Homework 3 Dry

{ Name: Roni Roitbord , ID: 313575599, email: roniro@campus.technion.ac.il

Name: Amit Gabay , ID: 206040768, email: amitg@campus.technion.ac.il }

שאלה 2 - 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.

התפקיד של פרוטוקול ARP הוא לתרגם כתבות IP לכתובת MAC, בהינתן שכתובות IP של מכונות אחרות ידועות (או ניתנות לגילוי באמצעות DNS), ע"י שידור נרחב של פקטת שאילתת ARP.

ב. **(4 נקודות)** איזה מידע הלקוח צריך לדעת על השרת לפני ההתחברות?

על הלקוח לדעת את שם ה-domain של השרת (בהנחה שהלקוח יכול להשתמש ב-DNS, אם לא אז יש לדעת את כתובת ה-IP) וכן את מספר הפורט שעליו השרת שהוא רוצה להתחבר אליו מאזין.

ג. **(4 נקודות)** איזה מידע הלקוח ידע על השרת אחרי ההתחברות?

ה-client מקבל חזרה socket שבאמצעותו ניתן לתקשר, socket זה מכיל את כתובת ה-IP של השרת, הפורט שעליו הוא מאזין, ואת פרוטוקול התקשורת שמאפיין את החיבור מול השרת הספציפי שאליו התחברנו.

ד. **(4 נקודות)** איזה מידע השרת צריך לדעת על הלקוח לפני ההתחברות?

נשים לב שבעוד שהלקוח חייב להיות מודע לקיומו של השרת ולדעת פרטים עליו מראש על מנת ליצור חיבור, השרת לא חייב לדעת אף מידע מראש על הלקוח, וכשהלקוח ישלח בקשת התחברות, השרת יאשר אותה ויקבל מהלקוח את הפרטים הנחוצים ליצירת החיבור (כתובת ה-IP של הלקוח ומספר הפורט).

ה. **(4 נקודות)** איזה מידע השרת ידע על הלקוח אחרי ההתחברות?

לאחר קבלת בקשת ההתחברות מהלקוח, נוצר socket שבאמצעותו ניתן לתקשר, socket זה מכיל את כתובת ה-IP של הלקוח, הפורט שעליו הוא מאזין, ואת פרוטוקול התקשורת שמאפיין את החיבור בין השניים.

ו. **(6 נקודות)** מה הבדל בין הפורט (port) שבשימוש השרת וזה של הלקוח. אין נבחר כל אחד מהם?

עבור פורט בשימוש השרת, פורט השרת (פורט המקור עבור השרת ופורט היעד עבור הלקוח) ב-socket שיווצר עבור כל חיבור יהיה אותו הפורט, בעוד שכל לקוח יכול להתחבר לאותו השרת, ובכל התחברות כזו, בין אם מאותו לקוח לבין אם מלקוח אחר, פורט הלקוח ב-socket שיווצר יהיה פורט ייחודי שונה.
כלומר – עבור לקוח יש האזנה על פורטים שונים פר connection בעוד שהשרת מאזין על אותו הפורט עבור כל ה-connections שנוצרים אליו.

ז. (6 נקודות) מה הבדל בין פרוטוקול TCP ו-UDP? הסבירו למה חלק מהאפליקציות מעדיפות TCP וחלק UDP.

ההבל הוא שפרוטוקול TCP מאפשר תקשורת אמינה בין שני תהליכים במכונות קצה ברשתות שונות (ברשת החיצונית), בעוד שפרוטוקול UDP מאפשר תקשורת לא אמינה בין שני תהליכים כנ"ל.
אפליקציות הדורשות תקשורת אמינה, כלומר **קבלת** הבקשה **פעם אחת** בדיוק **בסדר** בו היא נשלחה, כמו שליחת העברה כספית מאפליקציית הבנק, יעדיפו שימוש בפרוטוקול TCP, בעוד שאפליקציות שלא משנה להן "לאבד" פקטה או לקבל כמה פקטות לא בסדר בו הן נשלחו, כמו אפליקציית זום או אפליקציות VOIP למיניהן, יעדיפו להשתמש בפרוטוקול UDP.

