אוניברסיטת אריאל בשומרון, המחלקה למדעי המחשב
רשתות תקשורת, סמסטר א' תשפ"ג

Final Project - SQL

ליאור וינמן 213081763
יועד תמר 213451818

6 במרץ 2023

תוכן עניינים

1 הלקוח

1.1 תיאור המערכת

1.1.1 ממשק גרפי - GUI

1.1.2 חיבור בפרוטוקול TCP

1.1.3 חיבור בפרוטוקול RUDP

1.2 טסטים

1.3 תעבורה

2 שרת הקצאת כתובות

2.1 תיאור המערכת

2.2 טסטים

2.3 תעבורה

3 מערכת שמות תחומים

בפרק זה נתאר את מערכת שמות התחומים (Domain Name System) אשר בנינו. נסקור את התוכנית עצמה (הקוד), נציג את הטסטים שכתבנו לצורך בדיקת התוכנית, נראה את הפלט של התוכנית ונתבונן בתעבורה של המערכת ברשת (תעבורת Wireshark).

3.1 תיאור המערכת

כאן נעבור על הקוד של ה־Dns שכתבנו:
כאן אנו מייבאים ספריות לשימוש, ספריית socket מאפשרת לנו לפתוח שקעים בין לקוחות ושרתים ולבצע התקשרות (בשרת זה התקשורת תהיה בפרוטוקול Udp). ספריית datetime מספקת לנו גישה לזמן הנוכחי. ספריית validators נותנת לנו דרך לוודא קלט תקין מהמשתמש (הריי השרת אמור לקבל קישור לאתר ולחלץ ממנו את כתובת ה־IP של התחום, אין לנו הבטחה על תקינות הקלט). ספריית urllib מאפשרת לנו עבודדה נוחה עם קישורים לאתרי אינטרנט.

```
# imports
import socket  # for socket-programming
from datetime import datetime  # for time and data
import validators  # for url-validation
from urllib.parse import urlparse  # for domain extracting
```

כאן אנו מגדירים את הקבועים שאיתם נשתמש. מגדירים את הכתובת שבה השרת נמצא ובאיזה פורט הוא משתמש. מגדירים את מספר הבתים המקסימלי שניתן לקבל מהסוקט ומגדירים מספר שגיאה כללי (לצורך סימון בקוד).

```
# constants
SERVER_ADDR = ("127.0.0.1", 53)  # server address and port
BUFFER_SIZE = 1024  # maximal size of received message
ERR = -1  # global err code
```

כאן כתבנו פונקציה אשר מקבלת קישור, ראשית בודקת שהוא בפורמט תקין מהצורה:

$$< Protocol >://< SubDomain > . < Domain > . < Extention > / < Path >$$

כאשר SubDomain ו־Path הם אופציאונליים. וכדי להחזיר את השם של התחום מספיק שהקישור יהיה מהצורה הבאה:

$$< Protocol >://< Domain > . < Extention >$$

לכן, מספיק לבדוק האם יש פרוטוקול בתחילת הקישור, אם לא נוסיף באופן גנרי HTTP ולאחר מכן נחזיר את שם התחום. כמובן אם קרתה שגיאה, נחזיר 1- כמספר השגיאה שהגדרנו.

```
class DNS:
    """
    this class is a DNS server implementation
    """

    def get_domain(self, url):
        """
        this function gets an url and checks if it valid
        :param url: the url to check validation
        :return: url's domain if the url valid, -1 else
        """

        if not urlparse(url).scheme:  # checking if url is in legal format
            url = "http://" + url
        try:
            validation = validators.url(url)  # validating url
        except TypeError:
            return ERR
        if validation:
            return urlparse(url).netloc  # returning the domain name
        return ERR
```

כאן כתבנו את הפונקציה הראשית, ראשית אנו מגדירים מילון (קבוצת זוגות של שם תחום וכתובת IP), לאחר מכן פותחים שקע הפועל בפרוטוקול UDP, ומקבעים לו את הכתובת ואת הפורט לקבוע אותו הגדרנו בהתחלה.

```
def main(self):
    """
    this is main function, here we're receiving message from client,
    then checking if its valid url - if yes sending it domain's ip address, if not, sending error message
    :return: -1 - cannot create server's socket
    """

    dns_cache = {}  # dictionary that holding pairs: (domain, ip)

    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as server_sock:  # creating UDP socket

        try:
            server_sock.bind(SERVER_ADDR)  # socket binding
        except Exception as e:
            print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] {e}")

        print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] server bound on: {SERVER_ADDR}")
```

