Izrada mobilne aplikacije za sinkroniziranu reprodukciju zvuka među uređajima u istoj lokalnoj mreži

Vrdoljak, Karlo

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1563

IZRADA MOBILNE APLIKACIJE ZA SINKRONIZIRANU REPRODUKCIJU ZVUKA MEĐU UREĐAJIMA U ISTOJ LOKALNOJ MREŽI

Karlo Vrdoljak

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1563

IZRADA MOBILNE APLIKACIJE ZA SINKRONIZIRANU REPRODUKCIJU ZVUKA MEĐU UREĐAJIMA U ISTOJ LOKALNOJ MREŽI

Karlo Vrdoljak

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1563

Pristupnik:	**Karlo Vrdoljak (0036542068)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	prof. dr. sc. Lea Skorin-Kapov

Zadatak: **Izrada mobilne aplikacije za sinkroniziranu reprodukciju zvuka među uređajima u istoj lokalnoj mreži**

Opis zadatka:

U današnjem digitalnom dobu, mobilne aplikacije su postale nezaobilazan alat za različite svakodnevne aktivnosti. Jedna od interesantnih funkcionalnosti je mogućnost sinkronizirane reprodukcije zvuka među uređajima koji su povezani u istoj lokalnoj mreži. Ova tehnologija omogućava korisnicima da istovremeno reproduciraju isti audio sadržaj sa više razmaknutih mobilnih uređaja u zajedničkom fizičkom prostoru, pružajući dojam da zvuk dolazi sa više strana. Ovakav prostorno okružujući zvuk (engl. surround sound) često se koristi u kućnim zvučnim sustavima s više zvučnika za izlaz. Vaš zadatak je razviti nativnu mobilnu aplikaciju za platformu Android koja će omogućiti sinkroniziranu reprodukciju zvuka među uređajima u istoj lokalnoj mreži. Jedan uređaj treba biti postavljen kao izvorni, te se s tog uređaja strujanjem prenosi audio sadržaj ostalim povezanim uređajima putem protokola UDP. Za izvedbu sinkronizacije, potrebno je razviti vlastiti program za reprodukciju medija (engl. media player) te implementirati razmjenu informacija potrebnih za postizanje sinkronizirane reprodukcije među povezanim uređajima.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

1. Uvod

Pametni zvučnici s funkcijom sinkronizacije postaju sve popularniji te se danas ta tehnologija ugrađuje u skoro sve nove zvučnike koji nude korisnicima poboljšano iskustvo slušanja glazbe i drugih audio sadržaja. Jedan on najpopularnijih zvučnika s tom funkcionalnosti je Flip 6 proizvođača JBL [1] koji omogućava uparivanje s bilo kojim zvučnikom koji je kompatibilan sa *PartyBoost* načinom sinkronizacije. No takav sustav povezanih zvučnika nije pristupačan svakom korisniku zbog visoke cijene te se ovim radom pokušava dati rješenje koje umjesto kupnje novih uređaja pokušava utilizirati zbirnu moć već skoro svima dostupnim mobilnim uređajima. Koristeći te uređaje i dobro osmišljene algoritme sinkronizacije mogao bi se postići distribuirani sustav zvučnika koji sinkronizirano reproduciraju glazbu ili neki drugi audio zapis iz više smjerova (engl. *surround sound*).

U ovom radu istražujemo koncept sinkroniziranog zvučnog sustava koji koristi mobilne uređaje kao dodatne zvučnike koji provode naredbe „glavnog“ mobilnog uređaja kako bi se dobila sinkronizirana reprodukcija zvuka. Tehnologiju koja će se koristiti za komunikaciju uređaja i izmjenu podataka je lokalna mreža, poznata i kao LAN (engl. *Local Area Network*), koja povezuje računala i druge uređaje na manjim udaljenostima koristeći komutator kao centralni uređaj prosljeđivanja paketa. Za razliku od Bluetooth veze ona je puno stabilnija, ima puno veći domet veze i omogućuje povezivanje puno većeg broja uređaja [2].

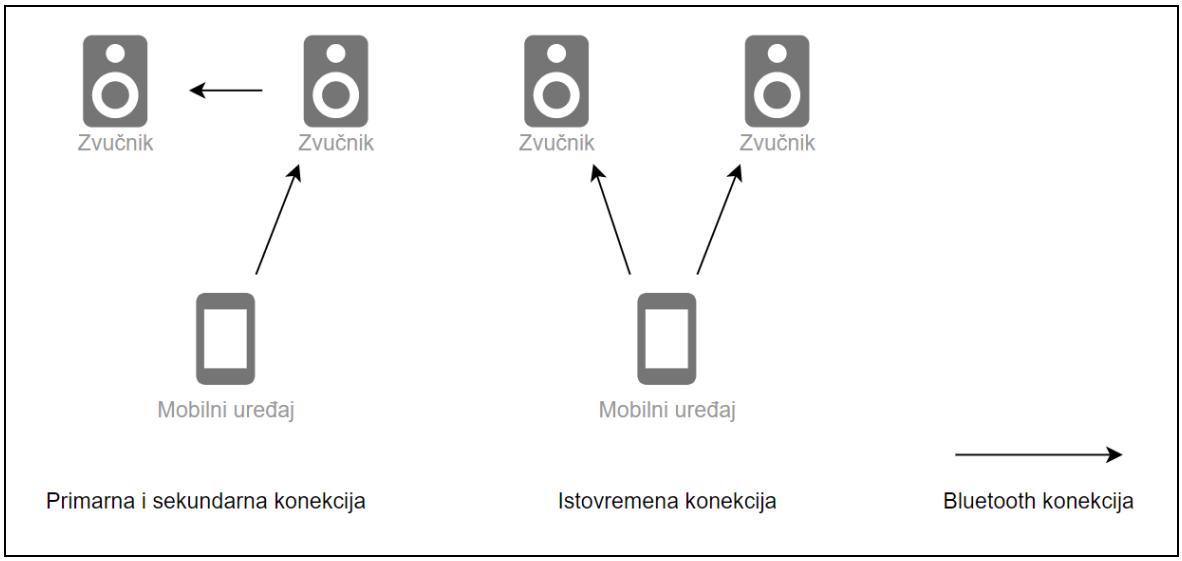
U radu se analiziraju tehnički izazovi koji se javljaju pri implementaciji takvog sustava, kao što su precizna sinkronizacija između uređaja, stabilnost veze i minimiziranje kašnjenja. Također se istražuju moguća rješenja, mehanizmi i tehnologije koje se mogu koristiti za savladavanje ovih izazova.

Prvotno će u 2. poglavlju biti predstavljeno kako se problem sinkronizacije pokušao riješiti dosad te će se dati uvid u konkretna rješenja koja također pokušavaju utilizirati mobilne uređaje kako bi došli do rješenja. U 3. poglavlju se predstavlja puni opseg problema i svi zahtjevi implementacije koje aplikacija mora zadovoljiti kako bi se dobilo unikatno rješenje. Nadalje, u 4. poglavlju bit će definiran model rješenja koje detaljno objašnjava koncepte, algoritme i korištene protokole za izvedbu. U 5. poglavlju predstavit će se

implementirano rješenje u obliku aplikacije u sklopu ovog rada. Poglavlje uključuje opis svih funkcionalnosti aplikacije te se prikazuju svi elementi korisničkog sučelja kao uputa za korištenje tih funkcionalnosti. U 6. poglavlju se analizira finalni rezultat projekta i objašnjava se eksperiment mjerenja preciznosti sinkronizacije sa i bez korištenja mehanizama sinkronizacije te se daju rezultati eksperimenta. U 7. poglavlju se daje sažeti zaključak koji donosi odluku o ukupnom uspjehu projekta te se u 8. poglavlju navode sve reference na literaturu, odnosno druge radove i projekte čija su otkrivanja ili informacije pomogla za izradu ovog rada.

2. Pregled literature i sličnih rješenja

Kao što je navedeno u uvodu, za distribuiranu sinkroniziranu reprodukciju se danas često koriste zvučnici s mogućnošću međusobnog povezivanja Bluetooth tehnologijom. Takvi zvučnici koriste TWS (engl. *True Wireless Sound*) [3] tehnologiju koja vrši sinkronizaciju na nekoliko mogućih načina, jedna mogućnost je da se svi zvučnici povezuju na centralni uređaj za reprodukciju te se sinkronizacija uspostavlja preko njega ili da su zvučnici povezani lančano jedan preko drugog, primarnim i sekundarnim konekcijama od centralnog uređaja (Slika 1). Ta tehnologija zasnovana je ne Bluetooth-u, no za potrebe ovog rada okrenut ćemo se uporabi računalne mreže kao prijenos podataka zbog toga što je takva konekcija jednostavnija za implementaciju i prilagodljivija za pametne uređaje.



