תיק פרויקט

סנכרון מוזיקה

# תוכן עניינים

# מבוא

Syncalong הינה תוכנה המאפשרת לנגן מוזיקה בצורה מסונכרנת בין מספר מכשירים. התוכנה מאפשרת לשתף קבצי מוזיקה בין מספר תחנות, ולהפעיל קבצי מוזיקה בו-זמנית בכל התחנות. באופן זה, מאפשרת syncalong להפוך מספר מכשירים המחוברים רשתית זה לזה למערכת סאונד שלמה.

המערכת מורכבת משרת יחיד, אליו ניתן לחבר מספר תחנות לקוח. ממשק המערכת בשרת מאפשר לנהל את החיבורים לשרת ואת רשימת השירים להשמעה, וכן לשלוט בנגינת המוזיקה בתחנות המרוחקות. ממשק המערכת בצד הלקוח מאפשר להתחבר לשירות או להתנתק ממנו, ומראה את המוזיקה המנוגנת כעת.

הפעולות הנתמכות על ידי המערכת הינן:

* שליחת קבצי מוזיקה לתחנות במחוברות לשרת, במידה והם אינם קיימים בתחנת הקצה
* ניגון, השהייה ועצירה של קובץ מוזיקה
* בניית רשימת השמעה והפעלתה באופן רציף בכל התחנות

# דרישות קדם

המערכת מותאמת לעבודה על כל מערכות ההפעלה, ונבדקה על מחשבי Windows 10 ו-Ubuntu 12.04. יש לוודא כי על תחנות הקצה מותקן כרטיס קול וישנם רמקולים מחוברים.

על מנת להריץ את התוכנה יש להתקין python בגרסה 3.7.4 ומעלה.

# הוראות התקנה

על מנת להתקין את המערכת, יש להיכנס לתיקייה הראשית (syncalong), ומשם להריץ את הפקודה: python setup.py install. הפקודה תתקין על המחשב הנוכחי הן את תוכנת השרת והן את תוכנת הלקוח, יחד עם כל התלויות הנלוות למערכת.

הגדרות המערכת ניתנות לשינוי ישירות בממשק התוכנה.

# הוראות שימוש

בלה בלה בלה

# ארכיטקטורת תוכנה

## סקירה

התוכנה בנויה מצד שרת, וצד לקוח. באחריות השרת לקבל חיבורים חדשים, לנהל את רשימת השירים להשמעה ולסנכרן את הפעולות השונות מול הלקוחות. בצד הלקוח ניתן לקבל קבצים מהשרת, וכן להשמיע מוזיקה על פי הוראות השרת. לצורך התקשורת בין השרת ללקוח, המערכת עושה שימוש במספר טכנולוגיות:

* התקשורת בין השרת ללקוחות עבור שליחת פקודות וסנכרון קבצים נעשית מעל פרוטוקול TCP. לצורך העברת המסרים יצרנו פרוטוקול משלנו, עליו יפורט בהמשך.
* לבניית ופרסור פקטות השתמשנו בספרייה scapy, המאפשרת להגדיר פרוטוקול ייחודי להגדרת המתכנת. הספרייה מאפשרת לייצר אובייקטים לפי מבנה ההודעה הרצוי, לבצע dispatch למידע הגולמי המתקבל מה-socket ולהכניסו לתוך אותם אובייקטים וכן לבצע סריאליזציה של האובייטקים הללו לבתים, כך שיוכלו להישלח מעל ל-socket.
* הסנכרון בין השרת ללקוחות, שבאמצעותו כלל התחנות מנגנות את המוזיקה באותו רגע בדיוק, נעשה באמצעות פרוטוקול NTP. השרת משמש גם כשרת NTP, כך שכלל התחנות מסונכרנות לפיו. בהמשך הפרק נציג הסבר נוסף על הפרוטוקול ואופן השימוש בו בתוכנה.
* קבלת חיבורים חדשים בצד השרת נעשית באופן א-סינכרוני, כך שניתן לבצע פעולות למול לקוחות קיימים גם בזמן טיפול בלקוחות חדשים. דבר זה נעשה באמצעות שימוש ב-thread ייעודי לקבלת לקוחות בצד השרת.

השימוש ב-TCP מאפשר למערכת לשמור על אמינות גבוהה הנדרשת בתהליך סנכרון הקבצים והעברת האותות. יחד עם זאת, משמעות הבחירה בפרוטוקול זה היא שהתקשורת בין השרת ללקוחות אורכת זמן רב, יותר מאשר לו היינו בוחרים להשתמש ב-UDP. בחרנו להעדיף אמינות על פני מהירות מכמה סיבות:

