UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET



**WebRTC komunikacija korišćenjem NodeJS programskog jezika i ugrađenih (native) API funkcionalnosti JavaScript ECMAScript 6 standarda**

SEMINARSKI RAD

PREDMET: Video sistemi [13e034VS]

Mentor: Kandidat:

Doc. Dr Ana Gavrovska Žarković Nikola 0434/2014

Beograd, 2019.

sadržaj

|  |  |  |
| --- | --- | --- |
|  |  | Str. |
|  |  |  |
|  | Spisak skraćenica | 3 |
|  | Spisak slika | 4 |
|  | Spisak tabela | 5 |
|  |  |  |
| 1. | Predgovor | 6 |
|  |  |  |
| 2. | Uvod | 6 |
|  |  |  |
| 3. | Postavka problema | 7 |
|  |  |  |
| 4. | Principi komuniciranja u realnom vremenu | 8 |
| 4.1. | Pojam realnog vremena | 8 |
| 4.2. | Benefiti komunikacije u realnom vremenu | 10 |
| 4.3. | Real time komunikacija u okviru mission-critical sistema u 5G mreži | 11 |
| 4.4. | Web komunikacija i realno vreme | 12 |
|  |  |  |
| 5. | Meahnizmi web komunikacije u realnom vremenu | 13 |
| 5.1. | HTTP Zahtevi i mehanizam status kodova | 14 |
| 5.2. | Web komunikacija pomoću standardnih AJAX zahteva | 16 |
| 5.3. | Pojam Web Socketa u web komunikacijama | 17 |
| 5.4. | WebRTC komunikacija | 18 |
| 5.4.1. | STUN Serveri | 20 |
| 5.4.2. | TURN Serveri |  |
|  |  |  |
| 6. | Implementacija jednog web RTC servisa | 21 |
| 6.1. | Razvoj Back-End Sistema | 21 |
| 6.2. | Izgled korisničkog interfejsa | 25 |
|  |  |  |
| 7. | Zaključak | 30 |
|  |  |  |
|  | Literatura | 31 |
|  |  |  |
|  |  |  |

SPISAK KORIŠĆENIH SKRAĆENICA

|  |  |  |
| --- | --- | --- |
| Skraćenica | Puni naziv | Objašnjenje (na srpskom) |
|  |  |  |
| WebRTC | Web Real-Time Communication | Web komunikacija u realnom vremenu |
| API | Application Programming Interface | Aplikativni interfejs (skup funkcionalnosti) |
| ICT | Information and Communication Technology | Informacione i komunikacione tehnologije |
| ISP | Internet Service Provider | Operater koji pruža internet usluge |
| PSTN | Public Switched Telephone Network | Javna fiksna mreža (mreža fiksne telefonije) |
| RTT | Round Trip Time | Ukupno vreme prenosa informacije |
| HTTP | Hyper Text Transfer Protocol | Protokol za razmenu teksta i hiperlinkova |
| POS | Point Of Sale | Pristupna tačka za plaćanje platnim karticama |
| CNS | Centranlni Nervni Sistem |  |
| HAS | Human Auditory System | Čovečiji Auditorni Sistem |
| HVS | Human Visual System | Čovečiji Vizuelni Sistem |
| ABS | Anti-lock Braking System | Kočioni sistem protiv zaključavanja kočnice |
| RT | Real Time | Realno Vreme |
| ISO/OSI | International Standardization Organisation / Open System Interconnection | Model komunikacionih sistema međunarodne standardizacione organizacije |
| HTML | Hypertext Markup Language | Programski jezik za kreiranje web stranica |
| CSS | Cascading Style Sheets | Stilizator web stranica |
| AJAX | Asynchronous Javascript And XML | Asinhroni mehanizam komuniciranja sa serverom |
| XML | eXtensible Markup Language | Format podataka koji se veoma često koristi za komuniciranje izmedju klijenta i servera |
| UDP | User Datagram Protocol | Non-reliable protokol na ISO/OSI sloju 3 |
| JSON | JavaScript Object Notation | Objektna notacija za JavaScript – format poruke isto kao što je i XML |
| MPEG | Motion Picture Expert Group | Grupa eskperata za pokretnu sliku |