Slika 1 - Prikaz različitih načina sinkronizacije TWS tehnologije, dijagram napravljen uz pomoć Draw.io alata [21]

Jedna od sličnih aplikacija koje koriste lokalnu mrežu je popularna aplikacija AmpMe [4]. Ona korisniku nudi mnogo dodatnih mogućnosti osim same sinkronizacije, poput podešavanja zvuka iz bilo kojeg izvora, na primjer iz YouTube-a i Spotify-ja ili mogućnost pisanja poruka uživo kroz aplikaciju. No, testiranje same sinkronizacije je dalo lošije rezultate, posebno uzimajući u obzir model pretplate aplikacije. Recenzije na Google Play trgovini ističu da korisnici često gube konekciju između uređaja te je kvaliteta zvuka reprodukcija na krajnjim uređajima smanjena [7].

Druga aplikacija je SoundSeeder [5], ona radi na sličnom principu, no kvaliteta zvuka je u ovom slučaju mnogo bolja. Za povezivanje dva uređaja sinkronizacija je vrlo visoke kvalitete, no već nakon tri spojena uređaja potrebno je ručno namještati odmak u vremenu zvuka radi sinkronizacije, što je za krajnjeg korisnika vrlo nepraktično.

Dosadašnje aplikacije vrše sinkronizaciju isključivo preko bežične lokalne mreže, no u sljedećem radu M. Urech, Distributed Speaker Synchronization [6], sinkronizacija je uspostavljena tako da uz inicijalnu sinkronizaciju preko mreže, mobilni uređaji ciklično osluškuju svoju okolinu te time pokušavaju usklađivati svoju reprodukciju s ostalim uređajima. Time je postignuta preciznost sinkronizacije ispod 14,4 ms te mehanizam sinkronizacije radi pouzdano bez gubitaka konekcije. Problem kod ovakvog pristupa, a koji se ne poklapa s definiranim zahtjevima aplikacije u ovom projektu, je da koristi eksterni poslužitelj za sinkronizaciju kojemu svaki uređaj mora imati pristup tijekom reprodukcije. Kao rezultat, kod svakog pokušaja sinkronizirane reprodukcije bilo potrebno dodatno pokretati poslužitelj na nekom od uređaja ili imati web-poslužitelj koji je konstantno pokrenut na istoj mašini, što uvodi nove probleme. Uređaji neće nužno biti spojeni na istu lokalnu mrežu kao poslužitelj te će morati imati pristup internetu kako bi mogli s njime komunicirati. Također, u slučaju velike udaljenosti od poslužitelja, usporio bi se proces sinkronizacije i upravljanja reprodukcijom zbog propagacijskog kašnjenja.

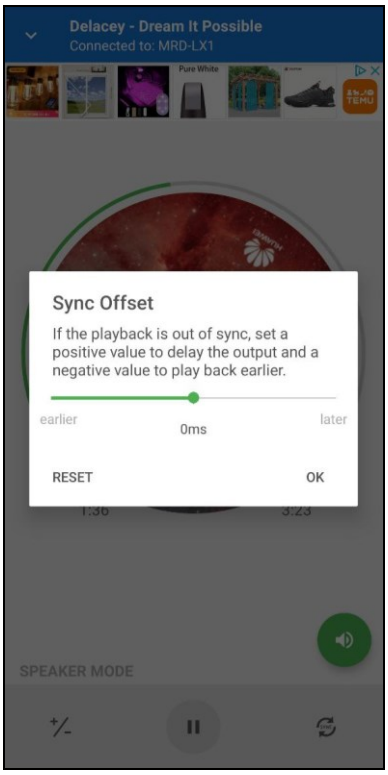
3. Opis problema i zahtjevi aplikacije

U ovom poglavlju bit će predstavljena problematika projekta koja se pokušava riješiti korištenjem određenih mehanizama i tehnologija. Bit će adresirani svi problemi sa prethodno razvijenim rješenjima te će se postaviti konkretni zahtjevi aplikacije koji moraju biti održani kroz model rješenja i konkretnu implementaciju danu u daljnjim poglavljima Model i Razvoj aplikacije MultiSync.

3.1. Kompleksnost korištenja

Kod nekih aplikacija se podešavanje sinkronizacije dodjeljuje kao odgovornost krajnjem korisniku tako da se ručno unosi vremenski razmak na temelju sluha korisnika kako bi se postigla sinkronizacija uređaja kao što se vidi na slici zaslona aplikacije SoundSeeder (Slika 2). Situacija je još gora ako je taj postupak potrebno ponavljati svaki put kada se reprodukcije prirodno desinkroniziraju zbog vanjskih faktora poput različite procesorske brzine mobitela. Stoga je jedan od ključnih zahtjeva minimalna odgovornost korisnika kako bi se osiguralo glatko i ugodno korisničko iskustvo.

Aplikacija bi trebala automatski upravljati sinkronizacijom i prilagođavati sve parametre u pozadini, omogućujući korisnicima da jednostavno pokrenu reprodukciju bez dodatnih koraka ili podešavanja. Dodatno, kao rješenje sustav ne bi trebao ovisiti o eksternom poslužitelju koji upravlja sinkronizacijom. Svi mehanizmi sinkronizacije bi trebali biti implementirani u okviru uređaja koji reproduciraju zvuk kako bi korisnici mogli koristiti aplikaciju i dok nemaju direktan pristup Internetskoj mreži.



Slika 2 - Slika zaslona aplikacije SoundSeeder uz skočni prozor Sync Offset

3.2. Performanse aplikacije

Poželjno je da aplikacija ima visoku preciznost sinkronizacije kako bi se održala visoka iskustvena kvaliteta, no osim sinkronizacije postoji još osobitosti aplikacije o kojima ovisi iskustvena kvaliteta. Potrebno je imati malo vrijeme odziva aplikacije na promijene stanja reprodukcije kao i kratko vrijeme učitavanja audio zapisa. Također, trebala bi smanjiti protok podataka između uređaja kako se stabilnost mreže ne bi urušila, odnosno da se smanji potrošnja baterije uređaja zbog slanja ili primanja prevelikog broja paketa. Kako bi se to postiglo potrebno je pravilno izabrati protokole uključene u izmjenu podataka odnosno upravljanje sinkronizacijom i reprodukcijom.

Dva važna transportna protokola u mrežnim komunikacijama su UDP (engl. *User Datagram Protocol*) i TCP (engl. *Transmission Control Protocol*) [8]. TCP omogućuje pouzdan prijenos podataka između uređaja na mreži. Prvo uspostavlja vezu između pošiljatelja i primatelja, osiguravajući da svi podaci stignu ispravnim redoslijedom i bez gubitaka, no ta razina kontrole i pouzdanosti dolazi s većom kompleksnošću i dužim odzivom. S druge strane protokol UDP ne zahtijeva uspostavu veze, što smanjuje vrijeme početnog kašnjenja i podržava multicast i broadcast prijenos, omogućujući slanje podataka

s pošiljatelja prema više primatelja istovremeno. Ovo je idealno za aplikacije koje trebaju slati isti set podataka većem broju primatelja, čime se smanjuje opterećenje mreže. Iako UDP ne osigurava pouzdanost isporuke i podaci mogu biti izgubljeni ili primljeni u pogrešnom redoslijedu, njegove prednosti uključuju smanjeno vrijeme odziva, efikasniji prijenos podataka i manje trošenje baterije zbog manje složenosti protokola.

3.3. Portabilnost na različite operacijske sustave

Veliki problem koji ostala rješenja ne adresiraju je mogućnost povezivanja uređaja različitih operacijskih sustava (Android, IOS, Windows te Linux). Iako će se glavnina korisnika povezivati mobilnim uređajima, potrebno je uzeti i prijenosna računala u obzir sa svojim operacijskim sustavima kako bi se broj mogućih korisnika mogao dodatno povećati.

Jedno rješenje je razvijati aplikacije zasebno za svaki operacijski sustav, što je vrlo složeno i zahtjeva mnogo vremena. Ovaj pristup također zahtijeva širok spektar znanja ili čak više timova za razvoj. Osim toga, potrebno je osigurati kompatibilnost tijekom upotrebe, odnosno omogućiti neometanu komunikaciju među uređajima bez problema i zastoja.

