מבוא לרשתות תקשורת

תרגיל מספר 3

להגשה עד ל 30.12 בשעה 22:00 דרך המודל בלבד

<u>חלק א:</u>

1. מחשב A שולח הודעה למחשב B, ובמסלול ביניהם יש 2 נתבים.

הMTU בין A ל R1, וכן בין R1 ל R2 הוא 1500B.

הMTU בין R2 לבין B הוא 660B.

מחשב A שולח למחשב B כמות של 1980 בתים של מידע (שכבת אפליקציה).

נתון: גודל תחילית UDP 10B, תחילית TCP של 20B ותחילית IP של 20B.

תארו כיצד ישלח המידע באמצעות UDP ובאמצעות TCP.

בתשובתכם יש לציין את ערכי השדות: IP-ID, Length, MF, DF, Offset.

2. נתון חיבור TCP רגיל (כלומר, כולל fast retransmit ו fast recovery) ולאחר ביצוע Handshake, אשר נשלחות בו 7 חבילות.

נתון:

- ערך ה threshold ההתחלתי הוא 10 חבילות.
- השהיית ההתפשטות היא d_p.
- משך זמן עבור timeout של חבילה הוא RTO.
- שאר ההשהיות זניחות.

א. ציירו דיאגרמת זמנים וחבילות כאשר החבילה השניה (בלבד) הולכת לאיבוד, חשבו כמה זמן לוקח לשדר את כל החבילות עד שהשולח יודע בוודאות שכולן הגיעו ליעדן (כפונק' של d_p ו RTO).

ב. ציירו דיאגרמת זמנים וחבילות כאשר החבילה הרביעית (בלבד) הולכת לאיבוד, חשבו כמה זמן לוקח לשדר את כל החבילות עד שהשולח יודע בוודאות שכולן הגיעו ליעדן (כפונק' של d_p ו RTO).

3. לקוח רוצה לשלוח הרבה מידע לשרת. נתון:

- האפר אצל השרת הוא 360B.
- משתמשים במספרים סידוריים יחסיים המתחילים מ 0.
- ה MSS=360B
- אין delayed Acks.
- האפליקציה קוראת בקצב שליש כאשר היא מקבלת מידע (על כל 3 בתים שהיא מקבלת היא מספיקה לקרוא 1), וכאשר היא לא מקבלת מידע (כלומר, בזמן שלוקח לack להתפשט ולbyte הראשון של החבילה הבאה להגיע), היא מספיקה לקרוא 40 בתים.
- אין שימוש במנגנונים שנועדו להתמודד עם silly window syndrome.

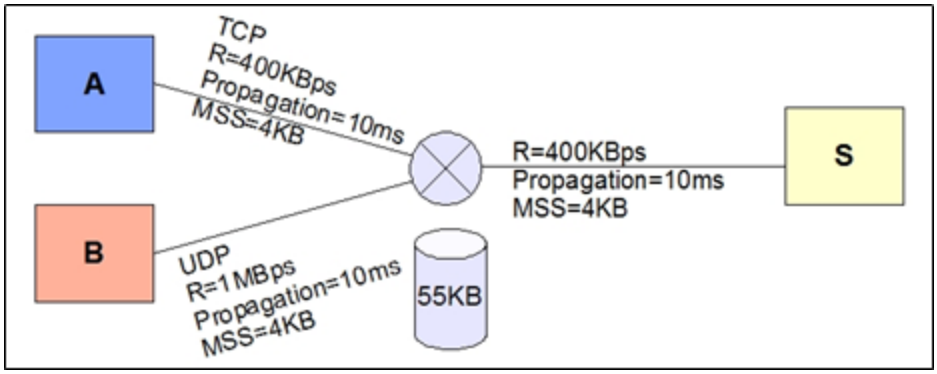
א. הראו באמצעות דיאגרמת חבילות (ללא חישוב זמנים) את הקמת החיבור ואת החבילות הנשלחות עד שהלקוח יודע בוודאות שהשרת קיבל לפחות 560 בתים. עבור כל החבילות יש לציין את השדות:

seq #, ack #, דגלי Syn ו Ack, ReceiveWindow, Data length

כמו-כן, יש לצייר את באפר השליחה של הלקוח ואת באפר הקבלה של השרת לאורך הדיאגרמה.

