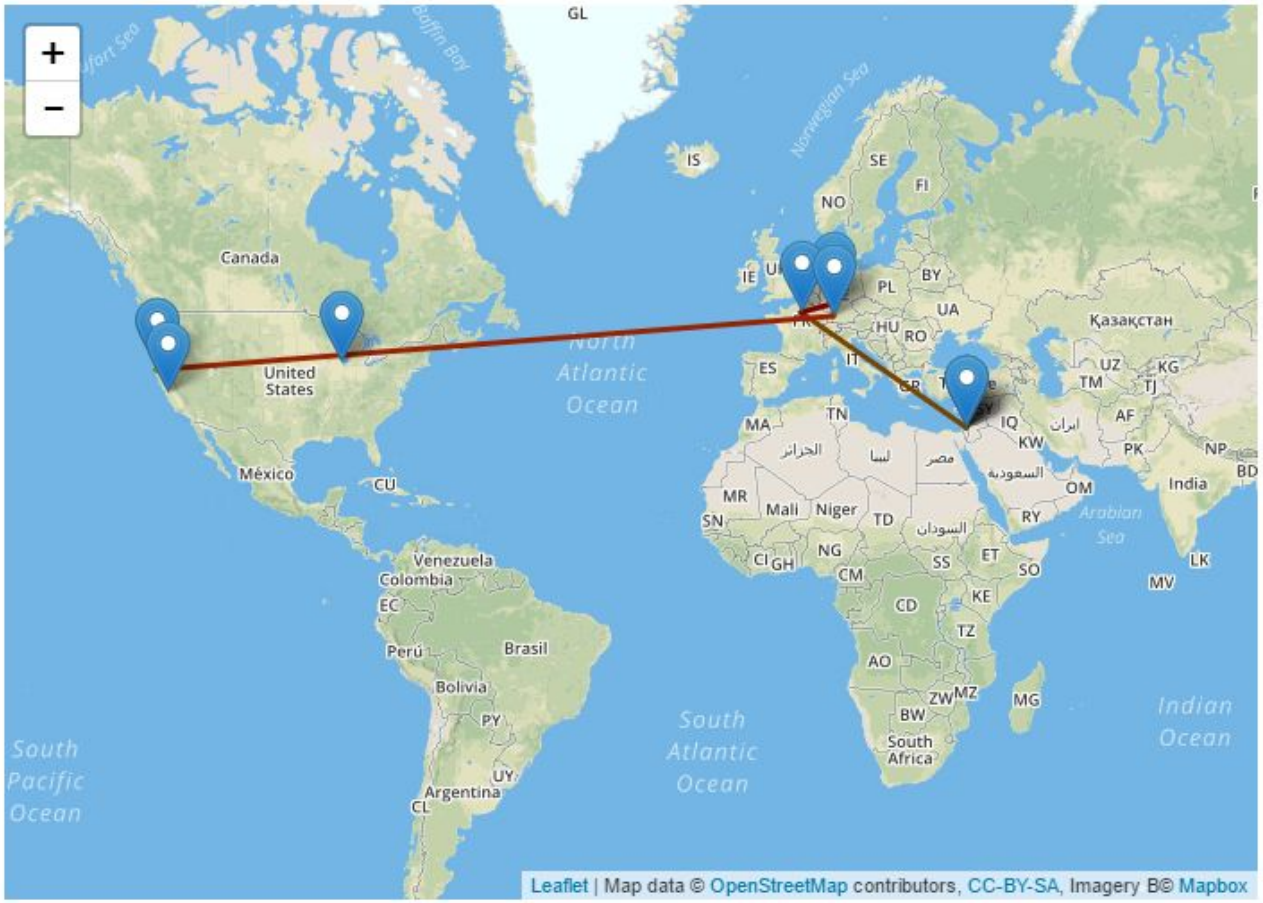
Visual Tracert



Enter an ip address or dns:

Akordi-online.com

Trace

שם מגישה: רשל סטריז'בסקי

תעודת זהות: 206457855

בית ספר: "מרכז חינוך ליאו-באק"

מקצוע: הגנת סייבר

שמות המנחים: אלון בר לב, שרית לולב

שם פרוייקט: Visual Tracert

תוכן עניינים

מבוא

הפרויקט מאפשר למשתמש למצוא דרך שבה עוברת חבילת מידע, דרך נתבי תקשורת שונים הממוקמים במקומות שונים בעולם. לכלי שתי אפשרויות עבודה, האחת, ייצוג גרפי של מסלול החבילה ברשת האינטרנט, והשנייה, הצגה טקסטואלית למסך, בעת הרצה ב-command line. הכלי בגרפי - מציג מפה, כאשר המשתמש מכניס כתובת בשדה הנתון ולוחץ על הכפתור לידו, תוך מספר שניות מוצג המסלול שעברה החבילה המבוקשת על המפה הנתונה. היתרון המשמעותי של המוצר הוא האסתטיקה – בניגוד לפקודת tracert רגילה שמציגה באופן מובנה את כתובות ה ip של הנתבים שעברה החבילה (רשימה של מידע על המסלול), במוצר V.Tracert, רואים על מפה בבירור את המסלול, מה שנותן מימד נוסף לפקודת הtracert.

פונקציונליות

המוצר מאפשר למשתמש לעקוב אחרי מסלול חבילת מידע בנוחות המקסימלית, המשתמש יכול לראות בקלות את המסלול המדויק אותו עברה.

