TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

רשתות תקשורת מחשבים
סמסטר א' 2020/21

תרגיל בית מעשי מס' 1

להגשה עד 4.11.2020

שימו לב: באתר הקורס תוכלו למצוא מסמך עם הנחיות מפורטות לגבי הגשת תרגילים מעשיים. על התרגילים המעשיים לעמוד בהנחיות הללו .

מטרות התרגיל

- מימוש של פרוטוקול.
- שימוש נכון בsocket programming.

מבוא

בתרגיל זה נתכנן פרוטוקול אפליקציה ונממש אפליקציית רשת במבנה שרת/לקוח. האפליקציה תממש גרסה פשוטה של המשחק Nim:

http://en.wikipedia.org/wiki/Nim

פתרון התרגיל מורכב משתי תוכנות. הלקוח (client) – מאפשר משחק אינטראקטיבי לשחקן בודד .

השרת (server) – מנהל את המשחק, ומשחק מול השחקן שמשתמש בתוכנת הלקוח.

בתרגיל מעשי מספר 2 נרחיב את האפליקציה כך שתתמוך במספר שחקנים.

המשחק

בסעיף זה נתאר את חוקי המשחק עבור גרסה פשוטה של Nim.

המשחק מיועד לשני משתתפים. הקלט למשחק הוא כמות הקוביות n_a, n_b, n_c של הערימות A,B,C . תוכלו להניח שגדלי הערימות הם מספרים שלמים בין 1 ל1000.

בכל תור לוקח אחד המשתתפים מס' קוביות (לפחות קובייה אחת) כרצונו מאחת הערימות. המנצח הוא האחרון שלוקח קוביות מהלוח.

להלן דוגמה למשחק עם הקלט $n_a = 4, n_b = 5, n_c = 6$.

A	B	C	
4	5	6	Alice takes 4 from A
0	5	6	Bob takes 4 from B
0	1	6	Alice takes 5 from C
0	1	1	Bob takes 1 from C
0	1	0	Alice takes 1 from B and wins.

הלקוח

תוכנית הלקוח מיועד לתקשורת בין השחקן לבין תוכנית השרת. הלוגיקה שבה מיועדת אך ורק לתקשורת עם המשתמש ועם תוכנית השרת. למשל, התוכנית לא "זוכרת" את גדלי הערימות של המשחק.

שורת הפקודה להרצה היא:

nim.py [hostname [port]]

כאשר hostname ו-port הם פרמטרים אופציונאליים. ערך ברירת המחדל הוא hostname = localhost, ו -port = 6444. לא ניתן לספק port ללא hostname. הפרמטר hostname יכול להיות שם או כתובת IP.

למשל:

nim.py nova.cs.tau.ac.il 45678

במהלך המשחק הלקוח מבצע את הצעדים הבאים אחד אחרי השני בלולאה, עד לסיום המשחק:

1. קבלת עדכון מהשרת לגבי גדלי הערימות הנוכחיים. יודפס:

Heap A: #

Heap B: #

Heap C: #

(כאשר # מוחלף בערכים המתאימים).

2. אם המשחק נגמר, הלקוח מקבל חיווי מהשרת לגבי זהות המנצח. יודפס:

Server win! –OR- You win!

אחרת, הלקוח מתבקש להזין את הצעד הבא שלו לconsole. במקרה זה יודפס:

Your turn:

הקלט הוא תו המייצג את הערימה הנבחרת (A, B, C) או Q עבור Quit ליציאה מהמשחק. אם הקלט אינו Quit התוכנית מקבלת גם מספר שלם המייצג את מספר הקוביות שברצון הלקוח להוריד מהערימה שבחר.

3. אם הקלט היה צעד- הודעה מתאימה תשלח לשרת.

במקרה שהקלט הוא Q, או שהודפס זהות המנצח- תוכנת הלקוח מסיימת.

4. הלקוח מקבל אישור מהשרת שהמהלך נקלט, או לחילופין התראה שהמהלך היה לא חוקי :

למשל הלקוח ניסה להוריד 5 קוביות מערימה שהיו בה רק 3 קוביות.

במקרה של מהלך לא חוקי, הלקוח מפסיד את התור, ולא מקבל הזדמנות להכניס קלט נוסף. יודפס:

Illegal move –OR- Move accepted

5. בכל שלב על הלקוח לזהות גם ניתוק מהשרת. במקרה כזה, המשחק מסתיים ומודפס
Disconnected from server

ההדפסות למסך חייבות להיות בדיוק כפי שמתואר בסעיף זה.

השרת

השרת מנהל משחק בודד. השרת "זוכר" את גדלי הערימות, מקבל צעדי משחק מהלקוח, ומשחק אחרי כל צעד של הלקוח.

