VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Decentralizuotas realaus laiko komunikavimas žiniatinklyje

Decentralized Real Time Communication in World Wide Web

Bakalauro baigiamasis darbas

Atliko: 4 kurso 6 grupės studentas
Rokas Ulickas (parašas)

Darbo vadovas: partn. doc. Vaidas Jusevičius (parašas)

Darbo recenzentas: asistentas dr. Vytautas Valaitis (parašas)

Vilnius, 2020

Padėka

Noriu padėkoti kurso draugams Jokūbui Lekevičiui, Pauliui Kalvaičiui, Adomui Poleninui, Danieliui Ritvui ir Vytautui Krivickui už tai, kad padėjo ištestuoti sukurtą decentralizuotą realaus laiko komunikavimo paslaugos prototipą.

Santrauka

Dabar neįmanoma įsivaizduoti pasaulio be realaus laiko komunikavimo žiniatinklyje paslaugų. Tai – patogi, greita ir nereikalaujanti daug kompiuterinio raštingumo įgūdžių interneto paslauga. Šiuo metu egzistuoja daugybė skirtingų realaus laiko komunikavimo žiniatinklyje paslaugų. Dažniausiai žiniatinklyje vaizdu, garsu ir duomenimis yra keičiamasi naudojant WebRTC technologiją. Šia technologija paremtas duomenų keitimasis vyksta lygiarangių ryšiu, tačiau sesijos užmezgimui reikalingas signalizuojantis kanalas. Praktikoje dažniausiai būna naudojamas paprasčiausias serveris.

Šiame darbe šalia realaus laiko komunikavimo, WebRTC technologijų nagrinėjimo yra apžvelgiama IPFS technologija, kuri leistų panaikinti centralizuoto serverio, kaip signalizuojančio kanalo, būtinybę.

Sukurtas decentralizuoto realaus laiko komunikavimo žiniatinklyje paslaugos prototipas, kuris sėkmingai buvo išbandytas ir naudojamas universiteto aplinkoje. Galiausiai yra pasiekiamas pagrindinis šio darbo tikslas – sukurta atvirai prieinama decentralizuoto realaus laiko komunikavimo paslauga.

Raktiniai žodžiai: decentralizuotas internetas, WebRTC, IPFS (IPFS), lygiarangių ryšys, realaus laiko komunikavimas žiniatinklyje.

Summary

It is impossible to imagine a world without real-time web communication services. It is a convenient, fast and easy-to-use web service. There are many different real-time web communication services. Generally, images, audio, and data are exchanged using WebRTC technology. Data exchange based on this technology takes place on a peer-to-peer basis, but a signalling channel is required to establish a session. In practice, the simple server is usually used.

In this work, in addition to real-time communication and the examination of WebRTC technology, author provides an overview of IPFS technology, which would eliminate the need for a centralized server as a signalling channel.

A prototype of a real-time web-based communication service has been developed and successfully tested, used in a university environment. Finally, the main goal of this work is achieved - to create an openly accessible real-time communication service.

Keywords: decentralized web, WebRTC, IPFS, peer-to-peer, real time communications in World Wide Web.

TURINYS

Įvadas

Realaus laiko komunikavimas (pokalbių kambariai, audio ir video konferencijos ir pan.) žiniatinklyje tampa vis populiaresnis. Šiuo metu egzistuoja technologijų, kurios įgalina vartotojų realaus laiko komunikavimą žiniatinklyje. Viena iš technologijų yra WebRTC. Šių sprendimų pasiekiamumas priklausomas nuo centrinio serverio egzistavimo. Pastarajam nustojus veikti realaus laiko komunikavimas tarp vartotojų nutrūksta.

Dabar pasaulyje yra milijardai puslapių, kuriuose interneto vartotojai naršo, bendrauja, perka prekes, žiūri filmus bei serialus. Populiariausi iš jų yra YouTube, Facebook, Amazon, Netflix, Alphabet priklausantys puslapiai. Pavyzdžiui, per sekundę yra įvykdoma 80 000 Google paieškos užklausų, beveik 2 trilijonai užklausų per metus. Kadangi visos užklausos yra kaupiamos centralizuotai, jos gali būti apdorotos – taip puikiai identifikuojant atskirų vartotojų pomėgius, amžių, gyvenamąją vietą ir kitus privačius asmens duomenis.

Tai sukelia privatumo problemų - visa informacija yra visiškai valdoma ir pasiekiama tų, kurie valdo arba turi prisijungimą prie tų puslapių serverių. Kartais įvyksta įsilaužimai į centralizuotus serverius ir pavagiama jautri informacija. Vienas didžiausių duomenų nutekėjimų buvo 2017 metais kompanijoje Equifax – nusikaltėliai pavogė 143 milijonų Jungtinių Amerikos Valstijų gyventojų asmeninius duomenis: sveikatos draudimo numerius, gimimo datas, adresus, vairuotojo pažymėjimo numerius [ZMM+18].

Kita problema – vienintelio trikties taško problema. Jeigu pradėtų nebeveikti, tarkime, Youtube serveriai, visa aplikacija nustotų funkcionuoti ir vartotojai nebegalėtų prisijungti ir peržiūrėti bet kokių įkeltų filmukų.

Galiausiai, grėsmę kelia paskirstyta paslaugos trikdymo ataka (angl. *DDoS attack*). Kadangi yra viena vieta į kurią nusikaltėliai gali atakuoti serverius, taip galima sukelti labai didelius nuostolius ir nuo to yra labai sunku apsisaugoti. Vienas iš pavydžių ataka prieš DNS paslaugos tiekėją Dyn, kas sukėlė ilgalaikius paslaugos sutrikdymus Jungtinėse Amerikos Valstybėse bei Europoje [Dic16].

Paskutinius metus pasaulyje vis daugiau kalbama apie decentralizuotą internetą, jo pritaikymo galimybes bei jo naudą. Dabar internete dažniausiai resursus pasiekiame per HTTP (*Hypertext Transfer Protocol*) arba HTTPS (*Hypertext Transfer Protocol Secure*) protokolo nuorodas, kurios mus nukreipia į konkrečią vietą. Decentralizuotame internete mes viską pasiekiame per adresus – ieškome, kuris tinklo narys turi mums reikiamą resursą, o ne iškart kreipiamės į serverį. Resursas gali būti net ir pas mūsų kaimyną arba darbdavį, jeigu jis yra prisijungęs prie decentralizuoto tinklo. Net žiniatinklio kūrėjas Tim Berners-Lee mano, kad

dabartinis sukurtas pasaulis žiniatinklyje nėra teisingas. Pats Berners-Lee šiuo metu kuria Solid projektą, kuris turėtų padidinti duomenų saugumą, privatumą [Cla18].

2020 metų kovo 16 d. Lietuvoje buvo paskelbtas karantinas dėl pasaulinės COVID-19 pandemijos. Didelė dalis visuomenės buvo paveikta: moksleiviai pradėjo mokytis, įmonių darbuotojai dirbti naudojant realaus laiko komunikavimo įrankius tokius kaip Zoom, Google Hangouts, Microsoft Teams, Skype. Šių įrankių egzistavimas tapo kaip niekad svarbus karantino metu. Buvo neapsieita ir be skandalų – skelbta, jog kompanijos Zoom įrankis palaiko end-to-end šifravimą, tačiau iš tikrųjų tai nebuvo tiesa [The20]. Tai teoriškai leidžia šnipinėti vykstančius pokalbius Zoom darbuotojams.

Šiems įrankiams tapus mūsų kasdieninio darbo arba mokymosi dalimi, žmonės susirūpino dėl savo privatumo. Šio bakalauro darbo vienas iš rezultatų gali būti sukurtas decentralizuoto realaus laiko komunikavimo paslaugos prototipas. Šį prototipą patobulinus, jis galėtų tapti naudojamas kaip bendravimo įrankis tarp žmonių, kurie yra susirūpinę savo asmenine informacija bei privatumu.

Šio darbo tikslas – palyginti realaus laiko komunikavimo metodus, įvertinti galimybes vykdyti decentralizuotą realaus laiko bendravimą žiniatinklyje pasitelkiant egzistuojančias technologijas bei sukurti realaus laiko komunikavimo paslaugos prototipą naršyklėje. Siekiant įgyvendinti darbo tikslą iškeliami tokie uždaviniai:

- apžvelgti realaus laiko komunikavimą ir skirtingas programų architektūras, kurios pastarąjį realizuoja;
- išnagrinėti WebRTC technologiją, kuri įgalina realaus laiko komunikavimą žiniatinklyje ir mobiliosiose aplikacijose;
- apžvelgti IPFS technologiją, kuri leidžia decentralizuotai keistis informacija;
- įvertinti WebRTC ir IPFS technologijų sintezės galimybes;
- sukurti realaus laiko komunikavimo paslaugos prototipą naršyklėje, panaudojant WebRTC ir IPFS technologijas.