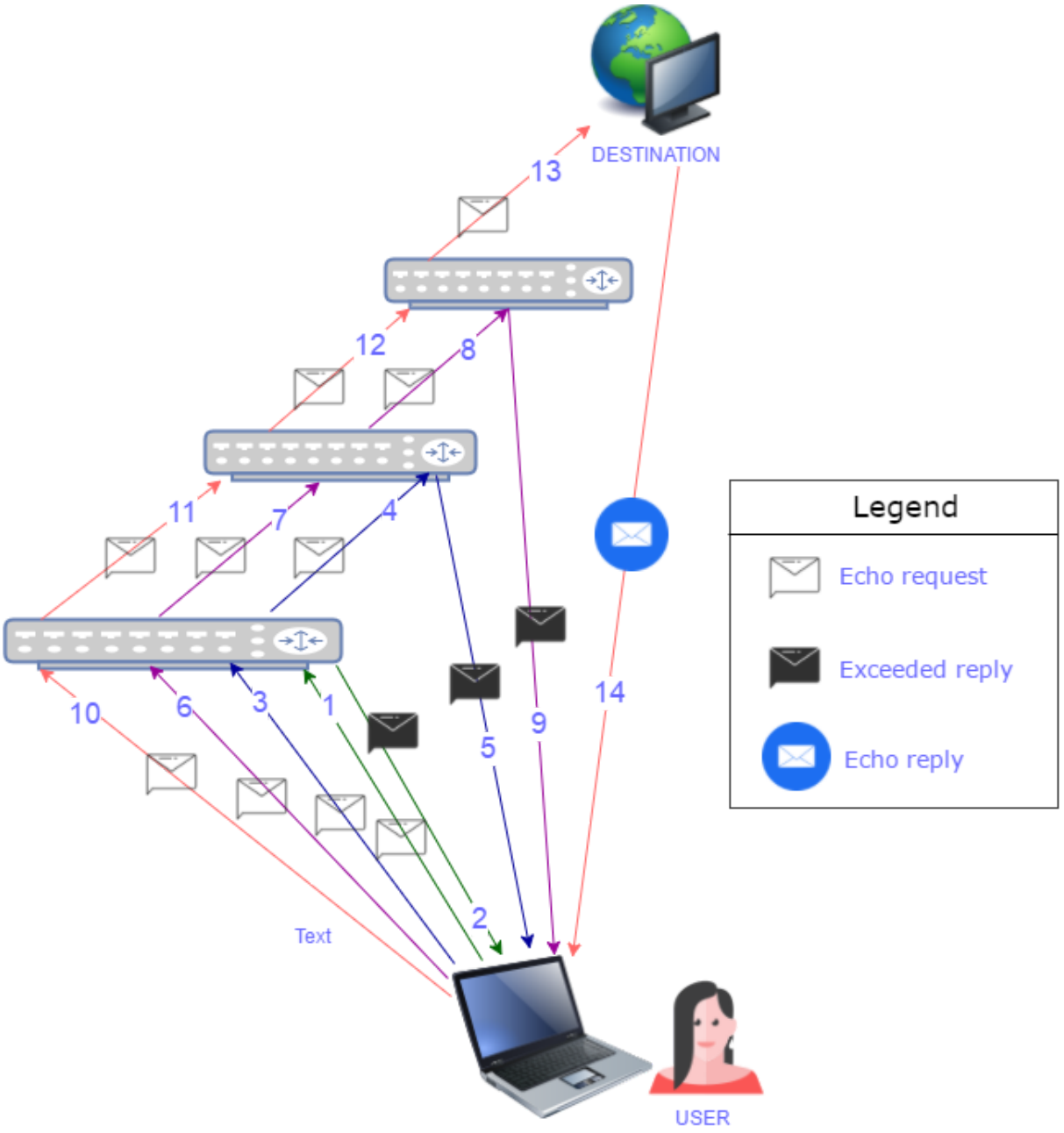
המסלול נוצר מנקודות של נתבים, דרכם עוברת החבילה, הנקודות מתחברות למסלול שלם. בנוסף למסלול שנראה, מוצגת אינפורמציה על זמני ההגעה של החבילה בין נקודה לנקודה. המוצר נכתב תוך חשיבה על איכות השירות שיקבל המשתמש, על ייצור סביבה בעלת נוחות מקסימלית ופשוטה, הן לייצור הבקשה עצמה, והן לייצוג הגרפי של הפלט למשתמש, בצורה המובנת ביותר. כל נקודה בעלת מספר סידורי, כדי לאפשר למשתמש לדעת מה סדר המסלול. נקודת הסוף, היעד, וההתחלה ניתנות לזיהוי משום שמהן יוצא רק קו אחד המחבר עם הנקודה הקודמת/ הבאה, בהתאמה.

הכלי השני, שמדפיס את המידע אודות המסלול באופן טקסטואלי, יכול לשמש ככלי לאיתור תקלות ובדיקת התקשורת בין שני רכיבים.

שני הכלים יכולים לשמש לצורכי לימוד והסברה על תקשורת על גבי שכבות שונות של מודל ה-OSI. ניתן לראות הסברים ודוגמאות בהמשך.

ארכיטקטורה

דיאגרמת ארכיטקטורה:



ICMP packets:

Stage	Description
1	Echo request - identifier, sequence number, ttl, type = 8.
2	Exceeded reply - identifier, sequence number, ttl expired, type = 11. id,sequence number are equal to request values.
3	Echo request - identifier, sequence number, ttl, type = 8.
4	Echo request - identifier, sequence number, ttl, type = 8.
5	Exceeded reply - identifier, sequence number, ttl expired, type = 11. id,sequence number are equal to request values.
6	Echo request - identifier, sequence number, ttl, type = 8.
	. . .
14	Echo reply - dentifier, sequence number, ttl, type = 0. id,sequence number are equal to request values.

הגורמים במערכת :

1. שרת http server - שרת המערכת הפועל על פרוטוקול http.
2. Geo IP - שירות שמאפשר לקבל את המיקום הגאוגרפי של נתב.
3. User Agent - תוכנית האחראית על הדפדפן המוצג למשתמש.

השירותים הניתנים במערכת:

1. קבלת כתובת ip \ שם דומיין מהמשתמש וטיפול בבקשה ע"י שרת ה http המבצע טיפול דומה להוראת tracert לגילוי הנתבים במסלול ולאחר מכן מציג אותם על פני המפה.
2. קבלת כתובת ip \ שם דומיין מהמשתמש והצגת המסלול באופן טקסטואלי בחלון ה cmd

הרקע התיאורטי

ip address

כתובת זו היא ייחודית, היא משמשת לייצוג נקודות קצה, ברשתות תקשורת שבהן משתמשים בפרוטוקול התקשורת IP, כגון רשת האינטרנט. הכתובת היא שדה מספרי באורך קבוע. לכל נקודת קצה ברשת קיימת כתובת ip, נקודה כזאת יכולה להיות למשל נתב רשת, מחשב וכו'. בעזרת הכתובת הייחודית, ניתן לבצע שליחה וקבלה מזוהה של מידע.

router

נַתָב, הוא רכיב תקשורת מחשבים שנועד לקביעת נתיבן והפצתן של חבילות נתונים ברשתות תקשורת נתונים.

ICMP

***I**nternet **C**ontrol **M**essege **P**rotocol*

פרוטוקול זה נמצא בשכבת הרשת (שכבה 3)לפי מודל OSI.

פרוטוקול שמשמש מנגנון לאבחון והתראה על שגיאות.

זהו פרוטוקול תקשורת, שהחבילות שלו נוצרות בשכבת ה-*IP*. החבילות נבנות מחבילת *IP* שעוטפת את הודעת ה-*ICMP* שנוצרה כתגובה. חבילות מסוג זה משמשות אותנו בלוגיקה המרכזית של הפרויקט ולכן יש להן מקום חשוב בו. בכל פעם נוצרת חבילת *ICMP* מסוג בקשת הד (echo request - type 8) שעוברת ברשת כתלות בערך ה-TTL שקטן בכל פעם שמגיע לנתב חדש. כשנתב מזהה TTL שערכו 0 הוא מחזיר הודעת *ICMP* מסוג exceeded reply - type 11 לשולח ההודעה ההתחלתית.

Tracert

ה-tracert נכתב על ידי ואן ג'ייקובסון בשנת 1987 - עזרו וייעצו סטיב דירינג, פיליפ וד, טים סיבר וקן אדלמן.

כותב הping, מייק מוס, טוען כי הtracert נכתב בזכות תוכנית שכתב, בה השתמש ג'ייקובסון.

