

תרגיל בית 2

תאריך הגשה: 15.5.2023 @ 23:59

האחראי על התרגיל: נדב, דוא"ל: nadav.adir@campus.technion.ac.il

נמקו היטב אך בקצרה את כל תשובותיכם. תשובה לא מנומקת לא תזכה במלוא הניקוד!

ההגשה הינה בזוגות בלבד, אלקטרונית באתר בפורמט PDF.

שאלה מס' 1 - DNS (20 נק')

שאלת מעבדה זו היא בנושא מערכת ה-DNS שנלמדה בכיתה. על מנת לבצע אותה תחילה עליכם לקרוא את

<https://www.ietf.org/rfc/rfc1034.txt> - **RFC 1034**

<https://www.ietf.org/rfc/rfc1035.txt> - **RFC 1035**

כמתואר ב-RFCs, מערכת ה-DNS מתרגמת hostnames לכתובות IP, וממלאת תפקיד קריטי בתשתית האינטרנט. במעבדה זו אנו נתבונן מקרוב על צד הלקוח במערכת ה-DNS. תפקיד הלקוח ב-DNS הוא פשוט יחסית – הלקוח שולח שאילתא (query) לשרת ה-DNS המקומי ומקבל תשובה (response) בחזרה. כפי שלמדנו בהרצאה, הרבה מפעולת מערכת ה-DNS שקופה ללקוח, כששרת ה-DNS מתקשרים אחד עם השני במורד ההיררכיה, בין אם בשאילתות רקורסיביות או שאילתות איטרטיביות. אולם, כאמור, במעבדה זו נתמקד בניתוח המערכת מצד הלקוח.

nslookup

במעבדה זו נבצע שימוש בכלי שנקרא nslookup, שזמין ברוב מערכות ההפעלה הנפוצות (Win/Linux/OSX).

על מנת להריץ כלי זה נקליד את הפקודה nslookup בחלון הפקודה (Run->cmd).

*במעבדה זו הדוגמאות יהיו תחת **Windows** אך השימוש בלינוקס ו**OSX** דומה.

הפעולה הבסיסית ביותר של כלי זה מאפשרת למחשב המריץ אותו לתשאל כל שרת DNS לגבי כל רשומת DNS כלשהי, הכלי שולח שאילתא לשרת המבוקש לגבי הhostname המבוקש ומציג למשתמש את התשובה.

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

C:\Users\ynezri>nslookup wikipedia.org
Server: dapp.csf.technion.ac.il
Address: 132.68.32.4

Non-authoritative answer:
Name: wikipedia.org
Addresses: 2620:0:862:edia::1
          91.198.174.192

C:\Users\ynezri>nslookup -type=NS wikipedia.org
Server: dapp.csf.technion.ac.il
Address: 132.68.32.4

Non-authoritative answer:
wikipedia.org nameserver = ns1.wikimedia.org
wikipedia.org nameserver = ns0.wikimedia.org
wikipedia.org nameserver = ns2.wikimedia.org

ns1.wikimedia.org internet address = 208.80.153.231
ns0.wikimedia.org internet address = 208.80.154.238
ns2.wikimedia.org internet address = 91.198.174.239

C:\Users\ynezri>nslookup wikipedia.org google-public-dns-a.google.com
Server: google-public-dns-a.google.com
Address: 8.8.8.8

Non-authoritative answer:
Name: wikipedia.org
Addresses: 2620:0:862:edia::1
          91.198.174.192

