ([]), הרי ש-`wlist` ו-`xlist` תהינהו לעולם ריקות.

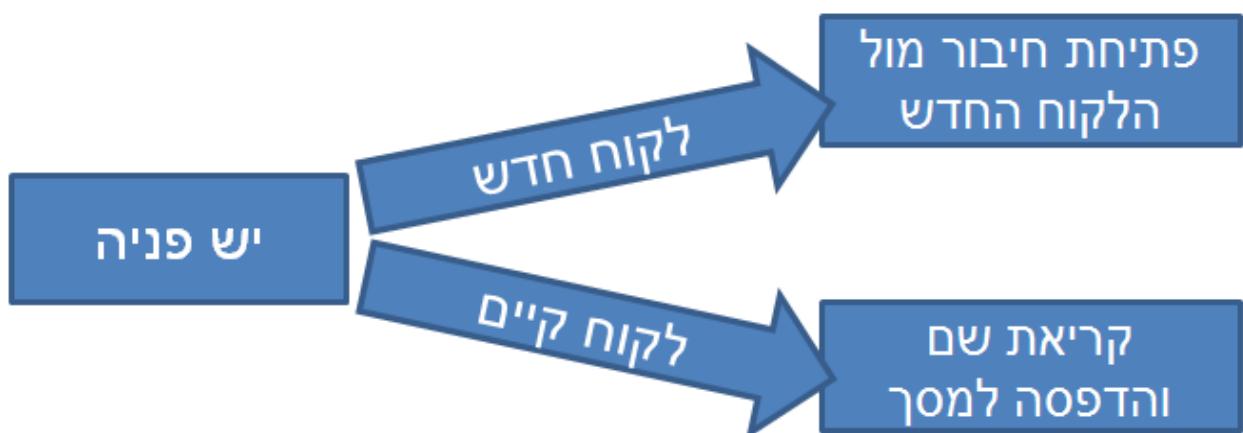


cutet נתמך במשתנה `rlist`, אליו כאמור תשמיר רשימת ה-Sockets מהם ניתן כרגע לקרוא. אלו שולחים אל הפונקציה `select` את ה-Sockets הבאים:

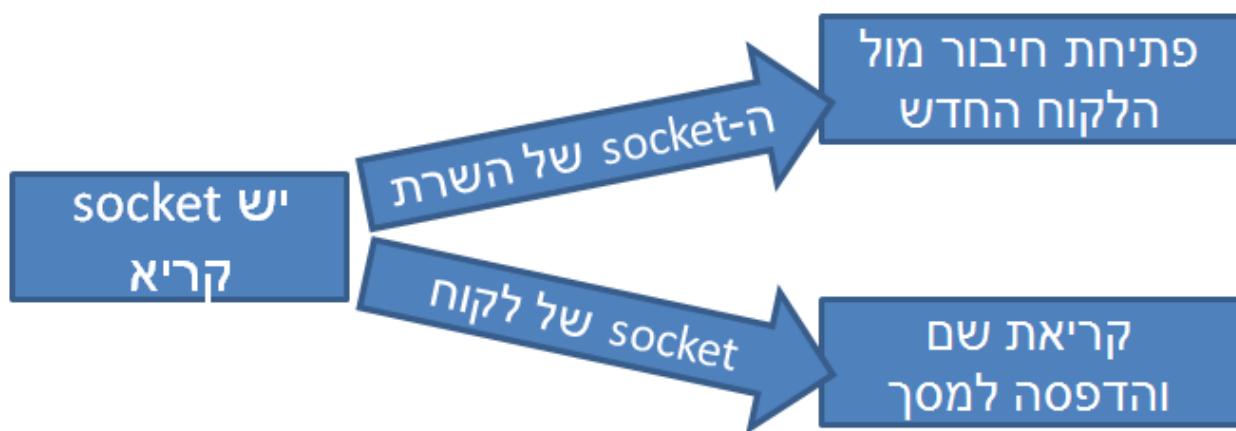
- Socket של השירות המאזין. במידה שנייתן לקרוא מ-Socket זה, המשמעות היא שיש פניה של ליקוח חדש. לעומת זאת, ניתן לקרוא ל-`accept` ולהקימם חיבור עם הליקוח החדש.
- כל ה-Sockets של הליקוחות. במידה שנייתן לקרוא מ-Socket של ליקוח, הרי שהואשלח מידע. לעומת זאת, ניתן לקרוא ל-`recv` ולקבל מידע מלוקוח קיימ.