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:
שמות מכונה נועדו להקל על בני אדם, יותר קל לזכור את השם "גוגל" מאשר את כתובת ה-IP של השרת שלהם, לכן כאשר משתמש מזין שם domain, המכונה תשלח שאילתת DNS לשרת ה-DNS (Domain Name Server), שרת ה-DNS יחפש את שם הדומיין ב-DB שלו כדי למצוא את כתובת ה-IP המתאימה לשם שהתקבל, ולאחר מכן יחזיר את כתובת ה-IP למכונה המבקשת, מה שיאפשר את יצירת החיבור עם השרת הנכון.

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:

יש מעט מאוד כתובות IP, כשפונים ל-Ethernet provider מקבלים מהם רק כתובת IP אחת, אך במקום שבו יותר ממכשיר אחד שרוצה לגשת לרשת, יש צורך לספק לכל אחד מהם מזהה יחודי – כתובת IP פנימית.
בהינתן מכשיר בעל יש כתובת פנימית IP1 שרוצה ליצור חיבור עם גוגל, אם הוא ישח פקטה מ-IP1, היא לא תוכל לצאת החוצה, מפני שאף מכשיר לא מכיר את ה-IP הזה בחוץ. לעומת זאת, לראוטר יש IP חיצוני - ip_public, הראוטר מחליף את הכתובת הפנימית ל-ip_public, מדבר עם גוגל, מקבל תשובה, ומעביר את התשובה ל-IP1. היציאה החוצה תמיד נעשית באמצעות IP חיצוני אחד ויחיד, אבל, בתוך הרשת הפנימית יש כמה כתובות IP שצריך עבור כל המכשירים, ה-NAT הוא בעצם הטבלה שיודעת לקשר בין תשובה שהתקבלה מבחוץ ל-IP הפנימי שביקש אותה.

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 זה שרוצה לקבל שירות מהשרת, ולכן על הלקוח להיות מודע לקיומו של השרת, ולדעת את הפרטים הנדרשים על מנת להתחבר אליו - שם ה-domain (או כתובת ה-IP) שלו ומספר הפורט שעליו הוא מאזין, בעוד שהשרת אינו חייב לדעת את פרטי הלקוח מראש, **הוא מקבל אותם מהלקוח** בחלק מפרטי בקשת ההתחברות של הלקוח.
נשים לב שהפורט הוא לאו דווקא קבוע לפי סוג האפליקציה, וישנם גם פורטים זמניים שמוקצים ע"י הקרנל, וכן המידע הנחוץ הוא קבוע ואינו תלוי במי יזם את החיבור.

שאלה 2 - סינכרון (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 מסויים 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).

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).

```
#include <pthread.h>

int members = 0;
mutex_t global;

void onLeave(int team) {
        mutex_lock(&global);
        members --;
        mutex_unlock(&global);
}
```

```
void onArrival(int team) {
        mutex_lock(&global);
        while (members > 0) {
                mutex_unlock(&global);
                sleep(10);
                mutex_lock(&global);
        }
        mutex_unlock(&global);
        members++;
}
```

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. 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).

נימוק:
c. הקוד משתמש ב-busy wait שפוגע בנצילות המעבד: נניח כי חבר קבוצה 0 מבצע כניסה לחדר, הוא קורא ל-onArrival , ומכיוון שאין אף אחד בחדר הוא מצליח לתפוס את המנעול ואינו נכנס ללולאה.
לכן, חבר קבוצה זה מצליח להיכנס לחדר מקדם את מספר ה-members ב-1, ומשחרר את המנעול.
כעת, נניח שחבר קבוצה 1 מעוניין להיכנס לחדר, הוא קורא ל-onArrival ותופס את המנעול, אך מכיוון שה-members גדול מ-0, הוא נכנס ללולאת המתנה שמבצעת:

- שחרור המנעול
- יוצא לשינה של 10 שניות
- נעילת המנעול

ושוב יכנס ללולאה שכן חבר קבוצה 0 עדיין לא יצא מהחדר.
פעולה זו חוזרת חלילה עד שחבר קבוצה 0 יצא מהחדר, וזהו בדיוק מצב של busy-wait שפוגע בנצילות המעבד, שהרי כל 10 שניות חבר קבוצה 1 (כלומר, החוט שהוא מייצג) משתמש במעבד, רוב הפעמים ללא צורך – מפני שחבר קבוצה 0 עדיין בחדר, ותנאי הלולאה עדיין מתקיים.
d. הקוד מפר את כלל הכניסה לחדר, אומנם חבר קבוצה 0 וחבר קבוצה 1 אכן לא יכולים להימצא בחדר בו זמנית, אך גם שני חברים מאותה קבוצה אינם יכולים להימצא באותו חדר יחד, מה שמפר את הכלל.
למשל: נניח כי חבר קבוצה 0 מבצע כניסה לחדר, הוא קורא ל-onArrival , ומכיוון שאין אף אחד בחדר הוא מצליח לתפוס את המנעול ואינו נכנס ללולאה.
לכן, חבר קבוצה זה מצליח להיכנס לחדר, מקדם את מספר ה-members ב-1, ומשחרר את המנעול.
כעת, נניח שחבר קבוצה 0 נוסף מעוניין להיכנס לחדר, הוא קורא ל-onArrival ותופס את המנעול, אך מכיוון שה-members גדול מ-0, הוא נכנס ללולאת המתנה, וזאת למרות שמותר לו לשהות באותו חדר עם חבר הקבוצה שלו.

המימוש של כניסה ויציאה שונה כך שישתמש במשתני תנאי:

```
int members[2] = {0};     // 2 counters
cond_t conds[2];          // 2 condition variables
mutex_t global;
void onArrival(int team) {
        mutex_lock(&global);
        int other = team? 0 : 1;
        while(members[other] > 0)
                cond_wait(&conds[team] , &global);
        members [team]++;
        mutex_unlock(&global);
}
void onLeave(int team) {
        mutex_lock(&global);
        members [team]--;
        int other = team? 0 : 1;
        cond_signal(&conds[other]);
        mutex_unlock(&global);
}
```

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

1. (7 נק') הסבירי את טענתו של עומר באמצעות דוגמת ריצה קונקרטית.

נניח כי חבר קבוצה 0 מבצע כניסה לחדר, הוא קורא ל-onArrival , ומכיוון שאין אף אחד בחדר הוא מצליח לתפוס את המנעול ואינו נכנס ללולאה.
לכן, חבר קבוצה זה מצליח להיכנס לחדר מקדם את members[0] ב-1, ומשחרר את המנעול.
כעת, נניח שחבר קבוצה 0 נוסף מעוניין להיכנס לחדר, הוא קורא ל-onArrival ותופס את המנעול, ומכיוון שאף אחד מקבוצה 1 לא נמצא בחדר, members[1]=0 ולכן גם הוא נכנס לחדר, מקדם את members[0] ב-1, ומשחרר את המנעול.
לאחר מכן, נניח שחבר קבוצה 1 מעוניין להיכנס לחדר, אך מכיוון ש-members[0]=2>0, הוא יוצא ההמתנה.
כעת, חבר קבוצה 0 הראשון שנכנס מעוניין לצאת מהחדר, הוא קורא ל-onLeave, תופס את המנעול, מוריד 1 מ-members[0], ושולח סיגנל לחוטים של חברי קבוצה 1, מה שיעיר את חברי קבוצה 1, שבודק את תנאי הלולאה (שלא מתקיים, מפני שעדיין יש חבר אחד מקבוצה 0 בחדר), וחוזר שוב להמתנה.
לכן, החוט של חבר קבוצה 1 מתעורר שלא לצורך, ומיד חוזר להמתנה.

1. (8 נק') כיצד ניתן לתקן את הבעיה שהציג עומר בסעיף הקודם?

ניתן לתקן את הבעיה הנ"ל ע"י הוספת השורה הבאה בין שורות 15 ו-16: $if(members[team] \leq 0)$.
באופן זה, רק אם אין בחדר חברים מקבוצה team שהתקבל כפרמטר לפונקציה onLeave, ישלח סיגנל לחברי הקבוצה השנייה, ולפי הכלל הם אכן יוכלו להיכנס לחדר ולא יתעוררו לחינם.

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.