Spisak slika

|  |  |  |
| --- | --- | --- |
| Br. |  | Strana |
|  |  |  |
| Slika 2.1. | Logoi WhatsApp, Skype i Viber servisa kao predstavnika RTC provajdera | 7 |
|  |  |  |
| Slika 4.1. | Servisi koji koriste komunikaciju u realnom vremenu | 9 |
| Slika 4.2. | Prinrip prijema i obrade signala prilikom direktne komunikacije | 9 |
| Slika 4.3. | ABS automobilski kočioni sistem | 11 |
| Slika 4.4. | Princip utilizacije dronova i real-time videa u 5G mrežama | 12 |
|  |  |  |
| Slika 5.1. | Uprošćena topologija i mehanizam rada globalne mreže | 13 |
| Slika 5.2. | Pojednostavljenje web komunikacije | 14 |
| Slika 5.3. | Primer HTTP zahteva | 15 |
| Slika 5.4. | Primer HTTP Odgovora | 16 |
| Slika 5.5. | HTTP Komunikacija klijent server i zahtev za multimedijalnim sadržajem | 17 |
| Slika 5.6. | Komunikacija dva klijenta preko Web Socket mehanizma | 18 |
| Slika 5.7. | Princip Web RTC komunikacije | 19 |
| Slika 5.8. | Princip korišćenja STUN i TURN servera kod webRTC-a | 19 |
|  |  |  |
| Slika 6.1. | Fajlovi i folderi koji se koriste za potrebe projekta | 24 |
| Slika 6.2. | Pozdravna korisnička poruka da je server aktivan na portu 4000 i da osluškuje zahteve | 25 |
| Slika 6.3. | Izgled Web stranice prilikom prvog pokretanja | 25 |
| Slika 6.4. | Prikaz korisničkog kontrolnog strima i čekanje na drugog Peer-a | 26 |
| Slika 6.5. | Notifikacija na serverskoj strani da je korisnik ostvario konekciju | 26 |
| Slika 6.6. | Stanje nakon konekcije drugog korisnika na server | 27 |
| Slika 6.7. | Ostvarena komunikacija između dva uređaja (PC levo i Pametni telefon desno) | 28 |
| Slika 6.8. | Izgled strane za redirekciju | 28 |

Spisak TABELA

|  |  |  |
| --- | --- | --- |
| Br. |  | Str. |
|  |  |  |
| Tabela 5.1. | Najzastupljeniji HTTP status kodovi i njihova značenja | 16 |
|  |  |  |

1. *Predgovor*

Još od najranijih perioda, čovek je težio ka tome da svoja mišljenja i stavove iskaže odnosno iznese drugom čoveku ili grupi ljudi. Jedini način komunikacije u to vreme bila je direktna međusobna komunikacija, što je rezultovalo osnivanjem prvih gradova i urbanih sredina gde su ljudi zbog međusobne bliskosti tu komunikaciju mogli ostvariti na direktan način. Napretkom tehnologije a posebno rapidnom ekspanzijom globalne mreže (Interneta) ljudi pomoću svojih pametnih uređaja mogu lakše nego ikada ostvariti vezu i razgovarati, razmenjivati stavove, mišljenja, ideje, bez obzira na prepreke koje se nalaze između njih (razdaljina, vremenske zone, tipovi uređaja koje poseduju, vremenske prilike, ...).

U ovom radu izneću svoja zapažanja i istraživanja koja se tiču metoda modernih komunikacija, kao i realizacije modernih video sistema odnosno načina razmene audio i video sadržaja između krajnjih korisnika a koji su vremenom postali *de facto* standard u ICT industriji u domenu komuniciranja u realnom vremenu.

1. *Uvod*

O korisničkim aplikacijama za razmenu audio/video sadržaja se uveliko priča i gotovo da ne postoji osoba koja nije čula za neku od njih i/ili koristila u svrhu komunikacije bez obzira na godine starosti. Danas gotovo da ne postoji osoba koja nije čula za servise poput *Skype-*a , *WhatsApp-*a i *Viber-*a i koja iste nema instalirane na svom pametnom telefonu. O ovim aplikacijama možemo pisati naširoko, ali ono što njih karakteriše jesu sledeće osobenosti koje su privukle milione korisnika koji ove aplikacije svakodnevno koriste, a neke od njih su:

1. Pouzdanost prenosa signala
2. Jednostavan i intuitivan korisnički interfejs
3. End-to-end enkripcija
4. Mogućnost prenosa podataka (uz audio i video sadržaje) – *data traffic*
5. Tarifiranje kod ISP kroz potrošeni internet saobraćaj (nema zasebnog tarifiranja)
6. Mogućnost integracije sa starim PSTN i javnim mobilnim mrežama (2G, 3G, 4G)

Takođe, kada govorimo o prednostima ovih sistema moramo napomenuti i koje su to neke od njihovih mana, a među koje spadaju:

1. Kompletna zavisnost o globalnoj Internet mreži
2. Nepostojanje kontrole kvaliteta signala (kvalitet signala zavisi od kvalitet veze)
3. Neminovno kašnjenje prenosa signala koji postoji u mreži
4. Platformska zavisnost (aplikacije su podržane samo na određenim platformama)



Slika 2.1 – Logoi najzastupljenijih servisa za audio/video komunikaciju

Kada sagledamo sve prednosti i mane ovih sistema, uviđamo koliko zapravo tu postoji prostora za poboljšanje i inovacije u domenu komunikacija u realnom vremenu, kreiranja novih koncepata i principa komunikacije, kao i ekspanzije postojećih servisa na platforme koje do sada nisu podržane.

Probleme zavisnosti o globalnoj Internet mreži kao i problem kašnjenja prenosa signala, koji su inače povezani problemi ne možemo rešiti na ovaj način i jedino pogodno rešenje predstavlja reorganizaciju sistema odnosno kreiranje nove potpuno nezavisne mreže koja je veoma skupa i ni na koji način nije isplativa pružaocu usluge.

Ovaj rad je organizovan u X celina, u kojima ćemo proći kroz neke od osnovnih pravila prilikom komunikacije u realnom vremenu, pokrićemo i neke starije metode komunikacije između uređaja na mreži, a na posletku ćemo proći kroz webRTC standardizaciju i na praktičnom implementacionom primeru prikazati kako se na veoma jednostavan način može kreirati platformski nezavisna aplikacija za razmenu audio/video sadržaja (Video Chat Application).

1. *Postavka problema*

Problem koji se pred nas postavlja može se formulisati na sledeći način. Neka su data dva krajnja korisnika koje žele da ostvare međusobnu vezu u realnom vremenu (Real-Time Communication). Ne ulazeći u detaljnu analizu oko toga koje tipove uređaja korisnici imaju, koje operativne sisteme koriste i kakve su njihove preference po pitanju korišćenja web pregledača odnosno kakve su njihove mogućnosti po pitanju instalacije posebnih (dodatnih) aplikacija.

Drugim rečima, potrebno je napraviti aplikaciju koja će raditi što pouzdanije na velikom broju platformi i pomoću koje će dva korisnika moći izvršiti razmenu audio/video/data sadržaja.

Kada rešimo problem komunikacije 2 korisnika, veoma lako možemo smisliti mehanizam koji bi dozvolio komunikaciju više grupa korisnika (više od 2 korisnika istovremeno) – *Video Conferencing.*

1. *Principi komuniciranja u realnom vremenu*

Komuniciranje u realnom vremenu je veoma kompleksna tematika i obuhvata mnogo više od prenosa video signala od tačke A do tačke B. Razvojem mobilnih mreža V generacije (5G mreže), uveliko se radi na razvoju sistema za autonomno upravljanje vozilima, a koje iziskuje strahovito brzu međusobnu komunikaciju. Ovo je samo jedna od primena, u kojoj je brzina faktor od značaja.

Dakle, kako bismo bili što precizniji možemo reći da je **vreme komuniciranja** – „vreme potrebno da prilikom komunikacije poruka sa uređaja koji predstavlja pošiljaoca uspešno stigne do svog konačnog odredišta odnosno do uređaja/grupe uređaja koji su primaoci poruke“.

Prilikom definicije komunikacije u realnom vremenu, moramo prvo pojam realnog vremena, što je tematika sledećeg poglavlja.