Drugo rješenje je implementacija web aplikacije što je iznimno portabilan način jer danas velika većina uređaja ima ugrađene pretraživače koji bi takvu aplikaciju mogli pokretati. Dodatno, prednost ovog pristupa je što korisnik ne bi trebao vršiti instalaciju softvera već samo otvoriti poveznicu na stranicu na kojoj će se pokretati aplikacija čime se dodatno smanjuju odgovornosti korisnika. Problem kod ovog pristupa je što zbog sigurnosti korisnika mnogi pretraživači onemogućavaju komunikaciju UDP protokolima [9]. Jedini način za slanje i primanje UDP paketa u ovom kontekstu je korištenjem WebRTC-a (Web Real-Time Communication). To je besplatan i otvoren skup standarda i API-ja (Application Programming Interface) koji omogućava web preglednicima i mobilnim aplikacijama da uspostave direktnu vezu između ravnopravnih čvorova za razmjenu audio, video i tekstualnih podataka u stvarnom vremenu, bez potrebe za posrednim poslužiteljem [22]. Nažalost, WebRTC je strogo definiran protokol i ne podržava mnogo prilagodbi. Specifičnost aplikacije može zahtijevati visoku razinu kontrole nad paketima koji se prenose mrežom, što WebRTC ne može adekvatno podržati.

Treće rješenje je implementacija aplikacije korištenjem cross-platformskih alata. To omogućava izradu aplikacije za više operacijskih sustava uz pisanje izvornog koda samo jednom. Međutim, zbog specifičnosti aplikacije, određene funkcionalnosti će ipak trebati

biti napisane u nativnim programskim jezicima odgovarajućih operacijskih sustava, jer za određene API-je sustava može nedostajati programska podrška alata.

Kao što će biti detaljno objašnjeno u 5. poglavlju, odabrano je treće, cross-platform rješenje putem popularnog Googleovog alata Flutter [10] za izradu aplikacija za razne operacijske sustave.

4. Model

U ovom poglavlju modelirat će se aplikacija koja prihvaća sve zahtjeve definirane u prethodno poglavlju. Bit će objašnjena arhitektura sustava tijekom reprodukcije, prijenos i format podataka koji struje mrežom te detaljno opisani primijenjeni mehanizmi sinkronizacije koji uzimaju u obzir nepredvidljivosti mreže i samih uređaja.

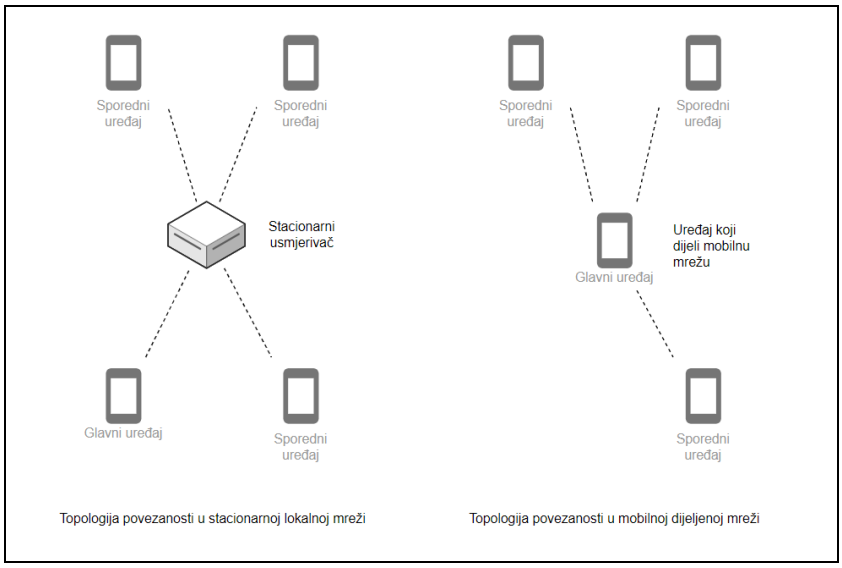
4.1. Arhitektura sustava

Sustav je osmišljen tako da je jedan uređaj iz grupe uređaja „glavni“ što znači da upravljanje reprodukcijom dolazi isključivo od njega dok su svi ostali uređaji efektivno njegovi zvučnici. U ovom modelu nije omogućeno da ostali uređaji upravljaju reprodukcijom kako bi se smanjila kompleksnost mehanizama sinkronizacije.

Komunikacija se bazira na bežičnom prijenosu u lokalnoj mreži (WLAN [11], engl. wireless local area network) i preko koje teče sva komunikacija. Za komunikaciju u lokalnoj mreži je odgovoran LAN komutator koji se koristi za usmjeravanje signala ili podataka između više uređaja ili mrežnih segmenata. Lokalna mreža se najčešće postiže na dva načina:

- povezivanjem na istu stacionarnu mrežu
- dijeljenjem hotspot mreže s jednog mobilnog uređaja.

U slučaju stacionarne mreže, ulogu komutatora ima stacionarni usmjerivač (engl. router), dok je kod hotspot mreže komutator onaj uređaj koji dijeli hotspot. Bitno je napomenuti da glavni uređaj u kontekstu aplikacije ne mora biti onaj koji dijeli hotspot. Topologija tako postavljenih mreža ja prikana na sliciSlika 3.



Slika 3 - Dijagrami stacionarne i mobilne lokalne mreže nakon spajanja uređaja, draw.io

4.2. Komunikacija uređaja

Tijekom rada aplikacije, odnosno reprodukcije zvuka, uređaji će mrežom komunicirati sa dva tipa paketa:

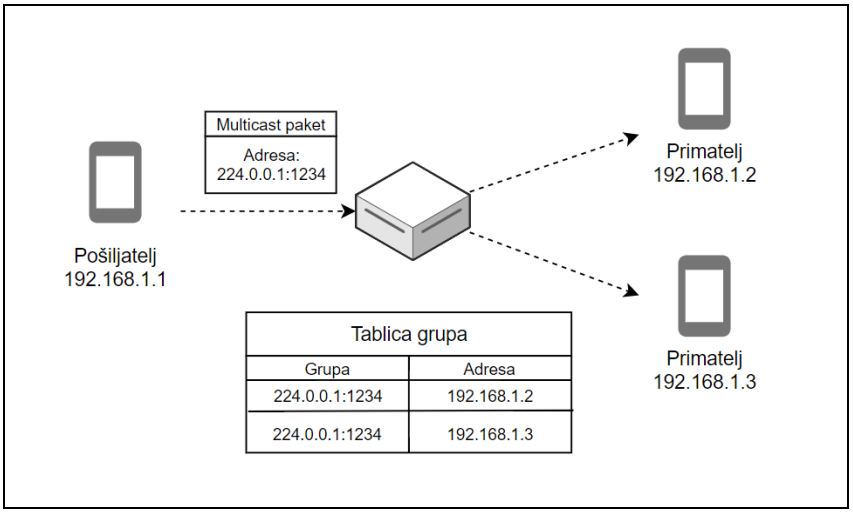
- naredbeni paketi koji služe upravljanjem reprodukcije
- podatkovni paketi za prenošenje audio zapisa za reprodukciju.

4.2.1. Protokol za prijenos paketa

Kako je već navedeno u 3. poglavlju, potrebno je implementirati komunikaciju protokolom UDP kako bi ubrzali rad aplikacije i smanjili opterećenje na mrežu. Iako protokol UDP ne osigurava dolazak svakog paketa, vjerojatnost gubitka paketa u lokalnoj mreži je izuzetno malena zbog male udaljenosti i malog broja komponenti mreže kroz koje paket treba proći.

Kako bi smanjili opterećenje na mrežu tijekom slanja podatkovnih paketa, koristi se višeodredišno razašiljanje (engl. *multicast)* [12] jer glavni uređaj treba isti paket audio podataka prenijeti svim sporednih uređajima. Multicast funkcionira na principu pretplate, gdje se uređaji prijavljuju u multicast grupu definiranu multicast adresom i vratima. Kada pošiljatelj šalje paket na multicast adresu, on se prosljeđuje samo onim uređajima koji su pretplaćeni na tu grupu. To se postiže korištenjem usmjerivača koji održavaju tablice pretplata i na temelju njih određuju destinacije paketa. Slika 4 prikazuje primjer odašiljanja multicast paketa u lokalnoj mreži, u tablici grupa komutatora su 2 primatelja pretplaćena

na grupu 224.0.0.1:1234. Kada od pošiljatelja dođe paket s odredišnom IP adresom koja odgovara grupnoj adresi, komutator ga prosljeđuje na sve adrese iz svoje tablice grupa na kojima je zapisana ta grupa, odnosno na primatelje s adresama 192.168.1.2 i 192.168.1.3. Zahtjevi za pretplatom i prekidanjem pretplate se vrše putem protokola Internet Group Management Protocol (IGMP) [23] tako da se taj zahtjev šalje iz uređaja koji se želi pretplatiti na adresu komutatora u mreži.