Paslaugos prototipui kurti yra panaudota javascript programavimo kalba. Kai kurie vartotojo sąsajos elementai yra sukurti panaudojant bootstrap 4 vartotojo sąsajos karkasą. Visa sistema ir jos programinis kodas yra vykdomas tik vartotojo naršyklėje – nereikia sukurti jokio serverio, kuris leistų vykdyti funkcijas, kurioms dažniausiai reikalingas serveris.

Yra panaudojamos IPFS technologijos įgyvendinimas skirtas žiniatinklio naršyklėms - js-ipfs versija 0.40. Reikia paminėti, jog ši realizacija yra dar tik alfa (angl. *alpha*) versijoje, tai yra, jos kūrėjai dar nerekomenduoja jos naudoti gamybinėje aplinkoje. Taip pat naudojama WebRTC technologijos įgyvendinimas, kuris yra kiekvienoje naujos kartos naršyklėje: Google Chrome, Safari, Microsoft Edge, Firefox, Opera.

Paslaugos talpinimui buvo pasirinkta Amazon Web Services statiniu resursų talpinimo S3 paslauga dėl savo patikimumo bei patirties naudojantis ja.

1. Realaus laiko komunikavimas

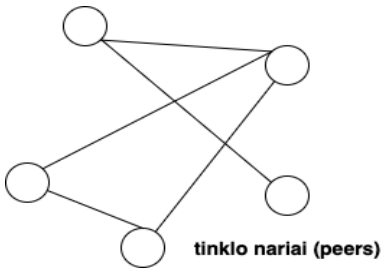
Realaus laiko komunikavimas (angl. *real-time communication*) – tai informacijos apsikeitimas per telekomunikacijos technologijas tarp siuntėjo ir gavėjo, kuris įvyksta be reikšmingo uždelsimo, vėlavimo. Svarbu paminėti, kad įgyvendinant realaus laiko komunikavimą būsena tarp siuntėjo ir gavėjo nėra niekur saugoma ir per jokį kitą įrenginį nepersiunčiama. Realaus laiko komunikavimas yra **gyvo komunikavimo** sinonimas (angl. *live communications*). Šio komunikavimo tipo pavyzdžiui:

- mobilieji skambučiai;
- radijo ryšys;
- susirašinėjimo programėlės;
- vaizdo skambučiai, vaizdo skambučių konferencijos.

Dažniausiai realaus laiko komunikavimas yra įgyvendinamas naudojant **lygiarangių ryšį** (angl. *peer-to-peer communications*)[1]. Taip pat tam pasiekti galima naudoti ir kliento – serverio modelio architektūra, tačiau uždelsimas ir vėlavimas bus žymiai didesnis, nei tiesioginis informacijos keitimasis tarp tinklo narių (angl. *peers*). Tai, pagal apibrėžimą, jau nebus tikras realaus laiko komunikavimas. Apie kliento – serverio ir peer-to-peer architektūrų privalumus ir trūkumus bus kalbama tolimesniuose skyriuose.

1.1. Peer-to-peer (lygiarangių ryšys)

Peer-to-peer leidžia dviem arba daugiau prietaisų komunikuoti tiesiogiai vienas su kitu ir dalintis informacija nesant centralizuoto serverio. Šis modelis pavaizduotas 1 paveiksle. Susijungę prietaisai yra lygiaverčiai, tai yra, priklausomai nuo situacijos kiekvienas iš jų gali veikti ir kaip klientas (paslaugos naudotojas), ir kaip serveris (paslaugos tiekėjas). Kiti autoriai peer-to-peer apibūdina labai paprastai - kaip priešingą kliento – serverio architektūrai. [Sin01] [Tho98].



1 pav. Peer-to-peer ryšių architektūra

[1] Toks terminas yra pasiūlytas VLKK (Valstybinės lietuvių kalbos komisijos), tačiau informacinių technologijų industrijos žmonėms jis labiau bus suprantamas kaip **peer-to-peer**. Todėl toliau tekste bus naudojamas angliškas terminas peer-to-peer. Koks angliško kompiuterijos termino „peer-to-peer communications“ lietuviškas atitikmuo? http://www.vlkk.lt/konsultacijos/7835-lygiarangiu-rysys-peer-to-peer-communications. (Tikrinta 2020-01-05)

Dažniausiai ryšio užmezgimui tarp dviejų tinklo narių yra naudojamas centralizuotas serveris, kuris dažniausiai tik valdo tinklo konfigūracija bei tinklo narių susijungimo sesijas, padeda surasti vienas kito adresus, kad būtų įmanomas tiesioginis komunikavimas.

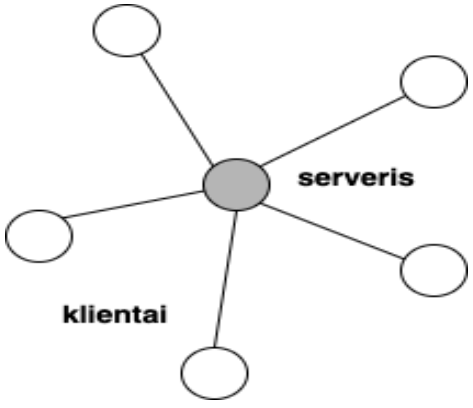
1.2. Kliento – serverio modelis

Kliento – serverio modelis (angl. *client – server model*) tai labai plačiai naudojama aplikacijos struktūra, kuri atskiria paslaugos tiekėjus, vadinamus serveriais ir paslaugos naudotojus, vadinamus klientais. Šis modelis pavaizduotas 2 paveiksle. Klientai pasinaudodami serveriuose saugomais duomenimis ir informacija gali pasiekti ir gauti resursus arba įvykdyti norimas funkcijas. Kasdieniai kliento – serverio architektūros pavyzdžiai:

- elektroninės bankininkystės paslauga;
- elektroninio pašto paslauga;
- automobilio dalijimosi paslauga.

Komunikavimas tarp kliento ir serverio vyksta užklausos – atsakymo tipo bendravimu. Klientas prašo konkretaus resurso – serveris grąžina tą resursą arba praneša apie įvykusią klaidą. Bendravimas turi vyksti abiems pusėms žinoma kalba bei taisyklėmis – **bendravimo protokolu** (angl. *communications protocol*).

Šiame architektūriniame modelyje serveris veikia kaip centralizuotos sistemos resursų tiekėjas, tai yra, jis vienintelis aptarnauja klientus sistemoje. Serverio procesorių skaičius, operatyvioji atmintis bei kiti parametrai turi būti žymiai didesni nei paprasto personalinio kompiuterio bei jie turi būti atitinkamai didinami arba mažinami priklausomai nuo prognozuojamo kliento, operacijų skaičiaus bei skaičiavim intensyvumo.



2 pav. Kliento - serverio architektūra

1.3. Kliento – serverio modelio ir peer-to-peer ryšių modelio palyginimas

Žemiau pateikiama kliento – serverio bei peer-to-peer ryšių modelių palyginimas, tam, kad skaitytojas geriau suprastų jų skirtumus ir panašumus.

1 lentelė. Kliento – serverio modelio bei peer-to-peer ryšių modelio palyginimas.

Savybė pagal kurią lyginama	**Klientas – serveris**	**Peer-to-peer**
Stabilumas	Stabili ir lengviau plečiama architektūra.	Gali kilti problemų ir sutrikimų, kai padaugėja tinklo narių (peers).
Išlaikymo kaina	Serveris bei jo išlaikymas kainuoja, be to, didesnis poreikis palaikymo komandai.	Pigesnis sukūrimas, nereikia išlaikyti serverių.
Serveris	Serveris gali nesugebėti apdoroti didelio skaičiaus užklausų, jeigu nebus pakankamai efektyvus.	Kadangi tinklo nariai dalinasi savo resursais su kitais tinklo nariais, visos užklausos apdorojamos.
Duomenys	Duomenys saugomi centralizuotame serveryje.	Kiekvienas tinklo narys turi savo duomenis arba jų kopiją.
Dėmesys	Resursams, funkcijų vykdymui.	Greitam duomenų perdavimui, pasiekiamumui.
Paslauga, resursai	Paslaugos resursai yra gaunami iš centralizuoto serverio.	Kadangi kiekvienas tinklo narys gali būti kaip serveris, kiekvienas gali apdoroti siunčiamas užklausas iš kitų tinklo narių.
Greitis	Informacija gaunama iš tarpininko – serverio, todėl uždelsimas yra ilgesnis nei peer-to-peer.	Bendraujama tiesiogiai su kitais tinklo nariais, todėl uždelsimas yra minimalus.