* 1. *Pojam realnog vremena*

U zavisnosti od tipa servisa i od primene sistema, pojmovi realnog vremena se razlikuju drastično i često pojam realnog vremena u jednom sistemu ne mora biti odgovarajući u drugom i obratno. Kada pričamo o realnom vremenu, sa tehinčkog stanovišta, najčešće mislimo na vreme potrebno informaciji da se prenese do željenog odredišta, a da se potom vrati ka svom izvorištu u vidu potvrde o uspešno primljenoj informaciji. U domenu mreža i mrežne infrastrukture, ovaj parametar naziva se RTT (Round Trip Time)

Za potrebe otvaranja web stranice, odnosno odgovora na HTTP zahteve, zadovoljavajuće vreme odgovora na zahtev je do 5*s.* Realni slučaj pokazuje da je vreme odgovora u opsegu od 200*ms* od 1*s,* i da se za ovaj vid komunikacije ove vrednosti smatraju ispunjenim zahtevom u realnom vremenu.

Za audio sisteme, realno vreme je okvirno i iznosi do 100*ms* a svako vreme nakon toga dovodi do osetnog kašnjenja sistema i do primetnog opadanja kvaliteta. Ukoliko usvojimo da je brzina zvuka u vazduhu pri nominalnoj temperaturi od 20°C okvirno iznosi 340*m/s* dobijamo da je optimalno RTT vreme jednako vremenu koje treba zvuku da pređe 34 metara.

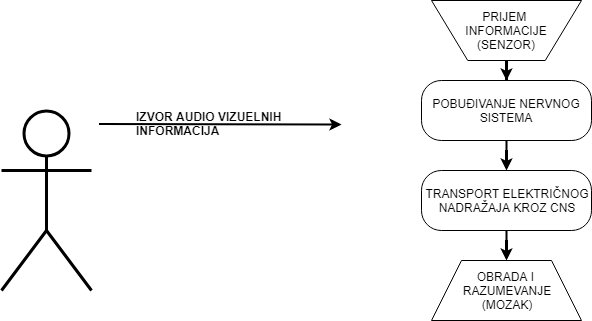
Video sistemi su znatno osetljiviji na kašnjenje, posebno ako se u obzir uzme činjenica da se pokretna slika generiše sa *frame rate*-om od 24 slika po sekundi (pa nadalje). Pri ovakvim uslovima moramo strogo voditi računa o kontinuitetu prenosa, kako bi se na prijemnoj strani (strani na kojoj se primljeni video reprodukuje) sadržaj mogao reprodukovati u kontinuitetu. Ukoliko dođe do prekira veze, pa makar i na kratko, kontinuitet se gubi i reprodukcija sadržaja postaje onemogućena.

Neki od primera komunikacije u realnom vremenu su komunikacija POS terminala sa matičnim (centralnim) sistemom, gde se u samo nekoliko sekundi izvrši autentifikacija, autorizacija i audit korisnka i njegove željene transakcije i izvrši se željena operacija. Takođe RTC se koristi i u vojne svrhe, kod kojih je moderna komunikaciona oprema spojena sa bojnom opremom (primopredajni sistem integrisan je u okviru vojnog šlema). U domenu video komunikacija, najzastupljeniji i možda jedan od najznačajnijih primera komunikacije u realnom vremenu jesu video sistemi za nadzor koji zahtevaju trenutni prikaz slike, i kod kojih svako vremensko odstupanje može dovesti do sigurnosnog problema.



Slika 4.1 – Servisi koji koriste komunikaciju u realnom vremenu

Ukoliko pak kao poredbeni kriterijum za termin realnog vremena uzmemo direktnu komunikaciju između 2 čoveka, možemo zaključiti da se ni prilikom direktne komunikacije dva čoveka komunikacija ne dešava bez kašnjenja i da postoji određena latencija koja je posledica rada i međusobne komunikacije CNS, AVS i HVS.



Slika 4.2 – Prinrip prijema i obrade signala prilikom direktne komunikacije

Kao što je prikazano na slici 4.2, kada se inicira komunikacija između 2 (ili više) čoveka, čovek koji prima informaciju mora sačekati da se ispoštuju sledeći procesi kako bi informacija uopšte došla do njega, i kako bi on razumeo tu informaciju.