שיטה למציאת המסלול בו עוברת חבילת מידע מאתר אחד לאחר, דרך נתבי תקשורת. אופן מציאת המסלול הוא שליחת רצף של הודעות מפרוטוקול *ICMP* מסוג echo request, עם ערך **T**ime **T**o **L**ive שגדל בכל שליחה ב-1. שדה ה-TTL מגדיר גבולות לחבילה ברשת. החבילה יכולה לעבור רק בכמות מוגדרת של נתבים, מה שמונע ממנה לנדוד ברשת באופן אינסופי. ערך זה נקבע בשדה ה-TTL.

echo request היא חבילת בקשה מפרוטוקול *ICMP* מסוג 8. בקשת כזו נקראת גם ping. חבילה כזאת משמשת לבדיקת התקשורת בין רכיבים ברשת.

ערך ה-TTL קטן, כל נתב מפחית אותו ב-1. כאשר נתב מקבל חבילה בה ערך זה הוא 0, הוא שולח הודעת exceeded reply אל השולח ההתחלתי. למשל, אם הערך ההתחלתי הוא 1, תוחזר הכתובת של הנתב הראשון במסלול, אם 2, השני וכן הלאה. כך שולח ההודעה, במקרה הזה המחשב ממנו מבוצעת פעולת הטרייס יודע להבדיל חבילה כזו מכל חבילה אחרת. שליחת ההודעות מתבצעת עד שהחבילה מגיעה לכתובת היעד המבוקשת על ידי המשתמש - מזוהה על ידי חבילת echo reply. אם נשלח מספר מוגדר של בקשות (לרוב 30) ולא התקבלה חבילה המעידה על סיום הפעולה, הפעולה מסתיימת. הפלט שנוצר הוא כתובות הנתבים המייצגים את המסלול שעברה החבילה, וזמני ההגעה של כל חבילה.

לפניכם דוגמה לפלט של הרצת tracert לכתובת www.ynet.co.il .

```
  1    46 ms    <1 ms    <1 ms  10.0.0.138
  2    10 ms    10 ms    10 ms  bzq-179-37-1.cust.bezeqint.net [212.179.37.1]
  3    18 ms    12 ms    12 ms  10.250.0.37
  4    15 ms    38 ms    11 ms  bzq-25-77-14.cust.bezeqint.net [212.25.77.14]
  5    10 ms    11 ms    11 ms  bzq-219-189-221.cablep.bezeqint.net [62.219.189
221]
  6    10 ms    10 ms     9 ms  2.20.154.121

Trace complete.
```

Ethernet

טכנולוגיה לתקשורת ברשתות מחשב מקומיות (LAN). לפי OSI (מודל המראה אופן העברת נתונים ברשת תקשורת, פעולות ואת הסדר שלהן), הethernet ממוקם בשכבת הקו שהיא השכבה השנייה.

IP

***I**nternet **P**rotocol*

פרוטוקול זה נמצא בשכבת הרשת (שכבה 3)לפי מודל OSI.

זהו אחד מהפרוטוקולים הנפוצים ביותר, בשל המהירות והיעילות שלו. אי אפשר להתעלם מהעובדה שבפרוטוקול זה אינה נבדקת ההגעה של חבילה לצד השני ואין אימות נתונים בין החבילות.

בפרוטוקול הנ"ל שני צידי התקשורת מזוהים על ידי כתובות ip.

HTTP

***H**yper**T**ext **T**ransfer **P**rotocol*

פרוטוקול זה נמצא בשכבת היישום (שכבה 7) לפי מודל OSI. זו היא השכבה הגבוהה ביותר.

זהו פרוטוקול שנועד להעביר דפי html ואובייקטים שהם מכילים (כמו למשל תמונות, קבצים, סרטונים וכו').

שרתים הבנויים על פרוטוקול זה הם הנפוצים ביותר ברשת והלקוחות הם בדרך כלל דפדפני אינטרנט.

התקשורת בין השרת ללקוח מתחילה בחיבור של צד הלקוח לכתובת ה-ip וה-port של השרת. הלקוח שולח בקשות והשרת מחזיר תשובות וזהו אופן התקשורת בפרוטוקול הנ"ל.

Python

פייטון היא שפת תכנות שהומצאה תוך מחשבה על פשטות הקוד ונוחות המשתמש. זוהי שפה דינמית ונפוצה מאוד. שפה זו מאפשרת תכנות מונחה עצמים.
בפרויקט, פייטון משמש כשפה המרכזית לכתיבת קוד התכנית - כל הפרויקט מבוסס על שפה זו, למעט ה-user interface. הנוחות של השפה מתבטאת בקלות של יצירת משתנים ושימוש בהם, כאשר אין צורך להגדרה ראשונית של משתנה הכוללת את סוגו ועוד, הזחה לשם הגדרת בלוקים לעומת שפות אחרות (בהן השימוש נעשה בסוגריים או הגדרות אחרות).

JavaScript

שפת תכנות המותאמת לשילוב באתרי אינטרנט. למעשה, רוב אתרי האינטרנט כיום משתמשים בשפה זו. ידועה בקשרה לתגיות ה-*html*.

מיושמת לעתים קרובות על מנת ליצור תכניות שיבוצעו בדפדפני אינטרנט.

גם בפרויקט, השפה משמשת להרצה ושליטה על דפדפן. היא שימשה עבור כתיבת הלקוח - user agent

שמקשר בין המשתמש לתכנית - שולח בקשה לכתובת חדשה, מקבל תשובה ומציג אותה על מפה אינטראקטיבית למשתמש.

html

***H**yper**T**ext **M**arkup **L**anguage*

שפת התגיות המרכזית בעולם, משמשת לתצוגה ועיצוב דפי אינטרנט ותוכן, לתצוגה בדפדפן.
בפרויקט V.Tracert מרכיבה בשילוב עם שפת *javascript* את ה-user agent האחרי על ייצוג הדפדפן המוצג למשתמש.

xml

*e**X**tensible **M**arkup **L**anguage*

אין היא שפת תכנות , אלא תקן לייצוג נתונים באופן נוח וטקסטואלי. השימוש בו נעשה באמצעות תגיות בדומה ל-*html*. שימוש ב-*xml* מקל החלפת נתונים בין מערכות שונות.
בפרויקט התקן הנ"ל ממומש ב-user agent יחד עם *DOM* (ראה בהמשך).

DOM

***D**ocument **O**bject **M**odel*

ממשק ל-*XML* ו-*HTML* המגדיר גישה למסמך מסוג כזה או אחר, בהתאמה. חשוב להבין כי כל דפדפן אינטרנט הוא גם מסמך שכזה, וה-*DOM* מאפשר לנו לעדכן את הדפדפן בהתאם לכל שינוי שקורה ברשת. במקרה שלנו, השימוש נעשה ב-*XML DOM* על מנת לחלץ מידע (למשל: הכתובת המוזנת על ידי המשתמש לשדה הנתון), עדכון המפה בהתאם לתוצאת ה - *tracert* וכו'.

