

Homework 3 Dry

סטודנטים מגישים:

אבו מוך ראמי – 324276765

ramiab@campus.technion.ac.il

עיאשי מוחמד – 213014061

muhammeda@campus.technion.ac.il

שאלה 1 - Networking - תקשורת (52 נק')

א. (4 נקודות) הסבירו מה תפקיד של פרוטוקול ARP.

התפקיד של ARP היא למצוא את ה MAC Address של מכונת היעד אם יודעים רק את ה IP Address שלו, אם התקן היעד אינו ברשת המקומית אז מחזירים את ה MAC Address של ה first-hop router.

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

את ה IP Address (או host name שניתן לתרגם אותו לכתובת IP באמצעות DNS) ואת ה Port של השרת.

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

את ה IP Address ואת ה Port של השרת.

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

כלום, השרת ידע מי שולח לו אחרי ההתחברות ולא לפני.

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

את ה IP Address ואת ה Port של הלקוח.

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

ההבדל ביניהם הוא שהפורט (port) של השרת קבוע וידוע מראש כך שהשרת מאזין לקשרים הנכנסות לעומת פורט (port) הלקוח שנבחר בצורה אקראית על ידי מערכת ההפעלה.

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

פרוטוקול UDP פשוט יותר, מהיר יותר אבל לא מבטיח שכל המידע יגיע באופן תקין ומסודר לעומת פרוטוקול TCP שמבטיח ודואג שכל הפאקטות מגיעות באופן תקין. ולכן אפליקציה שאכפת לה מאובדן מידע תעדיף TCP. ואפליקציה שמעדיפה רק מהירות תעדיף UDP.

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

- לשלוח פקטות (frame) מחשבי קצה בתוך אותה רשת (LAN connectivity).
- לתרגם כתובת IP לכתובת MAC.
- לתרגם שם השרת לכתובת IP.
- לתרגם שם השרת לכתובת MAC.
- לשלוח פקטות בין מחשבי קצה ברשתות שונות (WAN).
- לאפשר תקשורת בין שני תהליכים במחשבי קצה ברשתות שונות (WAN).

נימוק:

למען סיפוק פשטות למשתמש ה DNS מתרגם את שם השרת לכתובת IP.

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:

פרוטוקול NAT מספק פטרון לניהול ואופטימיזציה לכתובות IP של כל המכשירים שמחוברים לאותה רשת מקומית.

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 צריך לדעת את ה domain name שניתן לתרגם אותו לכתובת IP באמצעות DNS ואת ה Port של השרת, לעומת השרת הוא לא חייב לדעת כלום לפני החיבור.

שאלה 2 - סינכרון (48 נק')

לאחר הריב המתקשר של נוגה(מוכרת בעיקר על ידי השיר שלה, "חד קרנל") ודנה (המוכרת בעיקר על ידי השיר שלה, "צא להמתנה"), עולם הפופ הישראלי התחלק לשתי קבוצות, קבוצת נוגה וקבוצת דנה. 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).

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 דנה וקבוצת נוגה, בהתאמה.
- (הניחו שאמצעי הסנכרון עברו אתחול תקין והתעלמו מבעיות קומפילציה אם ישנן, שכן מטרת השאלה אינה לבדוק שגיאות אתחול/תחביר).

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

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 – במקרה שיש מעבד יחיד במחשב, אם יש שני חוטים כך שהראשון נכנס לפונקציה OnArrival ומצליח, ואז החט השני נכנס לאותה פונקציה ובפעם הראשונה שנכנס ללולאה ועושה Sleep ומיד קורת החלפת הקשר לחוט הראשון כשמבצע OnLeave אז החוט השני יצטרך להמתין את כל עשר השניות בכוח עד שיוכל להמשיך את פונקציית OnArrival ולכן זה ניצול זמן מעבד מיותר ולכן זה לכאורה busy waiting.

d – ננחל כי חוט ראדון מקבוצה מספר 0 נכנס לפונקציה OnArrival(0) וסיים בהצלחה כלומר נכנס לחדר, ואז חוט שני מקבוצה מספר 0 נכנס גם לפונקציה OnArrival(0) מבלי שייצא החוט הראשון מהחדר, אז הוא נכנס ללולאה אינסופית, יושן 10 שניות וחוזר ללולאה עוד פעם וכו' ואז לא יוכלו 2 חברים מאותה קבוצה לנכנס לחדר באותו זמן.

