Универзитет у Београду

Електротехнички факултет



Имплементација дистрибуираног система

за играње игара у облаку

Мастер рад

|  |  |
| --- | --- |
| Ментор: | Кандидат: |
| проф. др Марко Мишић | Александар Динчић 2022/3055 |

Београд, октобар 2025.

Садржај

[Садржај 2](#_Toc209780760)

[1. Увод 3](#_Toc209780761)

[2. Опис система 5](#_Toc209780762)

[2.1. Преглед постојећих сервиса 5](#_Toc209780763)

[2.2. Захтеви система 7](#_Toc209780764)

[3. Архитектура система 9](#_Toc209780765)

[3.1. Фронтенд 9](#_Toc209780766)

[3.2. Контролер 10](#_Toc209780767)

[3.3. Размена порука 10](#_Toc209780768)

[3.4. Агенти 10](#_Toc209780769)

[3.5. Складиште објеката 11](#_Toc209780770)

[4. Пренос слике и звука 12](#_Toc209780771)

[*4.1.* О библиотеци *GStreamer* 12](#_Toc209780772)

[*4.2.* О протоколу *WebRTC* 14](#_Toc209780773)

[4.3. Слика 15](#_Toc209780774)

[4.3.1. Дохватање 15](#_Toc209780775)

[4.3.2. Кодирање 15](#_Toc209780776)

[4.3.3. Пренос 15](#_Toc209780777)

[4.4. Звук 16](#_Toc209780778)

[4.4.1. Дохватање 16](#_Toc209780779)

[4.4.2. Кодирање 16](#_Toc209780780)

[4.4.3. Пренос 16](#_Toc209780781)

[5. Пренос команди 17](#_Toc209780782)

[6. Ток сесије 18](#_Toc209780783)

[7. Евалуација 19](#_Toc209780784)

[8. Закључак 20](#_Toc209780785)

[Литература 21](#_Toc209780786)

[Списак скраћеница 23](#_Toc209780787)

[Списак слика 24](#_Toc209780788)

[Списак табела 25](#_Toc209780789)

1. Увод

Развој видео игара уско је повезан са развојем рачунарског хардвера. Са порастом могућности хардвера расту визуелна и механичка комплексност игара, али и системски захтеви за њихово покретање. Ово значи да су савремени наслови ограничени на најновије играчке конзоле и рачунаре са напреднијим процесором и графичком картицом, као и већом количином радне и складишне меморије, чиме се значајно сужава скуп корисника којима су такве игре приступачне. Са друге стране, развој брзих и поузданих интернет конекција, напредних техника кодирања и преноса слике и скалабилних серверских инфраструктура омогућили су појаву првих сервиса за играње игара у облаку. При коришћењу ових система, игра се извршава на некој од серверских машина, док се слика и звук преносе до корисничког уређаја у реалном времену. Корисник интерагује са игром тако што се унете команде преносе и извршавају на серверу. Главне погодности оваквог приступа су загарантоване перформансе игре, без обзира на њену захтевност, као и тренутан приступ великом асортиману игара са било ког уређаја, без потребе за инсталацијом. Овај рад се бави имплементацијом једног таквог система. Циљ је развити прототип који кориснику омогућава играње одабраних игара из веб претраживача.

Рад је сачињен од осам поглавља. Уводно поглавље садржи излагање о концепту играња видео игара у облаку, његовом значају, као и кратко представљање сваког од поглавља рада.

Друго поглавље садржи преглед значајнијих постојећих решења, са циљем представљања одлика оваквих система. Затим ће, на основу тога, бити успостављени функционални захтеви система којег желимо да имплементирамо.

Треће поглавље представља сажети приказ добијеног решења и његову архитектуру. Такође ће бити описане технологије коришћене за његову реализацију.

Наредна поглавља детаљно описују имплементацију система, са освртом на најзначајнијим проблемима са којима смо се сусрели током имплементације. Четврто поглавље бавиће се преносом слике и звука. Биће приказане коришћене технике дохватања, кодирања и преноса до корисника, са нагласком на смањењу кашњења.

Пето поглавље посвећено је проблему преноса корисничких команди. Описаћемо начин на који се унос преноси до серверске машине и затим симулира у оквиру игре.

Шесто поглавље бави се самом корисничком сесијом, укључујући извршавање игре и управљање корисничким датотекама. Детаљно ће бити описана комуникација и проток порука између корисника, серверских компоненти и базе података, од отварања сесије до њеног затварања.

У седмом поглављу приказана је евалуација добијеног решења. Биће описан метод тестирања функционалности система, као и добијени резултати. Такође ће бити извршено мерење метрика од значаја за систем, попут кашњења преноса и времена покретања сесије.

На крају, резимираћемо рад и добијене резултате. Затим ћемо размотрити могућности за побољшање система.