שימוש לב Ci על מנת לעשות זאת, השתמשנו בשרשור רשימות של פיטונות:  
`[server_socket] + open_client_sockets`

את הלוגיקה שצירפנו קודם:



ניתן למעשה לרשום גם בצורה הבאה:



כלומר, 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:

נבדוק אם ה-socket הנוכחי הוא של השרת:

if current\_socket is server\_socket:

אם כן, הרי 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 נדפס שקיבלנו מידע מלקוח קיימ:

else:

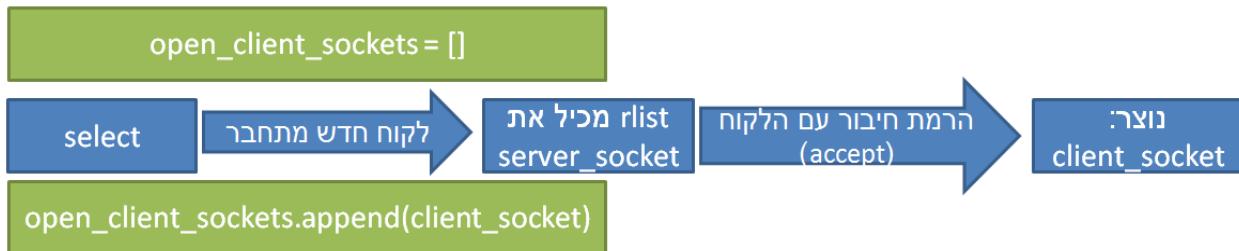
print 'New data from client!'

לולאת ה-while המלאה שלנו נראה这般階層如下：

```

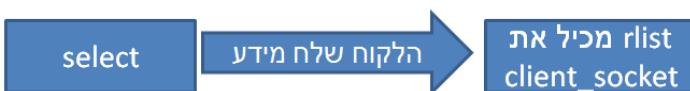
while True:
    rlist, wlist, xlist = select.select( [server_socket] + open_client_sockets, [], [] )
    for current_socket in rlist:
        if current_socket is server_socket:
            (new_socket, address) = server_socket.accept()
            open_client_sockets.append(new_socket)
        else:
            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 יתחבר 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 המתודה **accept** ולאחר מכן יוסיף אותו לרשימה **open\_client\_sockets**.



בפעם הבאה שתoruן לולאת ה-`while`, הפונקציה `select` תחזיר כאשר הלוקו שלח מידע לשרת (בהנחה שלא התחבר ביןתיים לוקו אחר). הפעם, היא תחזיר אל המשתנה `rlist` רשימה שמכילה את ה-Socket בין הלוקו לשרת, אותו Socket שהתווסף קודם לכן לרשימה `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 מהлокו. בטור התחלה, עלינו לקרוא את המידע:

```

data = current_socket.recv(1024)
  
```

כפי שלמדנו בפרק [תכנות ב-Sockets](#), יש להבין אם 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 בלאה אינסופית:
  - מקבל שם מהמשתמש (באמצעות `raw_input`).
  - שלוח את השם אל השרת.

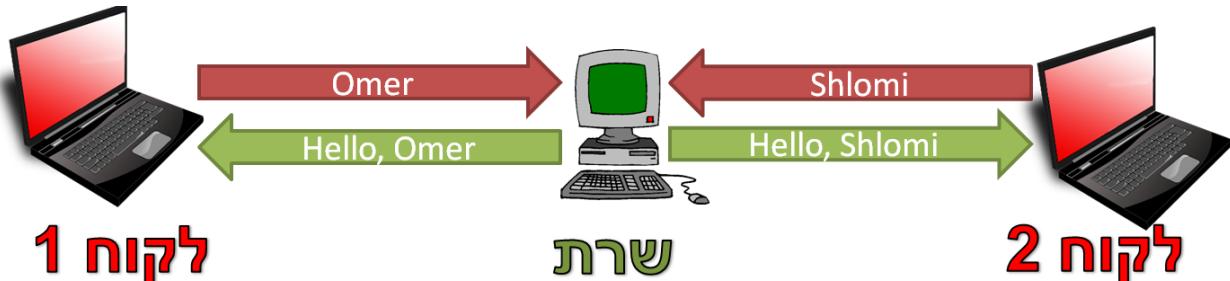
הפעילו את סקירהט הלוקוח מספר פעמים במקביל, וכתבו אל השרתשמות שונים, בכל פעם מלוקוח אחר. וידאו כי השרת מצליח להדפיס את ההודעות מהлокוחות שלכם.



## תרגיל 12.3 מודרך - שרת מרובה משתתפים עם תשובה לлокוחות

עד כה הצלחנו לקבל מידע מסווג לוקוחות מקביל. יכולת זה אمنם חשובה, אך אינה מספקת - علينا גם להצליח לתת שירות לлокוחות, כלומר לשלוח מידע אליהם.

בתרגיל זה נமמש שרת שמקבל חיבור מלוקה, מקבל את שמו של הלוקה, 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.



שים לב שהטיפול במקרה זה צריך להיות מקביל - כלומר, השרת יוכל להשאיר את החיבור עם הלוקה הראשון פתוח בעודו מספק שירות ללקוח השני.

לצורך התרגיל, נסתמך על הקוד שכתבנו בתרגיל המודרך הקודם, בו ביצענו רק קריאה של נתונים מהлокות. נשנה את הקוד באיזור שטיפל בהודעה שהתקבלת מלוקה 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 הלוקה. 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 שאתה Socket שעצמי קראתי ממנו את המידע מוקן לך שנשלח לו את המידע. במקרה זה, הפונקציה **send** עלולה להיתקע, ולא יוכל לתת שירות לשאר הלוקות. אכן אנחנו רואים מקרה פשוט, בו השרת מגיב לכל לקוח בנפרד, אך בהמשך נראה מקרים מורכבים יותר בהם חשוב במיוחד לוודא שכל Socket אליו אנו מעוניינים לשלוח מידע יהיה במצב שМОון לשליחה.

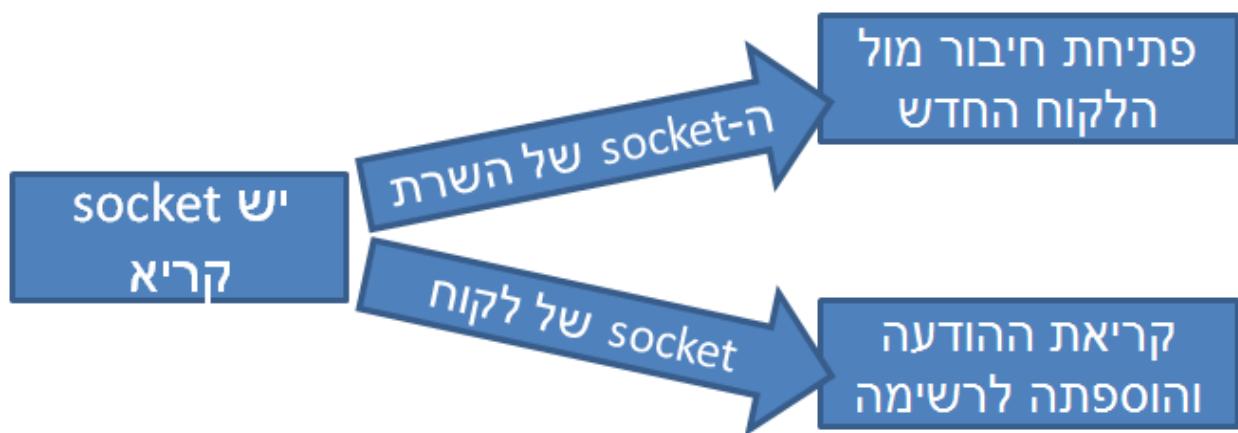
אי לכך, علينا להוסיף את הודעה לרשימה שתכיל את כל ההודעות שיש לשלוח, ולאחר מכן לשלוח אותה כנתיתן יהיה לעשות זאת:

```
if data == "":
    open_client_sockets.remove(current_socket)
    print "Connection with client closed."
else:
    messages_to_send.append((current_socket, 'Hello, ' + data))
```

שים לב שהוסףנו כאן לרשימה אובייקט מסוג tuple שמכיל את ה-*Socket* שאליו יש לשלוח את הודעה, ואת תוכן הודעה.מן הסתם, יהיה علينا להגדיר את הרשימה לפני שנוסיף אליו לתוכה, ולכן נעשה זאת לפני לולאת *the-while*:

```
messages_to_send = []
while True:
    ... (קוד הלולאה שתיארנו קודם)
```

זו הלוגיקה שמattaרת את התנהלות הנוכחיות של הלולאה:



בנוסף, בכל איטציה של הלולאה, ננסה לשלוח את כל ההודעות שעדיין לא נשלחו. נעשה זאת באמצעות פונקציה *send\_waiting\_messages*, אשר נממש בקרוב. הפונקציה מקבלת רשימת *Sockets* שנייה לשלוח אליהם מידע כרגע, ששמורה צצור במשתנה *wlist*:

```
messages_to_send = []
while True:
    ... (קוד הלולאה שתיארנו קודם)
    send_waiting_messages(wlist)
```

עם זאת, כאשר מימשנו את השרת שרק 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 נראהthus כה:

```
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 שנוצר לנו לעשות הוא למשוך את `send_waiting_messages`.

זכור, פונקציה זו מקבלת את רשימת ה-Sockets אליה ניתן לשלוח מידע, וכן 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 منها את שני האיברים שלה למשתנים נפרדים:

```
for message in messages_to_send:
```

```
    (client_socket, data) = message
```

שימוש בהפונקציה ה策ילה לגשת אל המשתנה `message` מכיוון שהוא משתנה גלובלי<sup>106</sup>.