GIS

***G**eographic **I**nformation **S**ystem*

מערכת מידע המאפשרת ניתוח וייצוג של מידע גיאוגרפי על מפה. על פי נתונים מסויימים ונתונים גיאוגרפים, נעשה ניתוח של אינפורמציה ולכן זהו כלי שימושי מאוד למשל לצרכים צבאיים, או אפילו לשם תכנון ערים, טבע ונוף ועוד.

בפרויקט אנו משתמשים בספריית Leaflet (הסבר בהמשך) ומייצגים נתונים אודות נתבים שונים על מפה.

Leaflet

זוהי ספריה בשפת javascript המאפשרת שימוש במפה. בעזרתה ניתן לייבא מפה אל הדפדפן ולהשתמש באופציות השונות שהיא מציעה כמו למשל סימון נקודות וציור קווים בים נקודות.

Geo ip

שימוש במערכת זו נעשה בעת ייצוג המיקומים הגיאוגרפים המרכיבים את המסלול המבוקש. בהינתן כתובת ip האתר מציג מידע אודות הנתב. המידע יכול להיות מיוצג במספר דרכים שונות כגון newline separated, serialized PHP, CSV, JSON,XML. בפרויקט נשתמש ב XML. לכן נקרא לכתובת האתר עם סיומת "\xml".

קיימות שני אפשרויות של תשובה כזאת:

1. status = fail : הנתב לא החזיר תשובה תקינה, לכן גם לא חזר מידע.
   מקרה כזה בהחלט יכול לקרות משום שחלק מהנתבים אינם יחזירו את כתובתם (ראה בטבלה מטה).

Error messages

value	description
private range	the IP address is part of a private range - more info
reserved range	the IP address is part of a reserved range - more info
invalid query	invalid IP address or domain name
quota	over quota

2. status = success : חזרה הודעה תקינה עם מידע על הנתב, למשל קורדינטות המיקום, עיר, ארץ וכו'

מהודעת התשובה אני מחלצת את הקורדינטות גאוגרפיות של מיקום השרת הנתון בעולם.

מצורף קישור לשירות הweb :

http://ip-api.com/xml/

gcap

python class שנכתב ע"י alonbl . באמצעותו ניתן לבצע האזנה לפקטות (הסנפה) וייצור של פקטות (packets) בשכבת *Ethernet*.

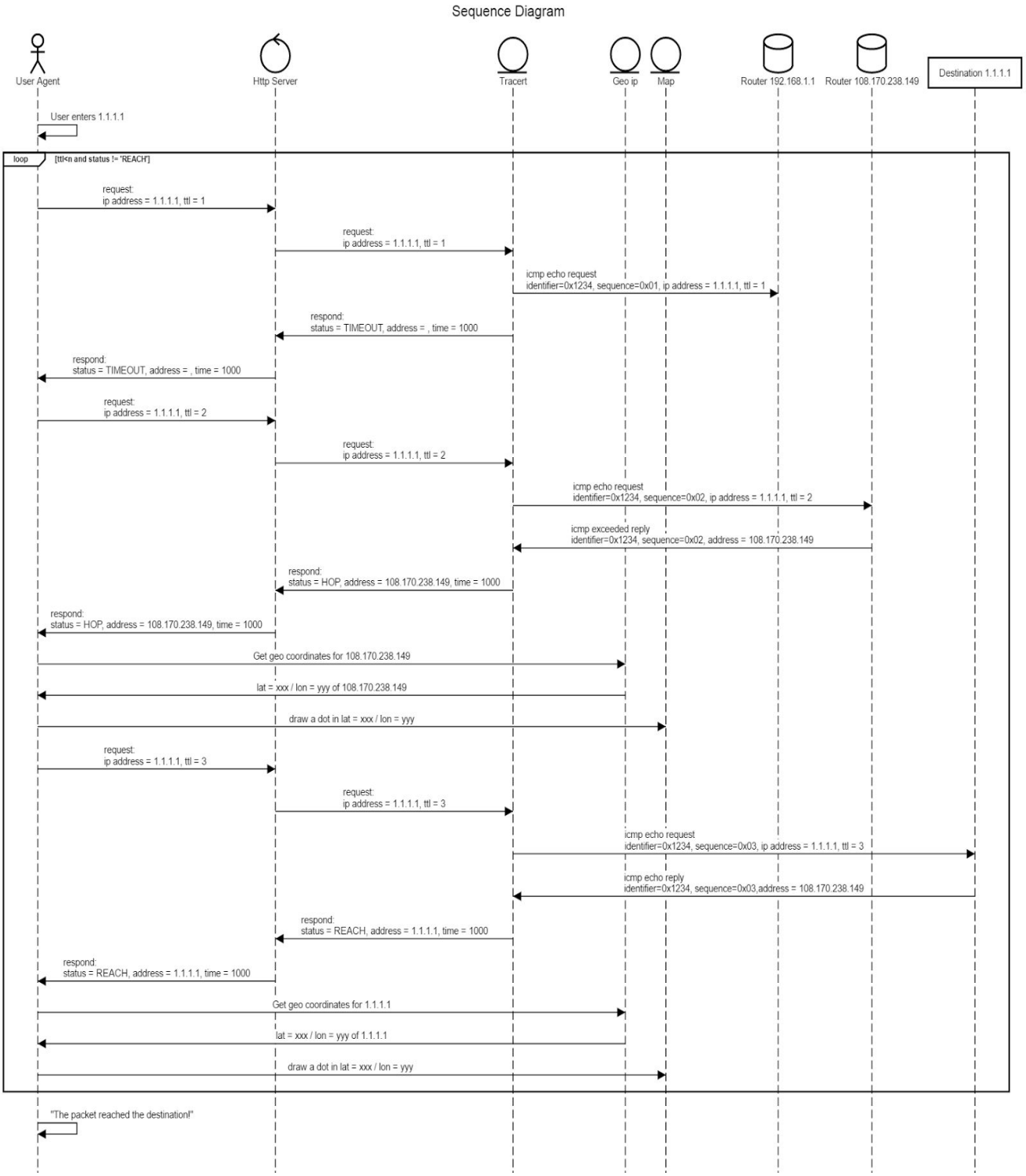
מצורף קישור:

