Operating Systems – 234123

Homework Exercise 3 – Dry

מגישים:
נועה פריאנטה 206200305
pariente@campus.technion.ac.il
ששון שמואל למעי 325172351
sason-shmuel@campus.technion.ac.il

שאלה 1 - Networking - תקשורת (52 נק')

חלק פתוח: הסבירו בקצרה (לא יותר מ2-1 שורות!)
לאחר מסע בחופי ישראל, יוליה כלבת הים הנזירית שמעה על מסיבת הקיץ של הפקולטה למדעי המחשב, ה-"טאוביץ", והחליטה לקפוץ לביקור. כשהגיעה לבניין טאוב, הבינה יוליה כי לא תמצא פה חוף ים וחיפשה דרכים לצנן את גופה. יוליה החליטה להזמין את קראנץ' הפיסטוק שכולם מדברים עליו וניסתה להזמין לעצמה אחד, אך אבוי, יוליה היא כלבת ים ולא עברה עדיין את הקורס מערכות הפעלה ולכן לא יודעת כיצד עובדת רשת האינטרנט, עזרו ליוליה להבין מושגים בסיסיים בעולם הרשתות על מנת שתוכל להזמין לעצמה קראנץ' פיסטוק.

א. (4 נקודות) הסבירו מה תפקיד של פרוטוקול ARP.
תשובה: פרוטוקול ARP הינו פרוטוקול בשכבת Data Link שתפקידו לאתר למפות כתובת MAC (כתובות המשמשות בשכבה 2) של תחנה ברשת לכתובת IP (כתובות המשמשות בשכבה 3) שלה.

ב. (4 נקודות) איזה מידע הלקוח צריך לדעת על השרת לפני ההתחברות?
תשובה: לפי התחברות לשרת הלקוח צריך לדעת שני נתונים:
1. כתובת IP של השרת או כתובת URL שלו אשר שרת DNS יוכל למפות לו לכתובת IP
2. מספר ה-port בו השרת מאזין לצורך השידור.

ג. (4 נקודות) איזה מידע הלקוח ידע על השרת אחרי ההתחברות?
תשובה: לאחר ההתחברות הלקוח אל השרת הוא ידע את כתובת ה-IP של השרת, את ה-port בו השרת מאזין (ומכאן שגם את פרוטוקול התקשורת) ואת ה-socket בו עוברת התקשורת ביניהם וכל מידע נוסף שהשרת יספק.

ד. (4 נקודות) איזה מידע השרת צריך לדעת על הלקוח לפני ההתחברות?
תשובה: השרת מאזין ב-port להתחברויות של לקוחות, אינו צריך מידע מקדים עליהם.

ה. (4 נקודות) איזה מידע השרת ידע על הלקוח אחרי ההתחברות?
תשובה: לאחר ההתחברות השרת ידע את כתובת IP של הלקוח, מספר ה-port בו הלקוח מתקשר וה-socket וסוג בקשת הלקוח וכן כל מידע נוסף שהלקוח ישלח לו.

ו. (6 נקודות) מה הבדל בין הפורט (port) שבשימוש השרת וזה של הלקוח. אין נבחר כל אחד מהם?
תשובה: מספר ה-port של השרת מציין את פרוטוקול התקשורת ולכן את סוג השירות אותו השרת מספק (HTTP SMTP וכו') על פי קונבנציות ידועות מראש לכל well-known port. ה-port של הלקוח נבחר באופן שרירותי על ידי מערכת ההפעלה ephemeral port עבור אותו חיבור ספציפי עם השרת וזה זמני עבור החיבור.

ז. (6 נקודות) מה הבדל בין פרוטוקול TCP ו-UDP? הסבירו למה חלק מהאפליקציות מעדיפות TCP וחלק UDP.
תשובה: פרוטוקול TCP מבטיח תקשורת אמינה (מבוסס ACK) והינו מבוסס חיבור (החלפת מידע לאחר ייסוד החיבור), UDP לא מבטיח תקשורת ואינו מבוסס חיבור (החלפת המידע בין הצדדים מתחילה מיד). אפליקציות Zoom ו-Skype ומשחקי מחשב online לדוגמה יעדיפו UDP מכיוון שיותר חשוב להם מהירות העברה מאשר אמינות המידע (תקיעות). לעומת זאת דפדפן האינטרנט יעדיף TCP מכיוון שאמינות המידע יותר חשוב ממהירות ההגעה.