Slika 4 – Prosljeđivanje multicast paketa u lokalnoj mreži, draw.io

4.2.2. Naredbeni paketi

Kako bi glavni uređaj mogao upravljati sporednima i održavati sinkronizaciju potrebno je definirati vlastite komande koje se šalju putem definiranog protokola. Njima se prenosi informacija o akciji koju bi sporedni uređaji trebali izvršiti kada prime takav paket. Svaki naredbeni paket ima preddefiniranu identifikacijsku oznaku kako bi se mogli identificirati kada dođu do sporednih uređaja.

U sklopu ovog rada su zatim definirane sve potrebne akcije korištene za ispravno upravljanje reprodukcijom na svim povezanim uređajima:

- *setAudio* – služi za zadavanje audio resursa koji će biti reproduciran. Njegov specijalni atribut je *asset* – definira url do audio zapisa, u slučaju da korisnik s glavnog uređaja odluči strujati lokalno spremljenu datoteku, ovaj paket označava početak strujanja te se sporedni uređaji moraju pripremiti za osluškivanje mreže i dohvaćanje datoteke
- *Resume* – služi za signalizaciju pokretanja reprodukcije

- *Pause* – služi za signalizaciju zaustavljanja reprodukcije
- *Seek* – postavljenje audio zapisa na točnu poziciju u vremenu, npr. korisnik želi preskočiti sljedećih 10 sekundi u reprodukciji. Ovaj paket također uključuje poseban atribut *position* koji predstavlja poziciju audio zapisa na kojoj je potrebno postaviti trenutnu reprodukciju
- *Sync* – ovaj paket služi za sinkronizaciju unutarnjih satova glavnog i sporednih uređaja, ovaj paket će biti detaljno objašnjen u sljedećem potpoglavlju.

Većina komandnih paketa će se slati ovisno o korisnikovoj aktivnosti u aplikaciji. Na primjer, ako korisnik na glavnom uređaju postavi reprodukciju na neku konkretnu pjesmu te zatim pokrene reprodukciju, poslat će se redom naredbeni paketi *setAsset* te kasnije *Resume*. Osim takvih akcijskih paketa, periodično se šalju i *Sync* paketi koji služe za sinkronizacija satova između uređaja.

4.3. Mehanizmi sinkronizacije

Za ispravan rad aplikacije nije dovoljno samo poslati kontrolne pakete i pokrenuti akciju nakon primanja paketa jer postoji mnogo trenutaka u kojem je točno vrijeme akcije teško odrediti. U ovom potpoglavlju odredit će se mogući uzroci desinkronizacije koji se mogu pojaviti tijekom reprodukcije te će pokušati dati algoritamsko rješenje tih problema.

4.3.1. Kompenzacija kolebanja kašnjenja LAN-a

Kašnjenje prijenosa paketa u lokalnim mrežama je vrlo niska [13] pogotovo u jednostavnoj topologiji s malenim brojem povezanih uređaja kao što će biti korištena za rad aplikacije. Problemi nastupaju kad je mreža pod velikim opterećenjem zbog prevelikog broja paketa koji se šalje zbog aktivnosti korisnika u mreži, npr. preuzimanje velikih datoteka s Interneta. U tim trenucima može doći do kolebanja kašnjenja (engl. *jitter*) što znači da će paketi koji se s izvora šalju u pravilnim intervalima, do svog cilja stizati u nepravilnim intervalima ili će čak dolaziti krivim redoslijedom. Za tu je svrhu je potreban mehanizam s kojim bi reprodukcija bila otporna na velike vremenske razlike u pristizanju paketa.

Jedan način za to je da se umjesto pokretanja akcije odmah nakon pristizanja paketa, akcija pokreće u točnom trenutku definiranim preko POSIX (engl. *Portable Operating System Interface*) vremena [14] u formatu koji uključuje milisekunde. Iako POSIX vrijeme

podržava i mikrosekunde, tolika preciznost nam nije važna zbog sposobnosti ljudskog uha da čuje vremensku razliku u zvuku na tek nekoliko desetaka milisekundi.

4.3.2. Sinkronizacija satova

Novi problem koji proizlazi je što se unutarnji satovi uređaja mogu razlikovati i do nekoliko sekundi, što znači da bi reprodukcija toliko kasnila/ranila ovisno o pomaku u satovima ako bi se oslonili samo na mehanizam pokretanja akcije u točnom POSIX vremenu. Rješenje je da svaki sporedni uređaj ima informaciju o tome kolika je razlika u satovima između njega i glavnog uređaja unutar jedne milisekunde kako bi znao koliko točno treba odgoditi reprodukciju.

To se može postići korištenjem Marzullovog algoritma [15] koji omogućuje sporednom uređaju da kontinuirano dobiva informaciju o razlici unutarnjih satova između sebe i glavnog uređaja. Marzullov algoritam osigurava preciznu sinkronizaciju vremena čak i u dinamičnim mrežnim uvjetima, minimizirajući odstupanja i omogućujući vršnom uređaju da točno odredi kada treba odgoditi reprodukciju kako bi se postigla željena vremenska sinkronizacija unutar tolerancije od jedne milisekunde.

Marzullov algoritam se primarno koristi u protokolu NTP (engl. *Network Time Protocol*) [16], koji je specifično dizajniran za sinkronizaciju vremena između računala u računalnim mrežama. NTP je jedan od najčešće korištenih protokola za održavanje točnosti vremena u distribuiranim računalnim sustavima i mrežama. NTP protokol može biti precizan i u redu veličine nekoliko mikrosekundi što je za potrebe aplikacije previše. Stoga će se za aplikaciju koristiti vrlo pojednostavljena verzija tog protokola.

4.3.3. Kalibriranje izlazne brzine zvuka

Dodatan problem koji se predstavlja je različita brzina procesiranja (engl. *app processing time*) i reprodukcije zvuka (engl. *audio output latency*) kod različitih uređaja [17], koja se dosta razlikuje kod uređaja na različitim operacijskim modelima pa čak istim modelima uređaja. Taj problem može pridodati nekoliko stotina milisekundi razlike što ima znatni utjecaj na kvalitetu korisničkog iskustva.

Rješenje problema je uvesti novi mehanizam sinkronizacije koji će na svakom uređaju mjeriti vrijeme potrebno za početak reprodukcije te će to vrijeme oduzeti od *defaultPlayTime* vrijednosti koja je identična za svaki uređaj, odnosno zadana je kao

unutarnja varijabla u programskom kodu. Svaki uređaj tada zasebno usporava odnosno ubrzava brzinu reprodukcije na određeno vrijeme kako bi se prilagodio predodređenom vremenu reprodukcije.

5. Razvoj aplikacije MultiSync

U sklopu ovog rada izrađena je aplikacija koja koristi protokole i algoritme navedene u prethodnom 4. poglavlju kako bi se dobila aplikacija jednostavnog sučelja s već definiranim značajkama. U ovom poglavlju će biti objašnjena konkretna implementacija zadanog modela kroz kratki opis u kojem će se dobiti uvid u mogućnosti aplikacije i koje su tehnologije bile korištene za samu izradu. Također će se kroz slike zaslona vidjeti izgled aplikacije u potpoglavlju te će biti objašnjene uloge pojedinih vizualnih elemenata aplikacije.

5.1. Opis aplikacije

Ime aplikacije je inspirirano ključnim komunikacijskim protokolom odgovornim za učinkovito prosljeđivanje paketa u mreži, multicast te čestim motivom ovog rada, sinkronizacija. Tada se spajanjem kratica tih pojmova dobiva ime MultiSync. Kao što će biti prikazano na slikama zaslona (Slika 5, Slika 6 i Slika 7), ime aplikacije nalazi se i na početnom zaslonu aplikacije.

Aplikacija ima tri moda reprodukcije zvuka:

- normalan mod - kada se aplikacija ponaša kao normalan program za reprodukciju zvuka
- sinkronizirajući mod, glavni uređaj – kada aplikacija preuzima ulogu glavnog uređaja u sustavu za sinkronizaciju
- sinkronizirajući mod, sporedni uređaj – kada aplikacija preuzima ulogu sporednog uređaja u sustavu za sinkronizaciju.

