מבוא לרשתות מחשבים חורף תשפ"ה

תרגיל בית 4 - Load Balancer (Simplified)

תאריך הגשה: יום ה' 23/01/2025 עד שעה 23:59.

האחראי על התרגיל: ערן תבור

שאלות והערות – דרך אתר הקורס במודל

ההגשה בזוגות בלבד (למעט אישור במייל מערן) והינה אלקטרונית בלבד דרך אתר הקורס.

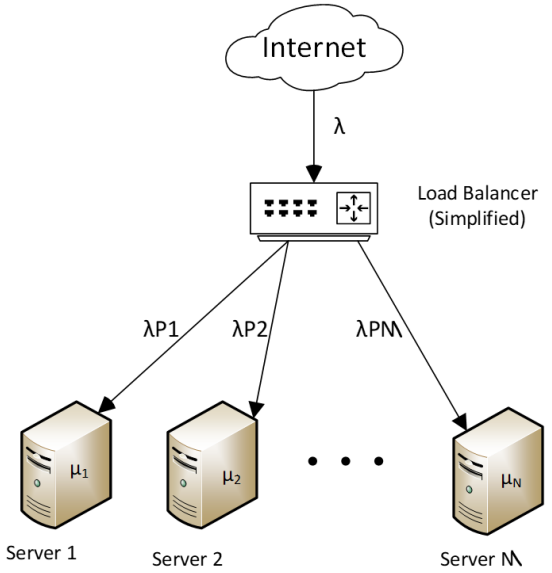
מבוא:

בתרגיל זה תממשו מודל מפושט של התקן חלוקת עומס (Load Balancer) בעזרת סימולציה של מערכת תורים.

התרגיל יבנה בשלבים כשבשלב האחרון התוכנית שתכתבו תסמלץ את המערכת הבאה:

Load Balancer

להתקן שלנו יש פורט כניסה אחד (Ingress) דרכו נכנסות הודעות בקשה לשירות מהאינטרנט. להתקן יש M יציאות (Egress) דרכן משדר ה-Load Balancer את ההודעות ל M שרתים שונים. במודל המפושט שלנו אין תעבורה החוזרת מהשרתים חזרה לאינטרנט.



- בקשות השירות כולן זהות.
- מופע בקשות השירות מהאינטרנט מפולג פואסונית עם פרמטר λ בקשות ליח' זמן.
- ה- Load Balancer מפנה את בקשות השירות באופן אקראי אל השרתים השונים בהסתברות המתאימה לכושר העיבוד של השרת. לדוגמה: בקשת שירות שמגיעה מהאינטרנט תופנה לקבלת שירות בתחנה i בהסתברות P_i $(0 \le P_i \le 1)$. P_i יהיו חלק מהקלט לסימולציה. יש לוודא ש $\sum_{i=1}^{M} P_i = 1$.

- זמן העיבוד של הודעה בתוך ה Load Balancer וזמן ההעברה של ההודעות מה load balancer לשרתים – זניח.
- כל שרת מחזיק בכניסה תור של בקשות לשירות, גודל התור של השרת ה- i הוא Q_i בקשות $(0 \leq Q_i)$.
- אם בקשה מגיעה לשרת והשרת פנוי היא תכנס לשירות מיידית.
- אם בקשה מגיעה לשרת והשרת אינו פנוי היא תכנס להמתנה בתור של השרת אם יש בו מקום.
- אם בקשה מגיעה לשרת כשהשרת אינו פנוי והתור מלא, הבקשה הגיעה היא תיזרק ולא תקבל שירות.
- בכל השרתים בקשות השירות מקבלות שירות לפי סדר ההגעה (FIFO).
- קצב השירות של שרת i מפולג פואסונית עם פרמטר μ_i בקשות ליח' זמן.

הנחיות כלליות למימוש:

- ניתן להשתמש בכל שפה שנוחה לכם
- יש לוודא את נכונות הקלט
- מומלץ לתעד את הקוד באופן הבא:
  (סעיף זה אינו מחייב אך יקל על הבודקים להבין מה עשיתם במקרה שווקטור בדיקה יכשל)
  - בתחילתו הסבר כללי על הרעיון המרכזי במימוש
  - לכל פונקציה הסבר קצר לגבי מטרתה
  - במקומות ספציפיים בהם קיים איזשהו תחכום שהקורא הסביר (=סטודנט אחר בקורס) עשוי להיתקל בקושי בהבנה מומלץ להוסיף הסבר

1. מימוש תור M/M/1/N

כתבו סימולציה לתור M/M/1/N (תור M/M/1 בעל N מקומות במערכת - כולל הבקשה שבשירות).