Iš 1 lentelės matome, kad pagrindiniai skirtumai tarp kliento – serverio ir peer-to-peer ryšių modelių yra tokie:

- kliento – serverio modelyje yra atskiri serveriai ir klientai, kurie atlieka tik vieną iš tų funkcijų, kur peer-to-peer ryšių modelyje kiekvienas tinklo narys gali veikti ir kaip klientas, ir serveris;
- kliento – serverio modelį yra **brangiau** sukurti;
- kliento – serverio modelis yra labiau **plečiamas** ir **stabilus**;

- kliento – serverio modelyje duomenys yra laikomi **centralizuotame** serveryje, o peer-to-peer ryšių modelyje kiekvienas tinklo narys **saugo savo** duomenis arba duomenų kopija;
- peer-to-peer modelyje informacija apsikeičiama **greičiau** nei kliento – serverio modelyje.

Galime daryti išvadą, kad peer-to-peer ryšių modelis yra labiau tinkantis realaus laiko komunikavimo aplikacijoms.

2. WebRTC – standartas, įgalinantis realaus laiko komunikavimą

Įsivaizduokite pasaulį, kuriama kiekvienas telefonas, kompiuteris, televizorius ir bet koks **daiktų interneto** (angl. *Internet of Things*) prietaisas gali bendrauti vienoje platformoje. 2011 metais Google kompanija išleido pradinę **WebRTC** projekto versiją, kuri įgalina realaus laiko komunikavimą žiniatinklyje bei mobiliosiose aplikacijose peer-to-peer ryšio pagrindu. Ši, WebRTC technologija, leidžia keistis garso, vaizdo bei bet kokio kito tipo informacija.

Nuo pat išleidimo 2011 metais, projektas yra prižiūrimas **W3C** (World Wide Web Consortium) ir **IETF** (Internet Engineering Task Force) organizacijų, kurios atsakingos už interneto standartų kūrimą. Prie projekto plėtojimo prisideda Apple, Microsoft, Mozilla, Opera inžinieriai, kurie įdiegia šį standartą į savo kuriamas interneto naršykles (Safari, Microsoft Edge, Firefox, Opera), bet visi kiti atviro kodo bendruomenės nariai.

Anksčiau, norint perduoti garsinę arba vaizdinę informaciją dažniausiai buvo naudojama Skype platforma arba tam tikslui reikėdavo įsidiegti įskiepius į interneto naršyklę. WebRTC technologija leidžia visiškai atsisakyti įskiepių ir įgalina tiesioginį duomenų apsikeitimą tarp naršyklių bei mobiliųjų įrenginių, kas atveria milžinišką potencialą realaus laiko komunikavimo aplikacijoms. Apie WebRTC reikia galvoti ne kaip apie aplikacija, bet kaip apie infrastruktūra, kuri leidžia pasiekti anksčiau išvardintus tikslus.

WebRTC yra naudojama Facebook Messenger mobiliojoje aplikacijoje, bei daugelyje kitų programų, startuoliuose kuriamų sprendimų. Naudojant šią technologiją aplikacijose, didėja žmonių produktyvumas (nes nebereikia diegtis įskiepių, siųstis programų), didėja vartotojų pasitenkinimas bei vartotojų patirtis. Taip pat mažėja interneto centralizuotumas, sudėtingumas.

Brendan Eich, JavaScript programavimo kalbos bei Mozilla technologijų direktorius, yra pasakęs, kad WebRTC yra naujas įrankis ir pagalba siekti laisvo, atviro ir neapkrauto interneto [Get20]. Kaip pamatysime toliau, tai gali būti viena iš svarbių technologijų, kuri gali padėti siekti vis didesnio interneto decentralizavimo.

2.1. WebRTC veikimo principas

Dažniausiai WebRTC technologijos naudojimas aplikacijoje susideda iš dviejų dalių: **signalizavimo** bei **komunikavimo** stadijų. Pirmojoje signalizavimo stadijoje yra apsikeičiama trijų tipų informacija:

- sesijos kontrolės informacija – inicijuoti arba baigti komunikaciją ir pranešti apie klaidas;
- tinklo konfigūracija – pasidalinama kompiuterio IP adresu bei prievadu;

- medijos galimybės – kokie kodekai bei rezoliucija bus reikalingi atvaizduoti mano siunčiamą informaciją bei priimti gaunamą.

Dažnai sakoma kad aplikacijos, kurios naudoja WebRTC, veikia tik tarp dviejų naršyklių, tačiau tai ne tiesa. Kaip ir buvo rašyta anksčiau, dažniausiai bet kokio peer-to-peer ryšio užmezgimui reikalingas serveris. Taip pat ir signalizavimo stadija turi būti atliekama su serveriu. Signalizavimo rezultatas yra arba užmegztas ryšys, tai yra, pradėta sesija tarp tinklo narių arba įvykusi klaida.

Antrajai, komunikavimo stadijai pradėti, yra svarbus serverio, kuris gali užtikrinti komunikavimo sesiją, egzistavimas. Į WebRTC technologiją nėra įtraukti arba siūlomi signalizavimo metodai. Juos pasirenka kompanijos bei programuotojai pagal savo nuožiūrą. Dažniausiai tai yra HTTPS užklausos, kurios naudojamos saugiai perduoti informaciją.

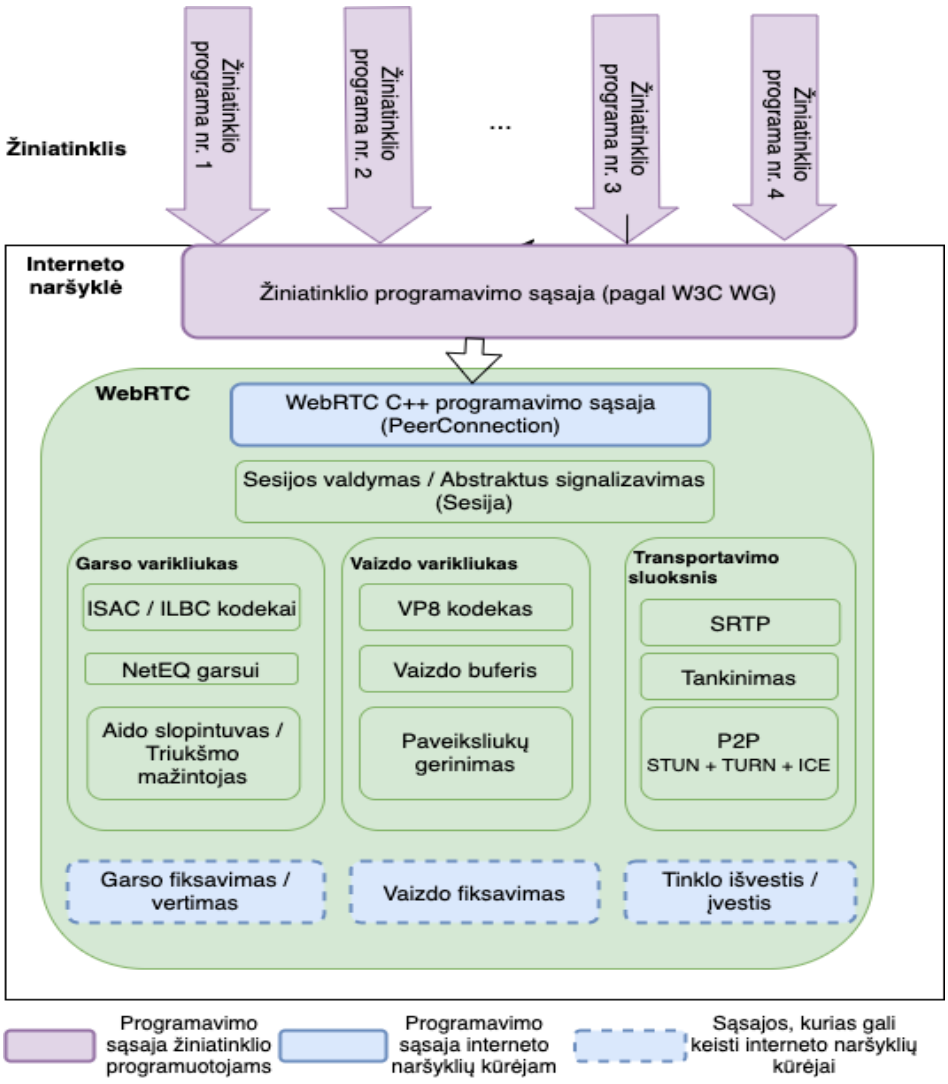
Užmezgus sesiją, jos dalyviai gali keistis informacija tiesiogiai. Ši, komunikavimo stadija skiriasi nuo klasikinio žiniatinklio programų, todėl, kad bendraujama tiesiogiai, o ne per centrinį serverį. Bendravimas tiesiogiai mažina vėlavimų trukmę bei didiną saugumą, nes nėra galimybės įvykdyti MITM (Man-In-The-Middle) atakos, kuri galėtų būti įvykdoma neteisėtai užvaldžius bet kokią tarpinę stotelę tarp siuntėjo ir gavėjo.