Kada uređaj prijeđe u mod sporednog uređaja, on se pretplati na multicast grupu na unaprijed zadanoj adresi i vratima čime se omogućava primanje paketa. Kada je aplikacija u ulozi glavnog uređaja, sve akcije koje korisnik napravi na tom uređaju se prevode u naredbene pakete koji se zatim šalju na adresu predodređene multicast grupe, iste one na koju su pretplaćeni sporedni uređaji te time dobivaju naredbene pakete i vrše akciju.

5.2. Korištene tehnologije

U ovom potpoglavlju će se objasniti uloga i obilježja tehnologija korištenih za izradu projekta, to su Flutter koji je poslužio kao glavna osovina za izradu aplikacije u čijem su programskom jeziku Dart pisane sve skripte za implementaciju korisničkog sučelja i algoritama. U svrhu testiranja integracije primanja i odašiljanja komandnih paketa i paketa audio podatak je poslužio i jezik Python čije su skripte simulirale rad glavnog i sporednog uređaja.

5.2.1. Flutter

Za izradu aplikacije korišten je popularni Googleov framework Flutter i programski jezik Dart [10]. Flutter omogućava brzi razvoj zbog toga što podržava jedinstvenu kodnu bazu (engl. one code base) što znači da je kod za sve operacijske sustave potrebno pisati samo jednom nakon čega se može kompilirati na sve operacijske sustave. Razlog te prilagodljivosti je što Flutter koristi visoko optimizirani 2D render engine, Impeller [18]. Tako Flutter ne koristi nativne widgete operacijskog sustava već sam projicira sliku na ekran te može kontrolirati svaki piksel. Također Flutter pruža apstrakciju platformno specifičnih funkcionalnosti, kao na primjer interakcija sa zvučnicima ili mikrofonom, kroz pluginove ili platformske kanale koji omogućuju komunikaciju Darta s nativnim programskim jezicima sustava. Za neke složenije funkcionalnosti ili funkcionalnosti za koje još nisu napisani pluginovi, potrebno je dodatno pisati nativni kod sustava za koji se aplikacija radi koji se kasnije nadovezuje u program.

Osnovni gradivni element korisničkog sučelja u Flutteru je widget. Ti widgeti se postavljaju u hijerarhijsku strukturu gdje je veliki naglasak na modularnost. Svaki widget može biti pojedinačna klasa koja ima svoje atribute i funkcionalnosti te ih se može duplicirati i pozicionirati po volji.

5.2.2. Python

Python [19] je programski jezik poznat po svojoj jednostavnosti i brzini pisanja koda. U ovom projektu je poslužio za testiranje komunikacije među uređajima i kompatibilnosti kontrolnih paketa za upravljanje reprodukcijom u aplikaciji. Na početku su se audio podaci odašiljali pomoću Python skripti s računala na ostale sporedne uređaje kako bi se testiralo primanje paketa i ispravnost kontrolnih paketa. U vrhu toga je bilo potrebno potpuno

simulirati finalno ponašanje uređaja u mreži što podrazumijeva pretplaćivanje na multicast grupu i primanje paketa te odašiljanje paketa s adresom multicast grupe. Međutim, u kasnijim fazama projekta, kada je odašiljanje audio podataka bilo implementirano kroz aplikaciju, potreba za ovim skriptama je nestala te se komunikacija mogla testirati direktno unutar same aplikacije.

5.3. Korisničko sučelje

U ovom potpoglavlju će biti predočen sam izgled finalne aplikacije te kako s pomoću elemenata korisničkog sučelja koristiti funkcionalnosti aplikacije. Izgled aplikacije je inspiriran sličnim rješenjima za sinkronizaciju reprodukcije koji su navedeni u poglavlju. Inspiracija je također dobivena iz danas popularne platforme za strujanje glazbe, podcasta i drugih audio sadržaja, Spotify.

Pri ulazu u aplikaciju korisnik se nalazi na početnom ekranu aplikacije koji se vidi na slici dolje (Slika 5). Na vrhu zaslona se nalazi naredbena traka na kojoj su pozicionirani:

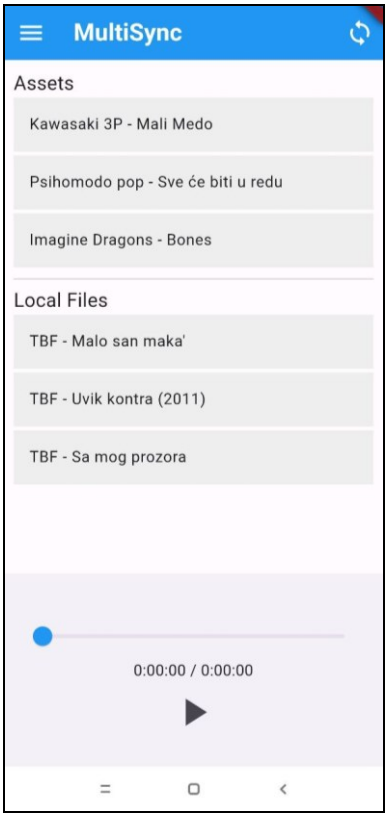
- lijevo – meni gumb koji služi otvaranju izbornika dodatnih mogućnosti
- sredina – ime aplikacije MultiSync
- desno – dugme za aktivaciju sinkronizacijskog moda, ovaj gumb aktivira uređaj koji se želi zadati kao glavni, odnosno onaj sa kojega će biti kontrolirana reprodukcija.

Srednji dio zaslona služi za prikazivanje mogućih datoteka audio zapisa, na gornjoj strani ispod teksta „Assets“ su prikazane pjesme kojima svaka aplikacija ima pristup, odnosno nazale se na svakom uređaju u sklopu aplikacije. Na donjoj strani ekrana ispod teksta „Local Files“ nalaze se svi audio zapisi koje je uređaj pronašao na postavljenoj putanji prema lokalno spremljenim audio zapisima na tom uređaju. Ta putanja se može promijeniti na svakom uređaju zasebno te će postupak za tu akciju biti predstavljen kasnije.

Na donjoj strani zaslona je predstavljeno stanje trenutne reprodukcije, sastoji se od više dijelova:

- imena trenutno reproducirane pjesme, na slici nije prikazano ime jer pjesma još nije učitana

- trenutna pozicija reprodukcije reprezentirana pomoću klizača koji je moguće pomicati kako bi se reprodukcija pokrenula u željenom trenutku
- trenutno vrijeme u pjesmi i sveukupno trajanje pjesme
- gumb za pokretanje ili zaustavljanje reprodukcije ovisno o trenutnom stanju pjesme.



Slika 5 - Početni zaslon aplikacije

Pritiskom na neku od pjesama iz „Assets“ ili „Local Files“ izbornika se pjesma učitava, prikazuje se ima pjesme te se postavlja ukupno trajanje reprodukcije (Slika 6).

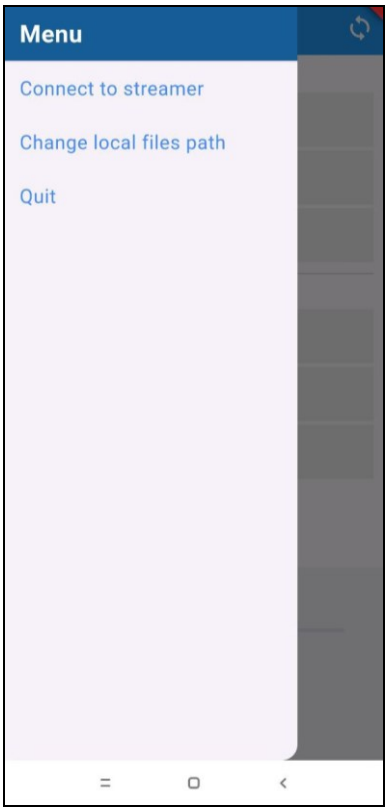
Slika 6 - Početni zaslon aplikacije s učitanom pjesmom

Pritiskom na gumb za početak sinkronizacije se mijenja boja naredbene trake i indikatora pozicije u reprodukciji koji se nalazi na klizaču (Slika 7). Time korisnik može jasno zaključiti da je sinkronizacija uređaja aktivirana te da je uređaj preuzeo ulogu glavnog uređaja u sustavu.