עכשווי, علينا לבדוק האם ניתן לכתוב אל ה-Socket, כלומר האם הוא נמצא ברשימה ה-Sockets שניית כתוב אליום:

---

<sup>106</sup> באופן כללי, לא טוב להשתמש במשתנים גלובליים כשבאים קוד. עם זאת, ספר זה נועד ללמד רשותות ולא תכנות נכון, וכן לא נתעכט על אלטרנטיבות נכונות יותר מבחינה פיתוח תוכנה.

```
if client_socket in wlist:
```

במידה שnitן לכתוב אל ה-Socket, נשלח אליו את ההודעה:

```
client_socket.send(data)
```

כמו כן, לא נרצה שההודעה תשלוח שוב ללוקוח. לשם כך נסיר אותה מרשימת ההודעות שיש לשולח:

```
messages_to_send.remove(message)
```

באם ה-Socket לא היה מוכן לכתיבה, הרוי שהפונקציה תצא מבל' לשלוח את ההודעה ללוקוח, וההודעה לא תמחק מרשימת ההודעות שעוד יש לשולח, וילך 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))

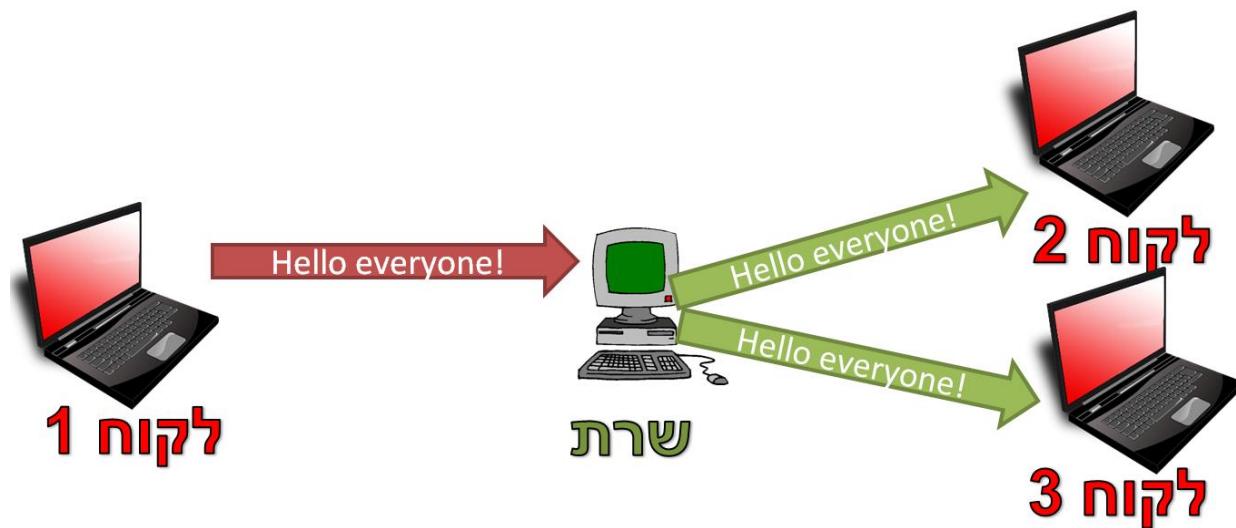
    send_waiting_messages(wlist)
```

### 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 שרך שולח מידע אל השרת ([פרק תכונות Sockets מתקדם: ריבוי משתמשים / תרגיל 12.2 - ל��וח לשרת מרובה משתתפים](#)).  
שפרו את הליקוח כך שגם יקרא מידע שנשלח אליו מהשרת וידפיס אותו למסך.  
הפעילו את סקריפט הליקוח מספר פעמים במקביל, וכתבו אל השרתשמות שונים, בכל פעם מליקוח אחר. וזאת כי כל לkekoch 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 הליקוח. כל לkekoch יכול לכתוב לשרת איזה מידע שהוא רוצה. השרת יdag לשאר הליקוחות את ההודעה שלו (ולא אליו בחזרה). כל לkekoch רואה את כל ההודעות של הליקוחות האחרים, אך לא יכול לדעת מי הגיעו ההודעה:



### הנחיות לתרגיל

יש להשתמש בספרייה `socket`, ולא בספריות עזר כגון `cgsocket`.

#### צד השרת

השרת צריך לשולח את הודעה לכל הלקוחות **מלבד** לזה ששלח אותה אליו. מקרה זה שונה מהשרותים שמיימנו קודם, ומורכב יותר. בכל פעם שתנסו לשולח את הודעה, עליכם לזכור למי כבר הצלחתם לשולח אותה, ולא לשולח פעמיים את אותה הודעה אףหาก.

#### צד הלקוח

סקריפט הלקוח צריך להצליח גם לקרוא מידע מהשרת, במידה שהוא רוצה לשולח הודעה לשאר משתמשים, וגם לקרוא מידע מהשרת. פעולה זו צריכה להתבצע במקביל, ולכן אסור להן להיות חוסמות (blocking).

בכל הפעמים בהן מיימנו ללקוח, על מנת לקרוא מידע מהשרת, השימושו פשוט במתודה `recv`. עם זאת, המתודה היא blocking, ולכן לא תוכל פשטוט לשימוש בה. במקרה זה, יהיה עליו להשתמש ב-`select` בדומה לכך.

כמו כן, בכל הפעמים בהן מיימנו ללקוח, קראנו מהשימוש 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 לשימוש בפונקציות `getch` ו-`kbhitz` של מודול זה.

## תרגיל 12.6 - צ'אט מתקדם



בתרגיל הקודם כתבתם צ'אט מרובה משתמשים. הצ'אט אמן אפשרות למשתמשים לתקשר, אך לא בצורה נוחה. אף משתמש לא יכול היה לדעת מי המשתמש שכטב 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. ככלmore, בכל פעם שעבדו רק על סעיף אחד. רק לאחר שהשלמתם סעיף מסוים ובדקתם כי הצ'אט פועל בהתאם למצופה, עברו לסעיף הבא.  
שים לב לקרוא את **תיאור הפרוטוקול שמופיע לאחר רשימת הסעיפים**, ולהסתמך עליו לאורך התרגיל כולו.

## **סעיפים**

1. כל הودעה תתחילה עם שם המשתמש שכתב אותה. לדוגמה:

talmid1: Hello everyone!

2. ליד כל הודעה פירשם הרשאה שהה ריא נכתבה. לדוגמה:

08:02 talmid1: Hello, the lesson is about to start.

3. אם משתמש שולח את המחרוזת "זעופ" – על השירות לנתק אותו.

4. במקרה שימוש עזב את הצ'אט (השרת נתק אוטו, או הוא החליט לעזוב בעצמו), על השרת לכתוב ללקוח שהוא יצא מהצ'אט. למשל:

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 כלום 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 לאחרים לחוש שהם מנהליים למידעיהם לא.

8. אפשרו למנהל למנות באופן דינامي מנהליים נוספים. **משמעות:** מי שאינו מנהל לא יכול למסות מנהליים.