נימוק:

a. חבר קבוצה 0 נכנס לחדר ע"י הרצת הפונקציה onArrivel במלואה: הוא תופס את המנעול m_arrival , מעדכן את הערך של members[0] להיות 1 ומשחרר את המנעול שתפס.
כעת, חבר הקבוצה 0 שנכנס מעוניין לצאת מהחדר והוא מבצע הרצה של הפונקציה onLeave באופן חלקי: הוא תופס את המנעול, m_leave ומעדכן את הערך של tmp להיות 1.
בסיום העדכון של המשתנה tmp, מתבצעת החלפת הקשר לחבר אחר מקבוצה 0 שנכנס לחדר ע"י הרצת הפונקציה onArrival = במלואה: הוא תופס את המנעול m_arrival (המנעול הזה פנוי! המנוי שתפוס ע"י התהליך שיצא להמתנה הוא m_leave), מעדכן את הערך של members[0] להיות 2 ומשחרר את המנעול.
כעת התהליך שיצא להמתנה חוזר לריצה וממשיך מאיפה שעצר: מעדכן את members[0] להיות 0 ומסיים.
כעת הערך שיש ב-members אינו משקף את מספר האנשים האמיתי מקבוצה 0 שנמצאים בחדר, מפני שעדיין נמצא אדם מקבוצה 0 בחדר.

נניח שעכשיו חבר מקבוצה 1 מעוניין להיכנס לחדר, הוא מריץ את הפונקציה onArrival במלואה, ובגלל שהמשתנה members[0] מעודכן לערך הלא נכון – 0 , הוא מצליח להיכנס.

כעת נמצאים יחד בחדר חבר מקבוצה 0 וחבר מקבוצה 1 בעקבות race condition למערך members, ובפרט למשתנה members[0].

b. חבר קבוצה 0 נכנס לחדר ע"י הרצת הפונקציה onArrivel במלואה: הוא תופס את המנעול m_arrival , מעדכן את הערך של members[0] להיות 1 ומשחרר את המנעול שתפס.

כעת, חבר קבוצה 0 נוסף נכנס לחדר ע"י הרצת הפונקציה onArrivel באופן חלקי: הוא תופס את המנעול m_arrival, ומעדכן את הערך של tmp להיות 1.

בסיום העדכון של המשתנה tmp, מתבצעת החלפת הקשר לחבר הראשון מקבוצה 0 שנכנס וכעת מעוניין לצאת מהחדר. הוא מבצע הרצה של הפונקציה onLeave במלואה: הוא תופס את המנעול m_leave ומעדכן את הערך של members[0] להיות 0 ומשחרר את המנעול.

כעת, החבר שיצא להמתנה חוזר לריצה, מעדכן את הערך של members[0] להיות 2 ומשחרר את המנעול.

לאחר מכן, אותו חבר מעוניין לבצע יציאה מהחדר, ומפעיל את הפונקציה onLeave במלואה: הוא תופס את המנעול m_leave ומעדכן את הערך של members[0] להיות 1 ומשחרר את המנעול.

נניח כעת שחבר מקבוצה1 מבצע כניסה לחדר, הוא תופס את המנעול אבל הוא לא יכול להיכנס, מפני ש-members[0]=1, למרות שבפועל אין אף אחד בחדר.

לכן, חבר קבוצה זה לא נכנס לחדר למרות שכלל הכניסה מתיר זאת (החדר ריק), עקב race condition למערך members, ובפרט למשתנה members[0].

d. חבר קבוצה 0 נכנס לחדר ע"י הרצת הפונקציה onArrivel במלואה: הוא תופס את המנעול m_arrival , מעדכן את הערך של members[0] להיות 1 ומשחרר את המנעול שתפס.

כעת, חבר קבוצה 1 נוסף נכנס לחדר ע"י הרצת הפונקציה onArrivel באופן חלקי: הוא תופס את המנעול m_arrival, ומגיע עד ללולאת ה-while.

בסיום בדיקת התנאי של לולאת ה-while, מתבצעת החלפת הקשר לחבר מקבוצה 0 שנכנס וכעת מעוניין לצאת מהחדר, הוא מבצע הרצה של הפונקציה onLeave במלואה: הוא תופס את המנעול m_leave ומעדכן את הערך של members[0] להיות 0, שולח סיגנל לחברי הקבוצה השנייה ומשחרר את המנעול.

כעת, החבר שיצא להמתנה חוזר לריצה, נכנס להמתנה (שכן התנאי בלולאה התקיים לפני ביצוע החלפת ההקשר), ויישאר בהמתנה כי הסיגנל שחבר קבוצה 0 שלח לו לא הגיע עליו.