פרוייקט תכנות – רשתות ואיטנרנט

מגישים – נועם סולן, רונן רוזין

תאריך הגשה – 03.01.2019

גרסה – 1.0

שם פרוייקט – תרגיל רטוב 2

מס׳ קורס – 83-455-01

שפת תכנות – פייטון 3.6

תיאור הבעיה

בניית ציר תקשורת בין שתי תחנות, בעזרת שרת. התחנות מקבלות IP של תחנת היעד ושל השרת, והשרת צריך להעביר את ההודעה של תחנה A אל B, ולהחזיר הודעה הנשלחת מתחנה B אל A.

הנחות:

1. תחנות A ו-B יכולות לרוץ על אותה מכונה
2. M רצה על מכונה אחרת
3. ניתן להניח כי ההודעה הנשלחת היא לא ריקה

הגבלות:

1. ניתן להשתמש רק בספריות סטנדרטיות

ממשקים:

1. ממשק CLI – CMD בווידוס / Terminal ב-UNIX.
2. מקלדת (אנגלית בלבד)

אימפלמנטציה

אנחנו משתמשים בשני קבצים:

1. קובץ לשרת – קובץ זה מכיל את כתובת הIP של השרת
2. קובץ לתחנות – קובץ זה מכיל לוגיקה בלבד, והוא נכתב בצורה גנרית עבור שתי התחנות – קביעת התחנה נקבעת בהרצה על ידי פרמטר. גם כתובת הIP של תחנת היעד וכתובות הIP של השרת מועברות כפרמטרים בהרצה.

את הקוד אנחנו מריצים בCLI של מערכת ההפעלה. במחשבי Windows, שמנו לב שצריך לתת הרשאה לFirewall. ב-MacOS לא נתקלנו בכך.

כתיבת הקוד

את התוכנית שלנו חילקנו לפונקציות כדי שנוכל לבדוק תקלות ביתר קלות. בכל תכנית ישנם 4 סוגי פונקציות עיקריות:

1. Connections\_handler – פונקציה זו היא פונקציה היררכית המנהלת את סדר הקריאה לשאר הפונקציות. היא תאפשר לנו להריץ את התוכנה יותר מפעם אחת אם נרצה.
2. Send – פונקציה זו יוזמת חיבור אל התחנה המרוחקת.
3. Receive - פונקציה זו מאזינה לIP ולפורט המצוין (הפורטים קודדו בצורה קשיחה בקוד – בהתאם להוראות). בגלל שאנחנו לא יודעים לאיזה IP להאזין, אנחנו מזינים 0.0.0.0, כדי להאזין לכל הIP שהclient מחובר אליהם.

בClients, ישנה גם פונקציה שיוצרת הודעה, היא קיימת רק שם מכיוון שאין צורך שהשרת ייצור הודעות.

בשרת הפונקציות מחולקות קצת אחרת, אבל הרעיון הכללי דומה:

1. Connections\_handler – פונקציה זו היא פונקציה היררכית המנהלת את סדר הקריאה לשאר הפונקציות. היא תאפשר לנו להריץ את התוכנה יותר מפעם אחת אם נרצה.
2. Send\_receive – פונקציה זו יוזמת חיבור לתחנה המרוחקת ושולחת את ההודעה שהשרת קיבל. היא ממתינה גם לקבלת ההודעה חזרה.
3. Listen – פונקציה זו מאזינה לIP ופורט מסוים (בשרת, גם הIP וגם הפורט מקודדים בצורה קשיחה בקוד – בהתאם להוראות).

כעת נפרט את מיפוי ההודעה ב-server:

1. msgAM-dataAB – ההודעה שמגיעה מA לM היא סטרינג בפורמט json. אנחנו הופכים את הסטרינג חזרה לאובייקט python, ומקבלים את תוכן ההודעה בקריאה msgAB[“dataAB”]
2. msgAM-msgMB – במקרה זה השרת הוא פשוט צינור, הוא מעביר את ההודעה שהוא מקבל לתחנה הבאה, כאשר אנחנו עושים שימוש בmsgAB[“partner\_ip”] כדי לקבל כתובת הIP של היעד
3. msgBMdataBA – גם כאן ההודעה שמגיעה מA לM היא סטרינג בפורמט json. אנחנו הופכים את הסטרינג חזרה לאובייקט python, ומקבלים את תוכן ההודעה בקריאה msgBA[“dataBA”]
4. msgBAmsgMA – גם כאן במקרה זה השרת הוא פשוט צינור, הוא מעביר את ההודעה שהוא מקבל לתחנה הבאה, כאשר אנחנו עושים שימוש בmsgBA[“partner\_ip”] כדי לקבל כתובת הIP של היעד

כעת נפרט את מיפוי ההודעה ב-client:

1. dataABmsgAM – אנחנו מכניסים את dataAB לאובייקט, אשר ממור בשליחה לסטרינג json.
2. msgMAdataBA – בקבלת ההודעה החוזרת, אנחנו מקבלים את dataBA דרך msgMA[“dataBA”]
3. msgMBdataAB - בקבלת ההודעה החוזרת, אנחנו מקבלים את dataAB דרך msgMB[“dataAB”]
4. dataBAmsgBM - אנחנו מכניסים את dataAB לאובייקט, אשר ממור בשליחה לסטרינג json.

בדיקות

בצענו מספר בדיקות על מנת שהתוכנות עובדות קשורה:

1. הרצה של התוכנות על אותו מחשב
2. הרצה של השרת על מחשב אחר וה-clients על מחשב אחד
3. הרצה של כל רכיב על מחשב שונה

נוסף על כך, בצענו בדיקות מקרי קצה:

1. סטרינג ריק
2. סטרינג ארוך במיוחד
3. רשתות שונות

מקורות:

<https://docs.python.org/3/library/socket.html>

<https://docs.python.org/3/library/json.html>