C:\Users\ynezri>
```

צילום המסך מראה תוצאות של 3 פקודות nslookup שונות. בדוגמא זו הרצתי את הכלי כאשר המחשב שלי היה מחובר לרשת הפקולטית, שבה שרת ה-DNS הדיפולטי הוא daap.csf.technion.ac.il. כאשר מריצים את nslookup ולא מציינים שרת dns הכלי מתשאל את שרת ה-DNS הדיפולטי.

כעת נתבונן בפקודה הראשונה:

nslookup wikipedia.org

במילים, הפקודה אומרת "שלח לי בבקשה את כתובת ה-IP של hostname wikipedia.org", הפלט שמוצג כתוצאה מפקודה זו הוא:

1. כתובת השרת שסיפק את התשובה.
 2. התשובה עצמה, שמכילה את הhostname וכתובות ה-IP של hostname זה (IPv4 ו IPv6).
- למרות שהתשובה הגיעה משרת ה-DNS המקומי בטכניון, יכול מאוד להיות שהשרת המקומי תשאל בצורה איטרטיבית כמה שרתי DNS אחרים על מנת לקבל תשובה זו.

נתבונן בפקודה השנייה:

nslookup -type=NS wikipedia.org

בדוגמא זו הגדרנו את המאפיין "-type=NS", מאפיין זה גורם לnslookup לשלוח שאילתא מסוג NS לשרת ה-DNS הדיפולטי, ובמילים "שלח לי בבקשה את hostnames של שרת ה Authoritative DNS עבור wikipedia.org". התשובה שמוצגת בצילום המסך מציינת דבר ראשון את השרת שסיפק את התשובה (השרת הטכניוני) ולאחר מכן את שמות של שלושה שרתי DNS של ויקיפדיה.

חדי העין מבינכם בטח שואלים את עצמם אם אמרנו שהכלי מבקש את שרתי ה Authoritative DNS מדוע מופיעה הכותרת Non-authoritative answer מעל התשובה?

התשובה היא שמכיוון שהתשובה הגיעה מcache של אחד השרתים בדרך ולא משרת אוטוריטטי של ויקיפדיה. לבסוף, התשובה גם כוללת את כתובות ה-IP של שרתי ה-DNS של ויקיפדיה.

נתבונן בפקודה השלישית:

nslookup wikipedia.org google-public-dns-a.google.com

בפקודה זו אנו מציינים שאנו מעוניינים לתשאל את השרת google-public-dns-a.google.com במקום את השרת הדיפולטי. אם ננסה לדוגמא לתשאל את אחד משרתי ה-DNS של ויקיפדיה כנראה שנענה בסירוב (נסו לשער למה...), לכן פנינו לשרת פומבי (במקרה זה פנינו לשרת של גוגל שמציעה שירותי DNS בחינם (<https://developers.google.com/speed/public-dns/?hl=en>))

התשובה שקיבלנו זהה לזו שקיבלנו מהשרת הדיפולטי.

כעת, שעברנו על השימוש הבסיסי בnslookup ענו על השאלות הבאות:

- a. השתמשו בכלי nslookup על מנת למצוא כתובת IP של שרת web של אוניברסיטה כלשהי באירופה. צרפו צילום מסך של התוצאה.
- b. כעת השתמשו בכלי על מנת למצוא את שרתי ה-DNS האוטוריטיים של אוניברסיטה כלשהי בארצות הברית. צרפו צילום מסך של התוצאה.

ipconfig

בפקודה ipconfig ניתן להשתמש על מנת לנהל את מטמון ה-DNS שנשמר על המחשב. לדוגמא אם נריץ את הפקודה:

ipconfig /displaydns

נקבל את כל הרשומות ששמורות במטמון ה-DNS, בכל רשומה נמצא בין היתר את הhostname, כתובת ה-IP, וכמה זמן נותר עד שנפנה את הרשומה מהמטמון (TTL – בשניות). מכיוון שבהמשך אנו הולכים לנתח שאילתות DNS באמצעות Wireshark, נרצה לנקות את המטמון הזה כדי שהתשובות יגיעו לשרתים עצמם ולא יחזרו מהמטמון. על מנת לנקות את המטמון נריץ את הפקודה:

ipconfig /flushdns

ניתוח פרוטוקול DNS באמצעות Wireshark

כעת שאנחנו יודעים כיצד לעבוד עם nslookup ו ipconfig, נוכל לנתח את שאילתות ה-DNS שהמחשב שלנו מבצע. בתור התחלה, "נתפוס" בעזרת Wireshark את חבילות ה-DNS שנשלחות בעקבות גלישה רגילה בדפדפן. בצעו את הפעולות הבאות:

- השתמשו ב ipconfig על מנת לנקות את מטמון ה-DNS של המחשב.
- פתחו את הדפדפן המועדף עליכם ונקו את מטמון ה-DNS שלו (כן, גם הדפדפן שומר אחד כזה...), בכרום היכנסו ל: <chrome://net-internals/#dns> ולחצו על הכפתור clear host cache
- פתחו את Wireshark, פתחו את חלון הממשקים בחרו ב Interfaces בעל חיבור לרשת האינטרנט.
- השתמשו ב ipconfig על מנת לגלות את כתובת ה-IP של Interface זה.
- בשורת ה Filter הכניסו את הביטוי הבא:

dns and ip.addr == x.x.x.x

כאשר במקום המתאים הכניסו את כתובת ה-IP שמצאתם, פילטר זה יגרום לתצוגה של חבילות מסוג DNS ושכתובת המקור או היעד שלהן היא הכתובת שהכנסתם.

- התחילו את פעולת ה-capture ב-Wireshark

- בדפדפן היכנסו לאתר <http://www.ietf.org>.
- כעת, עצרו את פעולת ה-Capture ב-Wireshark.

ענו על השאלות הבאות:

- c. אתרו את השאילתה והתשובה הרלוונטית, האם הן נשלחו מעל פרוטוקול UDP או TCP ?
- d. מהו destination port של השאילתה? מהו source port של התשובה?
- e. לאיזה כתובת IP נשלחה השאילתה? השתמשו בipconfig על מנת לקבוע את כתובות ה-IP של שרת ה-DNS המקומי שלכם. האם זו אותה הכתובת? (שימו לב כי ייתכן כי ה-router הביתי שלכם יתפקד כשרת ה-DNS המקומי עבורכם וזה בסדר, אנא ציינו בתשובתכם אם זה אכן המקרה).
- f. הסתכלו על הודעת התשובה שהתקבלה. כמה "תשובות" התקבלו? איזה מידע כל תשובה כזו מכילה?