https://github.com/alonbl/network-course/tree/master/gcap

מימוש

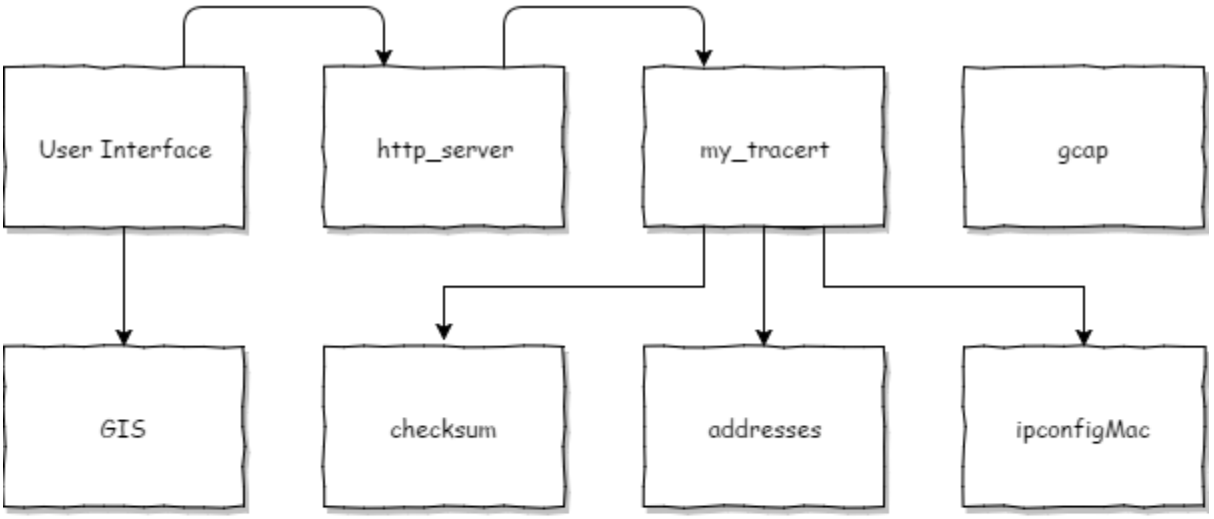
דיאגרמת רצף

sequence diagram מציגה את רצף האירועים העיקריים במערכת.



דיאגרמת בלוקים

הדיאגרמה מציגה את כל המודולים הקיימים במערכת ואת הקשרים ביניהם.



User Interface: an html file, which presents the web browser on the screen.

http_server: an http protocol server.

my_tracert: a module that traces for one hop, with a given destination and ttl.

checksum: a module contains a function that calculates checksum.

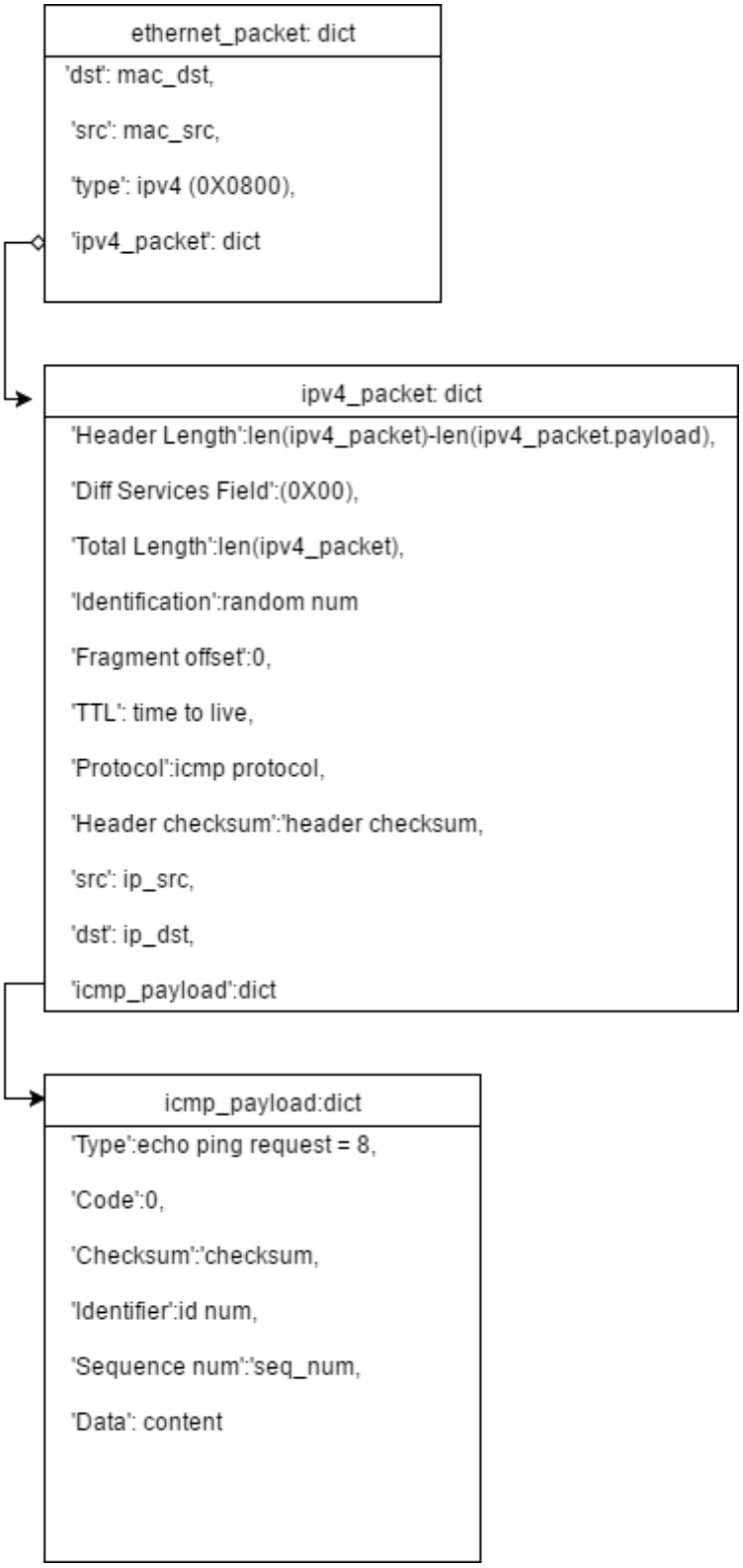
addresses: a method to obtain the ip addresses.

ipconfigMac: a method to obtain the local mac.

gcap: a python class wrote by alonbl. Enables catching and sending packets on the ethernet level.

GIS: Geo ip service I use in order to get the geo coordinates.

דיאגרמת מבני נתונים



User Agent-Http Server Protocol

Description

The user agent requests the server to trace one hop. The server uses the request to call the tracert function, gets the values from that function and sends an xml response to the user agent.

Request

```
/trace?ip_or_dns=ADDRESS&ttl=TTL
```

TTL: 'Time To Live' - the maximum amount of hops the packet can live through
ADDRESS: the destination address we trace for.

Response

There are 3 kindes of responds:
HOP: the tracert got an exceeded packet - means that we have a ip address of a hop.
REACH: the tracert got an echo reply - means that we reached the destination.
NONE: the tracert didn't get any related response for 3 tries.

```
<root>
      <result status="HOP"  time=TIME  ipAddr=HOP_ADDRESS/>
</root>
```

```
<root>
      <result status="REACH"  time=TIME  ipAddr=HOP_ADDRESS/>
</root>
```

```
<root>
      <result status="TIMEOUT"  time=TIME  ipAddr=''/>
</root>
```

HOP_ADDRESS: the hop address the tracert reached.

TIME: the time passed while tracing.

User Agent-Geo IP Protocol

Description

The user agent requests the coordinates of the ip address he got from the tracert.

Request

http://ip-api.com/xml/HOP_ADDRESS

HOP_ADDRESS: this is the ip address we got from the tracert.