Slika 7 - Početni zaslon s aktiviranom mogućnosti sinkronizacije

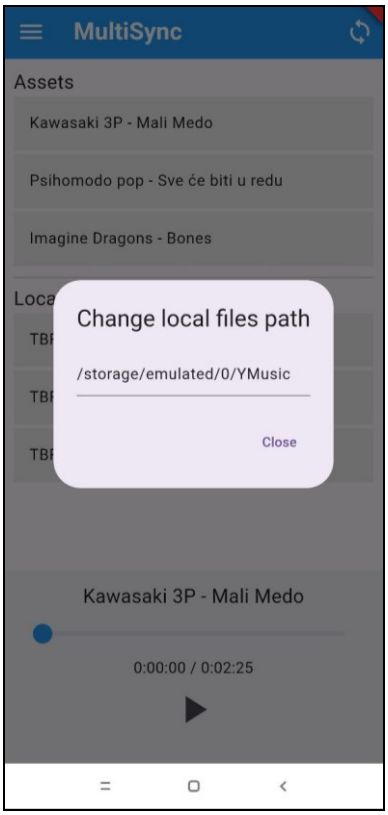
Pritiskom na gumb za otvaranje meni se otvara izbornik sa dodatnim mogućnostima sa lijeve strane zaslona (Slika 8). Na njemu su prikazane sljedeće mogućnosti:

- *Connect to streamer* - otvaranje ekrana za spajanje sa „glavnim“ uređajem kako bi se započela sinkronizirana reprodukcija
- *Change local file path* – promjena putanje do lokalno spremljenih audio zapisa
- *Quit* – izlazak iz aplikacije



Slika 8 - Početni zaslon s otvorenim izbornikom

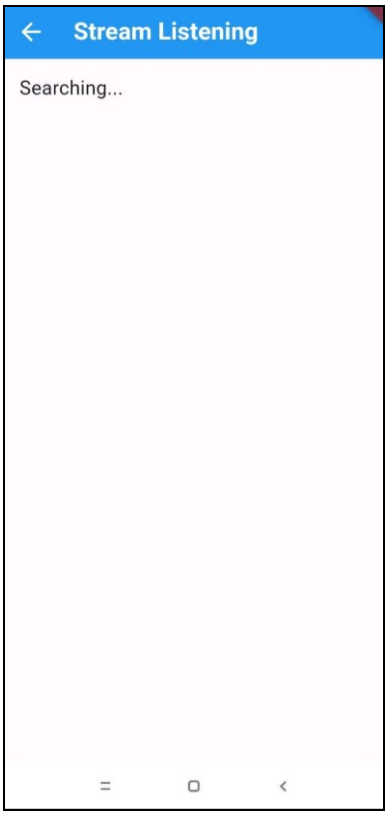
Pritiskom na mogućnost *Change local files path* se zatvara izbornik te se otvara novi skočni prozor na kojem je korisniku omogućeno da promijeni putanju do direktorija s lokalnih datoteka. Nakon pritiska gumba *Close,* skočni prozor se zatvara i ažurira se popis pjesama ispod teksta *Local Files* ovisno o odabranom direktoriju.



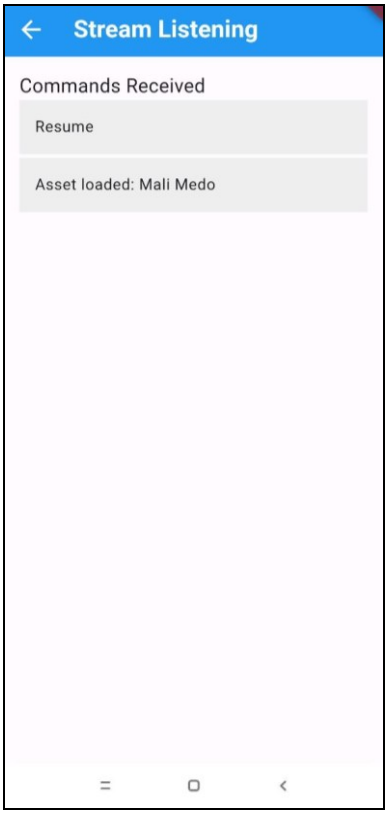
Slika 9 - Početni ekran s otvorenim skočnim prozorom za promjenu putanje

Pritiskom na mogućnost *Connect to streamer* iz izbornika mogućnosti se otvara novi prozor *Stream Listening* (Slika 10). On u svojoj naredbenoj traci ima mogućnost za povratak na početnu stranicu (strelica lijevo) te piše naziv stranice.

Nakon ulaska na stranicu, prikazuje se tekst *Searching...* koji korisniku daje informaciju da još nije pronađen glavni uređaj koji upravlja reprodukcijom. Tek nakon dolaska prvog komandnog paketa se registrira glavni uređaj te se tekst *Searching...* mijenja u *Commands Received*. Nakon toga se ispisuje svaki komandni paket kojeg aplikacija dohvaća te ga se stavlja u sivi pravokutnik kako bi korisnik mogao pratiti akcije glavnog uređaja. Svakom dobivenim paketom se u pozadini upravlja reprodukcijom ovisno o definiranom ponašanju komande. Na Slika 11 se vidi primanje komandnog paketa *setAsset* sa atributom *asset* = malimedo.mp3 kojim se postavlja audio zapis na malimedo.mp3. Nakon njega dolazi paket *resume* kojim se audio zapis pokreće u izračunato vrijeme te se kalibrira kako bi se postigla sinkronizacija.



Slika 10 - Zaslon oslušavanja mreže prije konekcije



Slika 11 - Zaslon oslušavanja mreže nakon primanja paketa

5.4. Format komandnih paketa

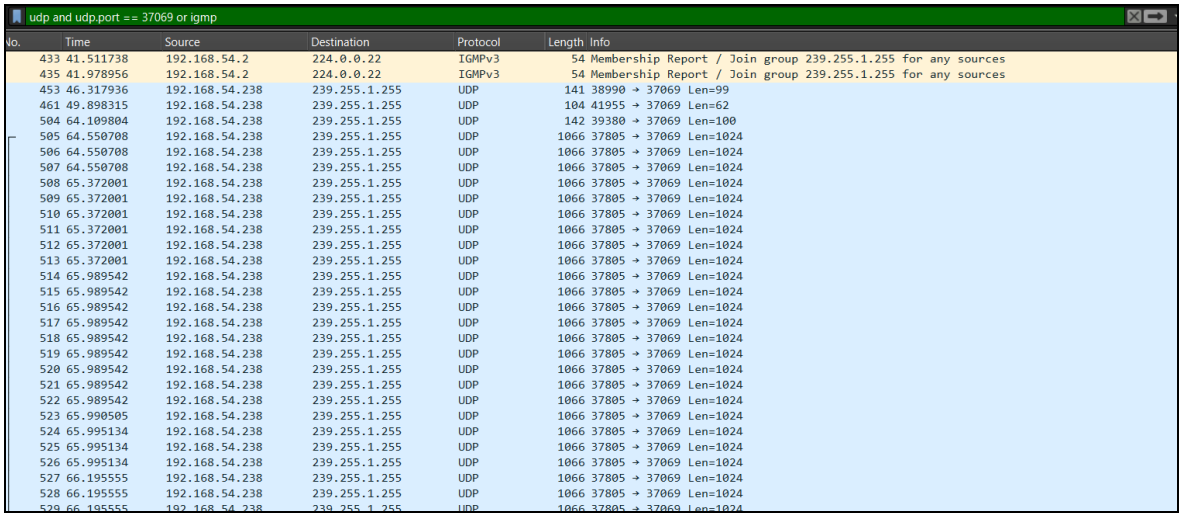
Komande za reprodukciju su zapisane u formatu JSON (engl. *JavaScript Object Notation*) datoteke. Koja se zatim kodira u niz bajtova te se šalje protokolom UDP. Svaka komanda ima svoju identifikacijsku oznaku kako bi ju se moglo odrediti pri dolasku na sporedni uređaj. Identifikacijske oznake za komande su sljedeće:

- *setPath* - 0
- *resume* - 1
- *pause* - 2
- *seek* - 3
- *sync* – 4.

U sljedećem tekstu se vidi izgled takve JSON datoteke koja predstavlja komandu *Resum* zbog koje će sporedni uređaji pokrenuti reprodukciju za 200 milisekundi gledajući po unutarnjem satu pošiljatelja:

```
{
    "command": 1,
    "senderDateTime": "2024-05-01T12:00:00",
    "actionDelay": 200
}
```

Koristeći alat Wireshark [24], možemo prikazati sam mrežni promet koji je snimljen na prijenosnom računalu nakon što se pridružio multicast grupi. Promet generira mobilni uređaj koji je spojen na istu mrežu te započinje rad aplikacije (Slika 12).