- g. צרפו צילומי מסך של השאילתה והתשובה המכילים את המידע הרלוונטי.

כעת, נשחק קצת עם nslookup, בצעו את הפעולות הבאות:

- התחילו את פעולת ה-capture ב-Wireshark (במידת הצורך נקו את מטמון ה-DNS והכניסו filter מתאים).
 - הריצו פקודת nslookup עבור הכתובת <https://www.reddit.com>
 - עצרו את פעולת ה-Capture ב-Wireshark.
- *שימו לב ש nslookup עלול לייצר עוד שאילתות על ווריאציות שונות של ה-hostname. התייחסו לזו המתשאלת עבור reddit.com עבור כתובת IPV4.

ענו על השאלות הבאות:

- h. אתרו את השאילתה והתשובה הרלוונטית, מהו destination port של השאילתה? מהו source port של התשובה?
- i. לאיזה כתובת IP נשלחה השאילתה? האם זו הכתובת של שרת ה-DNS המקומי שלכם?
- j. הסתכלו על הודעת השאילתה שנשלחה, איזה סוג שאילתה זו?
- k. הסתכלו על הודעת התשובה שהתקבלה, כמה תשובות התקבלו בהודעה זו? איזה מידע מכילה כל תשובה?
- l. צרפו צילומי מסך של השאילתה והתשובה המכילים את המידע הרלוונטי.

כעת, חזרו על אותו הניסוי אך הריצו את הפקודה: `nslookup -type=NS reddit.com`.

ענו על השאלות הבאות:

- m. לאיזה כתובת IP נשלחה השאילתה? האם זו הכתובת של שרת ה-DNS המקומי שלכם?
- n. הסתכלו על הודעת השאילתה שנשלחה, איזה סוג שאילתה זו?
- o. הסתכלו על הודעת התשובה שהתקבלה. איזה שרתי DNS של גוגל מופיעים בה? האם התשובה מכילה גם את כתובות ה-IP של שרתים אלה?
- p. צרפו צילומי מסך של השאילתה והתשובה המכילים את המידע הרלוונטי.

כעת, חזרו על אותו הניסוי אך הריצו את הפקודה

`nslookup reddit.com google-public-dns-a.google.com`

ענו על השאלות הבאות:

- q. מהי לדעתכם השאילתה הראשונה שנשלחת בעקבות הרצת פקודה זו?

- r. אתרו את השאילתה והתשובה הראשונה שנשלחו. לאיזה כתובת IP נשלחה השאילתה? האם זו הכתובת של שרת ה-DNS המקומי שלכם? מהי כתובת ה-IP שהתקבלה בתשובה (התייחסו לתשובה מסוג A)?
- s. אתרו את השאילתה והתשובה שנשלחו עבור reddit.com. לאיזה כתובת IP נשלחה השאילתה? האם זו הכתובת של שרת ה-DNS המקומי שלכם?
- t. צרפו צילומי מסך של השאילתות והתשובות המכילים את המידע הרלוונטי.

שאלה מס' 2 - ICMP (14 נק')

1. עיינו ב-RFC-792 והסבירו במקרים הבאים איזו חבילת ICMP (type+code) תשלח:
 - a. נתב קיבל חבילה שערך שדה ה-TTL שלה הוא 1 והוא לא היעד שלה.
 - b. נתב קיבל חבילה שגדולה יותר מה-MTU אבל דגל ה-DF של החבילה דלוק.
 - c. נתב קיבל חבילה אך החוצץ שלו לשליחת חבילות מלא.
 - d. מחשב היעד קיבל חבילה שה-destination port שלה לא זמין.
 - e. נתב מקבל חבילה ומזהה שכתובת המקור של החבילה והכתובת של ה-hop הבא שייכים לאותה רשת והוא מעוניין להודיע למחשב השולח שעדיף לו לשלוח את החבילה ישירות ל-hop הבא במקום דרכו.
 - f. נתב קיבל הודעת ping echo ומעוניין להחזיר תשובה.
 - g. הנתב קיבל חבילה שהרשת של כתובת היעד שלה לא מופיעה בטבלת הניתוב שלו.
2. הסבר בקצרה את עקרון פעולת מנגנון mtu discovery.
3. הסבר בקצרה את הקשר בין חבילת IP לחבילת ICMP.

מעבדת MININET (66 נק')

על מנת לקבל הבנה טובה יותר של הפרוטוקולים שאנחנו לומדים בכיתה (IP, TCP, RIP, OSPF, BGP...), נרצה לקבל ניסיון "hands-on" בתהליכים השונים שפרוטוקולים אלו מעורבים בהם. אבל ציוד הרשת שבו פרוטוקולים אלו ממומשים הוא יקר ומסובך, לכן נרצה ליצור סביבה וירטואלית בה נוכל לבנות רשתות - משתמשי קצה (hosts), מתגים (switches), נתבים (routers), וקישורים (links) שתאפשר לנו לבחון את התנהגות הפרוטוקולים הללו מקרוב...

שאלה 1: הסבירו בקצרה מהם ההבדלים בין נתב (router) למתג (switch)?

לשם כך נשתמש בMininet.

Mininet הוא אמולטור רשת, אשר יוצר רשת וירטואלית, מעל ליבת Linux אחת. התכנה משתמשת ב Network Namespaces על מנת לגרום למערכת הפעלה אחת לדמות רשת שלמה. הישויות השונות ב mininet מתנהגות כמו מחשבים אמיתיים, ניתן להתחבר אליהן ב ssh, ניתן להריץ עליהן תוכנות כגון ping, wireshark וכו' ובעצם לעשות כל מה שאנחנו מצפים ממחשב רגיל המריץ Linux. הקישורים ב mininet גם כן מדמים קישור אמיתי עם אפשרות להגדיר רוחב פס משתנה, אחוזי שגיאה, delay ועוד.

Mininet היא תכנה מורכבת שמצריכה מספר רב של הגדרות, על מנת להקל על ההתקנה ולצמצם את מספר השגיאות שנובעות מהגדרות אלו, אנו הולכים להריץ מכונה וירטואלית שעליה מותקנת סביבה מוכנה של mininet.

שלב 1- הורדה והתקנה:

הורידו והתקינו את התכנה VirtualBox מהקישור הבא:

<https://www.virtualbox.org/wiki/Downloads>

הורידו את קובץ התמונה שבו נשתמש מהקישור הבא:

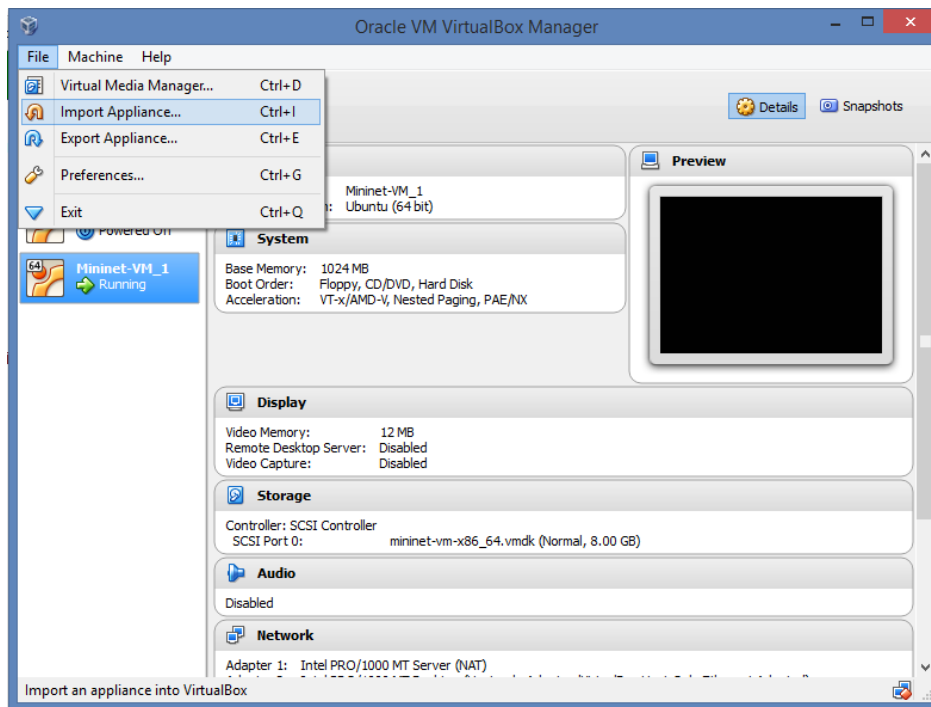
mininet VM

הקובץ מכיל תמונה של מערכת הפעלה 14.04 ubuntu שעליה מותקן האמולטור mininet וחבילת quagga שמממשת פרוטוקולי רשת כגון OSPF RIP BGP וכו'.

כעת בצעו import למכונה שהורדתם:

```
file->import appliance
```

ובחרו את הקובץ של המכונה שהורדתם.



שימו לב - לפעמים יש צורך לאפשר וירטואליזציה בתפריט **bios** של המחשב.

ההורדה והפתיחה עלולים לקחת קצת זמן...זה זמן טוב להפסקת קפה.

כעת הריצו את המכונה.

על מנת להתחבר תדרשו להזין את הנתונים הבאים:

Login: mininet

Password: mininet

כעת הריצו את הפקודה הבאה בחלון שנפתח:

ifconfig -a

הפקודה מציגה את ממשקי הרשת שזמינים במכונה הוירטואלית. עליכם לחפש ממשק שכתובת ה IP שלו מהצורה:

192.168.x.x

```

mininet@mininet-vm:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:40:d8:bc
          inet addr:192.168.56.102  Bcast:192.168.56.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7621 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6446 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:911913 (911.9 KB)  TX bytes:1289781 (1.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4684 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4684 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:607000 (607.0 KB)  TX bytes:607000 (607.0 KB)

mininet@mininet-vm:~$

```

זכרו את הכתובת הזו כי נשתמש בה בהמשך.

כעת נרצה לחבר את המכונה הוירטואלית לאינטרנט. הריצו את הפקודה הבאה:

sudo dhclient eth1

ודאו שהמכונה אכן מחוברת לרשת האינטרנט (ping google.com למשל).

על מנת להקל על העבודה במכונה הוירטואלית, ולאפשר עבודה בחלונות מרובים אנחנו הולכים להתחבר אליה דרך לקוח ssh .

על מנת לאפשר תצוגה גרפית דרך ssh נצטרך להתקין מערכת שתתמוך במערכת החלונות X11, לכן נתקין את התכנה xming. הורידו את הקובץ הבא:

<http://sourceforge.net/projects/xming/files/latest/download>

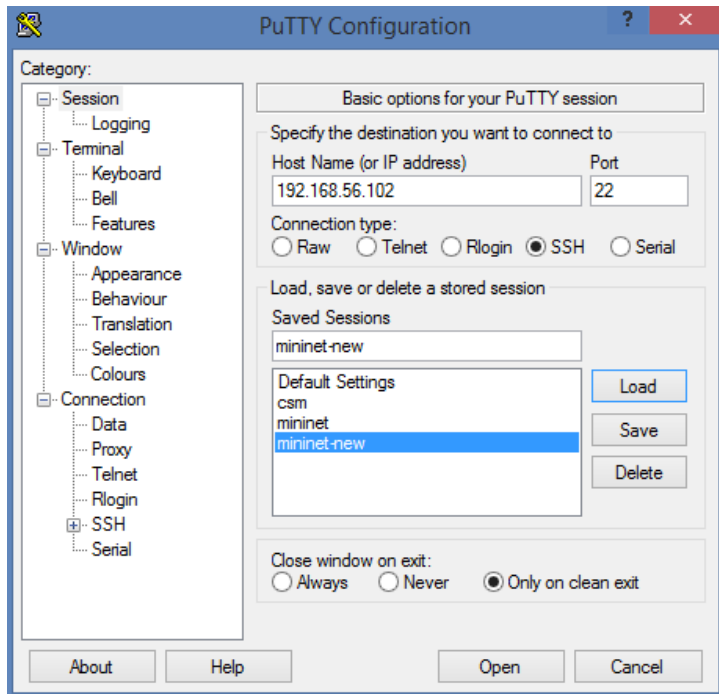
ובצעו את תהליך ההתקנה.

הדבר האחרון שנצטרך להוריד ולהתקין הוא לקוח ssh שתומך בX11, לאורך ההדגמות בקורס אנחנו נשתמש בPutty:

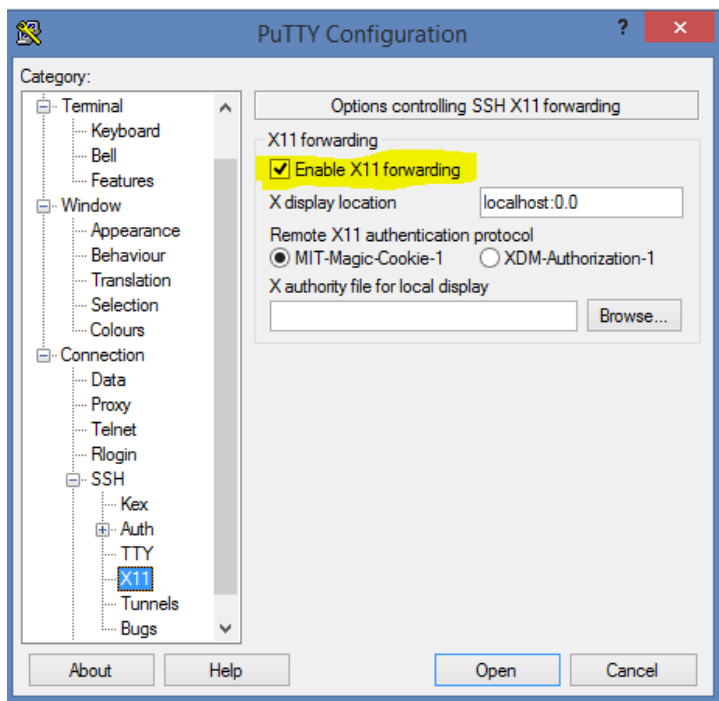
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

שלב 2 – הפעלה בסיסית

כעת הפעל את התכנה Xming, ואת לקוח הssh, התחברו לכתובת הקו שמצאתם קודם לכן:



ודאו שיש תמיכה בX11:



על מנת להתחבר למכונה תאלצו שוב להזין את שם המשתמש והסיסמא הבאים:

Login: **mininet**

Password: mininet

לאחר מכן הריצו את הפקודה הבאה:

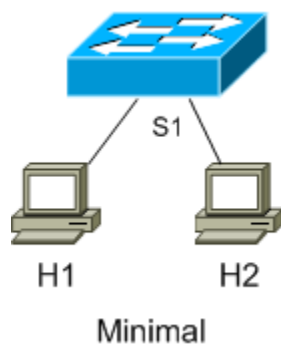
sudo mn

מכיוון שmininet משנה את הגדרות מערכת ההפעלה כל פעם שנריץ את האמולטור נצטרך לעשות זאת בהרשאות root (ע"י שימוש בsudo). כעת אתם אמורים לראות את הפלט הבא:

```
mininet@mininet-vm: ~  
mininet@mininet-vm:~$ sudo mn  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
c0  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet>
```

בואו נפענח את מה שקרה, בשורה הראשונה ניתן לראות שmininet מנסה ליצור רשת כלשהי. מכיוון שלא ציינו באיזה טופולוגיה להשתמש, mininet יצר רשת עם [הטופולוגיה](#) הדיפולטית שלו (על איך להגדיר לmininet באיזו טופולוגיה להשתמש נדבר בהמשך).

הטופולוגיה הדיפולטית של mininet נראית כך:



* מכיוון שmininet פותח במקור על מנת לחקור [רשתות SDN](#) לכל טופולוגיה מתווספת ישות שנקראת בקר (controller), אך בד"כ בקר זה יהיה שקוף לנו.

ניתן לראות שmininet אכן יוצר רשת המכילה שני משתמשי קצה h1 h2, ומתג s1. בנוסף mininet יוצר שני לינקים (s1,h1) (s1,h2) המחברים בין משתמשי הקצה למתג.

לאחר מכן ניתן לראות שנכנסנו ל-[CLI](#) של mininet, אפשר לראות זאת ע"י השינוי בprompt, בואו ננסה להריץ כמה פקודות.

על מנת להציג את הישויות ברשת הריצו את הפקודה **nodes**, אמור להתקבל הפלט הבא:

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
```