ח. (6 נקודות) מהו תפקיד פרוטוקול ה- DNS?

א. לשלוח פקטות (frame) מחשבי קצה בתוך אותה רשת (LAN connectivity).

ב. לתרגם כתובת IP לכתובת MAC.

ג. לתרגם שם השרת לכתובת IP.

ד. לתרגם שם השרת לכתובת MAC.

ה. לשלוח פקטות בין מחשבי קצה ברשתות שונות (WAN).

ו. לאפשר תקשורת בין שני תהליכים במחשבי קצה ברשתות שונות (WAN).

נימוק:

פרוטוקול DNS (Domain Name Server) משמש לתרגום שם שרת לכתובת IP. יתרונות: קל יותר לזכור שם שרת מאשר כתובת IP כמו כן היא יכולה להשתנות וככה שרתי DNS הופכים את זה לשקוף בפני המשתמשים.

ט. (8 נקודות) מהו תפקיד פרוטוקול ה- NAT?

א. וידוי של הצפנת המידע.

ב. שימוש של מספר קטן של כתובות IP עבור הרבה מכשירים בתוך הרשת.

ג. הסתרת זהות הלקוח.

ד. הסתרת זהות השרת.

ה. וידוי של הצפנת המידע + שימוש של מספר קטן של כתובות IP עבור הרבה מכשירים בתוך הרשת.

ו. וידוי של הצפנת המידע + הסתרת זהות הלקוח.

נימוק:

טכניקת ניתוב ברשת המחשבים הממושת בנתבים. לרשת הפנימית יש כתובת IP אחד פומבית ובתוך הרשת יש כתובות פנימיות שאין נראות מחוץ לרשת והנתב כחלק מתפקידו כשער בין הרשת הפנימית לחיצונית מנהל זאת וממחזר כתובות פנימיות. הצורך נבע מכך שכתובות IP4 התחילו להגמר.

י. (6 נקודות) מה נכון במודל תקשורת שרת/לקוח על מנת ליצור connection (חיבור)?

א. הלקוח חייב לדעת גם שם של ה-domain של השרת וגם מספר הפורט של השרת.

ב. שרת חייב לדעת כתובת IP של הלקוח, אך הלקוח לא חייב לדעת כתובת IP של השרת.

ג. שרת חייב לדעת כתובת IP של הלקוח, וגם הלקוח חייב לדעת כתובת IP של השרת.

ד. השרת חייב לדעת גם כתובת IP וגם מספר הפורט של הלקוח.

ה. הלקוח חייב לדעת כתובת שם של ה-domain של השרת. הפורט הינו קבוע לפי סוג ה-application

ו. המידע הנחוץ תלוי בצד שיוזם את החיבור.

נימוק:

הלקוח הוא הצד שיוזם את החיבור ולכן עליו לדעת את הפרטים הללו כדי לפנות לשרת וליצור חיבור. על ידי כתובת ה-IP או שם ה-domain (כפי שציינו לעיל יש DNS שממיר ביניהם במידת הצורך) הוא ידע לפנות לשרת ועל ידי כתובת ה-port הוא ידע איפה השרת מאזין לבקשות.

שאלה 2 - סנכרון (48 נק')

לאחר הפרידה המתוקשרת של נוגה(מוכרת בעיקר על ידי השיר שלה, "חד קרנל") ומרבי, עולם הפופ הישראלי התחלק לשתי קבוצות, קבוצת נוגה וקבוצת מרבי. בין הקבוצות שררה שנאה רבה ולא היו מוכנים לשהות באותו החדר, ולכן הוגדר כי כאשר חבר אחת הקבוצות רוצה להיכנס לחדר מסוים עליו לציית לכלל הבא: אם יש חברי קבוצה אחרת בחדר אזי אסור לו להיכנס ועליו להמתין עד שיעזבו (לעומת זאת, מספר חברים מאותה הקבוצה יכולים לשהות בחדר באותו הזמן).