Response

```
<query>
  <lat>
    <![CDATA[ LAT ]]>
  </lat>
  <lon>
    <![CDATA[ LON ]]>
  </lon>
</query>
```

LAT: a float number - **Latitude** is a geographic coordinate that specifies the north-south position of a point on the Earth's surface.

LON: a float number - **Longitude** is a geographic coordinate that specifies the east-west position of a point on the Earth's surface.

Example #1:

Request

http://ip-api.com/xml/8.8.8.8

Response

```
<query>
  <status>
```

```
  <![CDATA[ success ]]>
</status>
<country>
  <![CDATA[ United States ]]>
</country>
<countryCode>
  <![CDATA[ US ]]>
</countryCode>
<region>
  <![CDATA[ CA ]]>
</region>
<regionName>
  <![CDATA[ California ]]>
</regionName>
<city>
  <![CDATA[ Mountain View ]]>
</city>
<zip>
  <![CDATA[ 94035 ]]>
</zip>
<lat>
  <![CDATA[ 37.386 ]]>
</lat>
<lon>
  <![CDATA[ -122.0838 ]]>
</lon>
<timezone>
  <![CDATA[ America/Los_Angeles ]]>
</timezone>
<isp>
  <![CDATA[ Level 3 Communications ]]>
</isp>
<org>
  <![CDATA[ Google ]]>
</org>
<as>
  <![CDATA[ AS15169 Google Inc. ]]>
</as>
```

```
  <query>
    <![CDATA[ 8.8.8.8 ]]>
  </query>
</query>
```

Example #2:

Request

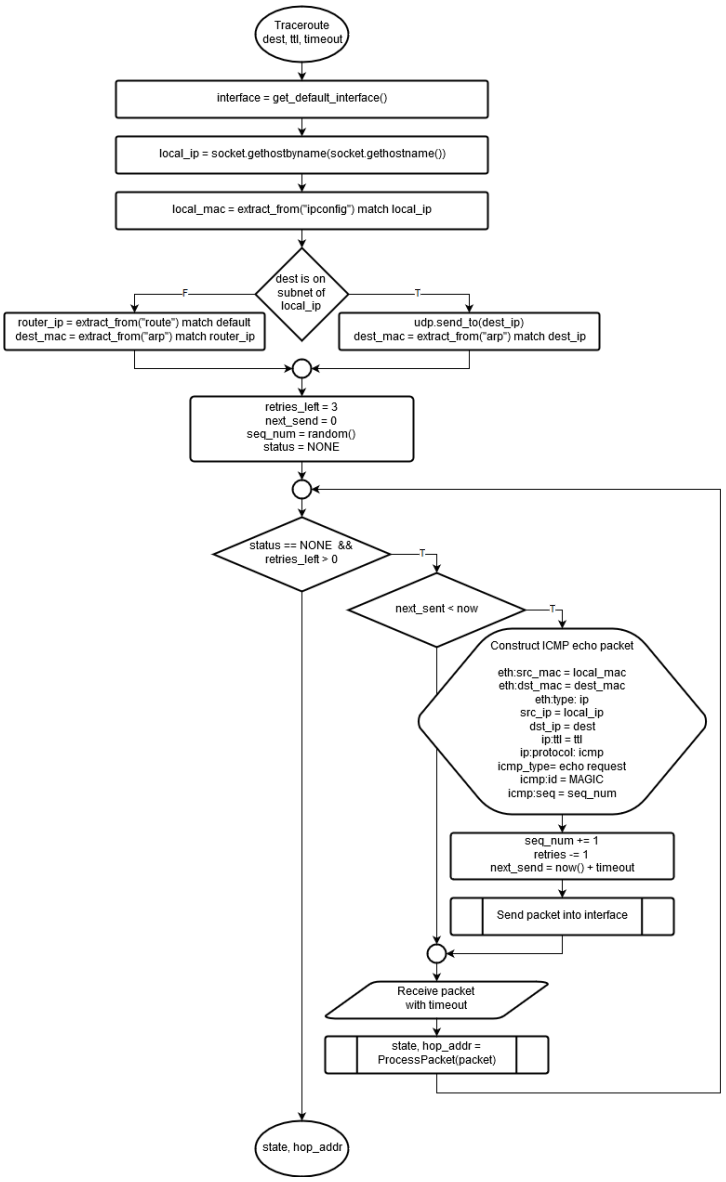
http://ip-api.com/xml/192.168.1.1

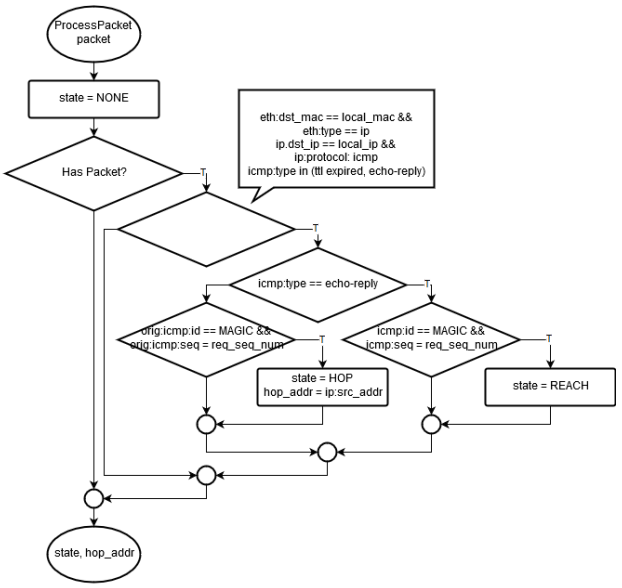
Response

```
<query>
  <status>
    <![CDATA[ fail ]]>
  </status>
  <message>
    <![CDATA[ private range ]]>
  </message>
  <query>
    <![CDATA[ 192.168.1.1 ]]>
  </query>
</query>
```

תרשים זרימה - לוגיקה של tracert

תרשים זרימה להסבר אופן מימוש ה tracert .





אתגרים במימוש ודרכי פתרון

האתגרים העיקריים בהם נתקלתי במהלך המימוש הוא למידה של שפה חדשה javascript, למידה על xml ו - html וכתיבת קוד משולב בין כל אלה. זה בהחלט היה מאתגר ללמוד וישר לממש בהתאם לצורך התכנית את הידע שיש לי בנושא. בנוסף, בשפת javascript אינם מיוצגים errors, בניגוד לpython למשל. כאשר היה ארור מסויים בתוכנית ה javascript, גם אם היה syntax error, התוכנית פשוט לא רצה כמעט או לא רצה בכלל. לגלות את הסיבה למצב כזה, היה מאתגר בכל פעם מחדש כי אין אפילו רמז מקום הבעיה. כאשר תוכנית רצה אך מפסיקה בשלב מסויים אפשר להיעזר בלוגיקה ולבדוק היכן נעצרה התכנית ולנסות לפתור את הבעיה, אך כאשר התכנית פשוט לא מגיבה, הרבה יותר מורכב למצוא את הטעות.