1. Опис система

У овом поглављу дефинисаћемо одлике система којег желимо да имплементирамо. У сврху упознавања са развојем и кључним могућностима оваквих система, најпре ћемо представити неколико историјски значајних примера. Затим ћемо дефинисати захтеве које наш систем треба да испуњава.

* 1. Преглед постојећих сервиса

Први пример сервиса за играње игара у облаку појавио се 2003. године под именом *Game Cluster* (*G-Cluster*). Нагласак је био на игрању игара преко сет-топ бокс уређаја, а корисницима су се нудиле одабране лиценциране игре. Ове игре су биле средње комплексности, а за њихово прилагођавање платформи неопходно је било лиценцирање и измена изворног кода. Како би се постигле гарантоване перформансе платформа је сарађивала са добављачима интернет услуга, који су обезбеђивали серверске машине за извршавање игара. [1]

Следећи корак у овој сфери направили су сервиси *OnLive* и *Gaikai* 2010. године. Развој паметних уређаја и техника кодирања слике омогућио је приступ овим сервисима из веб претраживача са личних рачунара, паметних мобилних уређаја и телевизора. Понуда игара била је већа и покривала је нове, комплексније игре. Док је *OnLive* платформа нудила комплетне игре уз претплату и њихово изнајмљивање, понуда платформе *Gaikai* је била ограничена на пробне верзије игара. Оба сервиса купила је компанија *Sony* и искористила их као темељ за свој *PlayStation Now* сервис, о којем ће бити речи касније. [2]



Слика 2.1.1. Сервиси G-Cluster, OnLive и Gaikai [3][4][5]

Компанија *NVIDIA* представила је 2013. године сервис *NVIDIA GRID*, који ће касније бити преименован у *GeForce NOW*. Као водећа компанија у сфери графичких картица, на располагању имају сопствени погон напредних серверских картица, као и ранији приступ технологијама попут виртуелизације и партиционисања графичких картица. Ово се одражава на високе перформансе које сервис нуди, а снага хардвера којим корисник располаже зависи од нивоа његове претплате. Поред игара које су у понуди у оквиру сервиса, једна од иновација јесте интеграција са постојећим платформама за играње игара као што је *Steam*, што омогућава кориснику да игра било коју игру коју већ поседује на тим платформама.[6] За серверске машине користи се модификовани *Linux* оперативни систем.[7] Пренос слике и звука користи *Real-time Transport Protocol* (*RTP*), док се за кодирање слике користи стандард *H.264*.[8]

Компаније које поседују играчку конзолу као један од својих главних производа често их користе као основу за сервисе оваквог типа. Ово је случај са компанијом *Sony* и њиховим сервисом *PlayStation Now*, у чијој инфраструктури се користе блејд сервери базирани на њиховој *PlayStation* конзоли.[9] Сличан приступ користи и компанија *Microsoft*, која за свој *Xbox Cloud Gaming* сервис користи хардвер базиран на *Xbox* конзоли.[10]. За коришћење оваквих сервиса обично је потребан контролер компатибилан са датом конзолом. За сву комуникацију и пренос података сервис *PlayStation Now* користи сопствени нестандардни протокол грађен преко *UDP* протокола.[8]

Последњи пример који ћемо размотрити је сервис *Stadia* компаније *Google*. Издат 2019. године, сервис је нудио одређен скуп игара са системом претплате. Сервис је имао висок ниво интеграције са *Google* производима, те је првенствено подржавао платформе попут *Chrome* и *Chromium* претраживача, или рачунара са *ChromeOS* оперативним системом. За серверску инфраструктуру, због своје распрострањености, користе се постојећи *Google* дата центри, са машинама које користе *Linux* оперативни систем и измењене *AMD* графичке картице.[11] Код овог сервиса за пренос се користи стандардни *WebRTC* протокол, са подршком за *H.264*, као и новији *VP9* стандард кодирања слике.[8] Сервис је престао са радом 2023. године.



Слика 2.1.2. Сервиси GeForce NOW, PlayStation Now и Stadia [12][13][14]