Slika 12 - Slika zaslona aplikacije Wireshark nakon snimanja mrežnog prometa

Prva dva paketa protokola IGMP šalje prijenosno računalo kako bi se pridružilo multicast grupi. Nakon što je uspješno pridruženo, od mobilnog uređaja privatne IP adrese 192.168.54.238 započinje prihvaćati pakete protokola UDP preko multicast grupe 239.255.1.255. Prva tri paketa koja prihvaća su manje veličine što ukazuje na to da prenose komande za reprodukciju. Ostali paketi su veličine 1066 bajtova što znači da su to podatkovni paketi koji prenose samu audio datoteku za reprodukciju.

6. Mjerenje kvalitete usluge

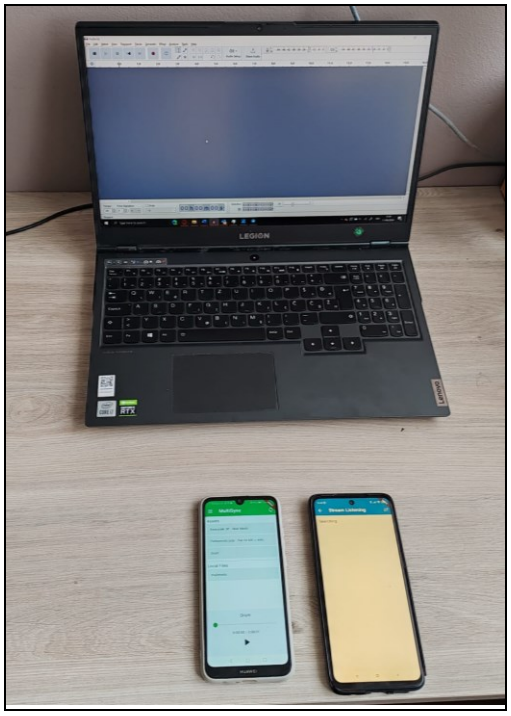
U ovom poglavlju opisana je metoda testiranja kvalitete usluge (engl. *Quality of Service*) konačne verzije aplikacije. Tijekom testiranja, fokus je bio na mjerenju jednog ključnog parametra: vremenske razlike u reprodukciji zvuka između dva uređaja.

6.1. Metoda mjerenja

Mjerenja su se vršila pomoć prijenosnog računala i dva mobilna uređaja:

- A) HUAWEI Y6 2019
- B) Redmi Note 9S.

Mobilni uređaji su postavljeni na istu udaljenost od računala (10 cm) te je započet rad aplikacija na oba uređaja. Uređaju A je dodijeljena uloga glavnog uređaja, dok je uređaju B dodijeljena uloga sporednog uređaja. Na prijenosnom računalu je pokrenuta aplikacija Audacity koja ima mogućnost snimanja zvuka. Svi uređaji pripremljeni za eksperiment se mogu vidjeti na slici 13.

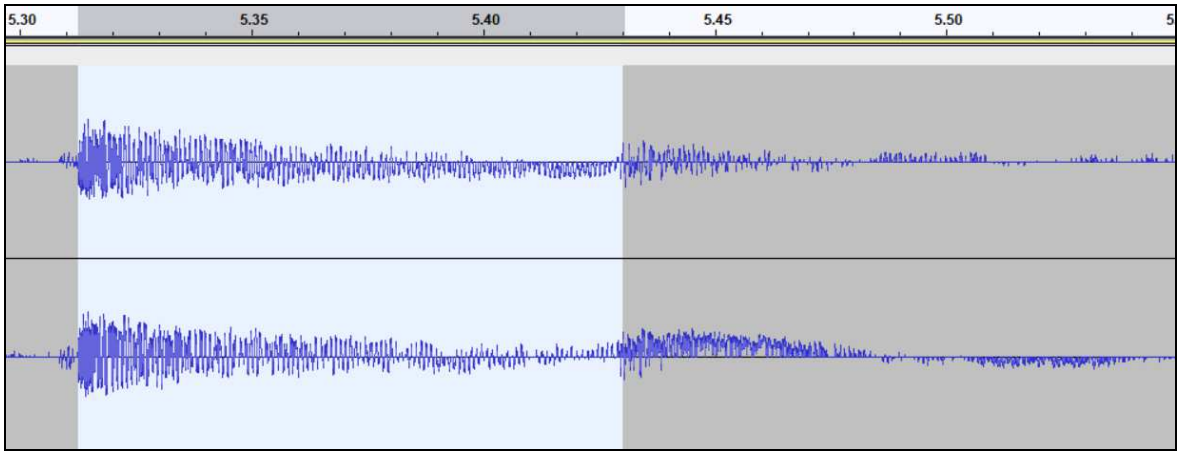


Slika 13 - Slika postavljenog eksperimenta

Eksperiment se ponavlja za 3 različite konfiguracije mehanizama kako bi se testirala efikasnost mehanizama sinkronizacije, a u svakom slučaju se izvodi 10 mjerenja:

1) slučaj u kojem niti jedan mehanizam sinkronizacije nije aktivan, glavni uređaj odašilje kontrolni paket i pokreće akciju, a sporedni uređaj pokreće akciju čim paket pristigne.
2) slučaj u kojem je aktivan jedino mehanizam koji sprječava pogrešku zbog kolebanja kašnjenjem na način da sinkronizira satove uređaja. Na taj način će uređaji pokrenuti akcije u približno isto POSIX vrijeme.
3) Slučaj u kojem su aktivni oba mehanizma sinkronizacije, upravljanje kolebanjem kašnjenja te različita brzina reprodukcije audio zapisa. Ovim slučajem se uzima u obzir da uređaji započinju akcije u isto vrijeme te usporavaju ili ubrzavaju reprodukciju ovisno o tome koliko vremena im je trebalo za sam početak reprodukcije.

Pri početku svakog mjerenja se započinje snimanje zvuka sa prijenosnog računala uz pomoć aplikacije Audacity [20], zatim glavni uređaj pokreće kratkotrajnu audio snimku Drum.wav, na kojoj se čuje jedino udarac bubnja. Pošto je audio kratak i ima nagle promjene u zvuku, lako ga je uvidjeti i analizirati na snimci zvuka koja se u aplikaciji Audacity može vidjeti na Slika 144. Na slici je označena razlika u vremenu snimljenog prvog i drugog udarca bubnja te se tako određuje razlika u vremenu reprodukcije dva uređaja mjerena s preciznošću 5 milisekundi.



Slika 14 - Slika zaslona aplikacije Audacity nakon snimanja zvuka eksperimenta

6.2. Rezultati mjerenja

Nakon ponovljenih 10 mjerenja za 3 različite konfiguracije mehanizama sinkronizacije, dobili smo sljedeća mjerenja koja su prikazana u tablici 1:

Tablica 1 – ispis mjerenja vremenske razlike u reprodukciji zvuka između dva uređaja (u ms) za 3 različite konfiguracije mehanizama s preciznošću 5 ms

Redni broj mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
(1) Bez mehanizama	70	70	120	45	15	100	135	80	50	80
(2) Prvi mehanizam	10	25	15	10	15	35	30	20	20	40
(3) Oba mehanizma	15	25	25	5	30	10	20	15	20	15

U tablici 2 su navedene prosječne vrijednosti, vrijednost standardne devijacije, minimalne vrijednosti te maksimalne vrijednosti za pojedinačne konfiguracije mehanizama:

Tablica 2 – prosječne vrijednosti, standardne devijacije, minimalne i maksimalne vrijednosti mjerenja

	Prosječna vrijednost (ms)	Standardna devijacija (ms)	Minimalna vrijednost (ms)	Maksimalna vrijednost (ms)
(1) Bez mehanizama	76,5	33,8	15	120
(2) Prvi mehanizam	22	9,8	10	40
(3) Oba mehanizma	18	7,1	5	30

Iz dobivenih rezultata možemo zaključiti da su implementirani mehanizmi sinkronizacije značajno poboljšali kvalitetu usluge. U slučaju kada mehanizmi nisu bili aktivni, prosječna razlika u vremenu reprodukcije bila je 76,5 ms, što ljudsko uho može prepoznati bez problema i može dovesti do neugodnog iskustva korisnika. Prva konfiguracija također ima relativno veliku standardnu devijacija što znači da kvaliteta usluge znatno varira. To se vidi i iz najmanje izmjerene vrijednosti 15 ms, što zapravo može pružiti ugodno korisničko iskustvo.