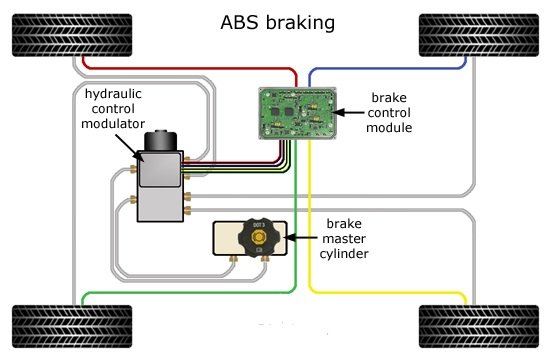
* Prvi korak je to da sadržaj koji biva prosleđen tom čoveku ima sintaksnu i semantičku vrednost
* Zatim u obzir moramo uzeti vreme propagacije signala od izvora pobude sve do fotoreceptora/slušnih receptora
* Signal se konvertuje iz jednog fizičkog oblika u električni i prosleđuje se kroz centralni nervni sistem u električnom obliku
* Sama propagacija impulsa kroz CNS takođe treba biti uzeta u obzir prilikom analize vremena trajanja signala
* Vreme obrade i vreme stvaranja slike u mozgu poslednji su korak ka potpunom primanju i razumevanju poslate informacije

Kao što možemo pretpostaviti svaki od ovih koraka unosi kašnjenje određenog reda veličine, a najkritičniji je drugi korak koji najviše utiče na vreme obrade i prenosa i jedini je na koji možemo uticati posredno ili neposredno.

* 1. Benefiti komunikacije u realnom vremenu

Komunikacija (RTC) sa sobom nosi veoma puno benefita, a mnogi od njih se ogledaju u servisima koji se već uveliko pružaju krajnjim korisnicima, ili su u procesu razvoja pa će se tek pružati. Neki od primera razvoja jesu u domenu eZdravlja (eng. *eHealth*) gde bi korisnici zdravstvenih usluga mogli da u realnom vremenu razgovaraju sa svojim lekarima bez odlaska u zdravstvenu ustanovu i zakazivanja. Takođe, domen u kome se ova komunikacija veoma često koristi i pomoću koje bi bilo jako teško zamisliti organizaciju velikih korporacija koje operišu iz više različitih država sa više različitih kontinenata jeste *Video Conferencing*, odnosno mogućnost povezivanja više fizički razdvojenih korisnika u jedan komunikacioni kanal pomoću kojega bi oni mogli da razmenjuju video sadržaj, audio sadržaj kao i digitalne podatke.

Takođe jedan primer brze komunikacije koristi se i u elektro-mehaničkim kočionim sistemima kod automobila novije generacije. Kako bi se implementirao ABS sistem, koji je najnoviji kočioni standard u automotive industriji, potrebno je izvršiti što bržu i što precizniju komunikaciju sa kontrolerom kočenja koji prima poruku, i na osnovu ulaznih parametara odlučuje koje će kočnice i na kojim točkovima aktivirati kao i silu koja će biti primenjena na kočione elemente kako bi automobil prilikom naglog kočenja pri velikim brzinama ostao stabilan na putu.



Slika 4.3 – ABS automobilski kočioni sistem

* 1. Real time komunikacija u okviru *mission-critical* sistema u 5G mreži

Kao što smo ranije pomenuli, sa razvojem novih mobilnih javnih mreža 5. generacije, pred provajdere je stavljen novi izazov, a to je omogućavanje *mission-critical* servisa korisnicima, odnosno omogućavanje servisa koji zahtevaju veliku brzinu realizacije zahteva i veliku preciznost.

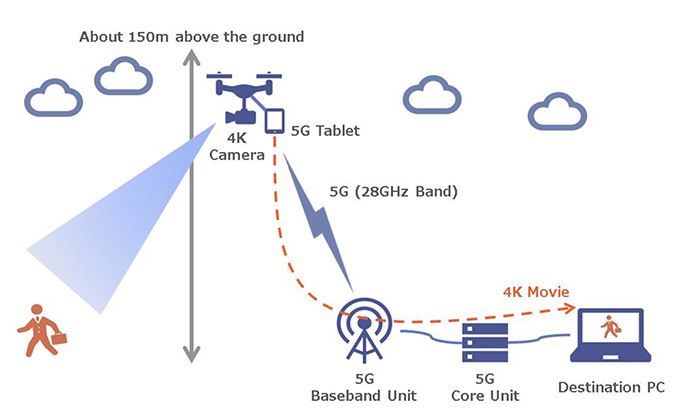
Primer korišćenja ovih RT servisa opet se ogleda u automotive industriji i to u domenu razvoja autonomnih vozila. Već sada imamo vozila koja omogućavaju vozaču udobnu vožnju u kooperativnom modu rada. Na primeru *Lane Assistent* funkcionalnosti, korišćenjem kamera na prednjoj i zadnjoj strani vozila, vrši se analiza saobraćaja i skeniranje saobraćajne trake čime se sprečava da vozač učestvuje u udesu ili skrene u suprotnu kolovoznu traku.