כאן זוהי הלולאה הראשית של השרת, הלולאה אינסופית כיוון שאנו רוצים שהשרת תמיד יעבוד ותמיד יחכה לתקשורת. בתחילת הלולאה, אנו מקבלים מהמשתמש כתובת ועלינו להחזיר את כתובת הIP של התחום, אם קיימת. אנחנו שולחים את מה שהתקבל מהלקוח לפונקציה הקודמת, אשר היא תחזיר לנו אם הכתובת קיימת או לא. אם אינה קיימת (כלומר הפונקציה החזירה את מספר השגיארה), נשלח ללקוח כי התרחשה שגיאה והתחום אינו קיים. אחרת, כלומר הפונקציה לא החזירה שגיאה, קודם כל נבדוק האם התחום עצמו כבר שמור לנו במילון שהגדרנו (שהריי תפקידו הוא להיות מטמון לשרת), אם שמור כבר, כלומר הבקשה הזו כבר התקבלה בעבר, נדע להחזיר את התשובה באופן מיידי ללקוח ללא חישובים, אחרת, נחשב את כתובת ה־IP של התחום ונשלח בחזרה ללקוח.

```
while True:

    try:
        client_msg, client_addr = server_sock.recvfrom(BUFFER_SIZE)  # receiving url
    except Exception as e:
        print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] {e}")

    print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] received message from: {client_addr}")

    dom = DNS.get_domain(self, client_msg.decode())  # getting domain name

    if dom == ERR:  # checking if there is an error
        try:
            server_sock.sendto("Non-Existent Domain".encode(), client_addr)  # sending error message
        except Exception as e:
            print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] {e}")

        print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] message has been sent to: {client_addr}")

        continue

    elif dom not in dns_cache:  # checking if current domain is in cache
        try:
            dns_cache[dom] = socket.gethostbyname(dom)
        except Exception as e:
            print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] {e}")

    rep = dns_cache[dom]

    try:
        server_sock.sendto(rep.encode(), client_addr)  # sending the domain's ip
    except Exception as e:
        print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] {e}")

    print(f"[{datetime.now().strftime('%d-%m-%Y %H:%M:%S')}] message has been sent to: {client_addr}")
```

3.2 טסטים

3.3 פלט

כאן נציג פלט לדוגמה של מערכת שמות התחומים שכתבנו.
כאן ניתן לראות פלט אשר השרת יציג לנו רוב הזמן, בפלט ניתן לראות חתימת זמן של הפעולה שקרתה ותיאור קצר של הפעולה, כמובן גם מופיעות הכתובות והפורטים שאיתם השרת מתקשר באותה הפעולה. כך נוכל לעקוב אחר פעילות השרת ובמקרה של משהו חריג, נוכל לדעת מתי זה קרה ומי גרם לכך.

```
[06-03-2023 02:41:20] server bound on: ('127.0.0.1', 53)
[06-03-2023 02:41:26] received message from: ('127.0.0.1', 50155)
[06-03-2023 02:41:26] message has been sent to: ('127.0.0.1', 50155)
[06-03-2023 02:41:26] received message from: ('127.0.0.1', 50156)
[06-03-2023 02:41:26] message has been sent to: ('127.0.0.1', 50156)
[06-03-2023 02:41:27] received message from: ('127.0.0.1', 50157)
[06-03-2023 02:41:27] message has been sent to: ('127.0.0.1', 50157)
[06-03-2023 02:41:29] received message from: ('127.0.0.1', 50158)
[06-03-2023 02:41:29] message has been sent to: ('127.0.0.1', 50158)
[06-03-2023 02:41:29] received message from: ('127.0.0.1', 50159)
[06-03-2023 02:41:29] message has been sent to: ('127.0.0.1', 50159)
[06-03-2023 02:41:30] received message from: ('127.0.0.1', 50160)
[06-03-2023 02:41:30] message has been sent to: ('127.0.0.1', 50160)
[06-03-2023 02:41:31] received message from: ('127.0.0.1', 50161)
[06-03-2023 02:41:31] message has been sent to: ('127.0.0.1', 50161)
[06-03-2023 02:41:34] received message from: ('127.0.0.1', 50162)
[06-03-2023 02:41:34] message has been sent to: ('127.0.0.1', 50162)
```

3.4 תעבורה

SQL APPLICATION 4