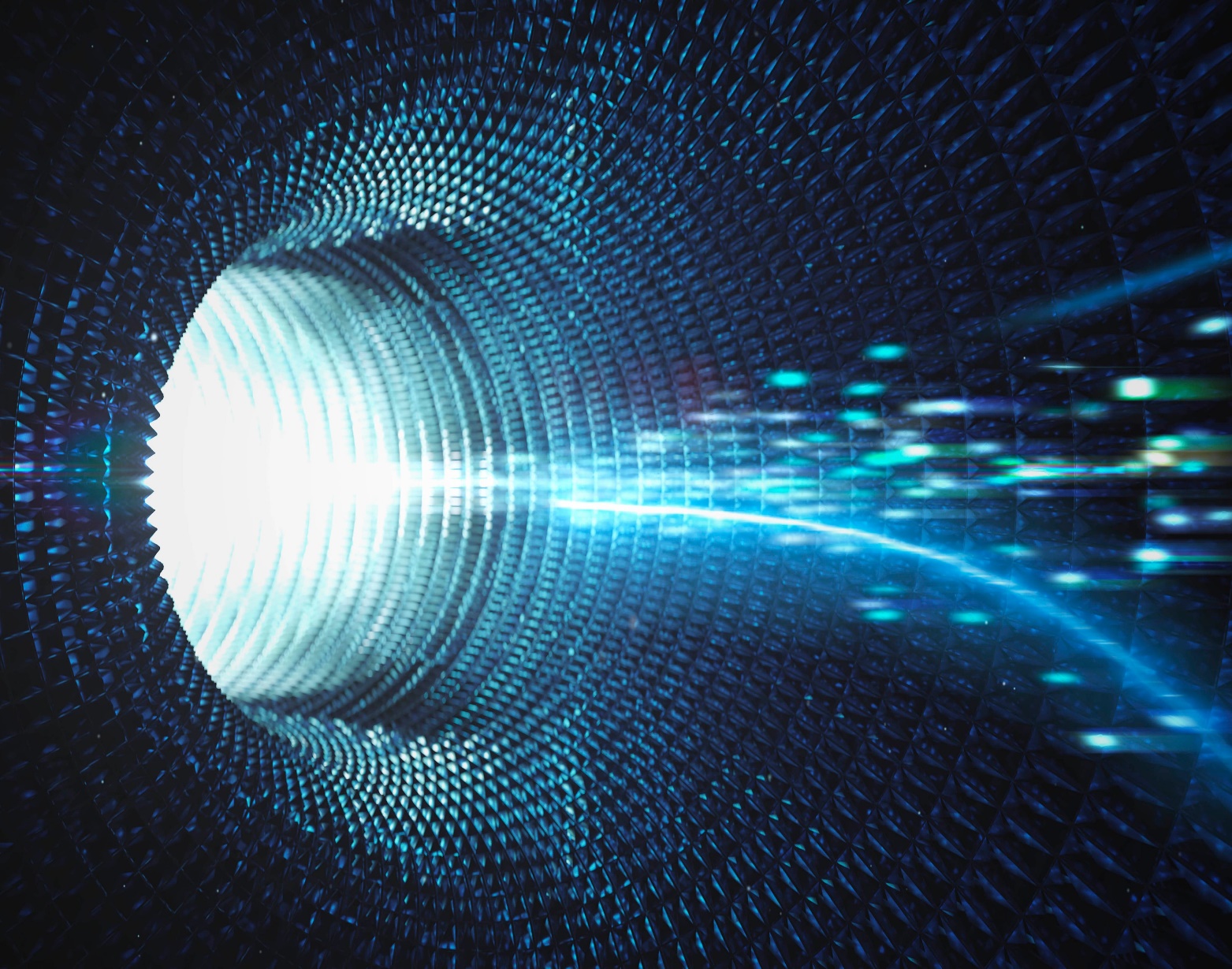
פרויקט גמר – TCP Over ICMP



סדנה בתקשורת מחשבים

מגיש: צח בקסיס - 206374951

מנחה: דני כלפון

רקע

מימוש Tunnel להעברת תקשורת TCP על גבי ICMP.