1. סמני נכון / לא נכון (אין צורך להסביר):

   1. (3 נק') יכולים להיות שני חברים מקבוצות שונות באותו חדר במקביל: נכון / לא נכון
   2. (3 נק') יכולים להיות שני חברים מאותה הקבוצה בחדר במקביל: נכון / לא נכון
   3. (3 נק') חברי קבוצה אחת עלולים להרעיב (כניסת) חברי קבוצה אחרת: נכון / לא נכון

בסעיפים הבאים מוצג קוד למימוש כניסה ויציאה של חברים בקבוצות השונות אל ומחדר מסוים, כאשר נתון כי:

- כל חוט מייצג חבר קבוצה כלשהי.
- בכניסה לחדר חבר הקבוצה קורא ל onArrival(int team), שמקבלת את הקבוצה אליה שייך.
- ביציאה מהחדר חבר הקבוצה קורא ל onLeave(int team) שמקבלת את הקבוצה אליה שייך.
- הערכים 0 ו-1 של team מייצגים את קבוצת מרבי וקבוצת נוגה, בהתאמה.
- (הניחו שאמצעי הסנכרון עברו אתחול תקין והתעלמו מבעיות קומפילציה אם ישנן, שכן מטרת השאלה אינה לבדוק שגיאות אתחול/תחביר).

```
1.  #include <pthread.h>
2.
3.  int members = 0;
4.  mutex_t global;
5.
6.  void onLeave(int team) {
7.          mutex_lock(&global);
8.          members --;
9.          mutex_unlock(&global);
10. }
11. void onArrival(int team) {
12.         mutex_lock(&global);
13.         while (members > 0) {
14.                 mutex_unlock(&global);
15.                 sleep(10);
16.                 mutex_lock(&global);
17.         }
18.         mutex_unlock(&global);
19.         members++;
20. }
```

2. (12 נק') בהתייחס לקוד הנ"ל, הקיפי את כל התשובות הנכונות (עשויה להיות יותר מאחת).
עבור כל תשובה שהקפת, תארי דוגמת הרצה המובילה לתשובה זו.

   a. קיימת בעיית נכונות עקב race condition למשאבים משותפים.
   b. קיימת בעיית DeadLock / Livelock בקוד.
   c. הקוד משתמש ב-Busy Wait שפוגע בנצילות המעבד.
   d. הקוד מפר את כלל הכניסה לחדר (שהוגדר בתחילת השאלה).

<u>נימוק:</u>

- קיימת בעיית נכונות עקב race condition למשאבים משותפים על ידי גישה ל-members אשר הוא משתנה משותף בשורה 20 ללא נעילה. דוגמת ריצה: שני אנשים שונים מקבוצות שונים מבצעים onArrival(0) ו-onArrival(1). התהליך הראשון לפני ביצוע שורה 19, ולכן ביצע כבר unlock למנעול. יש context switch

לתהליך השני, אשר עדיין members=0 ולכן הוא מדלק על ה-while אל שורה 18 ומבצע unlock ומצליח להכנס לחדש. שניהם מעלים ++ ל-members.

- הקוד משתמש ב-Busy Wait שפוגע בנצילות המעבד שכן כאשר שני אנשים מאותה קבוצה נכנסים לחדר, כלומר שתי קריאות onArrival(1) ו-onArrival(1) גורמות לבזבוז זמן מעבד כי אחת מהתהליכים יהיה ב-while וב-sleep(10) כל עוד האיש השני לא יצא מהחדר (זאת על אף שהם בכלל מאותה קבוצה וזה חוקי שיהיו באותו חדש ויש פה בזבוז מיותר) באופן כללי כנלמד בהרצאה הבדיקה החוזרת גורם ל-busy wait.
- הקוד מפר את כלל הכניסה לחדר מכיוון שאין הבחנה במשתנה members בין 2 הקבוצות והדוגמה בנקודה הראשונה תקפה גם פה.

