Tokovnik pri chiptune glasbi

Primož Bajželj, Fakulteta za računalništvo in informatiko, 2012

***Povzetek*—Včasih nas nostalgija lahko ponese nekaj let ali celo desetletje v preteklost. Chiptune je glasba, ki se je pojavila več kot desetletje v preteklosti in sega celo v prejšnje tisočletje. V nadaljevanju je predstavljen tokovnik za prenos že ustvarjene glasbe, ki je bil razvit kot del celotnega projekta.**

***Index Terms*—Tokovnik, chiptune, prenos podatkov**

***Abstract*—every once in a while, nostalgy can takes us few years or even a decade back into the time. Chiptune is an example of a music that appeared more than a decade ago and it goes back even into the previous millennium. In this paper streamer as the part of the whole project is presented.**

***Index Terms*—Streamer, chiptune, data streaming**

I. Uvod

STREAMANJE je način prenosa pri katerem končni uporabnik lahko neprestano dostopa do določenega vira katerega nek ponudnik ponuja. Pri tem velja, da se lahko sam prikaz začne še preden je prenesen celoten vir - torej predvajanje filma se lahko začne še predno imamo cel film na računalniku. Primeri različnih streamanj pa so npr. radio, televizija ali cene delnic.

Za pretok po omrežju je najbolj uporabljen referenčni model TCP/IP, ki velja za de facto standard. Razdeljen je na 4 sloje in v primerjavi s starejšem OSI modelom odstrani prezentacijski in sejni sloj ter združi fizično in povezavno plast. Višje plasti vedno uporabljajo funkcije nižjih plasti.

- Naloga fizične plasti je, da skrbi za fizični prenos podatkov.
- Mrežna plast skrbi za transparentno pošiljanje podatkov med mrežami. Dostava pri tem ni zagotovljena, niti vrstni red dostave. Povezava s povezavnim slojem je protok ARP, ki skrbi za pretvorbo med MAC in IP naslovi.
- Prenosna plast skrbi za povezavni in brez-povezavni način delovanja. Tu najdemo protokola TCP in UDP.
- Aplikacijska plast služi uporabi standardnih in nestandardnih aplikacij kamor spada tudi naša aplikacija

Streamanje se lahko izvaja na različnih protokolih, kot sta npr. UDP ali TCP. Za namene zvoka in videa pa tudi obstaja namenski standard RTP v povezavi z RTCP.

TCP se smatra za zanesljiv protokol s čimer so vsi paketi dostavljeni in so v pravem vrstnem redu. Pri tem se sicer lahko pojavi težava ob večjih motnjah na omrežju, saj se mora paket takrat še enkrat ali večkrat poslati s čimer se streamanje ustavi. Zato se v takem primeru ustvari nek buffer, ki to preprečuje - s tem se sicer ustvari določen zamik.

Druga možnost pa je UDP, ki velja za nezanesljiv protokol, kar pomeni, da ne zagotavlja, da bo paket zagotovo prišel na cilj kot tudi ne, da bodo prišli v pravem vrstnem redu. Te težave se potem rešujejo na nivoju aplikacije, kjer npr. pazimo, da pakete pred predvajanjem uredimo po vrsti.