Aktiviranjem mehanizma koji sprječava pogreške uzrokovane kolebanjem kašnjenja, prosječna razlika se smanjila na 22 ms, što predstavlja značajno poboljšanje. Istovremeno,

smanjenje standardne devijacije omogućava znatno veću pouzdanost pri sinkronizaciji reprodukcije.

Dodatnim aktiviranjem mehanizma za kalibriranje unutarnje brzine reprodukcije, prosječna razlika se smanjila na 18 ms, što nije znatno poboljšanje od prethodne konfiguracije, no svejedno pomaže iskustvu korisnika. Također se smanjuje i standardna devijacija mjerenja što dodatno pospješava pouzdanosti slušanja pri velikom opterećenju mreže ili velikoj razlici u procesorskoj brzini uređaja.

7. Zaključak

U današnjem svijetu, mehanizmi za sinkronizaciju su ugrađeni u mnoštvo uređaja, od slušalica do sofisticiranih zvučnika visoke kvalitete. Međutim, Bluetooth tehnologija, koja se često koristi za sinkronizaciju zvuka, ima svoja ograničenja u broju uređaja koji se mogu povezati i u postizanju besprijekorne sinkronizacije.

Ovaj rad se bavi izazovima sinkronizacije zvuka na više uređaja i nudi alternativno rješenje: distribuirani sustav za sinkroniziranu reprodukciju zvuka korištenjem lokalne računalne mreže. Prednost ovog sustava u odnosu na Bluetooth tehnologiju je skalabilnost. U današnjem svijetu, kada gotovo svatko posjeduje pametni uređaj, često se susrećemo s velikim brojem takvih uređaja na istom mjestu. Spajanjem i sinkronizacijom tih uređaja kroz lokalnu mrežu može se stvoriti praktički besplatan sustav zvučnika s višestrukim izvorima zvuka.

Danas već postoje mnoga gotova rješenja koja pokušavaju riješiti ovaj problem, no za potrebe ovog projekta su definirani stroži zahtjevi aplikacije koji su doveli do implementacije vlastitog rješenja kroz cross-platformu Flutter. Model rješenja uzima u obzir performanse mreže, kolebanje kašnjenjem paketa u mreži te različito vrijeme reprodukcije audio zapisa kod različitih uređaja.

Iako je prosječna razlika u vremenu reprodukcije nakon primijenjenih algoritama sinkronizacije ispod čujnosti ljudskog uha, ponekad se ta razlika poveća te ljudsko uho ipak zamijeti razliku što smanjuje iskustvenu kvalitetu slušatelja. Kao buduće proširenje projekta, trebalo bi proširiti model reprodukcije kako bi se razlika u reprodukciji smanjila na uistinu nečujnu razinu. Također, dodatan zahtjev bi bio uvesti mogućnost da i sporedni uređaji mogu upravljati reprodukcijom, što sporazumijeva kompleksniji mehanizam sinkronizacije.

8. Literatura

[1] JBL, *JBL FLIP 6,* https://www.jbl.com/bluetooth-speakers/FLIP-6-.html, pristup: 10.6.2024.

[2] Lifewire, *Bluetooth vs. Wi-Fi: What's the Difference?*, https://www.lifewire.com/bluetooth-vs-wi-fi-4088218, pristup: 10.6.2024.

[3] Telink, An Introduction to True Wireless Stereo (TWS) Technology, https://www.telink-semi.com/introduction-true-wireless-stereo/, prisput: 20.3.2024.

[4] Amp Me Inc., Aplikacija AmpMe, https://www.ampme.com, pristup: 30.3.2024.

[5] JekApps, Aplikacija SoundSeeder, https://soundseeder.com, pristup: 30.3.2024.

[6] M. Urech, "Distributed Speaker Synchronization," 2016., https://pub.tik.ee.ethz.ch/students/2016-FS/SA-2016-36.pdf

[7] Google Play stranica za dohvaćanje aplikacije AmpMe, https://play.google.com/store/apps/details?id=com.amp.android, pristup: 30.3.2024.

[8] BasuMallick, C. *TCP vs. UDP: Understanding 10 Key Differences,* https://www.spiceworks.com/tech/networking/articles/tcp-vs-udp/, pristup: 20.3.2024.

[9] Fiedler, G. *Why can't I send UDP packets from a browser?,*

https://gafferongames.com/post/why_cant_i_send_udp_packets_from_a_browser/, pristup: 29.3.2024.

[10] Flutter, *Build for any screen*, https://flutter.dev, pristup: 20.3.2024.

[11] B. P. Crow, I. Widjaja, J. G. Kim and P. T. Sakai, "IEEE 802.11 Wireless Local Area Networks," IEEE Communications Magazine, vol. 35, no. 9, pp. 116-126, Sept. 1997.

[12] – Firewall.cx, *Multicast - Understand How IP Multicast Works*, https://www.firewall.cx/networking/network-fundamentals/network-multicast.html, pristup 30.3.2024.

[13] EKTECH, *What Is Low Latency and How To Achieve It*, https://ektech.com.my/what-is-low-latency-and-how-to-achieve/, pristup: 13.4.2024.

[14] Bhargava, K. *What is Epoch time ? Also know as UNIX time or POSIX time,* https://medium.com/@kushal.bhargava01/what-is-epoch-time-also-know-as-unix-time-or-posix-time-bd8efd1a491, pristup: 13.4.2024.

[15] Wikipedia, *Marzullo's algorithm*, https://en.wikipedia.org/wiki/Marzullo%27s_algorithm, pristup: 13.4.2024.

[16] Wikipedia, *Network Time Protocol,*

https://en.wikipedia.org/wiki/Network_Time_Protocol, pristup: 13.4.2024.

[17] Android Developers, *Audio latency*, https://developer.android.com/ndk/guides/audio/audio-latency, pristup: 13.4.2024

[18] Flutter, *Impeller rendering engine*, https://docs.flutter.dev/perf/impeller, pristup: 21.5.2024

[19] Python Software Foundation, *python*, https://www.python.org, pristup: 30.5.2024.

[20] Muse Group & contributors, *Audacity*, https://www.audacityteam.org, pristup: 23.5.2024.

[21] JGraph, draw.io, https://www.drawio.com, pristup: 25.3.2024.

[22] WebRTC, *Real-time communication for the web*, https://webrtc.org, pristup: 20.3.2024.

[23] B. Cain, S. Deering, Cereva Networks, S. Deering, I. Kouvelas, Cisco Systems, B. Fenner, AT&T Labs – Research, A. Thyagarajan, Ericsson, „Internet Group Management Protocol, Version 3,“ RFC 3376, Oct. 2002., https://datatracker.ietf.org/doc/html/rfc3376

[24] Wireshark Foundation, *The world's most popular network protocol analyzer*, https://www.wireshark.org, pristup: 13.6.2024.

Sažetak

Ovaj rad istražuje izazove sinkronizacije zvuka na više uređaja i predlaže distribuirani sustav za sinkroniziranu reprodukciju zvuka putem lokalne mreže. Izlažu se slična rješenja i stroži zahtjevi koje nova implementacija mora ispunjavati. Definiran je model koji uzima u obzir performanse mreže i uređaja za reprodukciju. Opisana je implementacija u cross-platformi Flutter i jeziku Dart, te je dan uvid u korisničko sučelje aplikacije. Po završetku implementacije je izvedeno mjerenje kvalitete usluge u kojoj se mjerila razlika u vremenu reprodukcije po čemu se donio zaključak o uspješnosti implementacije te je dan zaključak rada u kojem se navode ideje za buduću nadogradnju.

Ključne riječi: sinkronizacija, distribuirani sustav, lokalna mreža, mobilni uređaj, Flutter, Dart

Summary

This thesis investigates the challenges of multi-device audio synchronization and proposes a distributed system for synchronized audio playback over a local network. Similar solutions are presented, as well as the stricter requirements that the new implementation must meet. A model is defined that takes into account network and playback device performance. The implementation is described in the cross-platform Flutter framework and the Dart language, and an overview of the application's user interface is provided. After the implementation was completed, a quality of service measurement was performed, in which the difference in playback time was measured, which led to the conclusion about the success of the implementation. A conclusion was also given, which includes ideas for future upgrades.

Keywords: synchronization, distributed system, local network, mobile device, Flutter, Dart