1.1. הקלט לסימולציה יהיה:

1.1.1. λ – קצב ההגעה [הודעות ליח' זמן]

1.1.2. μ – קצב השירות [הודעות ליח' זמן]

1.1.3. N – מספר ההודעות במערכת (כולל זו שמקבלת שירות)

1.1.4. T – זמן ריצת הסימולציה (באותן יחידות זמן כמו קצב המופע וקצב השירות)

הסימולציה תיעצר בזמן T והודעה שנמצאת בטיפול תחשב כאחת שלא טופלה.

הסימולציה נדרשת להיות מסוג event-driven. סימולציה מסוג זה מכילה תור של אירועים (כגון הגעת לקוח או עזיבת לקוח) שמנוהל באופן דינאמי (טיפול באירוע עשוי להכניס אירועים נוספים לתור או להוציא ממנו אירועים בהתאם לאפיון המערכת)

בסימולציה של התור אתם תגרילו מופע הודעות בעל פילוג פואסוני וזמן שירות בעל פילוג אקספוננציאלי. וודאו שבכל ריצה של הסימולציה מוגרל גרעין (seed) שונה. אחרת, כל הריצות עשויות להיות זהות זו לזו.

הנחיות מפורטות ודוגמה בפייתון ניתן למצוא ב:
https://www.linkedin.com/pulse/simulating-single-server-queue-python-chirag-subramanian-bpqne/

1.2. (10 נקודות) בדוגמה זו התור מומש בעזרת ערמה (heap). ניתן לממש תור אירועים גם באמצעות רשימה מקושרת. מהם היתרונות והחסרונות של כל צורת מימוש?

The main thing we need from an event list is that it would output events in a synchronized fashion i.e. in the correct order they happened. This thing is easy to achieve when using

heap and ordering by the time, but the heap implementation itself can be somewhat complicated and operations are done in $O(\log(n))$.
In an implementation with a linked list it would be harder to preserve order (insertions in $O(n)$) but implementation would be easier and deletions would be $O(1)$. Furthermore, we might be able to implement insertions with a probable complexity of less than $O(n)$

הערה: ניתן לכתוב את הסימולציה בכל שפה שנוחה לכם.

2. (10 נקודות) עבור $\lambda = 1$, $\mu = 2$, N=1000, T=5 הריצו את הסימולציה 5 פעמים רשמו בטבלה את:
2.1. מספר הצרכנים שקיבלו שירות
2.2. זמן ההמתנה הממוצע של צרכנים במערכת (כולל קבלת שירות)

	מספר הצרכנים שקיבלו שירות	זמן ההמתנה הממוצע
1	3	0.5409
2	4	0.4652
3	7	1.0411
4	3	0.5571
5	5	0.8596

3. (20 נקודות) עבור $\lambda = 1$, $\mu = 2$, N=1000 ובהתאם לחומר התאורטי שנלמד בהרצאות ובתרגולים:
3.1. מה ההסתברות שהודעה שמגיעה למערכת תמצא תור מלא ולכן תיזרק? (ניתן להיעזר בכלים נומריים לחישוב סכום הטור)

The probability the queue will be full is P_N and therefore the probability that a message will be discarded is also P_N (if a message arrives what is the probability that the queue cannot accept it).

Lets calculate P_N:

Our state machine is as shown in class but with N states.

We got the steady state equations:

1)

$$\lambda P_0 = \mu P_1$$

$$\dots$$

$$\lambda P_{N-1} = \mu P_N$$

2) and

$$\sum_{k=0}^{N} P_k = 1$$

We'll denote $\rho = \frac{\lambda}{\mu}$. from 1 we get

$$P_k = \rho^k * P_0$$

Then from 2 we will get

$$1 = \sum_{k=0}^{N} P_k = P_0 \sum_{k=0}^{N} \rho^k = P_0 * \frac{1-\rho^{N+1}}{1-\rho}$$

$$\rightarrow P_0 = \frac{1-\rho}{1-\rho^{N+1}}$$

So

$$P_{drop} = P_N = \rho^N * P_0 = \frac{(1-\rho) * \rho^N}{1-\rho^{N+1}}$$

In our case

$$P_{1000} = \frac{\left(1-\frac{1}{2}\right) * \frac{1}{2}^{1000}}{1-\frac{1}{2}^{1000+1}} = \frac{\frac{1}{2}^{1001}}{1-\frac{1}{2}^{1001}} \approx 0$$

3.2. חשבו את תוחלת מספר הצרכנים שקיבלו שירות ואת תוחלת זמן ההמתנה הממוצע במערכת (כולל קבלת שירות) בתור M/M/1 (בעל תור אינסופי), ניתן להיעזר בנוסחאות שנלמדו בכתה.