2.2. WebRTC programavimas

WebRTC turi keletą JavaScript kalbos programavimo sąsajų (angl. *API – Application Programming Interface*) [LR14]:

- MediaRecorder (arba getUserMedia()) – gauti prieigą ir fiksuoti garsinę, vaizdinę informaciją iš vartotojo kameros arba mikrofono;
- RTCPeerConnection – perduoti garsinę ir vaizdinę informacija tarp vartotojų, pavyzdžiui, palaikyti skambutį arba video konferenciją;
- RTCDataChannel – perduoti duomenis tarp vartotojų.

3 paveiksle pavaizduota WebRTC architektūra. Kaip matome iš paveikslėlio, žalių dalių yra daug ir jos labai kompleksiškos. Programuotojų laimei, jiems reikia išmokti naudotis WebAPI (violetine spalva pažymėta dalimi), o visu kitu yra pasirūpinta WebRTC ir interneto naršyklių kūrėjų [Web20].



3 pav. WebRTC architektūra.

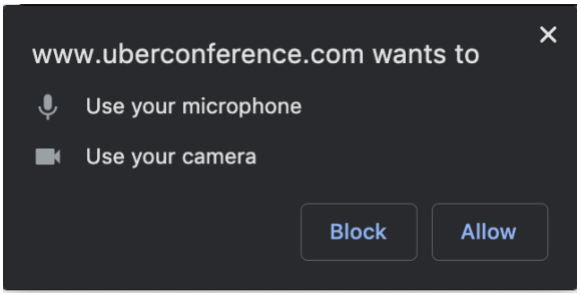
2.3. WebRTC saugumas

Nagrinėjant WebRTC standarto saugumą, turėtume išskirti kelias skirtingas grupes:

- serverio, kuris dalyvauja signalizavimo stadijoje, saugumas;
- kliento (peer) saugumas;
- komunikavimo kanalo saugumas.

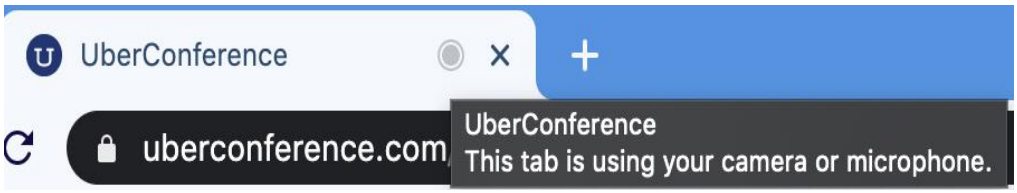
Populiariausios interneto naršyklės įgalina dviejų lygių saugumą:

- naršyklė prašo sutikimo naudoti prietaiso mikrofoną arba video šaltinį (4 paveikslas);



4 pav. Naršyklė prašo kameros bei mikrofono naudojimo leidimo

- naršyklė skirtingais identifikatoriais skirtukų juostoje parodo, kada jie yra naudojami, taip informuodami vartotoją.



5 pav. Naršyklė praneša vartotojui, kad šis skirtukas naudoja jo kamerą arba mikrofoną

Komunikacijos kanalas yra pati svarbiausia bei labiausiai apsaugota WebRTC dalis. WebRTC kūrėjai užtikrina, kad visa perduodama vaizdinė, garsinė bei kita informacija yra persiunčiama SRTP (Secure Real-time Transport Protocol) ir DTLS (Datagram Transport Layer Security) protokolais, kurie užtikrina žinutės šifravimą, saugumą bei vientisumą. WebRTC naršyklės API yra pasiekiamas tik iš apsaugotų šaltinių (localhost arba naudojant HTTPS duomenų perdavimo protokolą).

Feher, Sidi, Shabtai, Puzis ir Marozas savo straipsnyje [FSS+18], kuriame jie nagrinėja WebRTC saugumą akcentuoja, kad WebRTC yra puikus gerai suprojektuotos technologijos pavyzdys, bei WebRTC yra žymiai daugiau pažengusi realaus laiko komunikavimo technologija nei kad kitos VoIP technologijos. Ir išvadose priduria, kad dėl didelio dėmesio skirtu WebRTC saugumui, ši technologija yra viena iš saugiausių visoje rinkoje.

3. IPFS technologija įgalinanti decentralizuotą apsikeitimą duomenimis

Tarpplanetinė failų sistema **IPFS** (angl. *The InterPlanetary File System*) yra 2015 metais Juan Benet ir Protocol Labs įmonės pradėta kurti peer-to-peer, paskirstyta failų sistema skirta sujungti visus įrenginius į vieningą failų sistemą [Ben14]. Kartais IPFS gali būti panaši į tradicinį žiniatinklį, tačiau į tai reiktų žiūrėti kaip į vieną **BitTorrent spiečių** (angl. BitTorrent swarm), kurie keičiasi failais pasiekiamais vienoje **GIT** repositorijoje. IPFS sukuria didelio našumo blokų adresų modelį panaudojant IPLD (angl. *InterPlanetary Linked Data*) technologiją. Toliau pateiksiu pagrindines IPFS ypatybes, per kurias išvardinsiu IPFS technologijos sėkmę.

3.1. Paskirstytos dėstymo lentelės

Dėstymo lentelės (angl. *hash tables*) – tai duomenų struktūra, kurioje duomenys yra saugomi raktas – reikšmė formatu. Paskirstytos dėstymo lentelės – tai tokios dėstymo lentelės, kurių kopijas turi kiekvienas tinkle, tarkime IPFS tinklo vartotojas. Šios lentelės yra naudojamos koordinuoti ir sekti metaduomenis apie peer-to-peer sistemas.

3.2. Blokų apsikeitimai (BitTorrent)

BitTorrent tai yra plačiai naudojama peer-to-peer pagrindu paremta failų apsikeitimo sistema, kuri sėkmingai koordinuoja nepatikimus (untrusted) tinklo narius, kurie dalinasi failais vienas su kitu. Pagrindinės BitTorrent protokolo savybės, kurios darė įtaką IPFS kūrimui ir architektūrai:

- BitTorrent duomenų apsikeitimo protokolas naudoja strategiją, kuri apdovanoja tinklo narius, kurie padeda vienas kitam keistis duomenimis ir baudžia tuos, kurie tik siunčiasi duomenis sau, bet nepadeda kitiems;
- BitTorrent duomenų apsikeitimo protokolas seka, kurie duomenis yra mažiausiai aptinkami sistemoje ir juos prioretizuoja.

3.3. Versijavimo sistemos (Git)

Versijavimo sistemos kaip Git leidžia lengvai sekti besikeičiančias failų versijas ir jomis dalintis efektyviai su kitais tinklo nariais. Jos paremtos Merkle DAG duomenų struktūra. Git sistemos savybės, kurios darė įtaką IPFS kūrimui ir architektūrai:

- objektai yra pasiekiami pagal jų adresą, kurį sugeneruoja dėstymo funkcija priklausomai nuo jų turinio;
- saitai tarp objektų yra nekeičiami, įtvirtinti, kas leidžia užtikrinti objektų vientisumą;

- versijavimo metaduomenys yra adresai, kuriuos sukurti ir pakeisti nekainuoja brangiai.

3.4. IPFS architektūra

IPFS sugebėjo pasinaudoti geriausiomis idėjomis iš jau sukurtų peer-to-peer sistemų (BitTorrent, Git, paskirstytos dėstymo lentelės). IPFS supaprastino, pagerino sujungė patvirtintas technologijas į vieną darnią sistemą. IPFS pasiūlė naują sistemą skirtyti ir versijuoti didelius kiekius duomenų, o galiausiai ši technologija gali padėti internetui vystytis, tik dar iki galo nėra žinoma, kokia linkme ir kur tai nuves.