Ovakvi servisi u automotive industriji polako postaju standard, i podrazumevaju određenu vrstu autonomije, ali ipak je čovek odnosno vozač taj koji odlučuje o procesu upravljanja vozilom. Sa idejom 5G mreža razvija se i ideja autonomnih vozila, tj. vozila koja se samostalno kreću kolovozom i koja samostalno vode računa o pravilima saobraćaja, saobraćajnoj signalizaciji, drugim učesnicima u saobraćaju, vremenskim prilikama i ostalim parametrima koja veoma mogu uticati na kvalitet vožnje.

*Mission-critical* *automotive* servisi su novi koncept uveden za potrebe razvoja autonomnih vozila. Primera radi, na autoputu pri ograničenju brzine od 130km/h, u slučaju naglog kočenja, vreme reakcije i samo vreme kočenja trebalo bi svesti na minimalno moguće. U suprotnom dolazi do lančanih sudara koji su jako opasni i pri kojima nastaju teške telesne povrede. Drugim rečima, *mission-critical* servisi su oni koji zahtevaju minimalno mogući RTT u datoj situaciji.

Danas na tržištu imamo takav autonomni kočioni sistem kompanij Tesla Motors koji koristi poruku poslatu od strane drugog autonomnog vozila o tome da bi trebalo naglo zakočiti ukoliko želimo da izbegnemo sudar sa tim vozilom. Poruka primljena u realnom vremenu ima RTT reda nekoliko *ms* a ukupno vreme kočenja je svedeno sa ~10*s na ~*3.3*s* pri brzinama od 140*km/h*.

Takođe novi koncept koji bi trebalo zaživeti sa 5G mrežama i koji je u rapidnom razvoju jeste komunikacija u realnom vremenu koja se izvršava pomoću dronova (malih autonomnih letelica opremljenih tehnologijom za podršku 5G mreži). Takođe, principi upravljanja ovim dronovima bili bi autonomni i oni bi, kao i autonomni automobili morali komunicirati ekstremno brzo i veoma precizno kako bi se sprečile eventualne kolizije tj. sudari zbog neadekvatnog vremena prijema poruke i reakcije kao i zbog loše primljene poruke.



Slika 4.4 – Princip utilizacije dronova i real-time videa u 5G mrežama

* 1. Web komunikacija i realno vreme

Web komunikacija može se ostvariti na različite načine i pomoću raznih protokola, ali prvenstveno mehanizam koji nas interesuje jeste web komunikacija korišćenjem **HTTP** protokola. Prvobitno, ovaj vid komunikacije koristio se na globalnoj mreži za dobavljanje statičkih Web stranica. To su bile veoma jednostavne web stranice, a za pristup istima odnosno za pregled istih potrebno je imati instaliran web pregledač (*Web Browser*) koji primljenu stranicu ume da prikaže u željenom formatu. U tom periodu je za potrebe prikaza razvijen i prvi programski jezik za kreiranje web stranica – **HTML** *(Hypertext Markup Langage*).

Kako je tržište odnosno kako su krajnji korisnici želeli da web stranice budu primamljivijeg izgleda, razvio se i novi način stilizovanja web stranica – **CSS** (*Cascading Style Sheets*) jezik koji daje uputstvo pregledaču na koji način da prikaže web stranicu.