In class we got for infinite queue

$$\overline{T} = \frac{1}{\mu - \lambda} = 1$$

We say in the lecture that ρ can be seeing as utilization factor, thus the total amount of requests being served is the service rate when the server is busy:

$$avg\ total\ served = \mu * \rho * T = \lambda * T$$

3.3. הסבירו את הפער (השגיאה) בין החישוב התאורטי לתוצאות שרשמתם בטבלה בסעיף 2.

When computing the average waiting time we are considering the case when $T \rightarrow \infty$ (or more accurately- when we have infinite amount of requests (samples)) so the results are not the same as when $T = 5$. Same goes with the total served, the utilization time converges to ρ. If we were to calculate for $T = 500000$ we would get

```
Number of customers served: 500477
Number of customers dropped: 0
Average wait time: 1.0136944794407188
```

As expected.

3.4. הריצו את הסימולציה K פעמים, בכל פעם עם גרעין (seed) שונה ובידקו עבור איזה K השגיאה הממוצעת קטנה מתחת ל 5%, 1%. הסבירו את התוצאות.
לצורך התרגיל נגדיר את שגיאה בפרמטר x:

$$Error(x) \equiv \frac{|Theoretical\ Value(x) - x|}{Theoretical\ Value(x)} \cdot 100$$

As explained above when T increases we converge to the calculated avg because we are considering more samples.

At $T = 7000$ we get less than 5%:

```
Number of customers served: 7042
Number of customers dropped: 0
Average wait time: 0.9603118424950386
avg total served error: 0.6
avg wait time error: 3.968815750496135
```

At $T = 500000$ we get less than 1%:

```
Number of customers served: 500325
Number of customers dropped: 0
Average wait time: 0.9977020546219542
avg total served error: 0.065
avg wait time error: 0.22979453780458403
```

3.5. הציעו דרך נוספת (מלבד להריץ הרבה פעמים) שמאפשרת להגדיל את הדיוק בתוצאות בסימולציה.

We can increase the λ and μ so we would get more request in a shorter time and thus converge to the theoretical value. For example, $\lambda = 1000, \mu = 2000, T = 500$

```
Number of customers served: 500412
Number of customers dropped: 0
Average wait time: 0.0010001327619626312
avg total served error: 0.0824
avg wait time error: 0.013276196263113833
```

4. הרחיבו את הסימולציה כדי לממש את ה Load Balancer:

אתם נדרשים לממש את ה- Load Balancer המפושט בעזרת סימולציה מבוסס אירועים (Event Driven Simulation).

קלט הסימולטור (לפי הסדר):

- T - זמן הפעולה הכולל של הסימולוציה.
לאחר T יחידות זמן(כולל נקודת הזמן T) לא יגיעו עוד בקשות לשירות לפורט הכניסה של ה Load Balancer אולם הבקשות הקיימות בשרתים מטופלות.
- M – מספר השרתים (1 ≤ M)
- $P_1\ P_2\ \ldots P_i \ldots P_M$ - N ערכים מופרדים ברווח ומיצגים את ההסתברות שההתקן יעביר בקשה שנכנסת לשרת i. כל הבקשות שנכנסות ל Load Balancer מועברות לקבלת שירות ($\sum_{i=1}^{N} P_i = 1$)

- λ – קצב מופע בקשות השירות מהאינטרנט [בקשות ליח' זמן].
- $Q_1\ Q_2\ \dots Q_i \dots\ Q_M$ - M ערכים מופרדים ברווח ומייצגים את גדלי התורים של השרתים (מספר הבקשות המקסימאלי ששרת יכול לשמור לפני קבלת שירות)
  שימו לב שמספר ההודעות הכולל המקסימאלי שיכולות להיות בשרת ה- i הוא $Q_i + 1$
- $\mu_1\ \mu_2\ \dots\ \mu_M$ - M ערכים מופרדים ברווח ומייצגים את קצבי השירות של השרתים [בקשות ליח' זמן]