9. תנו למנהלים את האפשרות להשתיק משתמש. הכוונה היא שהמשתמש יוכל לראות מה כתוב בדף, אך לא יוכל לכתוב. אם הוא כותב, הוא יקבל תשובה – "You cannot speak here". Um zat, משתמש משוכנע כי על פניו אין אינטרנט, אז באפשרותו לשלוח הודעה ללקוח או ללקוח.

10. אפשרו שליחת הודעות פרטיות בין שני משתמשים (במקרה של הودעה פרטית), רק שני המשתמשים יכולים לקבל הודעתה (לפחות אחד מהם חייב להיות ידוע)

00:43 Italmid3: This is a private message

- את החלטה זו ניתן לראות בפינה הימנית של המסך, מתחת לכתובת "view manager".

## טיור הפרוטוקול

### צד הלקוח

על כל הודעה שהלקוח 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:**4Omer18This is an example**

הסבר:

- **באזטום** - 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".  
**שימוש 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 הראשון הוא "0"?

דוגמה למינוי מנהל:**4Omer26Shlomi**

הסבר:

- **באזטום** - 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 משמש:**4Omer33Avi**

הסבר:

- **באזטום** - 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 משמש, 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 הצעט.

דוגמה להשתקת משתמש:

4Omer43Avi

הסבר:

- **באודו** - 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:

- **באודו** - 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 הלוקחות רק במחוזות, כלומר הוא שולח מחוזת שתוצג ללקוח. כל הודעה נשלחת בפורמט הבא:

דוגמה	סוג	שדה
11	מספר	אורק המחוות
Hello world	מחוזת	המחוות

דוגמה לשילוח הודעה מהשירות ללוקות:

4409:47 talmid6 has been kicked from the chat!

הסבר:

- **באדום** - אורק כל הודעה שהשירות שולח. מכיוון שההודעה היא "09:47" .44 ."from the chat!"
- **בכחול** - ההודעה עצמה שהשירות שלח. בדוגמה זו - "chat!"

## תכנות Sockets מתקדם - סיכום

בפרק זה למדנו כיצד לתוכנת באמצעות Sockets שרת ולקוח היכולים לטפל בכמה בקשות במקביל. התחלנו בהבנת הצורך במימוש שירותים שטיפולים בכמה ליקוחות, ולאחר מכן הבנו את האתגר שבמימוש שירותים שכאה. 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 השרת 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 השרת 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, התרמודדנו 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 הודעה פרטיות.

בדרכו רכשנו כל' MERCHANTABILITY נוסף, שמאפשר לנו לתקשר עם מספר יישויות רשות במקביל מעל ממashing him - **Sockets**.

## פרק 13 - מילון מושגים

פרק זה כולל מונחים בהם נעשה שימוש לאורך הספר, והגדורותיהם. המילון מועד לשימוש במהלך הקריאה. ההגדורות מובאות בהקשר שלهن לפרק ולמידע שמצוין בספר, ולא מועדו לעמוד בזכות עצמן בכך להגדיר את המושגים.

### פרק 1 - תחילת מסע - איך עובד האינטרנט?

- **WWW** - ראשית תיבות של World Wide Web. אוסף עמודי האינטרנט אליום אנו גולשים בדף.
- **בקשה (Request)** - הودעה שנשלחת מהלקוח אל השרת בכך לבקש שירות כלשהו.
- **תגובה (Response)** - הודעה שנשלחת מהשרת אל הלוקה כמענה לבקשת.
- **כתובת מקור** - כתובת המצינית מי שלח חבילה מידע מסוימת.
- **כתובת יעד** - כתובת המצינית לאן חבילה מmoveנת.
- **ping** - כלי המאפשר לבדוק קישוריות לשוט 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

- **תקשרות 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 הרשת.
- **Wireshark** - תוכנת הסנפה.
- **פקטה (חבילה, Packet)** - חבילה מידע המכילה מוען, נמען ותוכן ההודעה. מונח זה מתאר גוש מידע בשכבה השרת.

- **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 השכבות.
- **ריבוב (Multiplexing)** - התהילהיר שבו מידע מספר מקורות משולב אל תוך משותף אחד.