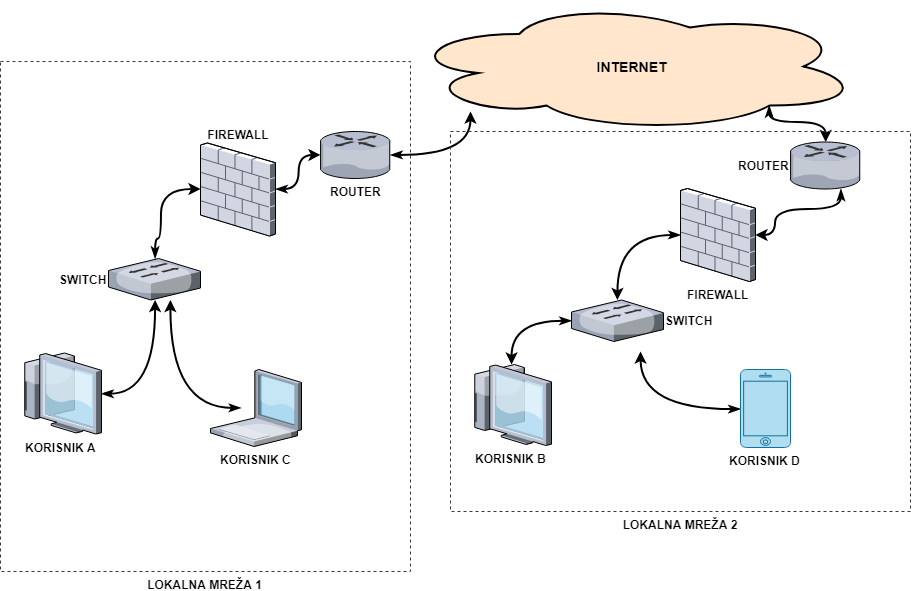
Poslednja tehnologija koja se koristila prilikom kreiranja web stranica, nastala je kao odgovor na zahtev korisnika za interaktivnošću. Razvijen je poseban *scripting* jezik koji je omogućio kreiranje interaktivnih web stranica. Naravno, ove web stranice bile su veoma jednostavnog dizajna i limitiranih mogućnosti. Primera radi, nismo podržavali polovinu servisa koji se koriste u modernim pregledačima pri otvaranju web stranica kreiranih modernim jezicima / tehnologijama.

Velika mana ovih sistema bila je statički prikaz stranica i nemogućnost prikaza poverljivih informacija korisniku, odnosno nemogućnost povezivanja web pregledača sa drugim sistemima (pozadinski sistemi poput baza podataka, web servisa, ... ). Pomenute tehnologije kreiranja statičkih web stranica često se združeno nazivaju jednim imenom i to kao *front-end* ili *front-facing*  tehnologije.

Pozadinske tehnologije (često referencirane i kao *back-end*) u fokus su došle ubrzo nakon realizacije da interaktivnost na web stranici sama po sebi nije dovoljna ukoliko se neka informacija ne može dostaviti nekom drugom korisniku odnosno prikazati na web pregledaču drugog korisnika. Ovakav vid komunikacije otvorio je vrata novim mogućnostima u domenu web komunikacije. Kreirani su prvi chat room-ovi i priv forumi koji su stekli ogromnu popularnost. Naravno, ništa ovo ne bi bilo moguće bez integracije pozadinskih sistema sa *front-end* sistemima i njihovim međusobno spregnutim radom.

1. Meahnizmi web komunikacije u realnom vremenu

Da bismo razumeli mehanizam web komunikacije, prvo je potrebno sagledati kako uopšte izgleda jedna internet mreža odnosno koji su to sve kanali i uređaji kroz koje treba da prođe jedan zahtev za preuzimanjem web stranice.

