Operating Systems – 234123

Homework Exercise 3 – Dry

Submitters:

Tom Guy

315155671

[tom.guy@campus.technion.ac.il](mailto:tom.guy@campus.technion.ac.il)

Nikita Ivanov

322245101

[Nikitaiv@campus.technion.ac.il](mailto:Nikitaiv@campus.technion.ac.il)

# **שאלה 2 - Networking - תקשורת (52 נק')**

**חלק פתוח:** הסבירו בקצרה (לא יותר מ1-2 שורות!)

1. **(4 נקודות)** הסבירו מה תפקיד של פרוטוקול ARP?

**תשובה:** פרוטוקול המשמש למציאת כתובת MAC לפי כתובת IP. משתמש א' משדר ARP query packet המכיל את כתובת ה-IP של משתמש ב' ומקבל בחזרה בתגובה ממנו את כתובת ה-MAC שלו.

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

**תשובה:** הלקוח צריך לדעת מה ה-port עליו מאזין השרת ובנוסף הוא צריך לדעת את כתובת ה-IP של השרת.

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

**תשובה:** לאחר ההתחברות, הלקוח צריך לדעת בנוסף לאיזה port שלו מערכת ההפעלה יידעה את ההתקשרות ואת ה-protocol (UDP\TCP) בו התקשורת תתבצע (כמובן שהוא יודע את כתובת ה-(IP.

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

**תשובה:** השרת לא צריך לדעת דבר על הלקוח לפני תחילת התקשורת.

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

**תשובה:** השרת צריך לדעת באיזה פרוטוקול תקשורת רוצה הלקוח להשתמש (UDP\TCP), באיזה port שלו ושל הלקוח תתבצע התקשורת ומה כתובת ה-IP של הלקוח (את כתובת ה-IP שלו הוא כמובן יודע

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

**תשובה:** port ההתחברות של הלקוח נבחר ע"י מערכת ההפעלה בעת יצירת תקשורת עם השרת בעת ה-port של השרת נקבע בעת יצירתו והוא תמיד מאזין עליו.

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

**תשובה:** פרוטוקול TCP דואג שהמידע שנשלח בפקטות מגיע לפי הסדר ובשלמותו בעוד שפרוטוקול UDP לא דואג לדבר ורק שולח את המידע. אפליקציה שחשוב לה שהמידע יגיע בזמן ולפי הסדר (בנק לדוגמה) תשתמש בפרוטוקול TCP כי לאיבוד מידע יש השלכות משמעותיות בעוד שאפליקציות אחרות יכולות להעדיף UDP כי הוא מהיר יותר (כשהמידע אכן מגיע) ואמינות פחות חשובה מביצועים (אפליקציות כמו VoIP).

**חלק אמריקאי:**

**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**:

**תשובה:** לפי המצגת הרצאה 8, שמות הכתובות הן לשימוש בני אדם ותרגומן לכתובות IP מתבצע על ידי פרוטוקול DNS.

**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**:

**תשובה:** בגלל שעם הפרוטוקול IPv4 יש רק עד כ-4 מיליארד כתובות IP אפשריות וכיום יש צורך ביותר, כל ראוטר "מאחד" תחתיו מספר מכשירים תחת אותן כתובות IP עם פרוטוקול NAT ובכך "חוסך" כתובות IP.

**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**:

**תשובה:** כדי ליצור חיבור ראשוני של שרת ולקוח, הלקוח פונה אל השרת (ע"י ה-IP שלו המתקבל באמצעות פרוטוקול DNS מהשם של ה-domain) אל מספר הפורט עליו מאזין השרת (שהוא קבוע ולא משתנה). השרת מאזין וממתין לפנייה של כל אחד על מספר הפורט הקבוע שלו ואינו חייב לדעת דבר על הלקוח לפני ההתחברות. כמובן שאחריה הוא צריך לדעת מה כתובת ה-IP של הלקוח ועל איזה פורט הוא מאזין.

שאלה 2 - סינכרון (48 נק')

המצאת המושג "פקולטה נחשבת" החמירה את הסכסוך בין הסטודנטים במדמ"ח ובהנדסת חשמל, ולכן הוגדר כי כאשר סטודנט מאחת הפקולטות רוצה להיכנס לחדר מסויים עליו לציית לכלל הבא:אם יש סטודנטים מפקולטה אחרת בחדר אזי אסור לסטודנט להיכנס ועליו להמתין עד שיעזבו (לעומת זאת, מספר סטודנטים מאותה פקולטה יכולים לשהות בחדר באותו הזמן).

