מבוא לתקשורת מחשבים

תרגיל 2

תוכנית שרת לקוח מעל שרות לא אמין (UDP) לצורך החזרת זמן בתצורות שונות

מגיש: אור קרן 315155531

שאלה תיאורטית 1:

שאלה: הריצו את התרגיל ובקשו בקשות GetClientToServerDelayEstimation רבות. האם ייתכן שהלקוח או השרת "ייתקעו"? אם לא, נמקו. אם כן, נמקו והציעו תיקון (ללא מימוש) בתוכנית שלכם בכדי שהלקוח והשרת לא "יתקעו".

תשובה: השאלה האם ייתכן שהרשת או הלקוח ייתקעו תלויה במחשב, ברשת, ובמהירות שבה המשתמש יוכל לבחור באופציה מספר 4.

כאשר מספר הבקשות רבה מאוד אכן השרת והלקוח יכולים להיתקע (לאחר שינוי ל1000 בקשות לכל חישוב המדמה תדירות גבוהה של בקשות, נראה כי השרת מתחיל להיתקע)

הצעתי לתיקון הינה להגביל את כמות הבקשות שניתן לשלוח לשרת בזמן נתון. אולי לבצע זאת על ידי buffer נוסף שמושך אליו כמו הודעות באינטרוול קבוע, אולי להגביל את גודל הbuffer של ספריית windows שממנה אנחנו קוראים את הבקשות וכו.

במידה ואנחנו רוצים לאפשר גישה לכל הלקוחות מלבד זה ששולח הודעות רבות, ניתן לבדוק מה כתובת הIP לכל בקשה שנשלחה ואם יש כתובת שממנה נשלחו כמות הודעות בתדירות רבה, ניתן לחסום אותה לתקופת זמן מסוימת.

שאלה תיאורטית 2:

שאלה: מבחינת ההשהיות השונות שאנו כבר מכירים (אותן 4 השהיות שנלמדו בכיתה), מה מבטא הגודל הממוצע אותו הלקוח מחשב על סמך תשובות השרת בבקשת GetClientToServerDelayEstimation?

(המלצה: נתחו גודל זה מבחינה תאורטית ולאו דווקא על סמך התוצאות המתקבלות מההרצה בפועל שעלולות להטעות, בפרט כאשר מבוצעות על אותו המחשב)

תשובה: לפי התוצאות שקיבלתי (בין 0.3 ל0.4 שניות) השווים לערך 300-400 מילי שניות,

נראה כי זמן זה מקורב להיות פעמיים timeduc (זמן הטריגר של מערכת ההפעלה לקריאת הודעה חדשה ומוגזר להיות עד 200 מילי שניות).

הסיבה שמדובר בפעמיים שכן פעם אחת השרת מחכה לבקשה ופעם נוספת הלקוח מחכה לתשובה.

פורמט שליחת הבקשות:

כלל הבקשות מוצגות בתפריט למשתמש לפי התפריט הבא:

```
=============== Menu ===============
A: get time
B: get time without date
C: get time since Epoch
D: get client to server estimated delay
E: get estimated RTT
F: get time without date or seconds
G: get year
H: get month and day
I: get seconds from start of the month
J: get week of year
K: get dayligh savings
L: get time lap
```

על המשתמש להזין את האות המתאימה על מנת לשלוח את הבקשה התאימה לשרת ולקבל את התשובה.
כלומר על מנת לקבל את השנה הנוכחית, המשתמש יזין את האות G ולאחר מכן Enter.

באופן זהה אותיות אלו יועברו לשרת הזמנים ולפי תפריט כמעט זהה, הפונקציה המתאימה תופעל (למעט בדיקת RTT ו delay ש"מכילים" גם מעין פונקציונליות של תוכנת הלקוח).
גודל הבקשות מוגדר להיות שני תווים, הchar המגדיר את הבקשה ולאחר מכן '0/'

גודל התשובה מהשרת מוגדרת להיות 255 בתים, כלל התשובות ממנו נכנסות בגודל זה ומוגדרות כתשובות טקסטואליות בלבד.