יש לוודא נכונות הקלט והפקודה אשר תריץ את הסימולציה הינה מהצורה:
(כל הערכים מופרדים ביניהם ברווח)

> ./simulator T M $P_1\ P_2\ \dots P_i \dots P_M\ \ \lambda\ Q_1\ Q_2\ \dots Q_i \dots\ Q_M\ \ \mu_1\ \mu_2\ \dots\ \mu_M$

דוגמה מספרית לקלט כזה הינה:

> ./simulator 5000 2 0.2 0.8 200 2 10 20 190

במקרה זה ,הסימולציה תרוץ למשך 5000 יחידות זמן עם שני שרתים. הסתברות שבקשת שירות תנותב לשרת מס' 1 היא 0.2 וההסתברות שבקשת שירות תנותב לשרת מס' 2 היא 0.8. קצב הגעת בקשות השירות הוא 200 בקשות ביחידת זמן. אורך התור בשרת הראשון הינו 2 ובשרת השני הינו 10. קצב השירות של שרת מס' 1 הוא 20 בקשות ביח' זמן וקצב השירות של השרת השני הוא 190 בקשות ליחידת זמן.

פלט הסימולטור הינו שורה אחת מופרדת ברווחים ומכילה את הפרמטרים הבאים לפי הסדר:

- A - מספר הבקשות שקיבלו שירות.
- B - מספר הבקשות שנתקלו בתור מלא ונזרקו ללא קבלת שירות.
- T_{end} – זמן סיום הטיפול בהודעה האחרונה
- $\overline{T_w}$ – זמן ההמתנה הממוצע של הודעה במערכת השרתים לפני קבלת שירות (רק עבור הודעות שלא נזרקו)
- $\overline{T_s}$ – זמן השירות הממוצע של הודעה במערכת השרתים

יש לעגל את התוצאות ל 4 ספרות מימין לנקודה העשרונית.

לדוגמה:

עבור קלט הדוגמה הפלט עשוי להיות:

871348 129125 5000.1400 0.0135 0.0201

או

871560 129357 5000.0319 0.0101 0.0237

'בדיקת שפיות' היא ריצה של הסימולציה עבור מקרים שאת תוצאתם ניתן לצפות ובכל לקבל חיזוק מסוים (גם אם מוגבל) לנכונות הסימולציה.

4.1. הריצו 'בדיקת שפיות' לסימולציה שכתבתם עבור המקרים הבאים, מהו פלט הסימולציה והאם הוא תואם את התאוריה?

4.1.1. בדיקת שרת בודד

4.1.1.1. > ./simulator 5000 1 1 20 1000 40

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 1 1 20 1000 40
99916 0 5000.0097 0.0254 0.0249
```

Matches theory. We got one server with size of 5000.

$$avg\ total\ served = \lambda * T = 20 * 5000 = 100000$$

And 0 drops as expected.

Also note that avg service time converges to $\frac{1}{\mu}$ as expected

4.1.1.2. > ./simulator 5000 4 1 0 0 0 20 1000 1000 1000 1000 40 40 40 40

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 1 0 0 0 20 1000 1000 1000 1000 40 40 40 40
99915 0 5000.3316 0.0248 0.0249
```

matches theory. We're in practice using only one server because of the probabilities so we should get the same as above and we did.

4.1.1.3. > ./simulator 5000 4 0 0 1 0 20 1000 1000 1000 1000 40 40 40 40

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 0 0 1 0 20 1000 1000 1000 1000 40 40 40 40
100261 0 5000.0268 0.0256 0.0251
```

Same here. We are using only one server so it should be the same.

הרחבת המקרה הקודם למספר שרתים אבל עם תוצאה דומה (שעדיין ניתן לצפות)

4.1.1.4. > ./simulator 5000 4 0.01 0.01 0.997 0.01 20 1000 1000 1000 1000 40 40 40 40

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 0.01 0.01 0.997 0.01 20 1000 1000 1000 1000 40 40 40 40
Input error: The probabilities must sum up to 1.
Usage: ./simulator T M P_1 P_2 ... P_M λ Q_1 Q_2 ... Q_M μ_1 μ_2 ... μ_M
```

Probabilities must sum up to 1.

If we change a bit, we should get almost the same result because we are mainly using one server (the one with the high probability), and indeed things stayed the same.