סמני נכון / לא נכון (אין צורך להסביר):

1. **(3 נק')** יכולים להיות שני סטודנטים מפקולטות שונות באותו חדר במקביל: **נכון / לא נכון**

הסבר: race condition – **יכול להיגרם מצב שבו שניים מגיעים במקביל, שניהם רואים שהחדר ריק ונכנסים יחד.**

1. **(3 נק')** יכולים להיות שני סטודנטים מפקולטות זהות בחדר במקביל: **נכון / לא נכון**
2. **(3 נק')** סטודנטי פקולטה אחת עלולים להרעיב (כניסת) סטודנטי פקולטה אחרת: **נכון / לא נכון**

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

* כל חוט מייצג סטודנט.
* בכניסה לחדר הסטודנט קורא ל (onArrival(int faculty, שמקבלת את פקולטת הסטודנט.
* ביציאה מהחדר הסטודנט קורא ל (onLeave(int faculty שמקבלת את פקולטת הסטודנט.
* הערכים 0 ו-1 של faculty מייצגים את הפקולטה להנדסת חשמל ומדמ״ח, בהתאמה.
* (הניחו שאמצעי הסנכרון עברו אתחול תקין והתעלמו מבעיות קומפילציה אם ישנן, שכן מטרת השאלה אינה לבדוק שגיאות אתחול/תחביר).

|  |  |  |  |
| --- | --- | --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10. | #include <pthread.h>  int students = 0;  mutex\_t global;  void **onLeave**(int faculty) {  mutex\_lock(&global);  students--;  mutex\_unlock(&global);  } | 11.  12.  13.  14.  15.  16.  17.  18.  19.  20. | void **onArrival**(int faculty) {  mutex\_lock(&global);  while (students > 0) {  mutex\_unlock(&global);  sleep(10);  mutex\_lock(&global);  }  students++;  mutex\_unlock(&global);  } |

1. **(12 נק')** בהתייחס לקוד הנ״ל, הקיפי אתכל התשובות הנכונות (עשויה להיות יותר מאחת).  
   עבור כל תשובה שהקפת, תארי דוגמת הרצההמובילה לתשובה זו.

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

נימוק:

**תשובה:** לכל מקרה נתאר תרחיש אחר.

תשובה ד': אין כלל התייחסות לאיזה פקולטה שייך כל סטודנט, אך בכל מקרה יכול להיות סטודנט יחיד בכל חדר בכל זמן – לא מתאפשר כניסת שני סטודנטים מאותה הפקולטה לאותו החדר. לדוגמה, סטודנט ראשון ממדמ"ח מגיע לחדר המונה מתעדכן. בעת הגעת הסטודנט השני, הוא לא יכול להיכנס עד שהראשון לא יוצא. דוגמת הרצה:

onArrival(1); // A student from cs enter the room.

onArrival(1); // Another student from cs want to enter the room but he can’t (condition in line 13 stands).

המימוש של כניסה ויציאה שונה כך שישתמש במשתני תנאי:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | int students[2] = {0}; // 2 counters  cond\_t conds[2]; // 2 condition variables  mutex\_t global;  void **onArrival**(int faculty) {  mutex\_lock(&global);  int other = faculty ? 0 : 1;  while(students[other] > 0)  cond\_wait(&conds[faculty] , &global);  students[faculty]++;  mutex\_unlock(&global);  }  void **onLeave**(int faculty) {  mutex\_lock(&global);  students[faculty]--;  int other = faculty ? 0 : 1;  cond\_broadcast(&conds[other]);  mutex\_unlock(&global);  } |

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

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

**תשובה:** דוגמה: נניח ושני סטודנטים מחשמל הגיעו אחד אחרי השני, המונה של סטודנטים בחשמל הוא כעת על 2. לאחר מכן הגיעו שלושה סטודנטים ממדמ"ח שניסו להיכנס ונכנסו להמתנה כי היה סטודנטים מחשמל בפנים. בשלב מסוים אחד הסטודנטים מחשמל יצא, הקטין את המונה ל-1 ואז בשורה 16 שידר לסטודנטים ממדמ"ח שהוא יצא. הם כולם חזרו מהמתנה וניסו להיכנס שוב אך מכיוון שהיה עוד סטודנט מחשמל בפנים הם לא הצליחו לנעול את המנעול וחזרו להמתנה. דוגמת הרצה:

onArrival(0); // First student from ee enters.

onArrival(0); // Second student from ee enters.

onArrival(1); onArrival(1); onArrival(1) // Three students from cs try to enter but fail.

onLeave(0); // One of the students from ee leaves the room. When the code reach line 16 all three students from cs woke up and try to enter again but fail because there is one more student from ee inside.