Kaip ir bet kokia paskirstyta sistema, IPFS yra peer-to-peer pagrindu sukurta sistema, kurioje jokie tinklo nariai nėra privilegijuoti. Tinklo nariai saugo objektus savo lokalioje saugykloje, o kad juos perduotų, jungiasi su kitais tinklo nariais.

IPFS protokolas yra padalintas į paprotokolius, kurie yra atsakingi už skirtingas funkcijas [Ben14]:

- objekto tapatybės – atlieka tinklo nario tapatybės sukūrimą bei verifikaciją;
- tinklas – valdo tarpusavio susijungimus tarp tinklo narių, tam gali naudoti daugybę tinklo protokolų, jie gali būti konfigūruojami;
- nukreipimas – saugo informaciją, kuri padeda tinklo nariams atrasti vienas kitą bei atrasti visą objektus tinkle;
- apsikeitimas – blokų apsikeitimo protokolas, kuris valdo efektyvų ir saugų blokų paskirstymą tarp tinklo narių;
- failai – versijuota failų sistemos hierarcija;
- įvardijimas – save sertifikuojanti besikeičiančių vardų sistema.

Kiekvienai šiai IPFS daliai paaiškinti reikėtų atskiro darbo, todėl tai paliksiu tolimesniems tyrimams.

4. Decentralizuoto realaus laiko komunikavimo žiniatinklyje prielaidos

2019 kovo mėnesį paskelbtoje studijoje apie WebRTC aplikacijas, paslaugas ir spėjimus 2019-2024 metams, yra rašoma, kad su 5G ryšio atėjimu, WebRTC bus vienas iš pagrindinių realaus laiko komunikavimo standartų [Res19]. Tai bevielius prietaisus leis padaryti dar labiau funkcionalius, patogesnius. Taip pat minima, kad WebRTC komunikacijų paslaugų tiekėjams bus viena iš pagrindinių technologijų, kuri leis sumažinti kaštus ir padidinti savo pelną. 2019 metais Google atskleidė, kad jos naršyklėje Chrome įvyksta 1.5 milijardo garso arba vaizdo bendravimo minučių per savaitę, naudojant WebRTC technologiją.

WebRTC technologija jau yra palaikoma labiausiai paplitusiose interneto naršyklėse: Microsoft Edge (nuo 76 versijos), Firefox (nuo 22 versijos), Google Chrome (nuo 28 versijos), Safari (nuo 11 versijos), Opera (nuo 18 versijos), telefonuose su Android arba iOS operacinėmis sistemomis ir naujausiomis interneto naršyklėse.

Jau dabar pasaulio WebRTC rinka yra vertinama 1.669 milijardo dolerių, o iki 2025 metų ji pasieks 21.023 milijardų dolerių vertę [Zio19].

Nors ir IPFS pradėta plėtoti 4 metais vėliau nei WebRTC, ji irgi pradeda įgauti pagreitį: jau yra sukurtos IPFS protokolo realizacijos Go, JavaScript, C, Python programavimo kalboms. 2017 metais dauguma svetainių susijusių su Katalonų nepriklausomybės referendumu buvo užblokuotos. Katalonų piratų partija (Catalan Pirate Party) panaudojo IPFS, tam, kad apeitų šį Konstitucinio Ispanijos teismo sprendimą [Tru17].

5. Decentralizuoto realaus laiko komunikavimo žiniatinklyje paslauga

Įvertinus visas prielaidas bei galimybes decentralizuoto realaus laiko komunikavimo žiniatinklyje sukūrimui, buvo nuspręsta sukurti paslaugos prototipą naršyklėje. Paslauga įgalins vartotoją komunikuoti realiu laiku su kitais vartotojais naudojantis tik naršykle, ir svarbiausia, nebus naudojamas centralizuotas serveris. Planuojami įgyvendinti funkcionalumai paslaugos prototipe:

- galimybė susirašyti su kitais vartotojais prisijungusiais prie paslaugos (vyksta per IPFS technologiją, naudojant publikavimo-prenumeravimo (angl. *Publish-Subscribe pattern*) programavimo šabloną;
- galimybė apsikeisti vaizdo bei garso ryšiu per WebRTC technologiją, naudojant IPFS technologiją kaip signalizavimo stadijai reikalingą serverį – **IPFS pakeičia signalizuojančio serverio būtinybę;**
- galimybė apsikeisti failais per IPFS tinklą.

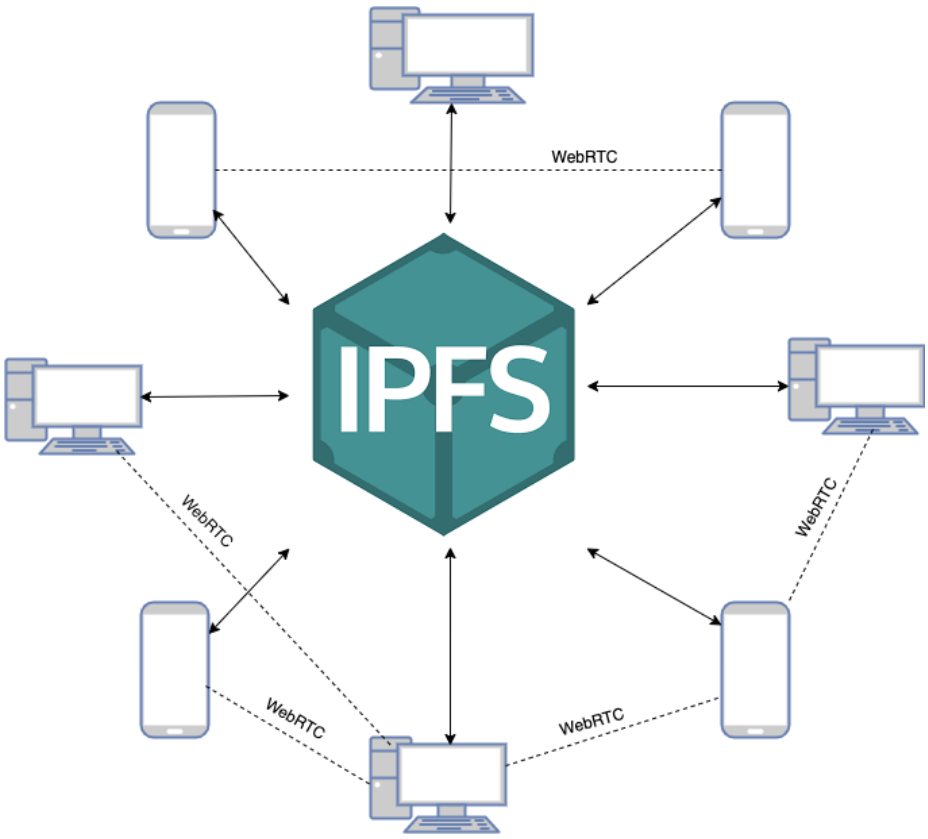
5.1. Paslaugos architektūra

6 paveiksle pavaizduota paslaugos architektūra. Kiekvienas besinaudojantis paslauga vartotojas prisijungęs prie paslaugos tinklalapio žiniatinklyje jungiasi prie IPFS tinklo ir tampa tinklo nariu.

Nuo prisijungimo prie IPFS tinklo momento, kiekvienas tinklo narys gali gauti žinutes, kas yra padaroma panaudojant publikavimo-prenumeravimo programavimo šabloną: prisijungę vartotojai iškart prenumeruoja visus pokalbių kambaryje gautus pranešimus (kai bet kuris prisijungęs vartotojas išsiunčia žinutę – jis publikuoja ją į kambarį, o kiti vartotojai ją gauna – taip atnaujinamas pokalbių langas).

Norėdami užmegzti vaizdo ir garso ryšį su kitu vartotoju, tai galima padaryti paspaudus mygtuką „Call“. Po paspaudimo pradeda vykti pirmoji WebRTC technologijos stadija – signalizavimas, per kurią apsikeičiama informacija bei užmezgiamas (arba įvyksta klaida) ryšys. Užmezgus ryšį, kitas vartotojas yra matomas gyvai bei galima pradėti bendrauti balsu. Vienu metu vartotojas gali užmegzti vaizdo ir garso ryšį su neribotu skaičiumi kitų vartotojų, tačiau realybėje dažniausiai tai būna iki 5 vartotojų, kadangi yra intensyviai naudojami kompiuterio resursai. Jeigu norima išjungti vartotoją, spaudžiamas mygtukas „Hang up“.