* 1. Захтеви система

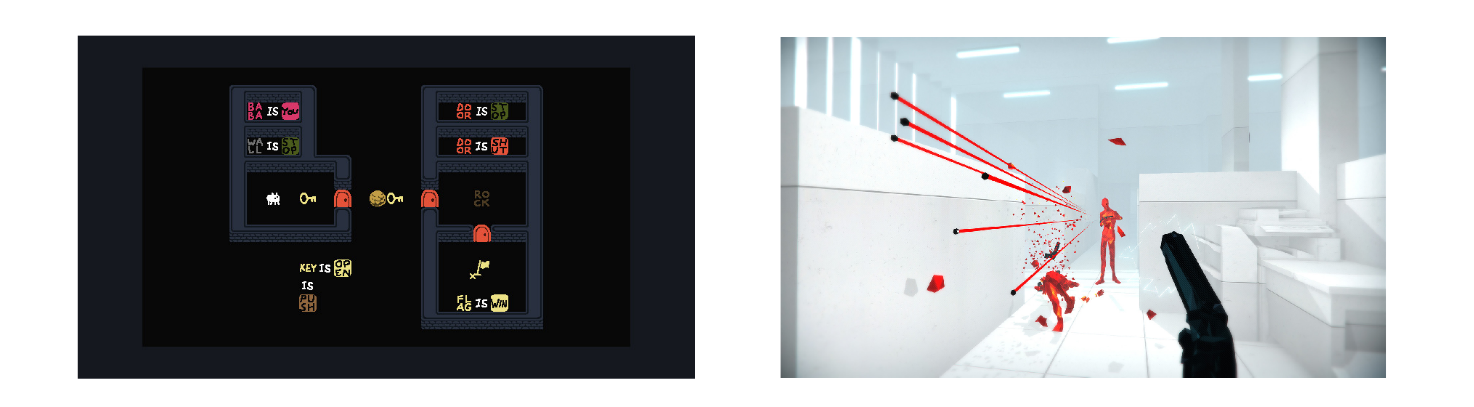
Тема овог рада јесте имплементација једног једноставног дистрибуираног система за играње игара у облаку, са циљем истраживања проблема удаљеног играња игара са системског становишта. Сервис треба бити реализован у виду веб апликације којој корисник приступа из претраживача на рачунару.

Довољно је да апликација садржи само једну страницу, која служи за играње игара. Два параметара која ова страница треба да прима јесу корисничко име и игра коју корисник жели да игра. Пошто регистрација и аутентикација корисника нису од великог значаја за елементе система које истражујемо, нећемо их имплементирати у оквиру овог рада. Уместо тога, корисничко име је довољно користити као кључ при дохватању и чувању корисничких података.

По приступу страници, игру треба покренути на некој од удаљених серверских машина и одмах започети пренос слике и звука до корисничког претраживача у реалном времену. Такође треба започети процес читања корисничког уноса, његовог преноса до серверске машине и симулације у оквиру игре. Од периферија за кориснички унос систем треба да подржава миш и тастатуру. Сесија треба да траје све док корисник не затвори прозор.

За игре које ће бити у понуди нашег сервиса изабрали смо следећа два наслова:

* *Baba is You* је логичка видео игра из 2019. године. У овој игри играч се, уносом на тастатури, креће по правоугаоној мрежи. Циљ је прећи сваки од нивоа пратећи правила, која су такође елементи мреже и којима се може манипулисати. Ова игра је изабрана због једноставног, дводимензионалног графичког приказа и ниских системских захтева, као и због своје потезне природе, што је чини толерантнијом на већа кашњења преноса и команди.
* *SUPERHOT* је акциона игра из првог лица из 2016. године. Играч користи миш и тастатуру за кретање по терену и пуцање. Циљ је елиминисати све противнике на сваком од нивоа, а играчу су на располагању одређене механике манипулације времена. Графички приказ ове игре садржи тродимензионалне елементе, што је чини захтевнијом за извршавање. Због своје акционе природе игра захтева прецизнији пренос уноса и мала кашњења. Ове особине чине ову игру меродавном за проверу могућности играња акционих и графички напреднијих игара на нашем систему.



Слика 2.2.1 Игре Baba is You и SUPERHOT [15][16]

Поред изабраних игара, систем треба бити лако проширив новим играма. Такође, за сваку од игара систем треба да чува корисников напредак. По завршетку сесије личне датотеке корисника треба сачувати како би, при покретању следеће сесије, корисник могао да настави са игром одакле је стао на претходној сесији.

Систем мора бити способан да опслужује више од једне корисничке сесије без сметњи. Важно је и да систем буде пројектован тако да буде скалабилан и да подржава једноставно проширење капацитета.

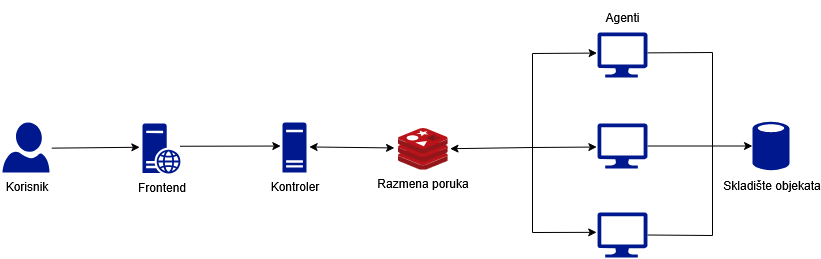
Најзначајнија метрика система за корисничко искуство јесте кашњење преноса, и од ње увелико зависи употребљивост система. Из тог разлога, приликом имплементације, неопходно је тежити што мањем кашњењу. Прихватљиво је жртвовати друге аспекте преноса, попут квалитета слике, како би се кашњење смањило на задовољавајући ниво.