על מנת לקבל תמונת רשת המכילה גם את הקישורים הקיימים בין הישויות הריצו את הפקודה **net**, אמור להתקבל הפלט הבא:

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

כאן בעצם אנחנו רואים פירוט של כל הישויות ברשת, ולאחר כל ישות יש פירוט של כל הקישורים שאליהם היא מחוברת. לדוגמה עבור הישות h1 מוצג הקישור h1-eth0:s1-eth1, השמות שמוצגים משני צדי הנקודתיים הם שמות הממשקים (Interfaces) בשני צדי הקישור. במקרה שלנו ממשק הוא בעצם כרטיס רשת (NIC) וירטואלי, לכל ממשק כזה מוגדרת כתובת mac, ובד"כ בעת יצירת קישור, mininet ידאג לשייך כתובת IP ו Subnet Mask לכל ממשק.

אם נזכר בהקדמה, אמרנו שכל ישות מדמה מחשב נפרד המריץ את מערכת ההפעלה Linux (אמירה זו אינה מדויקת, תכף נראה מדוע...), בואו נראה כיצד אמירה זו באה לידי ביטוי. על מנת להריץ פקודה על ישות מסוימת נכתוב את שם הישות ולאחר מכן את הפקודה ב-CLI של mininet. לדוגמה אם נרצה להריץ את הפקודה **date** (תכנית המחזירה את התאריך והשעה) על הישות h1 נריץ את הפקודה **h1 date**:

```
mininet> h1 date
Mon Aug 17 17:36:32 PDT 2015
mininet>
```

נקבל בחזרה את התאריך והשעה.

אם נרצה ש h1 יכתוב את השם שלו לקובץ נריץ את הפקודה **h1 echo "h1 says hello world" > h1hello**, בטח אתם שואלים את עצמכם למה צריך שיעור פקודות בלינוקס עכשיו, שימו לב מה קורה כאשר אנחנו מריצים את הפקודה **h2 cat h1hello**:

```
mininet> h1 echo "h1 says hello world" > h1hello
mininet> h2 cat h1hello
h1 says hello world
mininet>
```

אם כל ישות מדמה מחשב נפרד איך h2 קרא את הקובץ ש h1 כתב?

התשובה הפשוטה היא שכל ישות לא מדמה מחשב נפרד, על מנת לאפשר יצירת טופולוגיות גדולות ולחסוך במשאבים כל הישויות בmininet חולקות את אותה מערכת הקבצים (File System), מה שמאפשר ליצור בmininet טופולוגיות בעלות מאות ואלפי ישויות על מחשב ביתי רגיל.

אז אם כל הישויות חולקות את אותה מערכת קבצים מה בעצם שונה ביניהן?

Mininet משתמש במנגנון הנקרא [Linux Network Namespaces](#) על מנת להבדיל בין הישויות. מנגנון זה בעצם יוצר לכל ישות מרחב רשת מבודד המכיל "מחסנית רשת" משלו (Network Stack) – טבלאות ניתוב, הגדרות רשת, ממשקים ועוד.

בכדי לראות שזה באמת קורה, נריץ את הפקודה **h1 ifconfig**. שימו לב ש [ifconfig](#) היא תכנית שמספקת לנו מידע על ההגדרות של ממשקים הקיימים במערכת ההפעלה ומאפשרת לערוך אותן:

```
mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet  HWaddr 5e:b1:40:79:02:eb
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         inet6 addr: fe80::5cb1:40ff:fe79:2eb/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:7 errors:0 dropped:0 overruns:0 frame:0
         TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:558 (558.0 B)  TX bytes:648 (648.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:8 errors:0 dropped:0 overruns:0 frame:0
         TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:672 (672.0 B)  TX bytes:672 (672.0 B)

mininet>
```

שאלה 2: התבוננו בהגדרות והסטטסטיקות של הממשק h1-eth0 (interface) בפלט של הפקודה ifconfig.

2.1 מה כתובת ה MAC של interface זה?

2.2 מהי כתובת ה IP של ה interface הזה? מה ה Subnetmask? כמה מקסימום hosts יכולים להיות ברשת זו?

2.3 מהי כתובת ה IPV6 של ה interface הזה?

2.4 מה גודל המסגרת המקסימלי שה interface הזה יכול לשלוח ולקבל?

2.5 מה משמעות השדה txqueuelen?

שאלה 3: הסבירו בקצרה מהו Loopback Interface, מה השימוש בו ומה אתם יכולים להגיד על כתובת ה IP שלו?

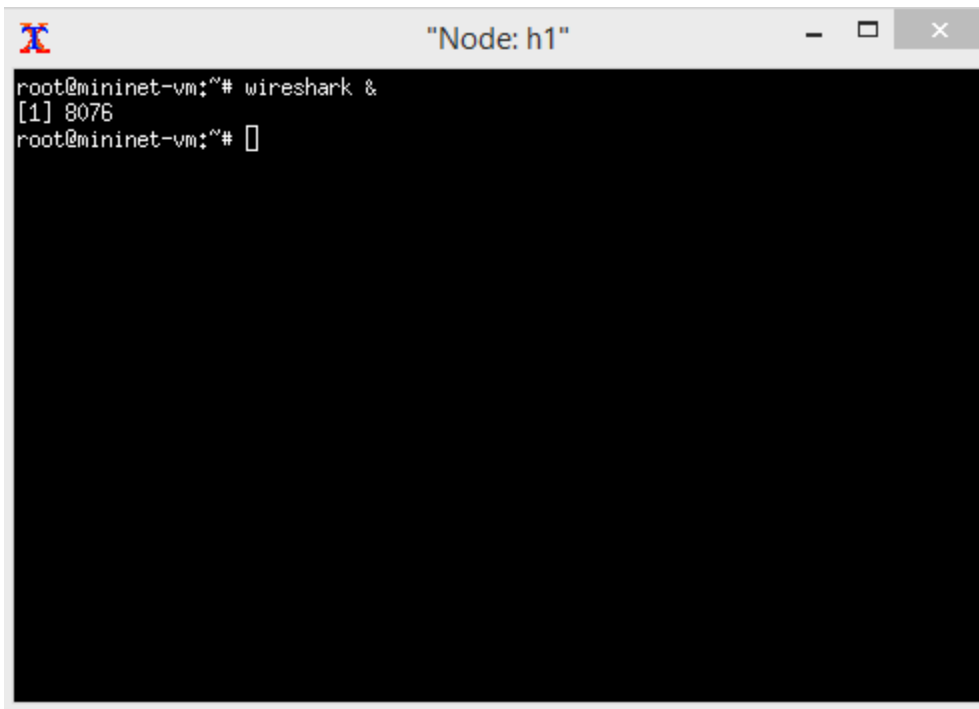
שאלה 4: הריצו את הפקודה ifconfig על h2 צרפו צילום מסך של הפלט. האם הפלט זהה לפקודת ifconfig שהרצנו על h1? מה משותף לכתובות ה ip של h1-eth1 ו h2-eth?