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_broadcast(&conds[other]);
17     mutex_unlock(&global);
18 }
```

אך נדב (יושב ראש מדמ"ח) טען שקוד זה גורם לחוטים להתעורר שלא לצורך ומיד לחזור למצב המתנה.
1. (7 נק') הסבירי את טענתו של נדב באמצעות דוגמת ריצה קונקרטית.

אם מבצעים OnArrival(1) 5 פעמים אחת אחרי השניה, ואז מנסים לבצע OnArrival(0) אז הוא ייתקע בתוך ה- while (הוא יחכה לחדר להתרוקן), בכל פעם שמישהו מבצע OnLeave(1) הוא ישלח סיגנל להעיר את כל החוטים שבצעו OnArrival(0) שזה במקרה שלנו חוט אחד, אבל החדר עדיין יש בו חוטים מקבוצה 1 ולכן הוא יתעורר 5 פעמים (ויחזור לחכות ב 4 מהם), למרות שהוא צריך להתעורר פעם אחת.

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

```
if(members[team] == 0){
    cond_broadcast(&conds[other]);
}
```

במקום שורה 16 בקוד הנ"ל נוסיף את הקוד משמאל. לפני שמעירים את כל החוטים הממתנים אנו בודקים אם התפנה החדר מחברי הקבוצה האחרת ואז לא גורמים לחוטים להתעורר ללא צורך ומיד לחזור למצב המתנה, כי אפשר לנכנס לחדר רק אחרי שכל חברי הקבוצה האחרת עזבו ורק אז מעירים את כל החוטים הממתנים (מכניסים לחדר) מוסיפים את מספרם ומשחררים המנועל.

נדב 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):

```
1 int members [2] = {0};           // 2 counters
2 cond_t conds[2];                 // 2 condition variables
3 mutex_t m_arrival, m_leave;     // there are *2* locks now
4 void onArrival(int team){
5     mutex_lock(&m_arrival);
6     int other = team? 0 : 1;
7     while(members [other] > 0)
8         cond_wait(&conds[team] , &m_arrival);
9     int tmp = members [team];
10    members [team] = tmp + 1;
11    mutex_unlock(&m_arrival);
12 }
13 void onLeave(int team){
14     mutex_lock(&m_leave);
15     int tmp = members[team];
16     members [team] = tmp - 1;
17     int other = team? 0 : 1;
18     cond_broadcast(&conds[other]);
19     mutex_unlock(&m_leave);
20 }
```

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. סיגנלים עלולים ללכת לאיבוד.

נימוק:

תרחיש בעייתי ל-a:

מבצעים onArrival(1) ומסתיימת בהצלחה.
מבצעים onLeave(1), ומתבצעת החלפת הקשר אחרי שורה 15 (ערך tmp הוא 1).
מבצעים onArrival(1), ומסיימים ואז ערך members[1] יתעדכן להיות 2.
ממשיכה ה- onLeave(1) מאיפה שהפסיקה, ומעדכנת את ערך members[1] ל-0.
מבצעים onArrival(0) ומכניסה אותו כי $members[1] = 0$, למרות שיש חבר קבוצה 1 בחדר.

תרחיש בעייתי ל-b:

מבצעים 2 פעמים onArrival(1) שעובדות כמצופה.
מבצעים onLeave(1) ומתבצעת החלפת הקשר אחרי שורה 15 (ערך tmp הוא 2)
מבצעים onLeave(1) ומתבצעת החלפת הקשר אחרי שורה 15 (ערך tmp הוא 2)
ואז ממשיכים onArrival(0) מאיפה שעצרנו שלא ייכנס לעולם כי members[1] ערכו 2, למרות שאין חבר קבוצה 1
בחדר, כתוצאה מה-race condition בעדכון ה-members[1] ב-onLeave(0).

תרחיש בעייתי ל-d:

מבצעים onArrival(1) ומסיימים בהצלחה.
מבצעים onArrival(0) ומתבצעת החלפת הקשר אחרי שורה 7 (אחרי בדיקת תנאי ה-while)
מבצעים onLeave(1) ומסתיימת בהצלחה.
חוזרים ל-onArrival(0) מאיפה שעצרנו ונמשיך לחכות לסיגנל למרות שהוא כבר נשלח ונעלם.