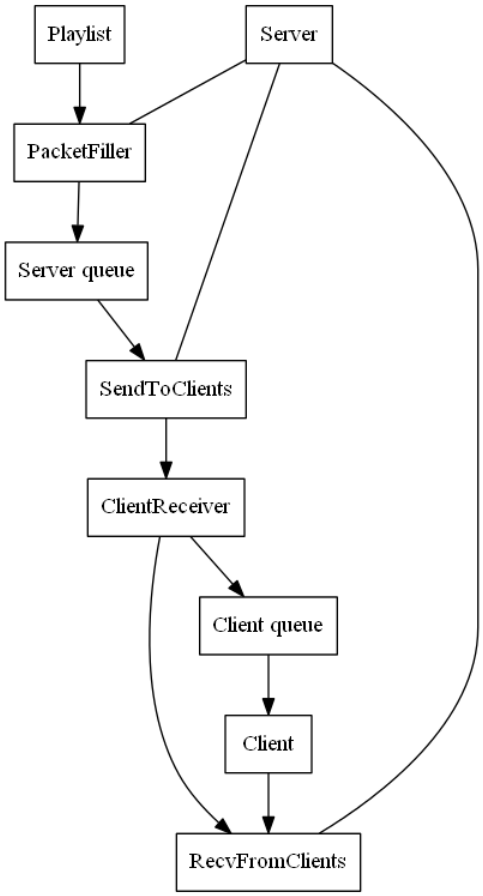

Figure 1. Prikaz sestave programske rešitve

Primeri drugih tokovnikov so Adobe Flash Player, Windows Media Player, realPlayer, Quick time in drugi. Različni ponudniki toka ponavadi ponujajo tako svoje rešitve tokovnikov in vsaj še 2 od prej naštetih za primer internetnega radia. Sami smo se odločili za izdelavo lastnega tokovnika, ki bo deloval s protokolom UDP.

II. Opis programske rešitve

Slika 1 prikazuje sestavo programske rešitve, ki je opisana v nadaljevanju.

A. Paket

Preko medmrežja pošiljamo serializiran objekt Packet, ki služi za prenos zvoka in formata. Med drugim je v paketu tudi nekakšen ID oz. zaporedna številka s katerim se preverja ali je to res naslednji paket. Ob sami konstrukciji se objekt z zvokom oz. formatom pretvori v zapis bajtov, ki se uporabljajo za kasnejše pošiljanje.

1) Zvok: Komponenta AudioInputStream nam omogoča, da pretok ne zgolj preusmerimo v izhod za zvok, ampak tudi v tok bajtov do katerih nato lahko dostopamo. S pomočjo vsebine formata nato lahko vse skupaj razdelimo na določene dele, ki jih nato lahko uporabimo za pošiljanje preko tokovnika. Pri tem je pametno, da delom dodelimo pametno velikost, kar lahko na strani klienta vpliva na kakovost zvoka. Pri tem moramo upoštevati izgube, kjer je dobro, da so paketi čim manjši (zaradi primera izgub oz. zamujanja paketov je na zvoku manj izgube, če so paketi manjši) ter število poslanih paketov, kjer je bolje, da so paketi čim večji (na ta način se vse skupaj pohitri).

2) Format: AudioInputStream poleg zvoka vsebuje tudi informacijo o formatu v razredu AudioFormat. Sam razred AudioFormat ni serializiran, kar pomeni, da ga ne moramo direktno pretvoriti v bajte ter poslati preko omrežja. Zaradi tega smo izdelali poseben razred, ki omogoča serializiranje podatov in s tem tudi pretvorbo ter prenos po omrežju.

III. Klient

Klient je sestavljen iz dveh delov:

- Client za inicializacijo in obdelavo paketov.
- ClientReceiver za prejemanje paketov.

A. Inicializacija

Klient ima nalogo, da poleg inicializacije ostalih objektov inicializira tudi nit ClientReceiver, ki služi, da se prejemanje paketov dogaja ločeno neodvisno od ostalega dela. Poleg tega pa ima tudi zelo pomembno nalogo, da pošlje določeno začetno kodo s čimer označi strežniku, da želi prejemati zvok.

B. Prejemanje paketov

Prejemanje paketov se dogaja v ločeni niti, kar nam omogoča ločeno delovanje predvajanja in prejemanja paketov. Na ta način se paketi poskušajo, najhitreje prejemati, kar je ključno za hitrost predvajanja. Sami prejeti paketi se dodajajo v določeno vrsto, ki je v drugi niti spraznjena. Poleg tega ima ta nit tudi nalogo, da v primeru, če je prejeti paket format, prejetje le tega sporoči strežniku.

C. Obdelava paketov

Prvi prejeti paket je paket formata. Le ta je v niti za prejemanje paketov potrjen strežniku, da ne pride do ponovnega pošiljanja. V tej niti pa se ob prejemu formata nastavi izhod na dobljen format, s čimer je v nadaljevanju možen vpis podatkov za zvok.

Naslednji paketi so paketi z zvokom, kjer je kot prvo potrebno preverjanje ali gre res za naslednji zaporedni paket. Vsi paketi so v osnovi uvrščeni v nek array s katerega se vsakič vzame najstarejši naslednji paket. Ob tem obstaja tudi možnost, da kakšen paket pride z zamudo, kar pomeni, da ta paket ne bo nikoli predvajan oz., da bo izrisan ob naslednji iteraciji.