* על העברת הקבצים מהשרת ללקוחות להיות אמינה. התוכנה תומכת בנגינת כל הפורמטים המוכרים לקבצי מוזיקה, והרבה מפורמטים אלו מסתמכים על שלמות המידע שבקובץ. אילו חלק מהקובץ יועבר באופן פגום או לא שלם, לא יהיה ניתן לנגן את הקובץ בצורה תקינה אצל הלקוח.
* קובץ שהועבר באופן חלקי עלול ליצור פערים בסנכרון שבין הלקוחות. אילו הקובץ נקטע לפני הסוף, למשל, הלקוח יעבור לנגן את השיר הבא לפני הלקוחות האחרים, וייצא מסנכרון.
* עלינו לוודא כי כל הלקוחות מקבלים את הפקודות מהשרת. מכיוון שהלקוחות מסתנכרנים מול שרת ה-NTP על מנת לבצע את הפקודה ברגע הנכון, עלינו לוודא רק שכלל הלקוחות אכן קיבלו את הפקודה. זמן ההגעה של האות מהשרת נלקח בחשבון בצד הלקוח בעת חישוב הזמן המתאים לביצוע הפקודה (ראו פירוט על אופן הסנכרון בהמשך).

## מבנה מחלקות

להלן תרשים של מבנה המחלקות בתוכנית והיחסים ביניהן:

<להשלים>

## תיאור המחלקות

### החבילה Common

#### המחלקה LengthSocket(socket.socket) והמודול length\_socket.py

המחלקה LengthSocket הינה האובייקט הבסיסי העוטף socket במערכת, לכתיבה ולקריאה הן בצד השרת והן בצד הלקוח. המחלקה יורשת מהאובייקט socket הפייתוני, ומשמשת לשליחה נוחה של הפרוטוקול שבשימוש המערכת. כל הודעה שנשלחת באמצעות LengthSocket נעטפת ב-header המתאים (ראו GeneralPacket), ולפניו נשלח גודל ההודעה; בכל קבלה המתבצעת על ידי LengthSocket, נקרא מה-socket גודל ההודעה ולפיו, ההודעה עצמה.

המחלקה חושפת שלוש פונקציות: send, recv ו-accept. פונקציות אלו דורסות את המימוש של socket הפייתוני, כך שיפעלו כמתואר לעיל (כמובן שפנימית, נעשה שימוש בפונקציות המקוריות על מנת לבצע פעולות למול ה-socket עצמו).

בנוסף, במודול בו מתוארת המחלקה (length\_socket.py) קיימת פונקציית עזר חופשית בשם "send\_to\_all", המקבלת רשימה של אובייקטים מסוג LengthSocket ורשימה של הודעות לשליחה. הפונקציה שולחת את כל ההודעות לכל ה-socket-ים, ומדווחת על שגיאות במידה וקרו.

#### המחלקה FileSyncPacket(scapy.packet.Packet) והמודול file\_sync\_packet.py

המחלקה FileSyncPacket יורשת מהמחלקה Packet הכללית של scapy, ומשמשת להעברת הודעות לסנכרון קבצי מוזיקה מהשרת ללקוחות. המחלקה מגדירה את מבנה ההודעה הבסיסי לסוג הודעות זה (ראו בהמשך).

במודול file\_sync\_packet.py מיוצאות מספר פונקציות המשמשות לבנייה נוחה של הודעות מסוג FileSyncPacket:

1. הפונקציות who\_has\_packet ו-who\_has\_answer\_packet – פונקציות לבניית הודעות שאלה / תשובה מהשרת / מהלקוח (בהתאמה), לבדיקה אצל מי מהלקוחות חסר קובץ מסוים ויש צורך בשליחתו מהשרת.
2. הפונקציה send\_file\_packet, הבונה הודעת FileSyncPacket המכילה תוכל של קובץ לסנכרון, תוך קריאת הקובץ מהדיסק.

#### המחלקה SignalPacket(scapy.packet.Packet)

מחלקה זו, הנמצאת במודול signal\_packet.py, יורשת אף היא ממחקלת Packet ב-scapy, ותפקידה להגדיר את מבנה ההודעות לשליחת איתותים בין השרת ללקוחות. ארבעת סוגי האיתותים האפשריים הם: play, stop, pause, unpause, ומתוארים בפירוט בהמשך העבודה, עם תיאור הפרוטוקול. לכמה מן האיתותים נדרשים פרמטרים נוספים. פרמטרים אלו מוגדרים גם הם במחלקה, וכן ערכיהם הדיפולטיים.

#### המחלקה GeneralPacket(scapy.packet.Packet) והמודול general\_packet.py

המחלקה GeneralPacket הינה מחלקת מעטפת עבור שתי מחלקות ההודעות המפורטות בסעיפים הקודמים. כל הודעה הנשלחת מה-socket נעטפת על ידי אובייקט זה. באמצעות הפרמטרים המוגדרים ב-GeneralPacket ניתן לחלץ את המידע הרצוי מתוך ההודעה, על פי סוגה (במקרה שלנו קיימים שני סוגי הודעות: FileSyncPacket, SignalPacket, אולם המחלקה כתובה כך שניתן להרחיבה לסוגי הודעות נוספים). חילוץ זה נעשה באופן אוטומטי על ידי scapy.