Наведени захтеви описују минимални скуп функционалности које један сервис за играње игара у облаку мора да испуњава. Имплементацијом система који испуњава ове захтеве стећи ћемо разумевање начина рада најважнијих аспеката оваквих система.

1. Архитектура система

Ово поглавље садржи кратак опис нашег решења. Представићемо архитектуру система, са освртом на сваку од компоненти система, њихову улогу, као и начин повезивања између њих. Такође ће бити набројане технологије коришћене у имплементацији.

Решење је израђено у виду једног дистрибуираног система сачињеног од више независних компоненти. На слици 3.1. дата је визуелизација архитектуре система са свим његовим компонентама. Сажето излагање о свакој од приказаних компоненти биће дато у потпоглављима која следе.



Слика 3.1. Архитектура система

* 1. Фронтенд

Фронтенд је компонента у виду веб сервера који представља приступну тачку систему за корисника. У питању је једноставна веб апликација која нуди једну страницу, преко које корисник са унетим корисничким именом игра игру по жељи. По отварању странице, она шаље захтев контролеру за започињање сесије, а затим успоставља *peer-to-peer* везу са додељеним агентом и започиње пријем приказа игре и слање корисничког уноса.

Ова компонента је имплементирана користећи радни оквир за веб апликације *Express.js*. Комуникација са агентом, као и слање уноса врши се коришћењем комуникационог протокола *WebSockets*, док се за пренос слике и звука користи протокол *WebRTC*, који омогућава слање мултимедијалног садржаја преко *peer-to-peer* везе у реалном времену. Оба протокола су стандардизовани у данашњим веб претраживачима. Детаљи о начину рада протокола *WebRTC* биће представљени у четвртом поглављу, док ће се о коришћењу протокола *WebSockets* за пренос команди говорити у петом поглављу.

* 1. Контролер

Контролер представља компоненту која управља сесијама и додељује агенте корисницима. По пријему захтева за почетак сесије од стране корисника, контролер шаље поруку са захтевом у ред порука, који служи као посредник између контролера и агената. Када један од агената прихвати захтев, контролер шаље кориснику одговор са информацијама о започетој сесији, како би корисник могао да успостави директну везу са агентом.

Попут фронтенда, и ова компонента је имплементирана у виду веб сервера. За имплементацију је коришћен радни оквир *Express.js*.

* 1. Размена порука

За потребу размене порука између контролера и агената користи се база података *Redis*. Као база која чува податке у радној меморији, због високих перформанси и флексибилности чест је избор за ову сврху. У понуди је мноштво апстракција за размену порука, од којих ћемо користити редове. Детаљи о коришћењу размене порука биће дати у шестом поглављу.

* 1. Агенти

Агенти представљају машине, физичке или виртуелне, на којима се извршавају игре. Ове машине најпре ишчекују захтев за покретање сесије из реда порука. По пристизању захтева, шаље се порука о прихватању захтева и агент успоставља директну комуникацију са корисником. Ако тражена игра не постоји на хард диску, довлачи се из складишта објеката, а исто се чини и са корисничким датотекама, уколико је корисник већ играо игру раније. Током трајања сесије, агент је задужен за слање приказа кориснику, као и за читање корисничког уноса и његово извођење. По завршетку сесије, агент чува корисничке датотеке у складишту објеката и ишчекује следећу сесију. Сваки агент извршава тачно једну сесију, те је број корисника који истовремено могу да играју игре ограничен бројем агената.

Агенти користе *Windows 11* оперативни систем, због највеће стопе компатибилности са данашњим играма. Управљачка логика агената је имплементирана у програмском језику *Python*, који је изабран због своје једноставности и развијених библиотека за комуникацију и рад са осталим компонентама система, као и због интеграција са неопходним системским позивима. Дохватање, кодирање и пренос слике и звука извршава се користећи мултимедијалну библиотеку *GStreamer*. Ова библиотека омогућава комплексну обраду слике и звука креирањем цевовода сачињених од компоненти које дефинишу сваки корак обраде. Детаљи рада ове библиотеке, као и коришћени протоколи хватања, кодирања и преноса биће приказани у четвртом поглављу.

* 1. Складиште објеката

Пошто игре и кориснички подаци представљају бинарне датотеке, за њихово чување користићемо складиште објеката. За наше потребе одабрано је складиште објеката *MinIO*. Главне одлике овог складишта су брзина и скалабилност, а поред основних операција чувања и дохватања датотека, оно нуди додатне могућности попут верзионисања датотека и репликације. О начину коришћења складишта објеката детаљније ће бити речи у шестом поглављу.

Описана архитектура је модуларна и испуњава услов скалабилности. Фронтенд и контролер су једноставни веб сервиси без стања, тако да их је могуће скалирати додавањем нових инстанци и вршењем распоређивања оптерећења. Нови агенти су одмах по покретању спремни за извршавање сесија, конзумирањем захтева из реда порука.