ב. הסבירו את בעיית ה silly window syndrome כפי שמתבטאת בחלק א', והסבירו כיצד המנגנון המתאים היה פותר אותה.

4. נתונה הרשת הבאה:



נתון:

- לקוח A מעלה קובץ בגודל 40KB לשרת S מעל TCP.
- לקוח B מעלה קובץ בגודל 20KB לשרת S מעל UDP.
- בזמן 0 לקוח A פונה לשרת S לצורך העלאת הקובץ בגודל 40KB מעל TCP, וכעבור 0.2 שניות הלקוח B מתחיל בהעלאת הקובץ שלו, קובץ בגודל 20KB לשרת S מעל UDP.
- התור בנתב בגודל 55KB.
- הבאפר בשרת S בגודל 100KB.
- ה-MSS בגודל 4KB.
- אין delayed ACK.
- קצב שידור של לקוח A, שרת S והנתב הוא 400KBps, וקצב שידור של B הוא 1MBps.
- timeout=0.1seconds.
- השהיית ההתפשטות בכל הערוצים 10 מילישניות.
- התעלמו מ-Headers ומהשהיית השידור של הודעות בקרה (ACK).

הדגימו באמצעות תרשים חבילות וזמנים מזמן 0 ועד שהלקוח A יודע בוודאות שהשרת S קיבל את הקובץ בשלמותו. הציגו חישוב זמנים מפורט וחשבו במדויק את הזמן עד הרגע ש-A
יודע בוודאות שהקובץ התקבל אצל השרת בהצלחה.

5. לקוח ושרת מחוברים ביניהם בעזרת ראוטר. לקוח פותח חיבור לשרת ושולח בקשה (קצרה) להורדת קובץ בגודל 16KB מהשרת (הניחו זמן שידור הבקשה זניח, ולכן ניתן להתעלם ממנו).

נתון:

- קצב שידור בשני הערוצים (בשני כיוונים) הוא 1MBps
- מהירות התפשטות 250000 ק"מ לשניה.
- מרחק בין הלקוח לנתב הוא 250 ק"מ ובין נתב לשרת 750 ק"מ.
- ה-MSS הוא 2KB (התעלמו מ-headers), ועם סיום קבלת החבילה תמיד נשלח ACK .
- הודעות ה ACK קטנות וקצב שידור שלהן זניח (לכן ניתן להתעלם מזמן השידור שלהן).
- התור בנתב בגודל 4KB ואין תעבורה אחרת ברשת.
- הניחו שהאפליקציה בלקוח קוראת מה-buffer בקצב קבוע של 0.5MBps. גודל ה-receiveBuffer הוא 7KB.

א. הראו דיאגרמת זמנים וחבילות בין השרת והלקוח כולל הקמת וסגירת החיבור (המספר הסידורי ההתחלתי בשרת הוא 20000 ובלקוח הוא 30000). על כל חבילה שנשלחת יש לציין את המספרים הסידוריים, גודל ה-receive Window, דגלי TCP, זמן וחלון בקרת עומס.

ב. חשבו כמה זמן יקח להעביר את הקובץ.

חלק ב:

עליכם לממש שרת TCP המתפקד באופן הבא:

הלקוח שולח לשרת שם של קובץ שהוא מעוניין להוריד ממנו (כלומר, שהשרת ישלח לו בחזרה).
הקבצים יושבים בתוך תיקייה בשם files אשר נמצאת באותה תיקייה שבה נמצא השרת.
שם הקובץ יכול לכלול גם נתיב. כלומר, אם הלקוח שלח רק את שם הקובץ, אזי הקובץ צריך להיות בתוך התיקייה files ברמה העליונה. במידה ושם הקובץ מכיל גם נתיב תיקייה, השרת מחפש את הקובץ בהתאם לנתיב בתוך התיקייה files.