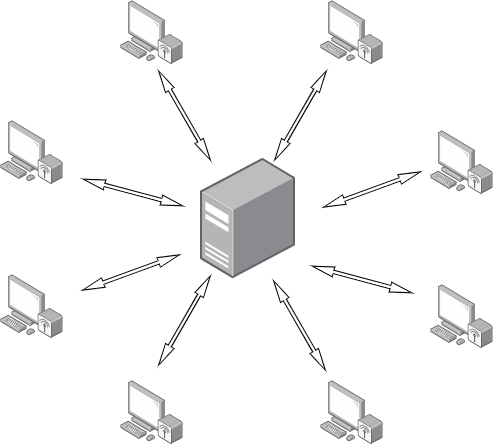
הפונקציה generate\_packet במודול משמשת לבנייה נוחה של הודעה מסוג GeneralPacket העוטפת הודעה מסוג אחר.

הפונקציה handle\_packet משמשת לעבודה נוחה מול הודעות המתקבלות ב-socket מסוים. הפונקציה מקבלת אובייקט מסוג GeneralPacket, ומילון הממפה בין סוג הודעה לבין פונקציה המטפלת בהודעה זו. הפונקציה דואגת להרצת לוגיקת הטיפול המתאימה לסוג ההודעה העטופה ב-GeneralPacket. דוגמת שימוש בפונקציה קיימת במחלה Client, בפונקציה start.

### החבילה Server

## פרוטוקול התקשרות

מבנה הרשת המוגדר על ידי המערכת הוא מבנה כוכב, בו כל הלקוחות מתחברים אל שרת מרכזי אחד, מסתנכרנים מולו ומקבלים ממנו פקודות לביצוע. כאמור בסקירה, כל התקשורת בין השרת ללקוחות וחזרה נעשית מעל פרוטוקול TCP.



איור 1 - מבנה הרשת המוגדר על ידי התוכנית: שרת מרכזי אחד, המסנכרן בין מספר קליינטים

עם עלייתו, השרת פותח thread המאזין לחיבורים חדשים. לקוח חדש המעוניין להתחבר לשרת שולח לו הודעת פתיחה. השרת מקבל את הלקוח ומוסיף אותו לרשימת הלקוחות שלו.

המבנה הבסיסי של כל הודעה הנשלחת במערכת בין השרת ללקוחות הוא כדלהלן:

|  |  |
| --- | --- |
| 8-15 | 0-7 |
| type | length |
| payload … | |

כאשר length מציין את גודל ההודעה כולה (כולל type), ו-type את סוגה: FileSyncPacket או SignalPacket. ה-payload מכיל את תוכן ההודעה מהסוג המתאים.

מבנה הודעת FileSyncPacket הוא:

|  |  |  |
| --- | --- | --- |
| 16-23 | 8-15 | 0-7 |
| file name length | file size | message type |
| file name … | | |

השדה file size מציין את גודל הקובץ לסנכרון. השדה file name מציין את שם הקובץ, ו-file name size מציין את אורכו של שדה השם. השדה message type מציין את סוג המסר המועבר מבין הארבעה:

* 2 – who has

הודעה מסוג זה נשלחת מהשרת לכלל הלקוחות, ומטרתה לתשאל כל לקוח האם הקובץ הרצוי שמור אצלו לוקאלית.

* 1 – have

הודעת תשובה להודעת who has, הנשלחת על ידי לקוח שברשותו הקובץ הרצוי במלואו.

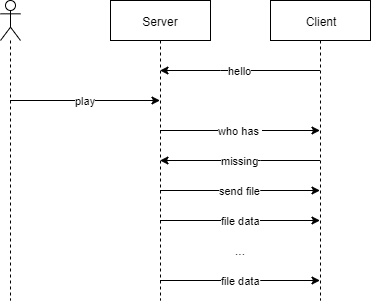
* 0 – missing

הודעת תשובה להודעת who has, הנשלחת על ידי לקוח שהקובץ הרצוי אינו קיים אצלו, או קיים אצלו באופן לא תקין (בגודל קובץ שונה).

* 4 – send file

הודעה מקדימה לשליחה של הקובץ עצמו. הודעה זו נשלחת מהשרת ללקוחות שיש לסנכרן עימם את קבצי השמע. לאחריה, ישלח השרת את המידע של הקובץ עצמו.

להלן תרשים זרימה המתאר את פעולות הפרוטוקול עבור סנכרון קובץ. האיור מתאר תקשורת למול לקוח אחד, אולם אותו התהליך משוכפל על ידי המערכת עבור יתר הלקוחות אשר התחברו לשרת.



איור 2 - סנכרון קובץ לנגינה בין שרת ללקוח. (1) הלקוח מתחבר אל השרת בהודעת פתיחה; (2) המשתמש מורה לשרת להתחיל לנגן; (3) השרת מתשאל את הלקוח: האם הקובץ קיים אצלו; (4) הלקוח משיב שהקובץ לא ברשותו; (5) השרת שולח הודעה לציון תחילת העברת הקובץ; (6) השרת שולח את הקובץ.