ה-Tunnel יכול להיות שימושי בסביבה בה תקשורת TCP חסומה או בתשלום, אך תקשורת ICMP (שמשמש בין היתר ל-ping) פתוחה.

הפתרון שלי

הפתרון שלי מורכב משני חלקים:

* הלקוח:
  + מאזין לבקשות TCP שאותן הוא רוצה לטנל (מלשון Tunnel), עוטף בקשות אלה ב-ICMP ושולח לשרת (באמצעות echo request).
  + מקבל פקטות ICMP מהשרת, מפרסר ומעביר חזרה ללקוח TCP.
* השרת:
  + מאזין לפקטות ICMP שמגיעות מהלקוח, עוטף אותן ב-TCP ושולח ל-Target.
  + מקבל פקטות TCP מה-Target, עוטף אותן ב-ICMP ושולח ללקוח (echo reply).

שפת תכנות – Python

בחרתי לכתוב את הפרויקט בשפת Python בשל יתרונותיה הרבים:

* עבודה עם raw sockets
  + בניית פקטות TCP, ICMP
  + מודל שרת-לקוח בסיסי
* גמישות ופשטות של השפה
* קהילת מפתחים גדולה

סביבת פיתוח – Pycharm

מויקיפדיה:

*PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python programming language. It is developed by the Czech company JetBrains (formerly known as IntelliJ).[5] It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.[6]*

*PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License,[7] and there is also Professional Edition with extra features – released under a subscription-funded proprietary license and also an educational version.*

הסביבה נוחה מאוד אם כי כבדה מעט.

תשתית – Docker

מויקיפדיה:

*Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.*[*[6]*](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-SYS-CON_Media-6)*The service has both free and premium tiers. The software that hosts the containers is called Docker Engine.*[*[7]*](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-what-is-a-container-7)*It was first started in 2013 and is developed by Docker, Inc.*[*[8]*](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-os4u-8)

על מנת לסמלץ את השרת והלקוח על אותה מכונה, השתמשתי ב-Docker.

כדי לבנות את ה-Image מהקוד, השתמשתי ב-Dockerfile בסיסי שמעתיק את הקבצים מה-Working Directory אל ה-Target Image.

כדי להריץ את ה-Serviceים, השרת והלקוח, השתמשתי ב-docker-compose.yml.

תכנון הפרויקט

המשתמש יהיה יכול:

* לבחור Host שאותו הוא רוצה לטנל. לדוגמא: ynet.co.il.
* לבחור את פורט הטינול.
* לבחור פורט לוקלי שממנו הוא רוצה להאזין לבקשות TCP מהלקוח.
* לפנות ממספר לקוחות לאותו שרת כדי לקבל את שירות הטינול (לא במקביל).
* לפנות מספר פעמים מאותו לקוח לשרת, מבלי לסגור ולהדליק מחדש את השרת.

מודל תקשורת

כאמור, התקשורת כולה בנויה סביב raw sockets ב-Python.

מויקיפדיה:

*A network socket is a software structure within a*[*network node*](https://en.wikipedia.org/wiki/Node_(networking))*of a*[*computer network*](https://en.wikipedia.org/wiki/Computer_network)*that serves as an endpoint for sending and receiving data across the network. The structure and properties of a socket are defined by an*[*application programming interface*](https://en.wikipedia.org/wiki/Application_programming_interface)*(API) for the networking architecture. Sockets are created only during the lifetime of a*[*process*](https://en.wikipedia.org/wiki/Process_(computing))*of an application running in the node.*

*Because of the standardization of the*[*TCP/IP*](https://en.wikipedia.org/wiki/TCP/IP)*protocols in the development of the*[*Internet*](https://en.wikipedia.org/wiki/Internet)*, the term network socket is most commonly used in the context of the Internet protocol suite, and is therefore often also referred to as Internet socket. In this context, a socket is externally identified to other hosts by its socket address, which is the triad of*[*transport protocol*](https://en.wikipedia.org/wiki/Transport_protocol)*,*[*IP address*](https://en.wikipedia.org/wiki/IP_address)*, and*[*port number*](https://en.wikipedia.org/wiki/Port_number)*.*

בפייתון השימוש ב-raw sockets נוח מאוד. כדי לעזור עם קריאות הקוד, יצרתי קובץ בשם sockets.py שמכיל את פונקציות העזר ל-socketים.