שורת הפקודה להרצה היא:

nim-server.py n_a n_b n_c [port]

כאשר n_a,n_b,n_c הם הפרמטרים למשחק המתוארים לעיל.

הפורט להקשבה הוא פרמטר אופציונלי עם ערך ברירת מחדל 6444 .

למשל:

nim-server.py 8 4 5 45678

לאחר שאותחל, השרת מציב את גדלי הערימות התחיליים ומחכה להתחברות מהלקוח .

לאחר התחברות המשחק מתחיל. השרת מגיב להודעות מהלקוח כפי שפורט בסעיף הקודם.

אחרי תורו של הלקוח, אם המשחק לא הסתיים מגיע תורו של השרת.

שימו לב: על השרת גם לשרת לקוחות מאותו המחשב, וגם לשרת לקוחות ממחשבים שונים.

לוגיקה של השרת

הלוגיקה של השרת תהיה פשוטה למימוש:

בכל שלב השרת יוריד קובייה אחת מהערימה עם הכי הרבה קוביות.

אם יש כמה ערימות עם אותו מספר של קוביות- נוריד קובייה מהערימה עם האינדקס הנמוך ביותר.

תוכנית השרת אינה מסיימת את ריצתה. לאחר שהלקוח הודיע לשרת על סיום השימוש- השרת סוגר את הsocket המתאים, וממתין ללקוחות חדשים להתחבר.

דוגמת ריצה

בצד השרת:

nim-server.py 2 3 4

בצד הלקוח (בפונט ירוק – קלט מהמשתמש):

nim.py

Heap A: 2

Heap B: 3

Heap C: 4

Your turn:

A 2

Move accepted

Heap A: 0

Heap B: 3

Heap C: 3

Your turn:

B 4

Illegal move

Heap A: 0

Heap B: 2

Heap C: 3

Your turn:

B 2

Move accepted

Heap A: 0

Heap B: 0

Heap C: 2

Your turn:

C 2

Move accepted

Heap A: 0

Heap B: 0

Heap C: 0

You win!

בונוס (5 נקודות)

בפרוטוקול TCP ישנה בעיה של שליחת ההודעה האחרונה בתקשורת בין שני המחשבים.

אם משתמשים בפקודה send ולאחר מכאן בפקודת close, פקודת close תסגור (לאחר זמן מה) את התקשרות. יתכן שלא יתקבל כל המידע אצל הצד השני, ולא יהיה ניתן לשלוח מחדש את המידע (עקב סגירת התקשרות).

במאמר הבא מתואר הבעיה, עם פתרון מתאים.

http://blog.netherlabs.nl/articles/2009/01/18/the-ultimate-so_linger-page-or-why-is-my-tcp-not-reliable

אנא מימשו את פתרון המתואר במאמר (שימוש בshutdown, ובפונקציה recv לאחר מכאן). הקוד בבלוג כתוב בשפת C, אך ניתן לבצע התאמות לפייתון.

נ.ב- על מנת להבין את המאמר לעומקו- הייתי ממליץ שתקראו בנספח את המצגות המתאימות על ההבדלים בין פקטות RST לFIN.

דרישות התרגיל

ראשית, עליכם לתכנן פרוטוקול אפליקציה מתאים שיעבוד מעל TCP. לאחר מכן, ממשו אותו כפי שנלמד

בתרגול. האפליקציה סינכרונית, כלומר אפשר להשתמש בפקודות חוסמות.

את פרוטוקול האפליקציה יש לתעד בבירור, באופן שיאפשר לכל אדם לממש לקוח או שרת ש"ידברו" עם

התוכנות שהגשתם.

על הפרוטוקול להיות בינארי, ועליכם להשתמש בפונקציית struct.pack .

הדגש בבדיקת הקוד שתגישו ינתן כמובן למימוש התקשורת ברשת, כך שעל השרת לשאת לקוחות ממחשבים שונים. על המימוש להיות יעיל ורובוסטי (robust).

אל תשכחו לבדוק שגיאות בערכי חזרה מפונקציות ולטפל בהם בהתאם.

ייתכן שזוגות רבים יבדקו את הפתרון שלהם על nova בו-זמנית. לכן, מומלץ להשתמש בפורט שרת שאינו פורט ברירת המחדל בעת בדיקת הקוד (אחרת, פונקציית bind תכשל).

בנוסף- אם הינכם מעבירים קבצים מוינדוס ללינוקס- עליכם לשנות את תו שורה חדשה "\r\n" לתו "\n". ניתן לעשות שינוי זה באמצעות notepad++.

בהצלחה :)