Vartotojas norėdamas pasidalinti failu su kitais vartotojais, spaudžia mygtuką „Send file“ pasirenka bei įkelia failą. Sistema jį įkelia į IPFS tinklą bei automatiškai pasidalina nuoroda pokalbių lange su visais kitais vartotojais.

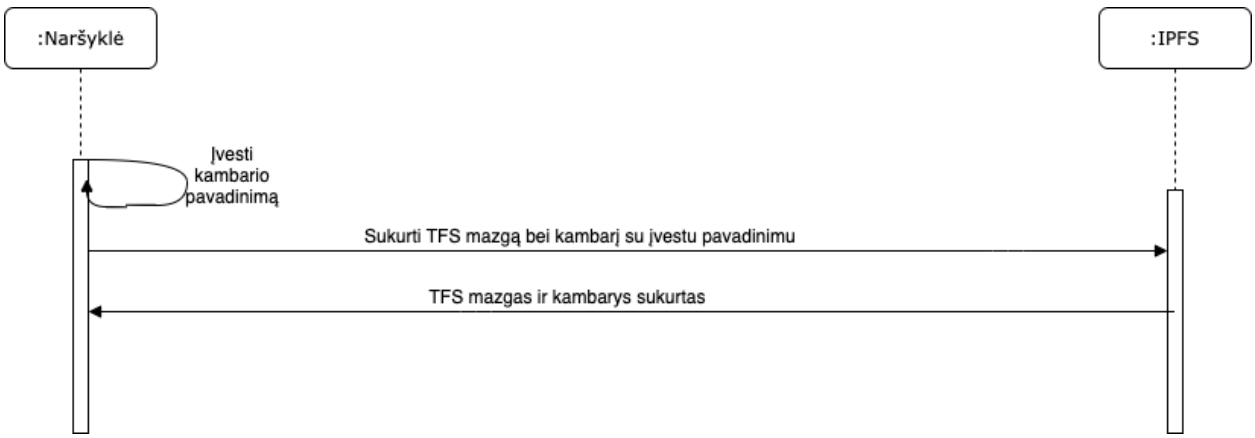


6 pav. Paslaugos architektūra

5.2. Paslaugos sekų diagramos

5.2.1. Vartotojo prisijungimo į paslaugą sekos diagrama

7 paveiksle pavaizduota vartotojo prisijungimas į paslaugą. Prisijungęs, vartotojas privalo įvesti kambario pavadinimą, į kuri nori prisijungti. Įvedęs kambario pavadinimą, naršyklė automatiškai sukuria IPFS mazgą, kuriuo vėliau manipuliuoja bei naudojasi: gauna žinutes, kai prisijungia naujas vartotojas, priima kitų žmonių žinutes, gali atlikti WebRTC signalizavimo veiksmus.

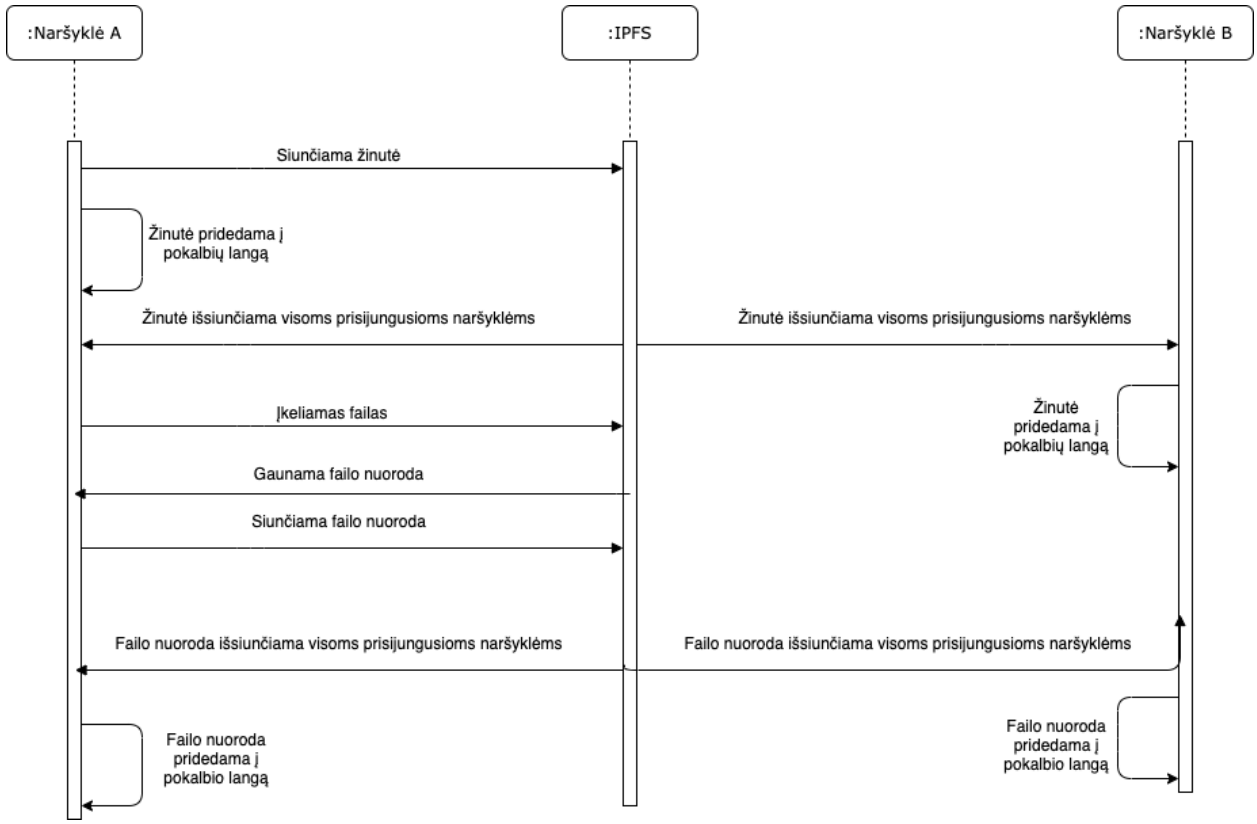


7 pav. Vartotojo prisijungimo į paslaugą sekos diagrama

5.2.2. Žinutės arba failo siuntimo/gavimo sekos diagrama

8 paveiksle pavaizduota žinutės arba failo siuntimo/gavimo sekos diagrama. Vartotojui sėkmingai įvykdžius „vartotojo prisijungimo į paslaugą“ veiksmą, jis gali siųsti žinutes bei siųsti failus kitiems pokalbio kambario vartotojams. Išsiuntus žinutę, vartotojo naršyklė nusiunčia žinutę į IPFS technologija paremtą kambarį, kuris visiems prenumeravusiems tą kambarį tą žinutę išsiunčia atgal. Visos žinutės turi identifikatorių **from,** kuris nurodo iš kurios naršyklės buvo siųsta žinutė. Jeigu žinutė yra gauta iš paties savęs, ji yra ignoruojama.

Tokiu pat būdu yra galimybė įkelti failą visiems pokalbio dalyviams. Failas yra įkeliamas į IPFS tinklą, tada gauta nuoroda yra nusiunčiama į IPFS technologiją paremtą kambarį, kuris visiems prenumeravusiems tą kambarį to failo nuorodą išsiunčia atgal.



8 pav. Žinutės arba failo siuntimo/gavimo sekos diagrama.

5.2.3. Vaizdo ir garso pokalbio užmezgimo sekos diagrama

9 paveiksle pavaizduota vaizdo ir garso pokalbio užmezgimo sekos diagrama. Tai yra standartine WebRTC technologijos procedūra. Kaip ir buvo minėta darbe, WebRTC negali užmegzti ryšio be serverio esančio tarp dviejų tinklo narių. Tą serverį mes vadiname signalizuojančiu kanalu ar signalizuojančiu serveriu. Tai gali būti bet kokia komunikacijos forma tarp tinklo narių, kuria įmanoma perduoti informacija prieš sukuriant ryšį: elektroninio pašto laiškas, atvirutė ar pašto balandis. Dažniausiai tam yra naudojama WebSocket technologija,