- **הרעבה (Starvation)** - תופעה בה תחנה אחת מציפה את הקו בקצב אורך של מידע, ובכך מונעת מהתחנות אחרות גישה לשדר על הקו.
- **Encapsulation (כימוס)** - עטיפת מידע בשכבות נוספות.
- **Decapsulation (קילוף)** - הוצאת המידע של שכבה מסוימת.
- **Header (תחלית)** - מידע שימושיפה כל שכבה לתחילת הפקטה, מכיל מידע שימושיפש 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 המשמשת לקוח (בין אם אפליקציה ייעודית ובין אם באמצעות הדפסה), 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 לחיי משאב ברשת האינטרנט (ראשי התיבות: Uniform Resource Locator). האופן שבו בניו URL:

<protocol>://<host>/<resource\_path>?<parameters, separated by &>

- **לדוגמה:**
- 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. מנגנים 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 הנשלחה 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 חילוק ה-path שבל-URL, מופרדים ממנה 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 "זוכר" 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 הלוקוט 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 בזיכרון המקומי של הלוקוט, ויובאו מחדש מהשרת רק אם הגresa שם השתנתה. מנגנון זה עושה שימוש בשדות header.
- **Conditional-GET** - הבקשת הנשלחת על-ידי הלוקוט 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 הגresa שב-*cache* של הלוקוט עדכנית, תוחזר תגובה עם קוד מצב 304 שמסמן שהמשaab לא שונה (והמשaab לא שונה בתגובה - כך נחסכה תעבורה "מיותרת").

- **מנגנון האוטנטיקציה הבסיסי** שנכשל ב프וטוקול HTTP עושה שימוש בשדות header וב code status כדי לבצע את זהות והאימות. המנגנון ד' חלש, משומם שהוא אינו מציין את שם המשתמש והסיסמה, אלא רק מקודד אותם.
- **שאילתת DNS (DNS Query)** - שאלה בפרוטוקול DNS עבור שם דומיין מסוים.
- **אזור (Zone)** - מערכת-DNS הינה הרכנית, והתחם המفرد שיוצר את ההיררכיה הוא התחם נקודת ("."). 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 - 5

- **Scapy** - ספריית פיתון המאפשרת לעבוד עם חבילות מידע בצורה מתקדמת. מאפשרת הסנהפה, יצירה ושליחה 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** - כלិ המאפשר לדעת על איזה פורטים המחשב משתמש, ואילו קישורים קיימים כרגע.
- **포רטים מוכרים (Well known ports)** - הפורטים מהווים שבין 0 ועד 1023 (כולל). פורטים אלו מוקצים בידיANA עבור אפליקציות ספציפיות.
- **תקורה (Overhead)** - מידע נוסף (יתיר) שנשלח מעבר למידע שורצים להעביר. לדוגמה, לשילוח Header יש תקורה - עבור מידע בראשת שהוא מידע נוסף 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 המשמש, הוא מתייחס ל קישור כמו לשופרפת 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 פשטוט לשולח את החבילה. בפרוטוקול מסווג זה, אין הבטחה שהחביבה תגיע ליעדה. כמו כן, אין הבטחה שהחבילות תגעהnas 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 צפוייה לקבל את החבילה.
- **Checksum** - תוצאה של פעולה מתמטית שמתבצעת על המידע. ה-Checksum מתיוסף לחבילה, בטור 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 ACK. מספר ה-ACK ב-TCP מצין את המספר הסידורי של הבית הבא שמצווה להתקבל.
- **Three Way Handshake (לחיצת יד ושלשת)** - הדרך להרמת קשר ב-TCP. כוללת שלוש חבילות: חבית SYN, חבית ACK+SYN וחבית ACK.
- **ISN (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 מנקודת A' לנקודת B' ברשת.
- **(Internet Protocol) IP** - פרוטוקול שכבה שלישית הנפוץ באינטראנט.
- **ipconfig** - כלי המאפשר לראות מידע מייד על הגדרות הרשות שלנו. למשל, הכלי מראה מה כתובות ה-IPינו.