У наредним поглављима бавићемо се најзначајнијим проблемима са којима смо се сусрели током имплементације система. Детаљи имплементације сваке од компоненти биће приказани приликом решавања проблема релевантних за ту компоненту.

1. Пренос слике и звука

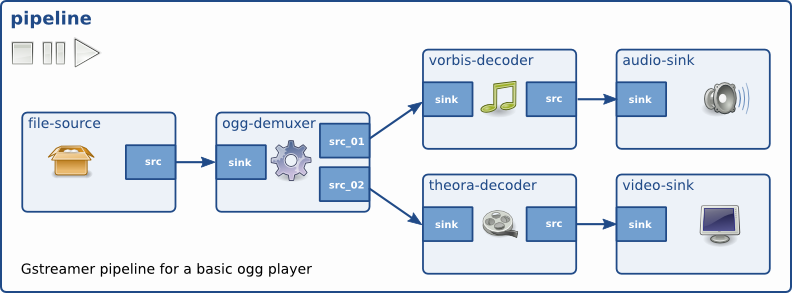
У овом поглављу бавићемо се проблемом преноса слике и звука, које емитује игра покренута на агенту, до корисничког веб претраживача. Проблем ћемо решавати засебно за слику и за звук, а рашчланићемо га на потпроблеме дохватања, кодирања, преноса и репродуковања. Најпре ћемо описати начин рада библиотеке *GStreamer* и протокола *WebRTC*, који представљају кључне технологије за решавање овог проблема. Затим ћемо детаљно описати решење сваког од потпроблема, са образложењем коришћених техника, формата и параметара у циљу компатибилности са веб претраживачима и минимизације кашњења.

* 1. О библиотеци *GStreamer*

Библиотека *GStreamer* омогућава обраду мултимедијалних токова дефинисањем цевовода. Цевоводи представљају усмерене графове, где сваки чвор графа дефинише један корак у обради података. У контексту библиотеке, чворови се називају елементима (*elements*). Сваки елемент има дефинисане прикључке (*pads*), који представљају интерфејсе за повезивање између елемената и могу бити изворни (*source pads*) и одводни (*sink pads*). Прикључци такође имају дефинисане могућности (*capabilities*), које дефинишу врсте података које могу да примају, уколико су изворни, или шаљу, уколико су одводни. Везе између елемената се успостављају преко изворног прикључка једног елемента и одводног прикључка другог, под условом да су им могућности компатибилне.

На основу своје улоге, елементи се могу класификовати у следеће групе:

* Изворни елементи (*source elements*) увозе податке у цевовод читајући их из датотека, уређаја попут камера и микрофона, или других екстерних извора података. Садрже искључиво изворне прикључке, и као такви представљају први корак обраде.
* Одводни елементи (*sink elements*) извозе податке из цевовода њиховим записивањем у датотеке, као и емитовањем преко излазних уређаја или преко мреже. Садрже искључиво одводне прикључке, и као такви представљају последњи корак обраде.
* Филтери (*filters*), врше обраду података, и као такви чине главни део цевовода. Поседују и изворне и одводне прикључке. Користе се за операције попут додавања трансформација и ефеката, кодирања и декодирања. Од подтипова филтера треба поменути и мултиплексере (*muxers*), који спајају податке из више одводних прикључака у један изворни прикључак, као и демултиплексере (*demuxers*), који раздвајају податке из једног одводног прикључка у више изворних прикључака.



Слика 4.1. Пример цевовода у библиотеци GStreamer [\_]

Библиотека сама по себи нуди велики број додатака, груписаних у оквиру званичних колекција, који садрже елементе за рад са често коришћеним форматима, протоколима, ефектима, као и улазним и излазним периферијама. Уколико је неопходно, корисник може креирати нове елементе имплементирањем понуђених интерфејса.

Сви програмски позиви које библиотека нуди представљају омотач око извршног окружења библиотеке. Извршно окружење се такође могуће покренути као самостални процес, са еквивалентним могућностима у поређењу са библиотекама омотачима. Пошто у тренутку писања овог рада постоје проблеми са радом омотача у програмском језику *Python* специфично за *Windows* оперативни систем, библиотеку ћемо користити директним покретањем извршног окружења. Иако се на овај начин губи на читљивости кода, за потребе нашег система нема губитка у функционалности, тако да је приступ прихватљив. У наставку поглавља описаћемо конкретне елементе коришћене приликом имплементације система, као и начин њиховог повезивања у цевоводе.

* 1. О протоколу *WebRTC*

Намена протокола *WebRTC* јесте комуникација у реалном времену преко *peer-to-peer* везе. Овај тип везе, у сврху слања слике и звука од агента до корисника, је оптималан за наш сервис због избегавања посредника у комуникацији, што значајно доприноси смањењу кашњења. Протокол је такође подржан од стране свих данашњих веб претраживача, чиме је покривена корисничка платформа коју желимо да подржимо, али и оставља могућности за ширење на друге подржане платформе, попут мобилних уређаја.