הפורמט שבו הלקוח שולח לשרת הוא הפורמט הבא:
בשורה הראשונה כתוב:

GET [Message] HTTP/1.1

כאשר במקום [Message] יהיה כתוב שם הקובץ.
שימו לב, כאשר כתוב שורה - הכוונה היא שורה ממש, ולכן מיד בסופה מופיע הסימן \n
נגדיר שבמידה ושם הקובץ הוא התו הבודד / (סלאש), אזי הכוונה לקובץ בשם index.html
הלקוח שולח שורות נוספות בהודעה, אך על השרת שלכם להתעלם מרובן, כפי שיוגדר בהמשך.
הלקוח סיים לשלוח את ההודעה כאשר הוא ישלח פעמים שורה חדשה, כלומר \r\n\r\n

אם הקובץ קיים, השרת יחזיר:

HTTP/1.1 200 OK
Connection: [conn]
Content-Length: [length]

ואז שורה ריקה, ואז את תוכן הקובץ.
כאשר, במקום [conn] יהיה רשום הערך של השדה connection שהופיע בבקשה מהלקוח, ובמקום [length] יופיע גודל הקובץ הנשלח.
למשל, הלקוח שלח:

GET / HTTP/1.1
...
Connection: close

...

(הלקוח שלח מידע נוסף - מסומן על ידי שלוש נק', אך הוא לא רלוונטי כי השרת שלנו מתעלם ממנו).

והשרת שלח בחזרה את התוכן של הקובץ index.html:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 11

hello world
```

במידה והערך של השדה connection יהיה close (כמו בדוגמא לעיל), סוגרים את החיבור לאחר שליחת הקובץ.
לעומת זאת, אם הערך הוא keep-alive, יש להשאיר את החיבור פתוח - ולקרוא את בקשת הקובץ הבאה של הלקוח.

מכיוון שאין לנו שליטה על הלקוח בתרגיל הזה (פרטים בהמשך) - אין לנו אפשרות להגביל את כמות החיבורים שהוא יפתח מול השרת.
אבל, מכיוון שהשרת שלנו אינו משתמש במקביליות, נגדיר במקום ל socket פרק זמן מקסימלי שבו הוא נתקע על recv (חפשו בגוגל על timeout בהקשר של סוקט TCP בפייתון).
במידה וה recv לא מקבל תשובה אחרי 1 שניות, יש לסגור את החיבור הנוכחי בשרת ולטפל בלקוח הבא (חיבור חדש).

זאת ההתנהגות לכל סוגי הקבצים, למעט קבצים עם סיומת jpg או ico. במקרה שכזה, יש לקרוא את תוכן הקובץ בשרת בצורה בינארית ואז לשלוח אותו. למשל:

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: [length]

[binary image data]
```

אם הקובץ לא קיים, השרת מחזיר:

HTTP/1.1 404 Not Found
Connection: close

אם הלקוח ביקש קובץ בשם

GET /redirect HTTP/1.1

השרת מחזיר בחזרה:

HTTP/1.1 301 Moved Permanently
Connection: close
Location: /result.html

(יש להקפיד להחזיר בתשובה שורה ריקה אחרי השורה של location. כלומר שורה ריקה ממש ולא רק שורה חדשה)

בנוסף, על השרת להדפיס למסך את הבקשות שהוא קיבל מהלקוח.