- **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 חבילות 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 שעביר 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 לשירות מחוץ לרשת. לשם כך, רכיב ה-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 חיצונית שנitin לנtab.
- **IPv6** - הגרסה החדשה של פרוטוקול IP. כתובות בגרסה זו של הפרוטוקול הן באורך 16 ביטים .(bytes)

## פרק 8 - שכבת הקו

- **Ethernet** - פרוטוקול של שכבת הקו, בו משתמשים כרטיסי רשת מסוג Ethernet (המחוברים באופן קווי).
- **כטובות MAC** - כטובות פיזיות של שכבת הקו. לכל כרטיס רשות יש כטובת צזו. כטובת MAC בפרוטוקול Ethernet הינה בגודל שווה בתים (bytes).
- **מסגרת (Frame)** - גוש מידע בשכבה השנייה.
- **ARP (Address Resolution Protocol)** - פרוטוקול שנועד למפות בין כטובות לוגיות של שכבת הרשת לכטובות פיזיות של שכבת הקו.
- **פורט (Port)** - " כניסה" ברכיב רשות 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 נדרש כדי לחבר כמה ישויות רשות יחד. ה-Hub אינו מכיר כטובות 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 נשלחה.
- **Switch (מtag)** - רכיב של שכבת הקו, השכבה השנייה. אי לך, ה-Switch מכיר כטובות MAC, מבין את המבנה של מסגרות בשכבה שנייה (למשל מסגרות Ethernet), וידע לחשב checksum. לאחר שה-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** - כטובות מסווג זה שייכות ליותר מישות אחת.
- **Unicast** - כטובת מסווג זה שייכת לישות אחת בלבד.

## פרק 10 – השכבה הפיזית

- **סיבית (Bit)** - קיצור של המושג ספירה בינארית, ספרה שיכולה להחזיק אחד משני ערכים: 0 או 1. סיבית היא תרגום של המושג הלועזי bit, שהוא קיצור לביטוי binary digit.
- **תווך (Medium)** - ברשתות תקשורת, תווך 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 אופן בו מתרגמים את הספרות 0 ו-1 לסימנים מוסכמים על גבי התווך.
- **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 אלקטרומגנטיים, שיכולים לנوع בvakuum ובתוווכים רבים אחרים).

- **הgal האלקטרומגנטי (Electromagnetic Wave)** - הוא סוג של gal שנע בתווים שונים במרחב (אוויר, מים, זכוכית, ריק/וакום, ועוד) באמצעות שינוי של השדות החשמליים והמגנטיים. האור שmagיע מהמשש ושהותו אנו רואים הוא gal אלקטרומגנטי 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, הampieude של gal הנושא (בתדר הקבוע) תשתנה לפי האמפליטודה של gal המידע.
- **אפנון מבוסס תדר (Frequency Modulation)** - בשיטת אפנון זו, משנה את התדר של gal הנושא על פי האמפליטודה של gal המידע.
- **מודם (Modem)** - קיצור (באנגלית) של Modulator & Demodulator - מכשיר שמאפן ומשחזר 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 ווק-טוקי (מכשיר קשר אלחוטי), בו רק צד אחד יכול לדבר בזמן שהצד השני מפסיק. 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 לדייק, ישנו מספר סוג כבלים אלו: CAT3, 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 המקטים תצטרכו כבל CAT5.
- **חיבור RJ-45** - השקע והתקע של כבלי הרשת הסטנדרטיים, כולל צורותם וסידור הכבלים הפנימיים לפוי צבע, מוגדר בטקן שנקרא Registered Jack RJ. עבור כבלי רשת Ethernet, התקן הוא RJ-45, אך ישנו תקנים דומים גם עבור כבלים אחרים כגון כבל הטלפון (RJ-11). יש לציין שהיבור זה נקרא גם חיבור RJ-45, 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 אורכי גל שנייתן למרחוק למדוד 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 לסייע לך לקרוא להתמצא כיצד להשתמש בכל מוסויים, או לך לקרוא המעניינים להכיר את החלופות לפקודות המוצגות מעל מערכת הפעלה מבוססת UNIX.

### הרשימה

שימוש ב-UNIX	שימוש ב-Windows	הין הציג בספר	מטרת הכל
ping -c 4 www.google.com	ping -n 4 www.google.com	<u>תחילת מסע - אין עובד<a href="#">האינטרנט?/ כתובות IP</a></u>	לבודק קישוריות, לשוט מרוחקת, ואת הזמן שלוקח לחבילה להגעה אליה ובחרזה.