Као *peer-to-peer* протокол, постоје два главна проблема које треба решити приликом успостављања везе између две крајње тачке. При проблем јесте међусобна спознаја крајњих тачака. Између њих је неопходно разменити намеру о успостављању везе, као и начин на који се то може извршити, користећи било који други метод комуникације.

Други проблем јесте присутност *Network Address Translation* (*NAT*) технологије у готово свим мрежама којима припадају кориснички уређаји. Ова технологија подразумева пресликавање приватне адресе уређаја у јавну адресу, уз јединствени порт, од стране рутера приликом слања одлазног саобраћаја ван мреже. Такође се врши обрнуто пресликавање приликом обраде долазног саобраћаја, а у зависности од коришћене врсте ове технологије могуће су рестрикције на основу досадашње комуникације примаоца са пошиљаоцем. Ово значи да крајња тачка мора бити свесна начина на који јој се може приступити путем јавне мреже, како би ту информацију могла доставити другој крајњој тачки.

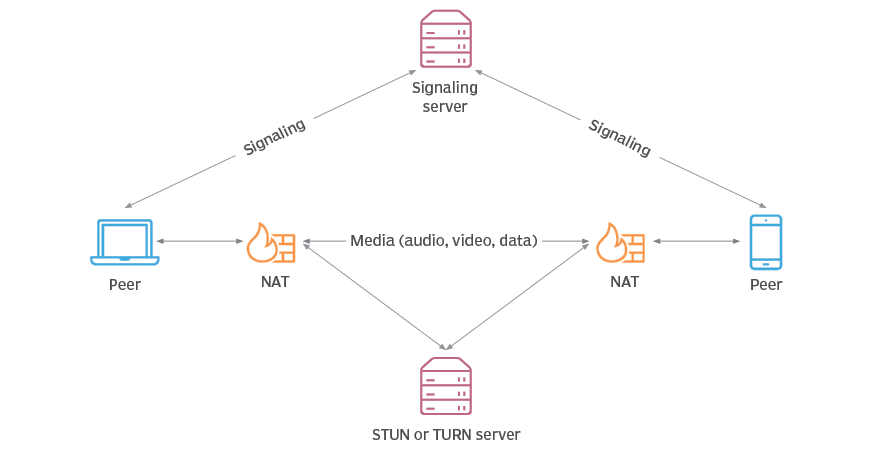
Поступак успостављања везе у протоколу *WebRTC* прилагођен је решавању ових проблема. Први проблем решава се иницијалним поступком сигнализације (*signalling*), који представља размену информација између крајњих тачака које су неопходне за успостављање везе. Крајње тачке оглашавају ове информације користећи *Session Description Protocol* (*SDP*). У питању је формат поруке у виду парова кључ-вредност која садржи податке попут списка подржаних типова медија и формата, као и адреса преко којих је могућа веза са пошиљаоцем. Како се крајње тачке у овом тренутку не познају, размена ових порука између учесника се врши преко посредника који се назива сервер за сигнализацију (*signalling server*). Сама комуникација са сервером за сигнализацију врши се на произвољан начин, где је најчешћи избор протокол *WebSockets*. Модел размене порука је такав да се *SDP* порука једне од крајњих тачака нуди другој као понуда (*offer*), а затим друга крајња тачка генерише своју *SDP* поруку и шаље је као одговор (*answer*). Овом приликом одбацују се понуђени формати који нису подржани од стране оба учесника, а може се одбити и целокупна веза уколико не постоји ниједан компатибилан начин за њено остваривање. Очекивани резултат овог поступка јесте да су крајње тачке усаглашене око начина извршавања сесије, те је могуће успоставити везу коришћењем понуђених адреса.

Други проблем решава се коришћењем протокола *Interactive Connectivity Establishment* (*ICE*). Намена овог протокола јесте утврђивање адреса и портова који представљају кандидате преко којих је крајња тачка доступна другој, а затим и избор кандидата са оптималном путањом. Размотрени кандидати, по опадајућој оптималности, су следећи:

* Приватна адреса и порт, који би се могли користити у случају да крајње тачке припадају истој мрежи. У том случају ово би представљало оптималан начин повезивања, без изласка у јавну мрежу.
* Адреса и порт добијени користећи протокол *Session Traversal Utilities for NAT* (*STUN*). Под овим се подразумева слање захтева екстерном серверу који имплементира овај протокол. Као одговор на овај захтев, сервер враћа адресу и порт пошиљаоца, што представља мапирање у јавну адресу које је *NAT* протокол извршио приликом слања захтева. Овако добијено мапирање валидно је и након завршеног *STUN* захтева, те се може користити за приступ крајњој тачки из јавне мреже и радиће са већином *NAT* верзија. У понуди су бесплатни, јавно доступни *STUN* сервери одржавани од стране компаније *Google*, и њих ћемо користити у нашем пројекту.
* Адреса и порт сервера који имплементира протокол *Traversal Using Relays around NAT* (*TURN*). Сервер који имплементира овај протокол користи се у улози посредника за сав саобраћај у случајевима кад права *peer-to-peer* конекција није могућа. Ово је случај код најрестриктивније верзије *NAT* технологије која се назива симетрични *NAT*. Код ове верзије, при слању порука ново мапирање се генерише за сваког примаоца и сав долазни саобраћај који не потиче од првобитног примаоца се одбија, те се вредност враћена од стране *STUN* сервера не може користити од стране друге крајње тачке. Додавање посредника у комуникацији може значајно утицати на кашњење преноса, док одржавање самог сервера захтева незанемарљиву количину рачунарских и мрежних ресурса, те због тога не постоји јавно доступне машине као за *STUN* и сервиси су одговорни за одржавање својих *TURN* сервера. Са друге стране, симетрични *NAT* се користи у изразито малом броју кућних мрежа, са мање од 5 процената удела, и чешћа је појава у корпоративним или мобилним мрежама. Стога ово не представља случај од значаја за наш систем, те нећемо користити *TURN* сервер у нашем систему, а захтеве корисника који користе симетрични *NAT* ћемо одбијати.

Генерисање кандидата врши се аутоматски на почетку сигнализације од стране протокола *WebRTC* на основу подешених адреса *STUN* и *TURN* сервера. Раније поменуте адресе и портови садржани у *SDP* поруци управо представљају откривене адресе кандидате. По завршеном поступку сигнализације, крајње тачке проверавају доступност добијених кандидата, редом по оптималности путање, и за везу користе најбољи доступни пар кандидата.

Након што је веза успостављена, омогућена је размена података, попут слике и звука, у виду произвољног броја токова, уз коришћење оптималне путање између крајњих тачака. За само слање података користи се протокол *RTP,* док се за пренос статистика о сесији, попут кашњења или количине изгубљених пакета, користи протокол *RTP Control Protocol* (*RTCP*). Коришћењем ових информација ток података аутоматски се прилагођава променљивим мрежним условима адаптирањем тренутног квалитета преноса.



Слика 4.2.1. Дијаграм рада протокола WebRTC []

* 1. Слика

Нешто нешто \_\_\_ слика без звука \_\_\_.

* + 1. Дохватање

Први корак у овом поступку јесте дохватање приказа игре. Пошто се игре извршавају на *Windows 11* оперативном систему, за дохватање слике користићемо *Windows Desktop Duplication API*. [] Ово је изворно подржан део графичког интерфејса *DirectX* чија је сврха пресликавање приказа првенствено у меморију на графичкој картици у виду сирових оквира у *RGBA* формату. [] Овај метод препоручен је од стране компаније *NVIDIA* као замена за њихов *Capture/GRID SDK* на овом оперативном систему. []

За коришћење овог интерфејса применићемо изворни елемент под називом *d3d12screencapturesrc*. Условљено интерфејсом који се интерно користи, могућности које изворни прикључак овог елемента нуди јесу сирови оквири у меморији графичке картице или у системској радној меморији, чиме се нуди компатибилност са већином филтер елемената.

d3d12screencapturesrc show-cursor=true window-handle={window\_handle}

Код 4.3.1.1. Елемент за дохватање слике

Од параметара, елемент прима ручку прозора којег дохватамо, као и опцију дохватања показивача, коју укључујемо. Начин дохватања ручке прозора игре за потребе дохватања описаћемо у шестом поглављу.

Дохватање слике је континуално, и траје све до експлицитног затварања цевовода од стране корисника, или затварања циљаног прозора.

* + 1. Кодирање

Следећи корак јесте кодирање сировог приказа у формат погодан за слање преко мреже. Од начина кодирања који су нам на располагању, постоје само два која су претраживачи у обавези да подрже у оквиру подршке за *WebRTC*, а то су *VP8* и *H.264*. []

Са друге стране, кодирање може бити рачунски захтеван посао, нарочито код виших резолуција и броја кадрова у секунди. Ово значи да се одређен део процесорског времена и радне меморије мора издвојити за кодирање, чиме се смањују ресурси посвећени игри, што може утицати на њене перформансе. Како машина на којој извршавамо агента садржи графичку картицу *NVIDIA RTX 3060*, у могућности смо да искористимо технологију *NVENC* својствену *NVIDIA* графичким картицама. Ова технологија омогућава вршење операције кодирања преко графичке картице, користећи посебно хардверско језгро на картици посвећено искључиво пословима кодирања. Ових језгара може бити једно или више, у зависности од модела картице. На овај начин растерећујемо остале рачунарске ресурсе и препуштамо им извршавање игре. Од два стандарда која разматрамо, *NVENC* подржава једино *H.264*, тако да ћемо њега користити за кодирање.