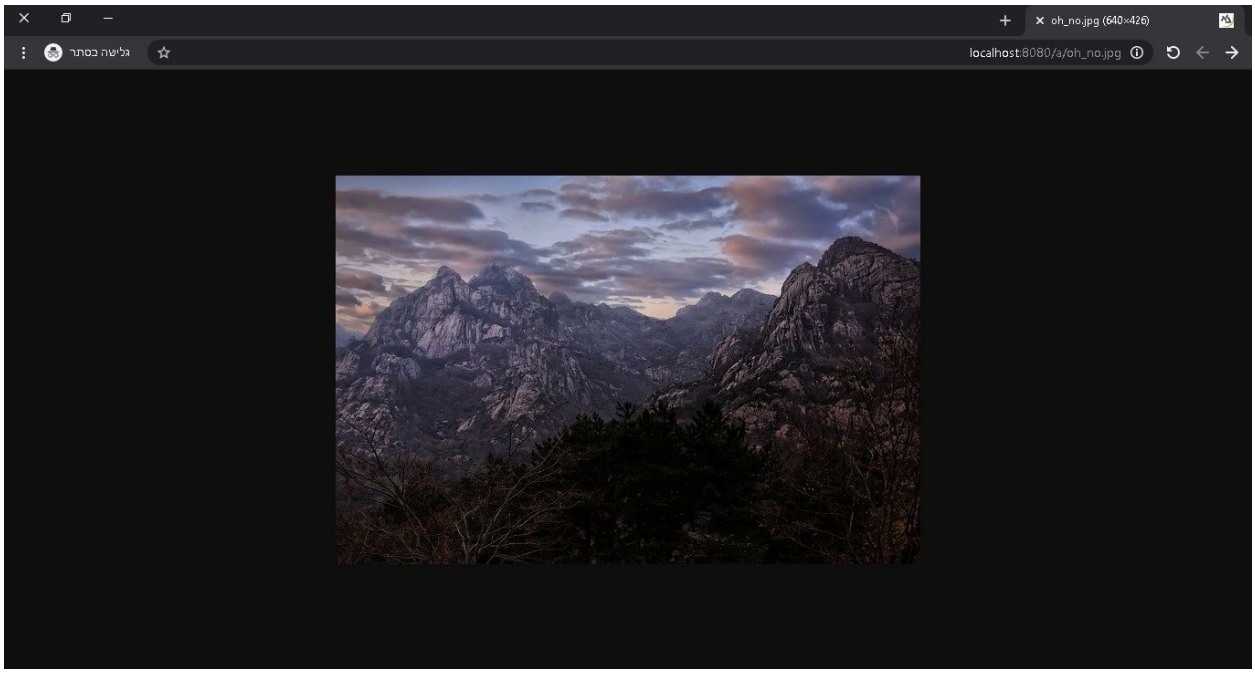
בתרגיל זה, אינכם כותבים לקוח. בתרגיל זה תשתמשו בלקוח קיים, והוא - הדפדפן שלכם (chrome). עליכם להקליד בשורת הכתובת של הדפדפן את הדבר הבא:
[Server IP]:[Server port][Path]
כלומר, כתובת ה IP של השרת, נקודותיים, ואז הפורט שהשרת שלכם מאזין לו ואז הנתיב של הקובץ. למשל:
1.2.3.4:80/
שורה זו פונה לשרת שנמצא בכתובת 1.2.3.4 ומאזין לפורט 80 ומבקשת את הנתיב / (כפי שהוגדר לעיל)
השרת שלכם מקבל כארגומנט לmain רק ארגומנט אחד - הפורט אליו הוא מאזין (כפי שהיה בתרגילים קודמים)
אם אתם מריצים את השרת על המחשב שלכם, אפשר במקום כתובת הIP המקומית (127.0.0.1) לכתוב localhost.