Poleg tega ima klient v tem delu nalogo, da sporoča ali je morda preveč zasičen. Ob določenem presegu minimalnega števila strežniku sporoči naj začne hitreje pošiljati pakete oz. ob presegu maksimalnega števila, da naj jih začne pošiljati počasneje. S tem se prepreči, da v primeru, ko strežnik pošilja oz. klient hitreje prejema pakete kot jih klient lahko predvaja, da bi število paketov raslo v neskončnost.

IV. Strežnik

Strežnik v osnovi sestoji iz treh glavnih niti, ki služijo za pravilno funkcioniranje aplikacije:

- PacketFiller, ki skrbi za dodajanje novih paketov.
- RecvFromClients, ki služi za prejemanje iz strani klientov.
- SendToClients, ki služi za pošiljanje klientom.

A. Dodajanje novih paketov

Za pridobivanje novih paketov služi poseben razred Playlist, katera med drugim omogoča tudi dodajanje novih datotek oz. imenikov. Po odprtju posamezne datoteke se s pomočjo razreda AudioInputStream kot prvo pridobi format. Za zbiranje paketov služi posebna vrsta kamor je paket formata v nadaljevanju tudi vrščen. Formatu sledi branje posameznih delov zvoka, ki se prav tako v obliki paketa zapišejo v prej omenjeno vrsto. Po koncu določene datoteke ima razred Playlist tudi nalogo, da odpre novo datoteko, kjer je tudi glavna možnost razširitve, ki bi npr. omogočala glasovanje za naslednjo skladbo.

B. Prejemanje s strani klientov

Zaradi različnih potreb komunikacije strežnik tudi prejema pakete iz strani klientov. Sami klienti so identificirani z ip naslovom, kar bi v primeru poslušanja z istega IP naslova prineslo težave. V nadaljevanju bi bila tako potrebna dodatna identifikacija, ki ne bi delovala zgolj na IP naslov, ampak bi se npr. klientu iz strani strežnika dodelila določena številka. V uporabi so naslednje naloge:

- Strežnik v osnovi ne ve komu mora pošiljati pakete, kar pomeni, da mu mora le to sporočiti klient. Znotraj te zanke zato strežnik prejema pakete in v primeru zahtevka za priključitev začne pošiljati pakete klientu.
- Dodatno se v tej zanki tudi prejema ali je klient uspešno prejel format. Potrditev le tega je potrebna, da ga strežnik ne pošilja ponovno, kar je popolnoma nepotrebno in bi zgolj obremenjevalo linijo.
- Zaradi napačne hitrosti pošiljanja se lahko na strani klienta pojavi problem nasičenja ali da mu primankuje paketov za predvajanje. V obeh problemih zaradi tega lahko pride do neželenega učinka počasnega igranja ali prekinjanja. V prvem primeru zaradi preobremenitve, medtem ko pri drugem ta učinek lahko nastane, ker se paketi prepočasi predvajajo. Zaradi tega ima strežnik tudi dodatno nalogo, da spremlja, če mu klient pošlje zahtevek za počasnejše ali hitrejše pošiljanje. Vse skupaj je pri tem omejeno na določeno minimalno in maksimalno vrednost, saj bi v nasprotnem primeru nekdo to lahko izkoristil kot varnostno luknjo ter zasičil ali zaustavil predvajanje pri klientih.

C. Pošiljanje klientom

Za vsakega priključenega klienta se v tej niti pošiljajo paketi. Posamezen paket se vzame iz vrste, kjer so paketi nato pa se le ta pošlje klientom. V primeru, ko se za klienta ugotovi, da ni potrdil formata se mu pošlje le ta.

V. Zaključek

Narejen tokovnik sicer ni popoln, a lahko služi kot osnova za nadaljnje delo. Ob sami izdelavi smo se sicer dotaknili glavnih težav, ki se pojavijo s tokovnikom, kjer pa se pokaže, da je pomembno, da pravilno obravnavamo lastnosti protokola, da je zvok čimbolj kvaliteten. Zanimiva je predvsem relativno enostavna možnost nadgradnje za glasovanje, ki ni velikokrat prisotna pri različnih drugih tokovnikih.

Literatura

[1] T. Denis, *Računalniška zvočna produkcija, kopije prosojnic*. FRI UL, 2011.

[2] Wikipedia, "Streaming media — Wikipedia, the free encyclopedia," 2012, [Online; accessed 20-Maj-2012]. [Online]. Available: en.wikipedia.org/wiki/Streaming_media