בעיות ידועות

קרו מספר מקרים בהם נראה כי מספר הנקודות האחרונות על המפה (ממקרים שנראו - **2**), מיוצגות אומנם כנקודות, אך אינן מחוברות בקו ביניהן. ייתכן כי הבעיה נמצאת בפונקציה שמציירת את הקו או בפונקציה שקוראת לה.

התקנה ותפעול

דרישות ואילוצים:

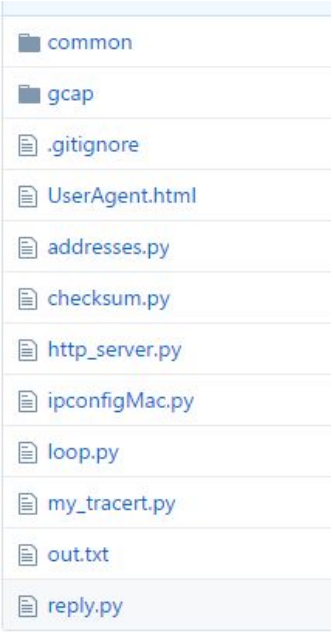
המערכת דורשת חיבור לאינטרנט

Visual Tracert

שלב 1:

הורד את התיקייה מהקישור הבא:

https://github.com/rashel1410/vtracert



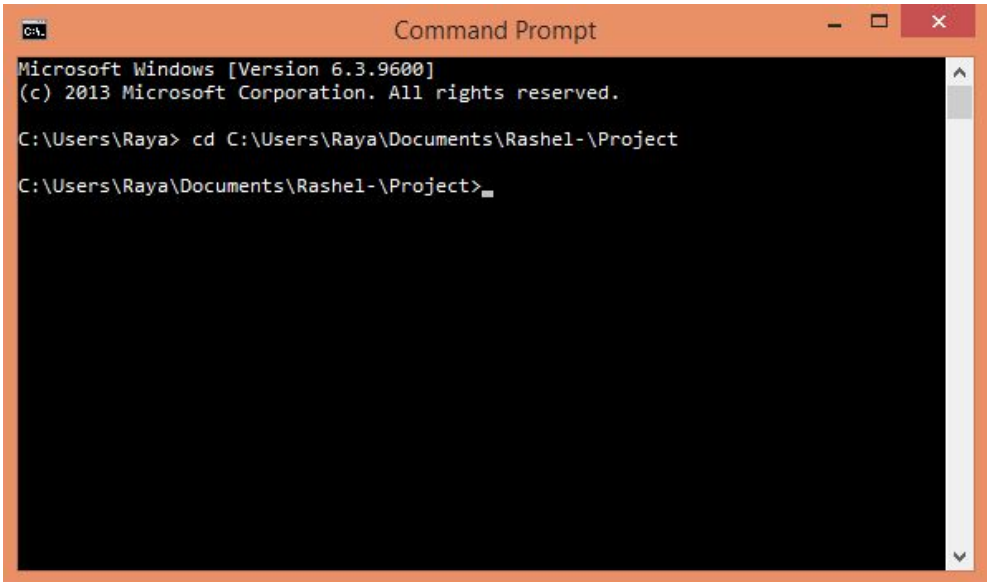
שלב 2:

פתח את התיקייה והעתק את מיקומה במחשב.

שלב 3:

פתח command line (cmd) - חפשו במחשב שלכם, זוהי תוכנה מובנית.

הקלידו ' cd' ולאחר מכן הדביקו את השורה שהעתקתם בשלב 2.



שלב 4:

הריצו את קובץ ה - http_server.py (פשוט כתבו את שם הקובץ, כמו שהוא)

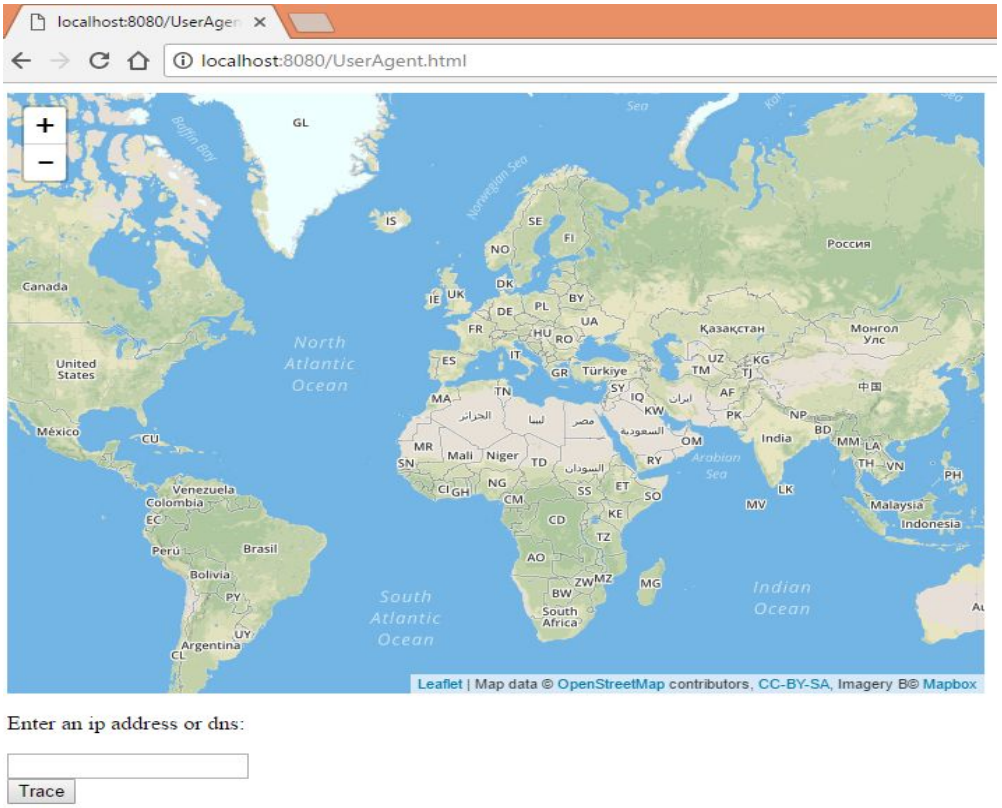
השרת רץ!