За кодирање овим стандардом коришћењем *NVENC* употребићемо филтер елемент под називом *nvh264enc*. Могућности одводног прикључка покривају мноштво формата, а међу њима су сирови оквири смештени на меморији графичке картице, чиме смо у могућности да се повежемо на изворни оквир елемента за дохватање без међукорака конверзије или преноса података, што значајно доприноси скраћењу трајања операције.

\_\_параметри\_\_оптимизовано за кашњење\_\_.

* + 1. Пренос

Кодирану слику на крају треба пренети до корисничког веб претраживача и приказати је. Ово вршимо користећи протокол *WebRTC*.

За потребе преноса коришћењем овог протокола доступан је одводни елемент *webrtcsink*,и њиме завршавамо нас цевовод. \_\_\_\_ захтева да сервер за сигнализацију испуњава специфичан интерфејс \_\_\_ произвођачи и потрошачи, што у нашем случају \_\_\_\_ респективно. Постоји једноставна имплементација сервера уграђена у елемент која се може користити, и она је довољна за наше потребе.

* 1. Звук

Нешто нешто

* + 1. Дохватање

Нешто нешто

* + 1. Кодирање

Нешто нешто

* + 1. Пренос

Нешто нешто

Са порастом хардверских способности \_\_\_ повећати параметри \_\_ подићи квалитет. Или не?

1. Пренос команди

У припреми.

1. Ток сесије

У припреми.

1. Евалуација

У припреми.

1. Закључак

У припреми.

Литература

1. Arto Ojala, Pasi Tyrväinen, „Developing Cloud Business Models: A Case Study on Cloud Gaming“, *IEEE Software (Volume: 28, Issue: 4, July-Aug. 2011)*
2. JP Mangalindan, *Cloud gaming’s history of false starts and promising reboots*, <https://www.polygon.com/features/2020/10/15/21499273/cloud-gaming-history-onlive-stadia-google>, 14.09.2025.
3. Cloud Gaming, Broadmedia Corporation, <https://www.broadmedia.co.jp/en/technology/cloudgame>, 14.09.2025.
4. OnLive, Logopedia, <https://logos.fandom.com/wiki/OnLive>, 14.09.2025.
5. Gaikai, Logopedia, <https://logos.fandom.com/wiki/Gaikai>, 14.09.2025.
6. Frequently Asked Questions for GeForce NOW, NVIDIA, <https://www.nvidia.com/en-us/geforce-now/faq>, 18.09.2025
7. Details about Geforce Now infrastructure - NVIDIA Developer Forums, <https://forums.developer.nvidia.com/t/details-about-geforce-now-infrastructure/237485>, 18.09.2025.
8. Andrea Di Domenico, Gianluca Perna, Martino Trevisan, Luca Vassio, Danilo Giordano, „A network analysis on cloud gaming: Stadia, GeForce Now and PSNow“, *Network 2021, 1, 247-260*
9. Richard Leadbetter, *Sony creates custom PS3 hardware for PlayStation Now*, <https://www.eurogamer.net/digitalfoundry-2014-sony-creates-custom-ps3-for-playstation-now>, 19.09.2025.
10. Sherif Saed, *Xbox Series X hardware will power xCloud servers next year – report*, <https://www.vg247.com/xbox-series-x-in-xcloud-servers-by-2021-report>, 19.09.2025.
11. Michael Larabel, *Stadia Is Google's Cloud Gaming Service Using Linux, Vulkan & A Custom AMD GPU*, <https://www.phoronix.com/news/Google-Stadia-Vulkan-Linux>, 20.09.2025.
12. NVIDIA GeForce NOW, Seeklogo, <https://seeklogo.com/vector-logo/457169/nvidia-geforce-now>, 21.09.2025.
13. Playstation Now, Seeklogo, <https://seeklogo.com/vector-logo/459914/playstation-now>, 21.09.2025.
14. Stadia, <https://stadia.google.com>, 21.09.2025.
15. Baba is You, Steam, <https://store.steampowered.com/app/736260/Baba_Is_You/>, 15.09.2025.
16. SUPERHOT, Steam, <https://store.steampowered.com/app/322500/SUPERHOT/>, 15.09.2025.

Списак скраћеница

Списак слика

[Слика 2.1.1. Сервиси G-Cluster, OnLive и Gaikai [3][4][5] 5](#_Toc209385094)

[Слика 2.1.2. Сервиси GeForce NOW, PlayStation Now и Stadia [12][13][14] 7](#_Toc209385095)

[Слика 2.2.1 Игре Baba is You и SUPERHOT [15][16] 8](#_Toc209385096)

[Слика 3.1. Архитектура система 9](#_Toc209385097)

Списак табела

**No table of figures entries found.**