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 0.01 0.01 0.97 0.01 20 1000 1000 1000 1000 40 40 40 40
100202 0 5000.0706 0.0226 0.0249
```

בדיקת מקרה קצה של תור בגודל 0

4.1.1.5. > ./simulator 5000 4 0 0 1 0 20 0 0 0 0 40 40 40 40

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 0 0 1 0 20 0 0 0 0 40 40 40 40
66323 33526 5000.095 0.0 0.0251
```

Every server can handle only one task (he cannot save any tasks for late processing) and so we can see avg waiting time is 0, because every request that's not dropped immediately gets processed.

Regarding the amount of dropped messages, it should conform to

$$P_{drop} = \frac{(1-\rho) * \rho^N}{1-\rho^{N+1}} = \frac{1}{3}$$

And indeed we get

$$\frac{dropped}{all} = \frac{33526}{33526 + 66323} \approx \frac{1}{3}$$

A third of the messages get dropped.

בדיקת מקרה של קצב שירות נמוך

4.1.1.6. > ./simulator 5000 4 0.25 0.25 0.25 0.25 20 100 100 100 100 0.5 0.5 0.5 0.5

```
matan@matans-laptop:~/itn/ITN_2$ ./simulator 5000 4 0.25 0.25 0.25 0.25 20 100 100 100 100 0.5 0.5 0.5 0.5
10437 89416 5204.4091 194.9161 1.9915
```

We have low service rate and so we can see high drop rate and high waiting time.

The avg service time ofc converges to $\frac{1}{\mu} = 2$ and after the time ends it takes the queues

$$avg\ service\ time * 100 = 200$$

And indeed we see that the last message is handled about 200 units after the end.

אופן הבדיקה:

הסימולציה שלכם תורץ בעזרת כלי אוטומטי. זמן הריצה יהיה 10000 יח' זמן וקצב המופע יהיה 200 הודעות ליח' זמן. יורצו 10 וקטורי בדיקה, בדומה לדוגמאות מעלה. השגיאה המותרת: 10% (וקטור תקין הוא וקטור שבו כל הפרמטרים בפלט נמצאים בטווח השגיאה המותר). כל וקטור מזכה ב 5 נקודות.

<u>הגשה</u>

- יש להגיש אלקטרונית דרך אתר הקורס קובץ zip יחיד בשם <id1>-<id2>.zip
(שימו לב רק zip ולא כל כיווץ אחר)
- בתוך קובץ ה- zip יימצא בין השאר קובץ makefile כך שהרצת הפקודה make לאחר פתיחת ה- zip תיצור את קובץ ההרצה בשם simulator
שימו לב שהדרישה ל- makefile הינה להקל עליכם לבחור שפת פיתוח (ראו הגבלה בהמשך)
ישנן שפות פיתוח שלא דורשות קומפילציה ועבורן יש להגיש makefile ריק שלא מבצע כלום ולהגיש ביחד איתו את סקריפט ההרצה simulator
- בתוך קובץ ה zip -יימצא בנוסף קובץ שייקרא dry.pdf אשר כולל את התשובות לשאלות בגיליון זה.
- את הסימולטור ניתן לכתוב בכל שפת תכנות שתרצו (פייתון מגרסה 3.10 ומעלה).
- התרגיל יבדק על מכונת Linux 22.04

- הסימולציה שלכם צריכה להסתיים תוך 2 דקות, כל ריצה שלא תסתיים תוך זמן זה תיחשב כריצה תקועה ותגרום להורדת ניקוד. המטרה של מגבלה זו הינה להימנע מלולאות אינסופיות בתוכניות שלכם, ולא לגרום לכם להשקיע זמן באופטמזציה של הקוד שלכם. כל עוד הקוד שלכם סביר ואין בו לולאות אינסופיות הקוד שלכם אמור להסתיים בזמן זה.
- עקב הבדיקה האוטומטית ,הגשות שלא יעמדו בתנאי ההגשה יקבלו ניקוד נמוך ,לכן בידקו היטב את תוצאותיכם.

כדי לוודא שכולכם מבינים היטב את התוכנית שכתבתם, חלק מהקבוצות עשויות להתבקש לבצע code review בפגישה אישית עם צוות הקורס. בפגישה זו כל אחד מחברי הקבוצה ידרש להציג הבנה מעמיקה של הקוד שכתבתם.

שאלות, הבהרות, הערות ובקשות, נא להפנות לערן: tavran@cs.technion.ac.il