שלב 5:

כעת פתחו את דפדפן האינטרנט וכתבו בשורת הכתובת את השורה הבאה:

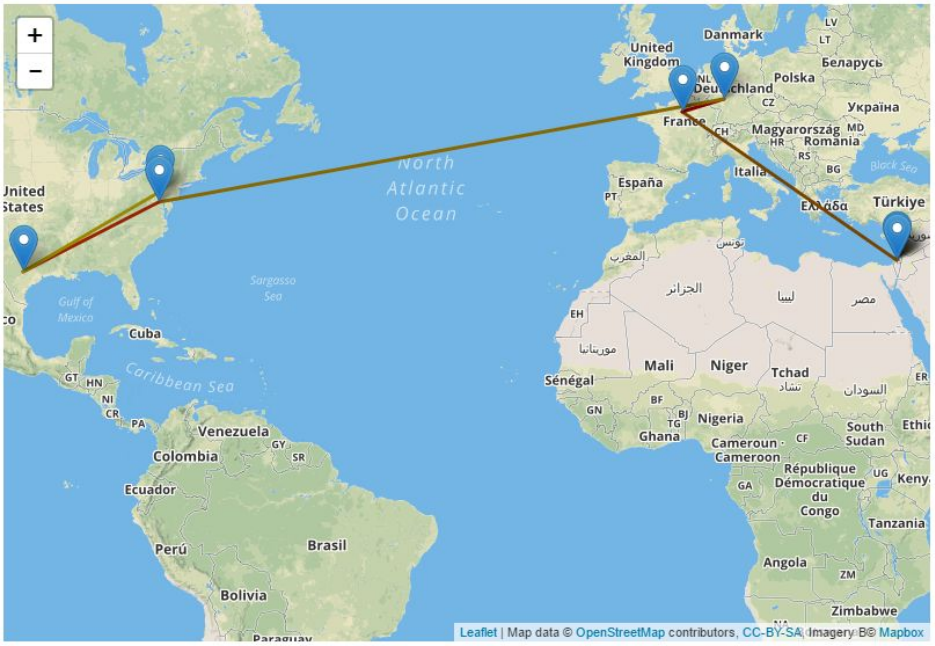
http://localhost:8080/UserAgent.html

תוצג בפניכם מפה ושדה קלט:



הכניסו כתובת ip או שם של אתר ולחצו על הכפתור כמובן :)

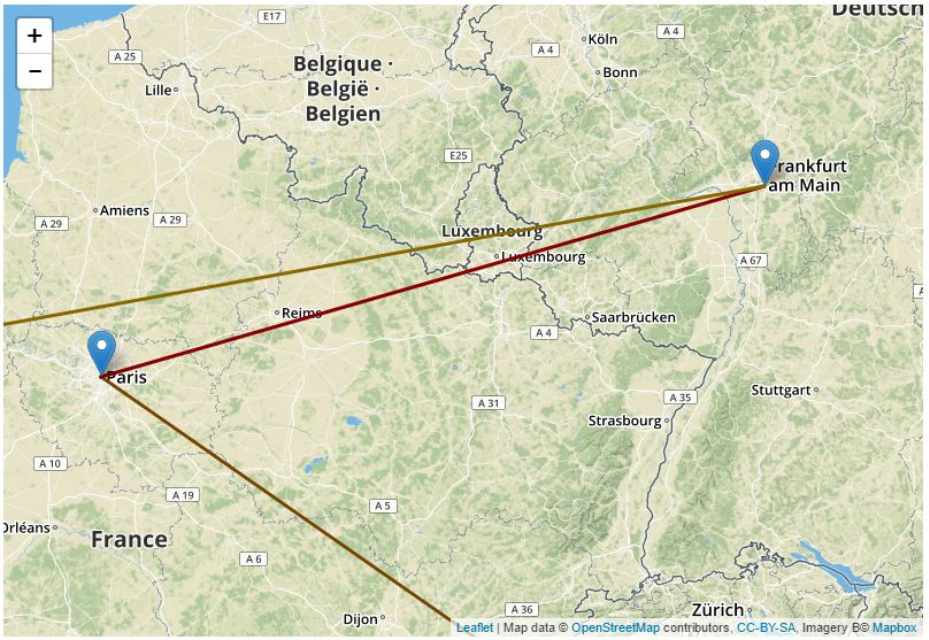
עכשיו נשאר רק לחכות מספר שניות והמסלול יתחיל להופיע! :-)



Enter an ip address or dns:

ruce.org

Trace



Enter an ip address or dns:

ruce.org

Trace

Textual Tracert

ראה שלבים 1-3 מעלה, תחת כותרת Visual Tracert.

שלב 4:

הרץ את הקובץ TextTracert.py בצורה הבאה:

```
C:\Users\Raya\Documents\Rashel-\Project>TextTracert.py --address 8.8.8.8
```

וחכה לתוצאות...

```
1       10.0.0.138
2       212.179.37.1
3       10.250.0.37
4       212.25.77.14
5       62.219.189.73
6       212.179.72.241
7       72.14.219.61
8       108.170.246.225
9       209.85.245.145
10      8.8.8.8
Trace complete. The packet reached the destination!
```

תוכניות עתיד

- server א-סינכרוני שיוכל לטפל במספר בקשות במקביל.
- הרצת ה-tracert מ-proccess אחר על מנת לאפשר עבודה מקבילית מבלי להפריע לריצת האפליקציה.
- קווי מסלול בעובי שונה/ צבעים שונים על פי קצב המעבר בין נתב לנתב, על מנת להשתמש בכל היכולות הגרפיות לטובת נוחות המשתמש.
- חוויה מעניינת בזמן הפעולה - קירוב וריחוק של המפה אל הנקודה החדשה.
- ביצוע tracert קבוע באופן מחזורי בכדי לעדכן במפה כל הזמן את המסלול עם הצבעים, כך שניתן לראות מתי פתאום מסלול משתנה או מהירות משתנה.

- ייצוג סטטוס התוצאות בטקסט.
- אפשרות להכניס מספר כתובות, כך שניתן יהיה להציג מספר מסלולים בו זמנית.
- אפשרות למחוק/ לא למחוק מסלולים לפני ביצוע tracert חדש.

פרק אישי

כתיבת הפרוייקט הייתה חוויה מאתגרת ומלמדת עבורי. הקשיים והאתגרים העיקריים (כפי שציינתי קודם) היו לימוד עצמי של נושאים חדשים לגמרי, ומציאת בעיות בשפת הjavascript שאינה מציגה errors. כמובן שחשוב לציין כי היו גם קשיים שונים במהלך המימוש של הלוגיקה הכללית של הtracert, מעקב בלתי פוסק אחרי פאקטים ב-wireshark, שלבסוף הפך לחלק בלתי נפרד מהבנת הפרוייקט ואופן המימוש עצמו. העובדה כי הפרוייקט כולל חלק גרפי ואסתטי ושניתן לבדוק הרצות בעזרת wireshark ולראות את מעבר הפאקטים גרם להנאה והתלהבות הרבה פעמים משום שרואים את התוצרים מול העניים. בנוסף, העובדה שהשתמשתי בכלים החדשים שלמדתי בדפדפן האינטרנט למשל, גרמו לי להבין איך הפעולות הפשוטות של חיי היומיום עובדות, פתאום הבנתי מה קורה כשאני מחפשת דבר פשוט באינטרנט, מה אומרים הרבה מהדברים שרשומים בשורת הכתובת לאחר חיפוש שכזה, והתחלתי לשים לב לפרטים אלה.
רכשתי מיומנויות כמו שיטות לחיפוש טעויות בקוד, חיפוש יעיל של מידע באינטרנט וכתיבת תיק רשמי עבור פרוייקט שכזה.
זו הייתה חוויה מלמדת ללא ספק, הודות הלימוד העצמי והודות עזרת המורים שליוו אותי לאורך כל הדרך.

תיעוד קוד

ב - github קישור בפרק הקוד

קוד פרויקט

https://github.com/rashel1410/vtracert/releases