להלן צילומי WireShark המציגים את תעבורת המידע בין הלקוח לשרת:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	33	49192 → 27015 Len=1
2	0.000456	127.0.0.1	127.0.0.1	UDP	56	27015 → 49192 Len=24
3	0.510672	127.0.0.1	127.0.0.1	UDP	33	49192 → 27015 Len=1
4	0.511132	127.0.0.1	127.0.0.1	UDP	40	27015 → 49192 Len=8

ניתן לראות כי שלחתי שתי בקשות וקיבלתי שתי תשובות בחזרה:

בקשה 1: האות A

```
0000  02 00 00 00 45 00 00 1d  93 ad 00 00 80 11 00 00   ····E··· ********
0010  7f 00 00 01 7f 00 00 01  c0 28 69 87 00 09 97 29   ******** ·(i····)
0020  41                                                 A
```

תשובה 1: הזמן הנוכחי

```
0000  02 00 00 00 45 00 00 34  93 ae 00 00 80 11 00 00   ····E··4 ********
0010  7f 00 00 01 7f 00 00 01  69 87 c0 28 00 20 dc 6e   ******** i··(· ·n
0020  53 61 74 20 44 65 63 20  32 31 20 32 33 3a 31 30   Sat Dec  21 23:10
0030  3a 33 37 20 32 30 32 34                           :37 2024
```

בקשה2: B

```
0000  02 00 00 00 45 00 00 1d  93 af 00 00 80 11 00 00   ····E··· ********
0010  7f 00 00 01 7f 00 00 01  c0 28 69 87 00 09 96 29   ******** ·(i····)
0020  42                                                 B
```

תשובה 2: הזמן הנוכחי ללא תאריך

```
0000  02 00 00 00 45 00 00 24  93 b0 00 00 80 11 00 00   ····E··$ ********
0010  7f 00 00 01 7f 00 00 01  69 87 c0 28 00 10 08 45   ******** i··(···E
0020  32 33 3a 31 30 3a 33 38                           23:10:38
```

המשימה

עליכם להרחיב את אפליקצית שרת הזמן בכדי שתתמוך בשירותים הבאים:

- הלקוח יכול לשלוח מספר סוגי בקשות שונות, בהתאם לקלט המשתמש. להלן סוגי הבקשות:

1. GetTime – החזר את הזמן בתצורה של שנה, חודש, יום, שעה, דקה ושנייה (כמו בדוגמא המקורית).
2. GetTimeWithoutDate – החזר את הזמן בתצורה של שעה, דקה ושנייה.
3. GetTimeSinceEpoch – החזר את הזמן בתצורה של שניות החל מ-1.1.1970.
4. GetClientToServerDelayEstimation – החזר אומדן של ההשהיה בין הלקוח לשרת (ראו הנחיות להלן).
5. MeasureRTT – מדידת RoundTripTime (RTT) (ראו הנחיות להלן).
6. GetTimeWithoutDateOrSeconds – החזר את הזמן בתצורה של שעה ודקה.
7. GetYear – החזר את השנה בלבד.
8. GetMonthAndDay – החזר את החודש ואת היום.
9. GetSecondsSinceBeginingOfMonth – החזר את מספר השניות שעברו מתחילת החודש הנוכחי.
10. GetWeekOfYear – החזר את מספר השבוע מתחילת השנה.
11. GetDaylightSavings – החזר 1 אם מוגדר שעון קיץ או 0 עבור שעון חורף.
12. GetTimeWithoutDateInCity – החזר את הזמן בתצורה של שעה, דקה ושנייה בערים שונות בעולם. על השרת לתמוך בערים דוחה (קטאר), פראג (צ׳כיה), ניו יורק (ארה״ב), ברלין (גרמניה) ועבור כל עיר אחרת שאיננה ברשימה להחזיר הודעה מתאימה עם הזמן האוניברסלי המתואם (UTC). בצד הלקוח, במידה והמשתמש בוחר באפשרות 12 יש להציג את רשימת הערים הזמינות ולקלוט מהמשתמש עיר ספציפית בה הוא מעוניין.
13. MeasureTimeLap – החזר את הזמן שעבר בין בקשת MeasureTimeLap הראשונה שהתקבלה מהלקוח לבקשת MeasureTimeLap השנייה. עבור הבקשה הראשונה תוחזר הודעה שמעדכנת את הלקוח כי הופעלה המדידה ועבור הבקשה השנייה יוחזר הזמן שעבר בין שתי הבקשות. אם הלקוח הפעיל מדידה ולא עצר אותה כעבור 3 דקות לכל היותר, יפסיק השרת את המדידה מבלי לעדכן את הלקוח.