המימוש של כניסה ויציאה שונה כך שישתמש במשתני תנאי:

```
1   int members[2] = {0};     // 2 counters
2   cond_t conds[2];          // 2 condition variables
3   mutex_t global;
4   void onArrival(int team) {
5           mutex_lock(&global);
6           int other = team? 0 : 1;
7           while(members[other] > 0)
8                   cond_wait(&conds[team] , &global);
9           members [team]++;
10          mutex_unlock(&global);
11  }
12  void onLeave(int team) {
13          mutex_lock(&global);
14          members [team]--;
15          int other = team? 0 : 1;
16          cond_signal(&conds[other]);
17          mutex_unlock(&global);
18  }
```

אך עומר (עתודאי במדמ"ח) טען שקוד זה גורם לחוטים להתעורר שלא לצורך ומיד לחזור למצב המתנה.

3. (7 נק') הסבירי את טענתו של עומר באמצעות דוגמת ריצה קונקרטית.

<u>תשובה:</u> הקוד אכן גורם לחוטים להתעורר שלא לצורך, כך שמודיע שחוט יכול להכנס לחדר על אף שלא. נתאר זאת בדוגמה הבאה:

1. onArrival(0) קריאה ראשונה אשר מעלה את ה-counter של team אפס ב-1. נכנס לחדר
2. לאחר מכן onArrival(0) אשר מעלה את ה-counter של team אפס ב-1. לכן כעת 2 ונכנס לחדר
3. לאחר מכן onArrival(1) אשר נכנסת למצב cond_wait מכיוון שיש 2 אנשים מהקבוצה השנייה בחדר.
4. לאחר מכן onLeave(0) אשר מורידה את ה-counter של team אפס ב-1, כעת המונה הוא 1 ושולחת סיגנל cond_signal למי שמחקה, שזה החוט משורה קודמת, אך אינו יכול עוד להכנס לחדר כי ה-counter של הקבוצה השנייה עוד גדול מ-0 ולכן שוב ב-while חוזר ל-cond_wait.

4. (8 נק') כיצד ניתן לתקן את הבעיה שהציג עומר בסעיף הקודם?

<u>תשובה:</u> נתקן על הבעיה על ידי הוספת תנאי לפני ה-cond_signal להאם ה-counter התאפס.

If(members[team])==0

עומר ניסה לשפר עוד את יעילות הקוד והחליט להשתמש בשני מנעולים: מנעול ראשון בעבור חברי קבוצה הנכנסים לחדר, ומנעול שני בעבור חברי קבוצה היוצאים מהחדר. להלן המימוש החדש (השינויים בקוד מודגשים):

```
int members [2] = {0};              // 2 counters
cond_t conds[2];                    // 2 condition variables
mutex_t m_arrival, m_leave;         // there are *2* locks now
void onArrival(int team){
        mutex_lock(&m_arrival);
        int other = team? 0 : 1;
        while(members [other] > 0)
                cond_wait(&conds[team] , &m_arrival);
        int tmp = members [team];
        members [team] = tmp + 1;
        mutex_unlock(&m_arrival);
}
void onLeave(int team){
        mutex_lock(&m_leave);
        int tmp = members[team];
        members [team] = tmp – 1;
        int other = team? 0 : 1;
        cond_signal(&conds[other]);
        mutex_unlock(&m_leave);
}
```

1. (12 נק') בהתייחס לקוד הנ"ל, הקיפי את כל התשובות הנכונות (עשויה להיות יותר מאחת). עבור כל תשובה שהקפת, תארי דוגמת הרצה המובילה לתשובה זו.

a) יתכנו 2 חברים מקבוצות שונות בתוך החדר ביחד, עקב race condition למשאב משותף.
b) יתכן מצב שחבר קבוצה כלשהי לא נכנס לחדר למרות כלל הכניסה שמתיר זאת, עקב race condition למשאב משותף.
c) קיימת בעיית DeadLock / Livelock בקוד.
d) יתכן מצב שחבר קבוצה כלשהי יחכה למרות כלל הכניסה שמתיר זאת, כאשר אין race condition למשאב משותף.