traceroute -n www.google.com	tracert -d www.google.com	<u>תחילת מסע - אין עובד<a href="#">האינטרנט? / ענן האינטראנט</a></u>	למצוא את הדרך שעוברת חבילה בין המחשב שלי לנקודות קצה שונות.
nslookup www.google.com	nslookup www.google.com	<u>תחילת מסע - אין עובד<a href="#">DNS/האינטרנט?</a></u>	לבצע תשאלות DNS.
telnet google.com 80	telnet google.com 80	<u>שכבת האפליקציה/<a href="#">התקבוננות</a> מודרכת בתגובה HTTP</u>	התחברות כלוקוט לשירות מרוחק.
nslookup set type=<TYPE> <host/address>	nslookup -t<TYPE> <host/address>	<u>שכבת האפליקציה/<a href="#">תרגיל 4.15</a> - תשאלות רשומות מסוגים שונים</u>	תשאלות DNS עם סוג שאלות מסוים.
תלוי בגירסה הספציפית. הסביר ניתן למצוא כאן: <a href="http://goo.gl/AF0Ui3">http://goo.gl/AF0Ui3</a>	ipconfig /flushdns	<u>עדרוף/<a href="#">תרגיל 5.1</a> מודרך - הסופה של DNS</u>	לאפס את מתמונן רשומות ה-DNS.
netstat -na	netstat -na	<u>שכבת התעבורה/<a href="#">תרגיל 6.1</a> מודרך - אילו פורטים פתוחים במחשב שלי?</u>	לספק מידע על חיבור רשת, ספציפית חיבור בשכבת התעבורה.
ifconfig -a	ipconfig /all	<u>שכבת הרשת/<a href="#">מה כתובת ה-IP</a> של?</u>	להציג מידע על כרטיסי רשת.

שימוש ב-UNIX	שימוש ב-Windows	היכן הוצג בסופר	מטרת הכל'י
route -n	route print	<u>שכבת הרשת/ מהי טבלת הניתוב שלי?</u>	להציג טבלאות ניתוב.
dhclient -r	ipconfig /release	<u>שכבת הרשת/ תרגיל 7.8 מודרנ'</u> <u>-קיבלת IP באמצעות DHCP</u>	להתנתק משרת DHCP ולוותר על פרטי הרשת.
dhclient	ipconfig /renew	<u>שכבת הרשת/ תרגיל 7.8 מודרנ'</u> <u>-קיבלת IP באמצעות DHCP</u>	לקבל את פרטי הרשת משרת ה-DHCP.
arp -n	arp -a	<u>שכבת הקו/ מטמון (Cache) של ARP</u>	להציג את מטמון ARP.
arp -d 192.168.1.100	arp -d 192.168.1.100	<u>שכבת הקו/ מטמון (Cache) של ARP</u>	למחוק רשומה ממטמון ARP.

## זכויות יוצרים - מקורות חיצוניים

- [http://commons.wikimedia.org/wiki/File:World%E2%80%99s first dual-core smartphone comes to europe.jpg](http://commons.wikimedia.org/wiki/File:World%E2%80%99s_first_dual-core_smartphone_comes_to_europe.jpg)
- [http://commons.wikimedia.org/wiki/File:Ethernet\\_RJ45\\_connector\\_p1160054.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_flat.svg.med.png)
- [https://www.iconfinder.com/icons/23912/router\\_wifi\\_icon#size=128](https://www.iconfinder.com/icons/23912/router_wifi_icon#size=128)
- [https://www.iconfinder.com/icons/47998/history\\_qualification\\_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-itzikgur/Videos-1-icon.html>
- [http://commons.wikimedia.org/wiki/File%3AInternational\\_Morse\\_Code.PNG](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: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:Amplitude-modulation_he.svg)
- [http://commons.wikimedia.org/wiki/File:Frequency\\_Modulation.svg#mediaviewer/File:Frequency\\_Modulation.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\\_.546jpg](http://commons.wikimedia.org/wiki/File:TP_Neostrada_Thomson_SpeedTouch_.546jpg)

[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:CAT5e_Cable.jpg#mediaviewer/File:CAT5e_Cable.jpg)

[http://commons.wikimedia.org/wiki/File:Cat\\_5.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%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:Parabolic_antennas_on_a_telecommunications_tower_on_Willans_Hill.jpg)

[http://commons.wikimedia.org/wiki/File:Multimode\\_stepindex\\_optical\\_fiber.svg](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.