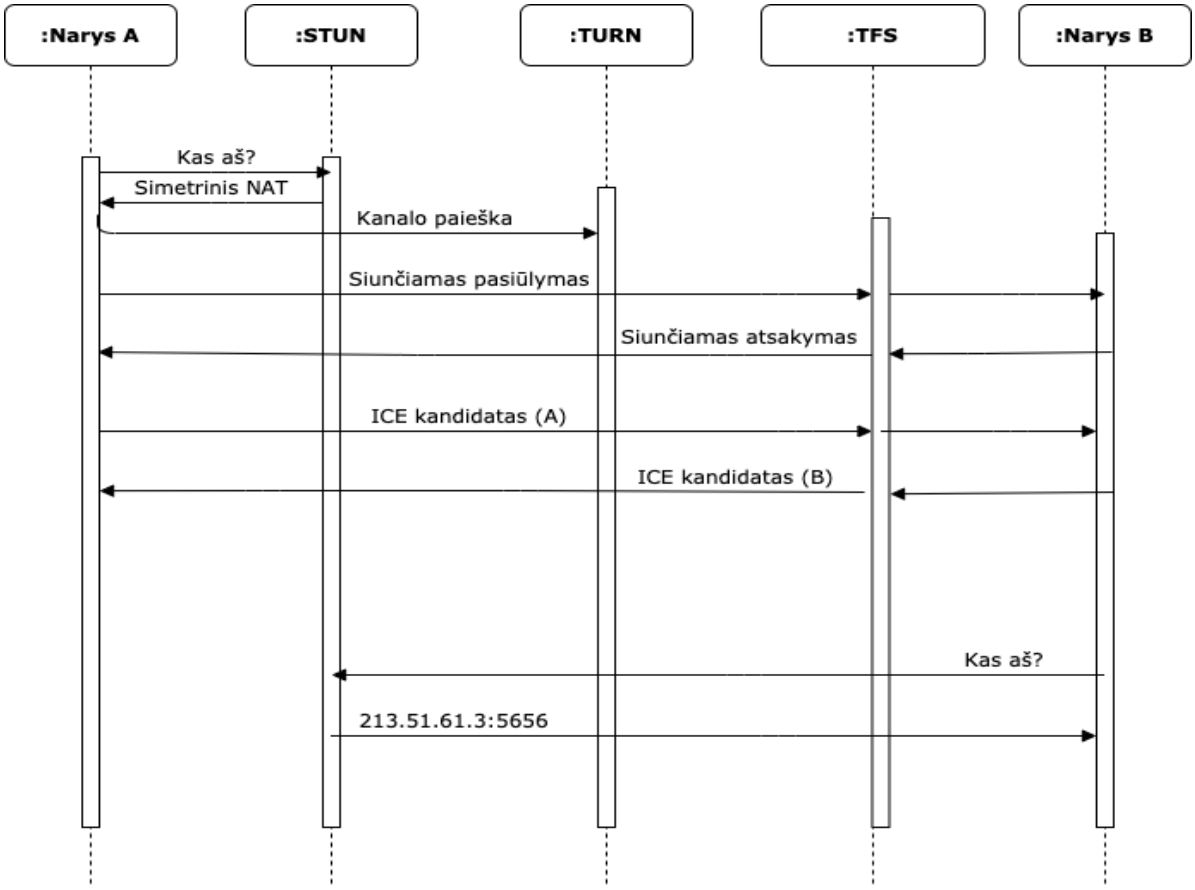
pasitelkiant atskirą serverį. Problema kyla tada, kai tas serveris yra nepasiekiamas. Autoriaus kuriamos paslaugos atveju, signalizuojantis kanalas bus paremtas IPFS technologija.

Tarkime, narys A yra ryšio iniciatorius, jis sukurs pasiūlymą (angl. *offer*). Tada jis išsiunčia pasiūlymą nariui B signalizuojančio kanalu. Narys B gauna pasiūlymą ir sukuria atsakymą (angl. *answer*). Tada narys B siunčia atgal atsakymą nariui A.

Pasiūlymą sudaro visa informacija apie nario pasiūlytą konfigūraciją kuriamam ryšiui, atsakymą sudaro analogiški duomenys. Užmezgus ryšį abu nariai turi savo bei kito nario duomenis (pasiūlymą arba atsakymą).

Diagramoje matosi ir kiti agentai bei esybės:

- STUN – tai protokolas, kuris leidžia rasti kompiuterio viešą IP adresą. Klientas siunčia užklausą į STUN serverį, kuris atsako su kliento viešu IP adresus ir ar klientas "nesislepia" už maršrutizatoriaus IP adreso.
- TURN – jei klientas slepiasi už maršrutizatoriaus IP adreso, reikalingas TURN serveris, kuris gali apeiti šį apribojimą persiųsdamas tinkle paketus reikiamu IP adresu.
- ICE – karkasas, kuris leidžia žiniatinklio naršyklėms prisijungti prie kitų naršyklių. ICE karkasas naudoja STUN ir TURN serverius.



9 pav. Vaizdo ir garso pokalbio užmezgimo schema.

5.3. Paslaugos realizacija

5.3.1. Programavimas

Kadangi sistemos apimtis nedidelė, nutarta kuriant sistemą nenaudoti jokių javascript programavimo karkasų, kurie palengvina komponentinį programavimą. Funkcionalumą stengtasi izoliuoti failuose pagal funkcijas. Stipriai remiamasi pirmuoju SOLID pricipu – vienos atsakomybės principas. Pavyzdžiui, faile WebRTC.js įvykdytas ryšio užmezgimo funkcionalumas, faile IPFS.js įvykdytas mazgo kūrimo, failo įkėlimo funkcionalumas. Failai gali vienas kitą naudoti.

Kuriama paslauga naudoja js-ipfs biblioteką, kuri yra pasiekiama per viešą npm (Node Package Manager) bibliotekų valdyklę. Projekto pradžioje buvo pradėta naudoti 0.40 versija, o šiuo metu ji jau yra atnaujinta iki 0.44 versijos. Reikia paminėti, kad bibliotekos funkcionalumas yra tik alfa stadijoje, dar nerekomenduojama jos naudoti gamybinėje aplinkoje. Bibliotekos funkcijų aprašymas yra peteikiamas viešoje GitHub bibliotekos saugykloje.

Buvo pasinaudota keliomis bibliotekos programavimo sąsajomis:

- files – failo įkėlimui į IPFS;
- pubsub – įvykdyti signalizavimo funkciją WebRTC technologijoje, dalintis žinutėmis tinkle, pasidalinti įkelto failo nuoroda pokalbių lange;
- swarm – prisijungimui prie spiečiaus

Pirmiausia yra kuriamas IPFS mazgo objektas, kurį kuriant būtina nurodyti vietinės saugyklos adresą. Šiuo atveju svarbus saugyklos pavadinimo unikalumas, todėl pasirinktas toks adreso generavimo algoritmas: Math.random() + Date.now(). Norint įgalinti vartotojus burtis į skirtingus privačių pokalbių kambarius, buvo nuspręsta kurti atskirus pokalbių kambarius. Programiškai tai yra padaroma sukūrus bibliotekos ipfs-pubsub-room 1.2.1 versijos (kuri irgi yra pasiekiama per npm) objektą, kuriam reikia nurodyti IPFS mazgo objektą bei kambario pavadinimą. Vėliau programoje naudojamas sukurtas kambario objektas:

- Naudojama funkcija broadcast(message) – kuri pasidalina žinute su visais vartotojais, kurie prisijungę prie kambario. Ši funkcija naudojama dalintis įkelto failo nuoroda arba išsiųsta žinute.
- Naudojama funkcija sendTo(peer, message) – kuri pasidalina žinute tik su vienu vartotoju. Šį funkcija naudojama vykdyti vaizdo/garso ryšio užmezgimui, tai yra, kaip WebRTC signalizuojantis kanalas.

Taip pat prototipo įgyvendinimo metu buvo sukurtos tokie javascript failai:

- LocalMedia.js – gaunamas vartotojo kameros vaizdas bei kompiuterio garsas;
- Messages.js – faile sukurta žinutės įdėjimo į pokalbio langą funkciją;

- Users.js – failas atsakingas už prisijungusių vartotojų rodymą, „Call“ ir „Hang up“ mygtukų sukūrimą, atitinkamų programinio kodo įvykdymą paspaudus juos;
- Videos.js – failas atsakingas už video langų pridėjimą arba išėmimą įvykus arba nutrūkus WebRTC susijungimui.

5.3.2. Paslaugos viešas pasiekiamumas

Suprogramavus paslaugą, buvo pasinaudota „webpack“ modulių surinkimo programa. Ji transformuoja, sujungia bei supakuoja visus projekto failus į tris:

- index.html, kuriame nurodoma puslapio struktūra;
- bundle.css, kuriame nurodomi puslapio elementams suteikiami stiliai;
- index.js, kuriame sudėti visos projekte panaudotos bibliotekos bei sukurti failai.