Graphical user interface, text, application

Description automatically generated

מבנה הקוד

הקוד מחולק לארבעה חלקים עיקריים:

* קובץ client.py. הקובץ מכיל את הלוגיקה העיקרית של הלקוח. פרסור הפרמטרים הקונפיגורבליים, יצירת ה"שרת" הלוקלי שמאזין לבקשות TCP נכנסות מהמשתמש, עטיפת הבקשות ב-ICMP ושליחתן לשרת, קבלת הודעות ICMP מהשרת ושליחתן כהודעות TCP חזרה למשתמש.
* קובץ server.py. הקובץ מכיל את הלוגיקה העיקרית של השרת. קבלת פקטות ICMP מהלקוח, הפיכתן להודעות TCP ושליחתן ל-Target המבוקש, קבלת התשובה ב-TCP ושליחתה כ-ICMP חזרה ללקוח.
* חבילה (package) של icmp. מכילה קבצי עזר לקוד:
  + Icmp.py. בניית ופרסור של פקטת ICMP.
  + Icmp\_checksum.py. חישוב checksum (מבוסס RFC-1071) של פקטת ICMP.
  + Icmp\_type.py. מכיל Enum של סוגי ההודעות – echo request & echo reply.
* חבילה של utils. מכילה את קובץ העזר sockets.py שתואר בפרק הקודם.

תיעוד הקוד

נעשו מאמצים רבים כדי להפוך את הקוד לקריא ככל האפשר, עם כמה שפחות תיעוד. קוד שדורש תיעוד הוא קוד רע, ובתיעוד עצמו ישנן בעיות רבות – התיעוד דורש עדכון, וכשנהפך ללא מעודכן יכול להרתיע אנשים אחרים שנכנסים לפרויקט; תיעוד עלול לגרום למתכנת להסתמך עליו ולהפסיק לכתוב קוד נקי שמסביר את עצמו. לכן ננקטה גישה של לכתוב תיעוד רק במקומות שבהם באמת נעשית פעולה מסובכת לוגית או עוסקת באזור רגיש. הפרויקט עצמו מתועד היטב במסמך זה ובREADME- שלו.

תכנית טסטים

System tests: כדי לבדוק שהשרת והלקוח עובדים כמצופה, ייעשו בדיקות ידניות. הבדיקות ייכללו הרצה של שניהם במקביל. תיבדק היכולת של טינול Hostים שונים, טיפול בשגיאות Socketים, מספר בקשות מאותו לקוח לשרת אחד (ללא סגירתו), לוגים אינדיקטיביים וחזרת תשובה תקינה (לרוב קובץ html).

מדריך למשתמש

כפי שמופיע בREADME הרשמי של הפרויקט:

Graphical user interface, text, application, email

Description automatically generated

הרצה לדוגמא:

* הרצת השרת:

sudo SERVER=server LOCAL\_PORT=12345 TARGET\_HOST=ynet.co.il TARGET\_PORT=443 docker-compose up

הרצה עם sudo נדרשת בשל עבודה עם raw sockets בלינוקס.

* הרצת הלקוח:

sudo docker-compose exec client curl https://127.0.0.1:12345/home/0,7340,L-8,00.html --header 'Host: www.ynet.co.il' --insecure

* קבלת התשובה בקובץ ynet.html:
* sudo docker-compose exec client curl https://127.0.0.1:12345/home/0,7340,L-8,00.html --header 'Host: www.ynet.co.il' --insecure > ynet.html

תכולות עתידיות

1. הרצה במקביל (multi-threaded) מספר לקוחות עם אותו שרת.
2. תמיכה במספר טינולים במקביל (לדוגמא ynet ו-stackoverflow על אותו שרת).
3. מערכת לוגים נוחה (במקום stdout, שמירת הלוגים בקובץ עם timestamp).

קישורים

נציג כעת מקורות שונים הקשורים לפרויקט:

הפרויקט ב-Github: <https://github.com/tzachbaksis/tcpovericmp>

על Docker: <https://www.docker.com/>

על פייתון: <https://www.python.org/>