- על השרת לעבד בקשות אלו ולהחזיר תשובה מתאימה, לפי סוג <u>השירות</u> המבוקש.
- על השרת לרוץ בלולאה אינסופית ולחכות לקבלת בקשות מהלקוחות.
- על הלקוח לרוץ בלולאה ולהציג תפריט למשתמש המפרט את הבקשות השונות האפשריות ובנוסף מציע אפשרות יציאה.

- בדומה לאפליקצית שרת הזמן, הסוקטים יהיו Blocking ואין צורך לממש את האפליקציה כ-Non-Blocking.
- לסעיפים ד ו-ה בלבד יש להשתמש בפונקציה **GetTickCount()**[1] (במקום פונקציית **time()**).
- מומלץ ל"הציץ" ולהבין כיצד אפשר להשתמש בפונקציית **.localtime()**

הנחיות לאומדן ההשהיה בין הלקוח לשרת (בקשה 4)

בסעיף זה אנו מעוניינים למדוד את אחת ההשהיות אותן למדנו בשיעור הראשון.

- השתמשו בלקוח ה-TimeServer כך שישלח 100 בקשות "מה הזמן" (מותאמות לסעיף זה) רצופות. ולאחר מכן יקלוט 100 תגובות מהשרת.
- התשובה לבקשת "מה הזמן" בסעיף זה צריכה להשתמש ולהחזיר את הערך שמוחזר ע"י פונקציית **GetTickCount**() (כפי שהוסבר לעיל).
- הלקוח יעשה מיצוע (יחשב ממוצע) של הפרשי חותמות הזמן, כפי שנמדדו בשרת.

הנחיות למדידת RTT (בקשה 5)

- השתמשו בלקוח ה-TimeServer כך שישלח בסה"כ 100 בקשות "מה הזמן?" (לא רצופות!) ויקלוט 100 תגובות מהשרת. כלומר, בקשה, תשובה, בקשה, תשובה ובסה"כ 100 זוגות כאלו.
- עבור כל אחת מ-100 הבקשות, הלקוח יחשב את הזמן שחלף מרגע שליחת הבקשה ועד מועד קבלת התשובה המתאימה וימצע זמנים אלו עבור כל 100 ההודעות שנשלחו.

שאלות תיאורטיות [10 נק']

- הריצו את התרגיל ובקשו בקשות GetClientToServerDelayEstimation רבות. האם ייתכן שהלקוח או השרת "ייתקעו"? אם לא, נמקו. אם כן, נמקו והציעו תיקון (ללא מימוש) בתוכנית שלכם בכדי שהלקוח והשרת לא "ייתקעו".
- מבחינת ההשהיות השונות שאנו כבר מכירים (אותן 4 השהיות שנלמדו בכיתה), מה מבטא הגודל הממוצע אותו הלקוח מחשב על סמך תשובות השרת בבקשת GetClientToServerDelayEstimation? (המלצה: נתחו גודל זה מבחינה תאורטית ולאו דווקא על סמך התוצאות המתקבלות מההרצה בפועל שעלולות להטעות, בפרט כאשר מבוצעות על אותו המחשב)

מסמך נלווה [15 נק']

---

[1] המוגדרת ב- http://msdn.microsoft.com/en-us/library/ms724408%28VS.85%29.aspx. שימו לב כי יש צורך ב: #include<windows.h>

- יש להגיש בנוסף לקוד התכנית (קבצי ה-cpp) מסמך נלווה המתאר כיצד בחרתם לממש את האפליקציה, כלומר, סוגי ההודעות השונות, המבנה שלהן והפעולות השונות הנובעות מהן (זהו למעשה הפרוטוקול). בנוסף יש לצרף תצלומי מסך (print screen) של תוכנת ה-wireshark אשר יראו את סוגי ההודעות השונות וההתאמה שלהן למבנה שתיארתם באופן מילולי.
- כמובן שניתן גם לצרף את התשובות לשאלות לעיל לאותו מסמך.

ההגשה היא ביחידים בלבד. בהצלחה!

- חובה להגיש תרגיל זה (אי הגשת התרגיל "תזכה" בציון 0). אין לעשות תרגיל זה בקבוצה (זוג מוגדר כקבוצה).
- הגשה באיחור גוררת **הורדת ציון** של 15% לאחור של עד שלושה ימים, 30% עד שישה ימים וכך הלאה.