Tokį failų rinkinį yra labai patogu įkelti į bet kokią objektų saugojimo paslaugą (angl. *hosting*). Autorius pasirinko AWS S3 paslaugą, kadangi ji yra beveik nemokama bei turima patirties dirbti su ja. Planuojama, kad paslauga bus ilgą laiką pasiekiama adresu https://vu-bakalauras-ipfs-webrtc.s3.eu-central-1.amazonaws.com/index.html.

5.3.3. Paslaugos išbandymas ir testavimas

Kadangi šią paslaugą sunkiai galima ištestuoti individualiai, autorius subūrė kurso draugų komandą. Žinučių siuntimas ir failų siuntimas veikė kaip ir tikėtasi, 30 vartotojų (naršyklių) veikė nepriekaištingai Google Chrome 81.0, Safari 13.1 naršyklėse.

Kitame etape buvo vaizdo/garso skambučių testavimas. Paslauga Google Chrome naršyklėje veikė kaip ir tikėtasi. Tačiau kilo problemų su kitomis naršyklėmis. WebRTC signalizavimo stadijoje nepavykdavo užmegzti ryšio, jeigu buvo naudojamasi ne naujausia Google Chrome versijos interneto naršykle. Autorius mano, jog tai yra netobulas WebRTC susijungimo algoritmo išpildymas, tačiau klaidos surasti nepavyko, kadangi naršyklės klaidos pranešimai nėra informatyvus.

Per Google Chrome susijungus keturiems vartotojams į bendrą pokalbį buvo galima puikiai vienas kitą matyti, girdėti bei susirašinėti pokalbių lange. Prisijungus penktajam vartotojui, visi vartotojai pradeda jausti padidėjusi kompiuterių procesorių darbą: buvo naudojama nuo 60% iki 100% procesorių pajėgumo. Tai galima paaiškinti tuo, kad kiekvienas vartotojas sukuria atskira ryšį su kiekvienu vaizdo/garso pokalbio dalyviu ir kiekvienam atskirai siunčia, bei turi apdoroti gautą vaizdo/garso ryšį. Tam reikia didelių kompiuterio resursų.

Gavus atsiliepimų iš sistemą testavusių vartotojų, nuspręsta, jog dar reikia tobulinti vartotojo sąsają bei patirtį, sukurti stabilias ir atkartojamas metrikas, kurios galėtų būti naudojamos matuojant kliento resursų naudojimą bei komunikacijos laikus tarp klientų. Be to, reikia išsiaiškinti priežastis, kodėl pavyksta sukurti vaizdo/garso ryšį tik tarp Google Chrome naršyklių.

Rezultatai ir išvados

Šiame darbe pasiekti rezultatai:

1. Apžvelgtas realaus laiko komunikavimas ir palygintos skirtingos programų architektūros, kurias taikant galima įgyvendinti realaus laiko komunikavimą.
2. Išanalizuota WebRTC technologija, kuri įgalina realaus laiko komunikavimą žiniatinklyje ir mobiliosiose aplikacijose.
3. Apžvelgta IPFS technologija, kuri leidžia decentralizuotai keistis informacija.
4. Įvertinta WebRTC ir IPFS technologijų sintezės galimybė.
5. Sukurtas ir ištestuotas realaus laiko komunikavimo paslaugos prototipas naršyklėje, panaudojant WebRTC ir IPFS technologijas.

Atlikus darbą prieita prie šių išvadų:

1. WebRTC ir IPFS technologijos yra suderinamos – IPFS gali būti naudojamas kaip signalizuojančio kanalo įgyvendinimas WebRTC ryšio užmezgime.
2. WebRTC ir IPFS technologijų pagalba įmanoma realizuoti decentralizuotą realaus laiko komunikavimą vien tik interneto naršyklės pagalba.
3. Naršyklių WebRTC realizacijos yra skirtingos brandos stadijose. Tik naujausioje Google Chrome versijoje (81.0) WebRTC signalizavimo stadijoje pavyko užmegzti ryšį. Kitose naršyklėse vaizdo/garso ryšio užmezgimas neveikė.
4. WebRTC standartas įgyvendintas ir palaikomas visose naujos kartos interneto naršyklėse, tuo tarpu js-ipfs bibliotekos realizacija naršyklėje yra tik alfa stadijoje, todėl šią biblioteką reiktų labai atsargiai naudoti gamybinėje aplinkoje, kur gali būti atskleisti privatūs arba konfidencialūs duomenys.

Šaltiniai

[Ben14] Juan Benet, IPFS - Content Addressed, Versioned, P2P File System. arXiv:1407.3561, 2014. https://arxiv.org/abs/1407.3561 (tikrinta 2020-01-05).

[Bra15] Dragan Posarac Branislav Sredojev Dragan Samardzija. WebRTC technology overview and signaling solution design and implementation. 2015. url: https://pdfs.semanticscholar.org/b0ab/3531023a8ec659169196903ee94b38dc78c6.pdf (tikrinta 2017-05-11).

[Cla18] Kate Clark. Tim Berners-Lee is on a mission to decentralize the web, https://techcrunch.com/2018/10/09/tim-berners-lee-is-on-a-mission-to-decentralize-the-web/ (tikrinta 2020-01-11).

[Dic16] Ben Dickson. The internet's weaknesses will get worse before they get better, https://www.dailydot.com/layer8/ddos-mirai-iot-botnet-broken-internet/ (tikrinta 2020-01-05).

[FSS+18] Feher, Ben & Sidi, Lior & Shabtai, Asaf & Puzis, Rami & Marozas, Leonardas. (2018). WebRTC security measures and weaknesses. International Journal of Internet Technology and Secured Transactions. 8. 78. 10.1504/IJITST.2018.09213.

[JSI17] JS-IPFS, red. Ipfs/js-ipfs. GitHub. 2020. https://github.com/ipfs/js-ipfs (tikrinta 2020-04-02).

[Get20] What is WebRTC? Everything you need to know about WebRTC in an easy to understand way. https://gettalkative.com/what-is-webrtc (tikrinta 2020-01-11).

[Gor17] Svetlana Gordiyenko. WebRTC Browser Security: Real-Time Communications for B2B, https://xbsoftware.com/blog/webrtc-browser-security-real-time-communications-for-b2b/ (tikrinta 2020-01-05).

[LR14] Salvatore Loreto, Simon Pietro Romano. Real-Time Communication with WebRTC. Peer-to-peer in the browse, O'Reilly. ISBN: 978-1-449-37187-6.

[Res19] Research And Markets. WebRTC Market by Software, Applications, Services, Solutions, and Devices with Global and Regional Forecasts 2019-2024.

[Tru17] IPFS and Distributed Technology Aids Catalonia as Rubber Bullets Are Fired. https://www.trustnodes.com/2017/10/01/ipfs-distributed-technology-aids-catalonia-rubber-bullets-fired (tikrinta 2020-01-05).

[Web20] WebRTC official page. https://webrtc.org/ (tikrinta 2020-01-05).

[Zio19] ZioMarketResearch. WebRTC Market by Solution (Gaming, Messaging & File Sharing, Social Networking, Video Calling & Conference, and Voice Calling &

Conference), by Service (Consulting Services, Implementation & Integration Services, and Others), and by End-Use Industry (Retail, BFSI, IT & Telecom, Healthcare, Government Sector, Media & Entertainment, and Others): Global Industry Perspective, Comprehensive Analysis, and Forecast, 2018–2025.

[ZMM+18] Yixin Zou, Abraham H. Mhaidli, Austin McCall, and Florian Schaub, School of Information, University of Michigan. Proceedings of the Fourteenth Symposium on Usable Privacy and Security, Baltimore, MD, USA, 2018. ISBN 978-1-939133-10-6.

Sąvokų apibrėžimai

HTTP (Hypertext Transfer Protocol Secure) - pagrindinis metodas informacijai pasauliniame tinkle (WWW) pasiekti. Pradinė protokolo paskirtis – pateikti standartinį būdą HTML puslapiams skelbti ir skaityti.

HTTPS (Hypertext Transfer Protocol Secure) – panašus į HTTP, tik yra užtikrinamas duomenų šifravimas ir saugumas juos perduodant.

APS – Aplikacijų programavimo sąsaja (API – Application Programming Interface) – tai sąsaja, kuria kitos programos gali pasiekti kitos programos funkcijas, duomenis.

VoIP (Voice over Internet Protocal) – interneto protokolas, kuris įgalina perduoti balso ryšį.

MITM (Man-In-The-Middle) ataka – ataka, kuri įvykdoma, kai nusikaltėliai pakeičia arba pasisavina duomenis, kurie buvo besikeičiami tarp dviejų pusių, kurios mano, kad bendrauja privačiai.