<u>נימוק:</u>

- יתכנו 2 חברים מקבוצות שונות בתוך החדר ביחד, עקב race condition למשאב משותף (members): נתאר את התרחיש הבא:
  - onArrival(0) נכנס איש מקבוצה 0 לחדר אשר ריק (נניח תחילת ריצת המערכת) נועל את המנעול, מעדכן בשורת קוד 10 את ערך members[0]=1 ומשחרר את המנעול.
  - onLeave(0) אותו איש טוב משורה מקודם החליט לצאת מהחדר, תופסת את המנעול השני (m_leave) ומאכסן ב-tmp=1 שזה אכן מספר האנשים כעת מהקבוצה בחדר. יש context switch לתהליך הבא:
  - onArrival(0) נכנס איש נוסף מקבוצה 0 לחדר. הוא נועל עם מנעול arrival אין אף אחד מקבוצה שנייה בחדר ולכן הוא עובר לשורה 9, מעדכן tmp=1 ובשורה 10 מעדכן members[0]=2 משחרר את המנעול ונכנס לחדר.
  - חוזרים חזרה לחוט מנקודה 2 שכעת מעדכן members[0]=0 בשורה 16 ויוצא מהחדר.

- מגיע בוב מקבוצה 1 תופס את מנעול arrival ושואל "האם יש מישהו מקבוצה 0 בחדר?" בשורת קוד 7, אך למרות שכן שמור ב-members[0] הערך 0 בגלל ה- race condition לכן נכנס לחדר בחופשיות.

- יתכן מצב שחבר קבוצה כלשהי לא נכנס לחדר למרות כלל הכניסה שמתיר זאת, עקב race condition למשאב משותף, נתאר את התרחיש הבא: באופן דומה לתיאור הקודם, עם שינוי קטן:
  - onArrival(0) נכנס איש מקבוצה אפס ומעדכן members[0]=1 יוצא מהפונקציה.
  - onArrival(0) נכנס איש נוסף מקבוצה 0 לחדר. tmp=1 כי יש איש מהקבוצה בחדר לפי משתנה members ואז יש context switch.
  - onLeave(0) האיש מנקודה אחד החליט לצאת מהחדר, בשלב זה המשאב המשותף members[0]=1 הוא יוצא ומעדכן אותו ל-0 בשורה 16.
  - חוזרים לחוט מנקודה שנייה, מעדכן כעת members[0]=tmp+1=2. כעת הוא מבצע onLeave(0) שבסופו אנו נשארים עם members[0]=1 על אף שהחדר ריק.
  - onArrival(1) מגיע בוב מקבוצה 1 תופס את מנעול arrival ושואל "האם יש מישהו מקבוצה 0 בחדר?" בשורת קוד 7, אך למרות שכן שמור ב-members[0] הערך 1 בגלל ה- race condition אין אף אחד בחדר ולכן הוא לא נכנס למרות שמותר לו על פי הכללים.

- יתכן מצב שחבר קבוצה כלשהי יחכה למרות כלל הכניסה שמתיר זאת, כאשר אין race condition למשאב משותף. דוגמת הריצה היא
  - onArrival(0) נכנס איש מקבוצה אפס ומעדכן members[0]=1 יוצא מהפונקציה.
  - onLeave(0) אותו איש לעיל רוצה כעת לצאת מהחדר, בזמן ריצת פונקציית העזיבה הוא מקבל context switch לחוט הבא לפני ריצת שורד קוד 16 (עדכון members)
  - onArrival(1) נכנס אדם מהקבוצה השנייה ותופס את המנעול (אפשרי כי שני מנעולים שונים להגעה ויציאה) הוא בודק בשורה 7 אם יש מישהו בחדר, מקבל שכן ונכנס ללולאה. לפני שמבצע cond_wait יש context switch נוסף חזרה לאיש הראשון
  - האם מסיים את ריצת onLeave אך מכיוון שהחוט השני לא נרשם עדיין לקבלת סיגנל, הסיגנל ששולח הולך לפח והוא יוצא מהחדר.
  - כעת החדר ריק והחוט השני (איש 1) תקועה כי לא מקבל סיגנל שיכול להכנס והוא מחכה.