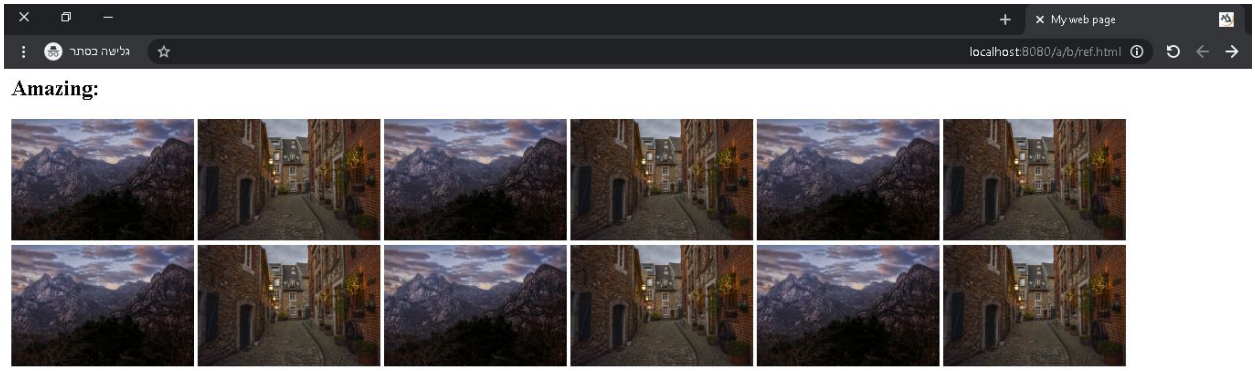
במודל תמצאו תיקיית files לדוגמא.
בבדיקה, התרגיל ייבדק מול תיקייה אחרת.

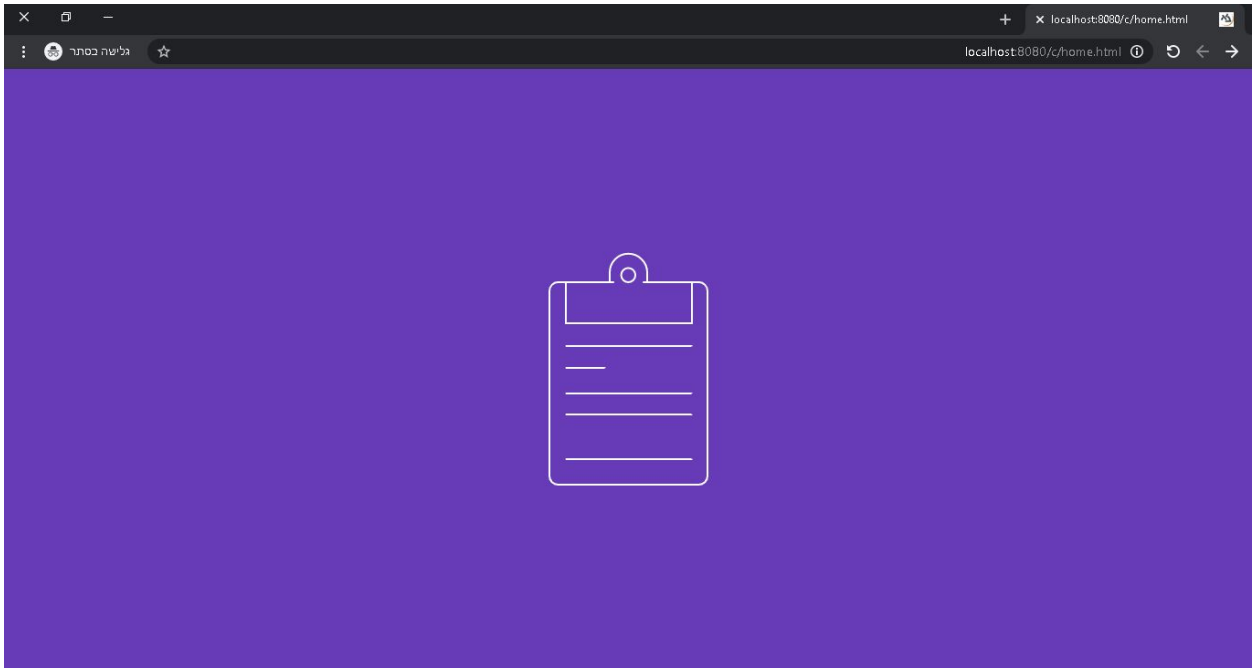
אסור להשתמש בספריות **בכלל** בתרגיל (כל ספריה או מודול אסור), למעט הספרייה socket.