כעת נרצה לבדוק האם באמת ניתן לשלוח מידע ברשת שלנו, הריצו את הפקודה **h1 ping h2**, וודאו שחבילות באמת עוברות בין משתמשי הקצה. שימו לב ש mininet מחליף לבד בין השם h2 לבין כתובת ה IP שלו.

כעת נריץ את התכנה wireshark על מנת לקבל מבט טוב יותר על מה שקורה ברשת. הריצו את הפקודה:

xterm h1

אם Xming פועל, והכל מוגדר כראוי, אמור להפתח חלון terminal חדש, תחת network namespace של h1 שממנו נוכל להריץ תוכנות בעלות ממשק גרפי. בחלון זה הריצו את הפקודה **wireshark &**:



```
root@mininet-vm:~# wireshark &
[1] 8076
root@mininet-vm:~#
```

מכיוון ש mininet רץ בהרשאות root נקבל אזהרה על פעולה תחת הרשאות אלה, סגרו חלון אזהרה זה. וודאו כי בחלון בחירת הממשקים של wireshark מופיעים הממשקים של h1.

מכיוון שכולכם כבר מומחי wireshark אין צורך להרחיב על היכולות של תכנה זו, אתם יכולים ליצר קצת תעבורה (**ping** או **iperf** לדוגמא) על מנת לוודא שהכל עובד כמו שצריך.

מה שנותר לנו לעשות זה ללמוד כיצד ליצור טופולוגיות חדשות ב mininet, ישנן 2 דרכים ליצור טופולוגיה חדשה:

- דרך א'- שימוש בשורת הפקודה
 - דרך ב'- כתיבת קוד פייתון
- מכיוון שאין לנו אינטרס ללמד אתכם פייתון, או שנשתמש בדרך א', או שנספק לכם קוד פייתון מוכן של טופולוגיה מוכנה.

מי שמעוניין להעמיק בנושא מוזמן לקרוא ב: <http://mininet.org/walkthrough/>

עד עכשיו ראינו כיצד לדמות בעזרת mininet משתמשי קצה ומתגים. אבל הרבה מהפרוטוקולים שאנו מעונינים לחקור (RIP, OSPF, BGP) מיושמים על גבי נתבים. לצערנו Mininet לא מגיע עם ישות נתב מובנת, אולם ניתן להשתמש בחבילות תכנה שממשות פרוטוקולי ניתוב מעל linux. אנחנו נשתמש בחבילה בשם [quagga](#) (על שם החיה שנכחדה [quagga](#)). החבילה מותקנת כבר על המכונה הוירטואלית שברשותכם.

חבילה זו מאפשרת לנו להריץ כמה [daemons](#) (תהליך שרץ ברקע), אחד לכל פרוטוקול ניתוב. ו daemon אחד שנקרא zebra.

לכל daemon יש קובץ קונפיגורציה וממשק CLI (Command Line Interface) שניתן להתחבר אליו באמצעות [telnet](#) (תכף נראה בדיוק איך). ממשק CLI זה דומה לממשק CLI של מערכת ההפעלה [Cisco IOS](#), אולם חלק גדול מהפונקציות ממומש באופן שונה.

מדריך מלא לכל הפקודות הנתמכות (עבור כל daemons) ניתן למצוא בקישור הבא:

<http://www.nongnu.org/quagga/docs.html>

במקרה הראשון, נראה דוגמא פשוטה לאיך ניתוב מתבצע בעזרת הגדרת נתיבים סטטיים בלבד.

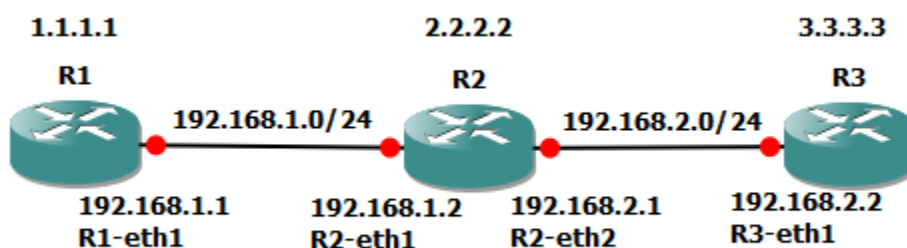
לשם כך נוריד את הקוד למעבדה זו. בתיקית הבית הריצו את הפקודה:

git clone <https://github.com/cs236341/staticRoute.git>

כעת הכנסו לתיקיה שנוצרה, והריצו את הפקודה:

sudo python staticRoute.py

אם הכל עובד כמו שצריך, נוצרה הטופולוגיה הבאה (וודאו זאת באמצעות הפקודה net):



בעצם יצרנו שלושה ראוטרים, כל ראوتر הוא switch שמריץ daemon של zebra, מכיוון שאנו נגדיר רק ניתוב סטטי, אין צורך להריץ את daemons של פרוטוקולי הניתוב השונים. שלושת הראוטרים מחוברים בטור, כאשר כל חיבור הוא רשת פרטית (בכתובות השייכות ל 192.168.1.0/24 ו 192.168.2.0/24). בנוסף לכל ראوتر יש כתובת נוספת, שמשויכת ל loopback interface שלו ומשמשת לניהול הראוטרים. כרגע לא ניתן להעביר חבילות מה Loopback interface של R1 לזו של R3. וודאו זאת, פתחו טרמינל ל R1 והריצו את הפקודה:

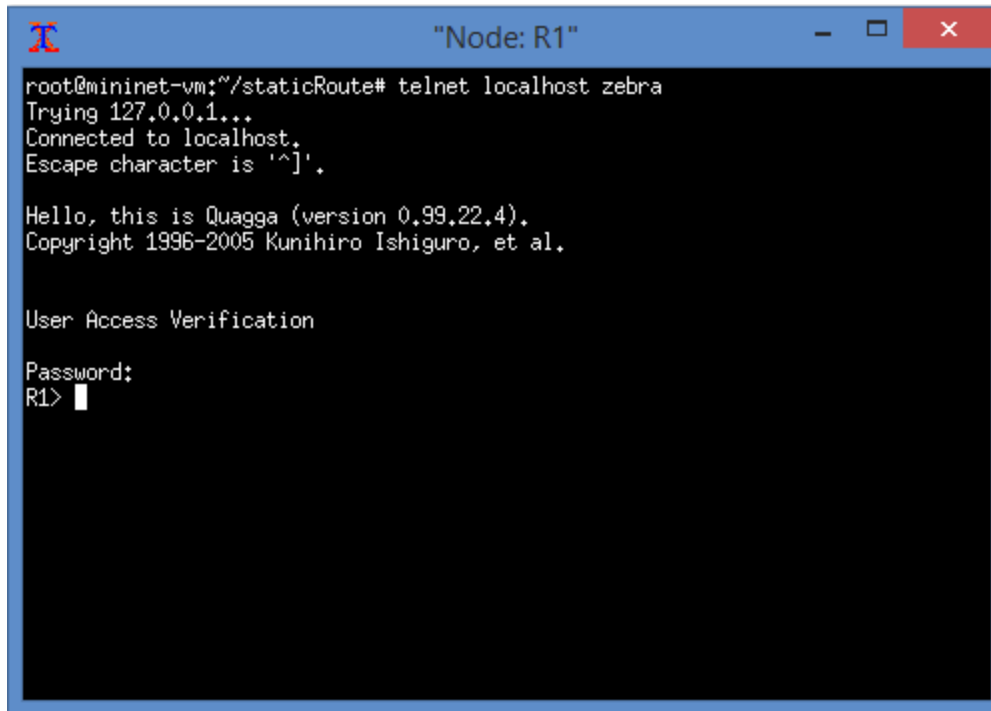
ping -I 1.1.1.1 3.3.3.3

דגל ה -I אומר לפינג שכתובת המקור תהיה 1.1.1.1.

לפני שניתן לכם לפתור את הבעיה, ניתן לכם כלים להתמודד איתה. דבר ראשון שנצטרך לעשות הוא להתחבר לdaemon של zebra. נעשה זאת, כאמור, בעזרת חיבור telnet. אם נרצה להתחבר לdaemon של R1, מתוך טרמינל של R1 נריץ את הפקודה:

telnet localhost zebra

ונקבל את הפלט הבא:



```
root@mininet-vm:~/staticRoute# telnet localhost zebra
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
R1> 
```

אם לא צוין אחרת הסיסמא לכל daemons היא 'en'.

כעת אנחנו מחוברים לdaemon. על מנת לראות את רשימת הפקודות האפשריות הקישו ?, ניתן לעשות זאת בכל שלב בהכנסת פקודה, לדוגמא הכניסו את הפקודה show ואז הקישו ?, תקבלו כפלט את כל תת הפקודות של show עם הסבר על כל אחת. בדומה ללינוקס, ניתן לבקש השלמת פקודה בעזרת tab. ניתן גם לכתוב קיצורים של פקודות (כל עוד ההשלמה יחידה) למשל הכנסת הפקודה:

sh ip rou

תושלם לפקודה:

show ip route

הריצו פקודה זו, היא בעצם מראה לנו את טבלת הניתוב של הdaemon. כרגע אנחנו נמצאים במצב View Mode שמאפשר לנו רק להציג מידע ולא לשנות הגדרות.

על מנת להשלים את המשימה, מעבר להתבוננות בטבלאות הניתוב של הראוטרים השונים, תצטרכו להוסיף נתיבים סטטיים, בכדי לעשות את זה תצטרכו לעבור לConfiguration Mode בעזרת הפקודות הבאות:

enable

configure terminal

הפקודה הראשונה מעבירה אותנו ל Enable Mode (תתבקשו להכניס סיסמא: 'en'). הפקודה השניה מעבירה אותנו ל Configuration Mode.

על מנת לחזור ל Enable Mode נשתמש בפקודה **end**.

על מנת לחזור ל View Mode נשתמש בפקודה **disable** (מתוך Enable Mode).

כשאתם ב Configuration mode על מנת להוסיף נתיב סטטי יש להשתמש בפקודה הבאה:

ip route network gateway [Command]

network is destination prefix with format of A.B.C.D/M. *gateway* is gateway for the prefix. When *gateway* is A.B.C.D format. It is taken as a IPv4 address gateway. Otherwise it is treated as an interface name. If the interface name is *null0* then zebra installs a blackhole route.

```
ip route 10.0.0.0/8 10.0.0.2
```

```
ip route 10.0.0.0/8 ppp0
```

```
ip route 10.0.0.0/8 null0
```

First example defines 10.0.0.0/8 static route with gateway 10.0.0.2. Second one defines the same prefix but with gateway to interface ppp0. The third install a blackhole route.

שימו לב שאם אתם בוחרים לציין את שם ה Interface ולא את כתובת ה IP של ה gateway אז הראוטר מניח שה network מחוברת ישירות ל Interface שצוין (האם זה מה שקורה בטופולוגיה שלנו?)

על מנת להסיר נתיב סטטי יש להשתמש בפקודה:

no ip route network gateway

עבור ה network וה gateway המתאימים.

בעצם ניתן לבטל כל הגדרה ע"י הוספת no לפני הפקודה ששימשה להגדרה עצמה.

כעת שאנחנו יודעים להתבונן בטבלאות הניתוב של הראוטר, להוסיף ולהוריד נתיבים סטטיים, עליכם להגדיר את הראוטר כך שחבילות יוכלו לעבור R1 ל R3, או במקרה שלנו שפקודת:

ping -I 1.1.1.1 3.3.3.3

שרצה על R1 תעבוד. שימו לב שאם אתם לא מבינים לאן מגיעות החבילות שלכם, ביכולתכם לפתוח wireshark שינטר כל interface בטופולוגיה.

שאלה 5: רשמו את פקודות הניתוב הסטטי שהוספתם לכל ראוטר על מנת שהחבילות יעברו כהלכה. צרפו את פלט פקודת ה ping לאחר הוספת הניתוב.