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

**תשובה:** כדי לתקן את הממתנה יש להוסיף לפני שורה 16 את שורה הבאה:

If(students[faculty] == 0)

לאחר מכן, רק אם לא נשארו סטודנטים מהפקולטה שהייתה בפנים נעיר את הסטודנטים מהפקולטה השנייה.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | int students[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 faculty){  mutex\_lock**(&m\_arrival);**  int other = faculty ? 0 : 1;  while(students[other] > 0)  cond\_wait(&conds[faculty] , **&m\_arrival**);  **int tmp = students[faculty];**  **students[faculty] = tmp + 1;**  mutex\_unlock**(&m\_arrival);**  }  void onLeave(int faculty){  mutex\_lock**(&m\_leave);**  **int tmp = students[faculty];**  **students[faculty] = tmp – 1;**  int other = faculty ? 0 : 1;  cond\_broadcast(&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.
2. יתכנו 2 סטודנטים מפקולטות שונות בתוך החדר ביחד, עקב race condition למשאב משותף.
3. יתכן סטודנט שלא נכנס לחדר למרות כלל הכניסה שמתיר זאת, עקב race condition למשאב משותף.
4. קיימת בעיית DeadLock / Livelock בקוד.
5. סיגנלים עלולים ללכת לאיבוד.

נימוק:

**תשובה:** נתאר לכל מקרה בנפרד. נתייחס לקטע הקוד הבא בשני המקרים:

onArrival(1);

onArrival(1); onLeave(1);

onArrival(0);

onLeave(1)

כאשר הקוד בשני הצדדים רץ בשני חוטים שונים. בשני המקרים הבאים נתייחס להחלפת הקשר שבוצעה בין שני התהליכים בין הקריאה השנייה ל-onArrival(1) בין שורות 9-10 ובין הקריאה ל-onLeave בין שורות 15-16.

כמו כן, למקרה ב' נתייחס גם לשורה onLeave(1) המצורפת בצד.

**מקרה א':** הפקודה בשורה 16 בוצעה אחרי השורה בפוקדה 10. בגלל שבעת יציאה להחלפת ההקשר, temp = 1 כי היה סטודנט אחד בפנים, בעת החזרה מהחלפת ההקשר students[1] = 2 אך אחרי שורה 16 יתעדכן ל-students[1] = 0 למרות שבעצם יש סטודנט ממדמ"ח בפנים. לאחר מכן, הסטודנט מחשמל יראה ב-onArrival(0) כי הוא יכול להיכנס לחדר ויכנס. כך יהיו שני סטודנטים מפקולטות שונות בחדר באותו זמן.

**מקרה ב':** במקרה זה נניח כי סידור הביצוע היה הפוך – הפקודה בשורה 16 בוצעה לפני השורה בפקודה 10. כך יתעדכן אחרי ביצוע הפקודה בשורה 10 נקבל students[1] = 2. לאחר מכן, כאשר תבוצע הפקודה onLeave(1), כלומר שהסטודנט היחיד שנשאר ממדמ"ח יעזוב, החדר יהיה ריק אך בעת onArrival(0) הכניסה של הסטודנט מחשמל לא תתאפשר כי students[1] = 1 > 0.  
**מקרה ד':** נתייחס לקטע הקוד הבא:

onArrival(1); onArrival(0);

onLeave(0);

נתייחס לכל צד כתהליך (חוט) שונה, ולהחלפת הקשר המבוצעת בין הפקודה onLeave(0) לבין הפקודה onArrival(1) בין השורות 7-8 ונניח כי שורה 7 מבוצעת, לאחר מכן החלפת הקשר לפונקציה onLeave(0) ואז חזרה לשורה 8.

בהתחלה נכנס סטודנט יחיד מחשמל לחדר ואז students[0] = 1. לאחר מכן, כאשר פונקציה onArrival(1) מתבצעת students[0] = 1 > 0 ונעבור את התנאי בשורה 7 ונצא להחלפת הקשר. אז פונקציה onLeave(0) מתבצעת, בשורה 18 נשלח הסיגנל הרלוונטי שלא מגיע לאף אחד, students[0] = 0 והפונקציה מסתיימת. אז מתבצעת החלפת הקשר חזרה לשורה 8 והתהליך יוצא להמתנה. הוא פספס את הסיגנל שהיה אמור להחזיר אותו מההמתנה כי כעת students[0] = 0.