להלן מספר דוגמאות להרצת התרגיל בעזרת תיקיית ה files אשר במודל:

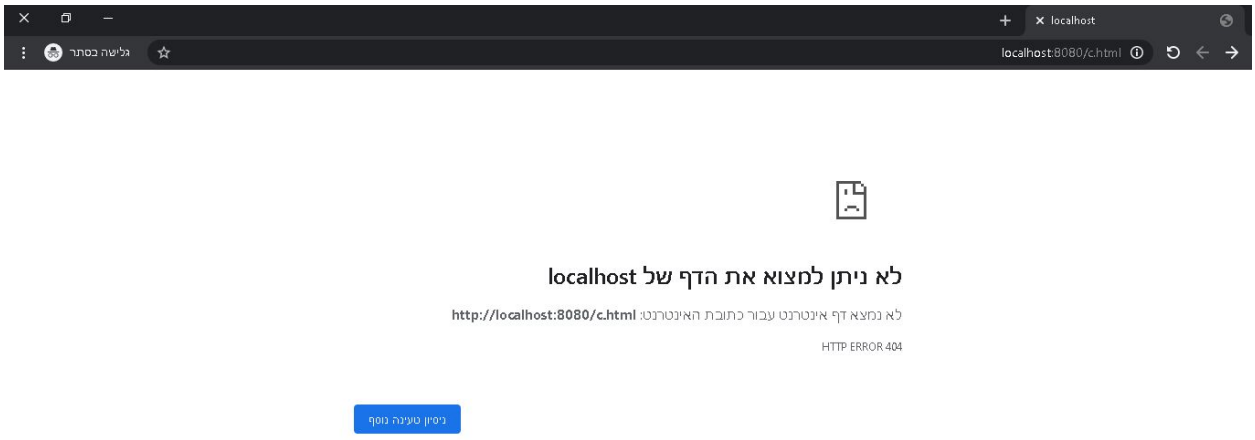
hello







דוגמא לקובץ שלא קיים:



דוגמא לגלישה redirect/ (הדפדפן שינה את הכתובת בהתאם ללוגיקה שהוגדרה לעיל):

מה יש להגיש?

1. תשובות לכל השאלות בחלק א. יש לסרוק/לצלם באיכות ברורה או להגיש קובץ דיגיטלי. אין הגשה פיזית לתיבה.
3. קובץ השרת של חלק ב, בשם server.py עם תיעוד בסיסי.
4. קבצי pcap של הרצת התרחישים המופיעים בדוגמת ההרצה לעיל (קובץ לכל תרחיש), יש לסנן ולייצא את התעבורה בהתאם.

5. מסמך PDF המסביר את התעבורה בקובץ של סעיף 4. **<u>שימו לב!</u>** אין צורך להיכנס לפרטים ולהסביר כל אחת מהחבילות בנפרד או את השדות של כל אחת מהחבילות. הדגש העיקרי של המסמך הזה הוא לנתח באופן כללי את התעבורה, בדגש על:

- כמה חיבורים היו?
- מה הועבר בכל חיבור?
- באילו חיבורים נשלחו יותר מבקשה אחת מהלקוח?
- מה קרה כשהוזן הקישור לדפדפן (מה נשלח לשרת, מה השרת החזיר)
- כאשר הדפדפן קיבל תשובה מהשרת - האם הוא שלח בקשה נוספת? איזה?

6. קובץ טקסט בשם details.txt עם שמות ות.ז. של המגישים.
7. הגשה בזוגות (**<u>רק אחד</u>** מבני הזוג מגיש את התרגיל בפועל)

בהצלחה