שאלה 6: מהם החסרונות העיקריים בשימוש בניתוב סטטי?

שלב 4 - ניתוב דינמי - פרוטוקול RIP

כעת, נרצה לחקור כמה מהתהליכים שמאפיינים את פרוטוקול הניתוב RIP.

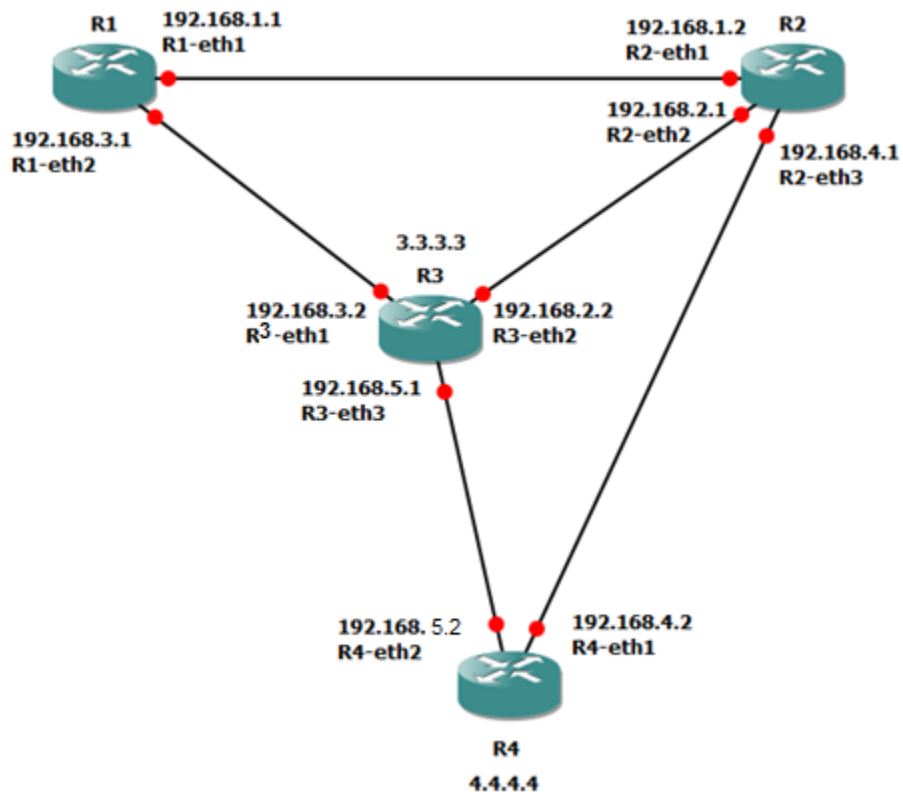
לשם כך נוריד את הקוד למעבדה זו. בתיקית הבית הריצו את הפקודה:

git clone <https://github.com/cs236341/rip.git>

כעת הכנסו לתיקיה שנוצרה, והריצו את הפקודה:

```
sudo python rip.py
```

נוצרה הטופולוגיה הבאה (וודאו זאת באמצעות הפקודה **net**):



כל הראוטרים מריצים את הdaemon zebra ומעליו daemon שמממש את פרוטוקול RIP שנקרא ripd.

המטריקה של כל Link היא 1.

בכיתה למדנו על מנגנון שנקרא Split Horizon על מנת למנוע מצב שנקרא Count to Infinity.

שאלה 7: הסבירו בקצרה מה היא בעיית ה Count to Infinity ואיך Split horizon עוזר למנוע אותה.

ישנם מנגנונים נוספים אשר עוזרים להתמודד עם הבעיה. עיינו ב [RFC 1058](#), בפרק 2.2 מוזכרים מנגנונים נוספים.

שאלה 8: הסבירו בקצרה מה הוא מנגנון ה-poisoned reverse.

שאלה 9: הסבירו בקצרה מה מנגנון ה-Triggered Updates ואיך הוא עוזר להתמודד עם בעיית ה-Count to Infinity מה החיסרון בשימוש במנגנון זה?

שאלה 10: הסבירו מהן המגבלות של פרוטוקול RIP.

שאלה 11: מהו התפקיד של שדה ה-command בחבילות שנשלחות במסגרת פרוטוקול RIP?

כעת, פתחו טרמינל ל R2, הריצו **wireshark** שמנטר את התעבורה על הממשק R2-eth1. ענו על השאלות באות:

שאלה 12: כל כמה זמן בממוצע שולח R2 הודעת response?

שאלה 13: מעל איזה פרוטוקול נשלחת הודעה זו? לאיזה פורט?

שאלה 14: לאיזה כתובת IP נשלחת הודעה זו? מה מיוחד בכתובת זו? (חפשו בגוגל)

שאלה 15: פרטו איזה מידע מכילה כל הודעה כזו (מידע רלוונטי ל-RIP)? צרפו צילום מסך.

כעת, נפיל את הלינק בין R2 ל R4 (המשיכו לנטר את התעבורה של R2-eth1), עברו ל CLI של mininet והריצו את הפקודה:

link R2 R4 down

שאלה 16: מהי המטריקה שמייצגת אינסוף (המטריקה המקסימלית)? מה זה מעיד על אורך מסלול מקסימלי ברשת המשתמשת ב RIP בהנחה שמחיר כל קשת שלם?

שאלה 17: הסבירו כיצד בא לידי ביטוי מנגנון ה-Triggered Updates.

בנוסף:

כעת נחזיר את הלינק בין R2 ל R4, ב CLI של mininet הריצו את הפקודה

link R2 R4 up

ונפעיל את מנגנון ה-poison reverse על הממשק R2-eth1, לשם כך נתחבר לdaemon של ripd על R2.

מתוך טרמינל של R2 הריצו את הפקודה:

telnet localhost ripd

עברו למצב קונפיגורציה (enable -> configure terminal) והריצו את הפקודות הבאות:

```
interface R2-eth1
ip rip split-horizon poisoned-reverse
```

סגרו את החלון, התבוננו בתעבורה על ממשק זה, וענו על השאלות הבאות:

שאלה 18: הסבירו כיצד בא לידי ביטוי מנגנון ה Poisoned Reverse ? צרפו צילום מסך של הודעת response שנשלחת ע"י R2 המעידה על כך שהמנגנון פועל.

שאלה 19: הסבירו מה ההבדל בין Poisoned Reverse ל Route Poisoning .

https://en.wikipedia.org/wiki/Route_poisoning

בהצלחה!!!!