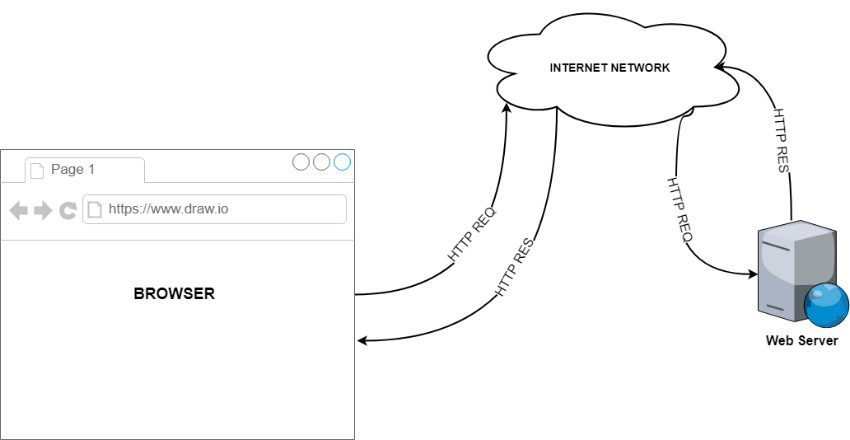


Slika 5.1 – Uprošćena topologija i mehanizam rada globalne mreže

Topologija mreže koja je prikazana na slici zapravo je osnova od koje počinjemo pri analizi jedne mreže i web komunikacije. Korisnički uređaji prvo se povezuju na čvorište (*Switch*) *–* uređaj zadužen za multipleksiranje i demultipleksiranje korisničkih signala. Zatim se signal šalje kroz *Firewall*, koji je zadužen za filtraciju odlazno / dolaznog saobraćaja i koji rešava prava korišćenja određenih servisa i dozvoljava odnosno zabranjuje dalju propagaciju mrežnog saobraćaja unutar jedne lokalne mreže. Poslednji uređaj o kojemo možemo govoriti na topologiji jes ruter (*Router*) čiji je zadatak da ispropagira mrežni saobraćaj do željenog uređaja na mreži a sve to metodom komutacije paketa (segmentacija poruke na delove i slanje delova različitim putanjama, a zatim ponovna kompozicija poruke na prijemu).

Danas su često ova tri mrežna uređaja (*Switch*, *Firewall* i *Router*) integrisana u jedinstveni uređaj koji se združeno naziva Ruter, ali on ima ugrađene funkcionalnosti i ostala dva mrežna uređaja, tako da treba biti pažljiv prilikom odabira terminologije.

Kada uzmemo u obzir da prilikom web komunikacije, odnosno slanja web zahteva ne želimo da zalazimo na niže slojeve ISO/OSI strukture, već pričamo o aplikativnom sloju, šema odnosno ilustracija komunikacije se znatno pojednostavljuje.



Slika 5.2 – Pojednostavljenje web komunikacije

Ono što moramo primetiti da je u ovom slučaju konceptualno izmenjeno jeste da, umesto korisnika odnosno korisničkog uređaja kao podnosioca zahteva imamo web pregledač (aplikativni sloj – layer 7). On pomoću *IP Adrese* (adrese uređaja na mreži) odnosno simboličke adrese (koja predstavlja sliku 1:1 između čoveku razumljive adrese i niza brojeva) kreira zahtev pomoću HTTP protokola i šalje ga na mrežu. Web Server (Uređaj čija je namena da opslužuje HTTP zahteve) odgovara na zahtev po protokolu i prosleđuje sadržaj web stranice. Web stranica se krajnjem korisniku dostavlja jednom i konekcija se prekida.

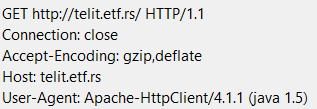
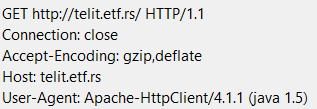
U slučaju da web stranica prikazuje neki video stream, konekcija između browser-a i web servera ili video streaming servera se ostavlja sve dokle postoji potreba za dostavljanjem sadržaja, a što predstavlja zasebne izazove koji su prevaziđeni određenim tehnologijama i o kojima će biti reči.

Treba imati u vidu da je u okviru praktičnog dela ovog rada autor kreirao Web Server kao i web stranicu koja će biti dostavljena na korisnički zahtev, a sve u svrhu demonstracije rada webRTC video komunikacije.

* 1. HTTP Zahtevi i mehanizam status kodova

Razvojem HTTP protokola, jedan od ozbiljnijih problema bio je način obrađivanja zahteva različite namene (kako razlikovati zahtev koji za cilj ima pouzdan prenos ka Web serveru od zahteva koji od Web servera traži statičku stranicu ili neku drugu informaciju). Tada je nasatala ideja uvođenja HTTP Glagola (*HTTP Verbs*) koji mogu biti GET,POST,PUT,DELETE a koji se često nazivaju i HTTP operacijama. U zavisnosti od željene akcije, korisnik može nešto zahtevati sa Web servera, može mu nešto poslati, može izmeniti neki postojeći atribut ili zahtevati da se nešto sa Web servera ukolni. Često se ove četiri operacije nazivaju CRUD operacijama (*Create Read Update Delete*).

Primer jednog HTTP zahteva (*HTTP Request*) dat je na sledećoj slici:

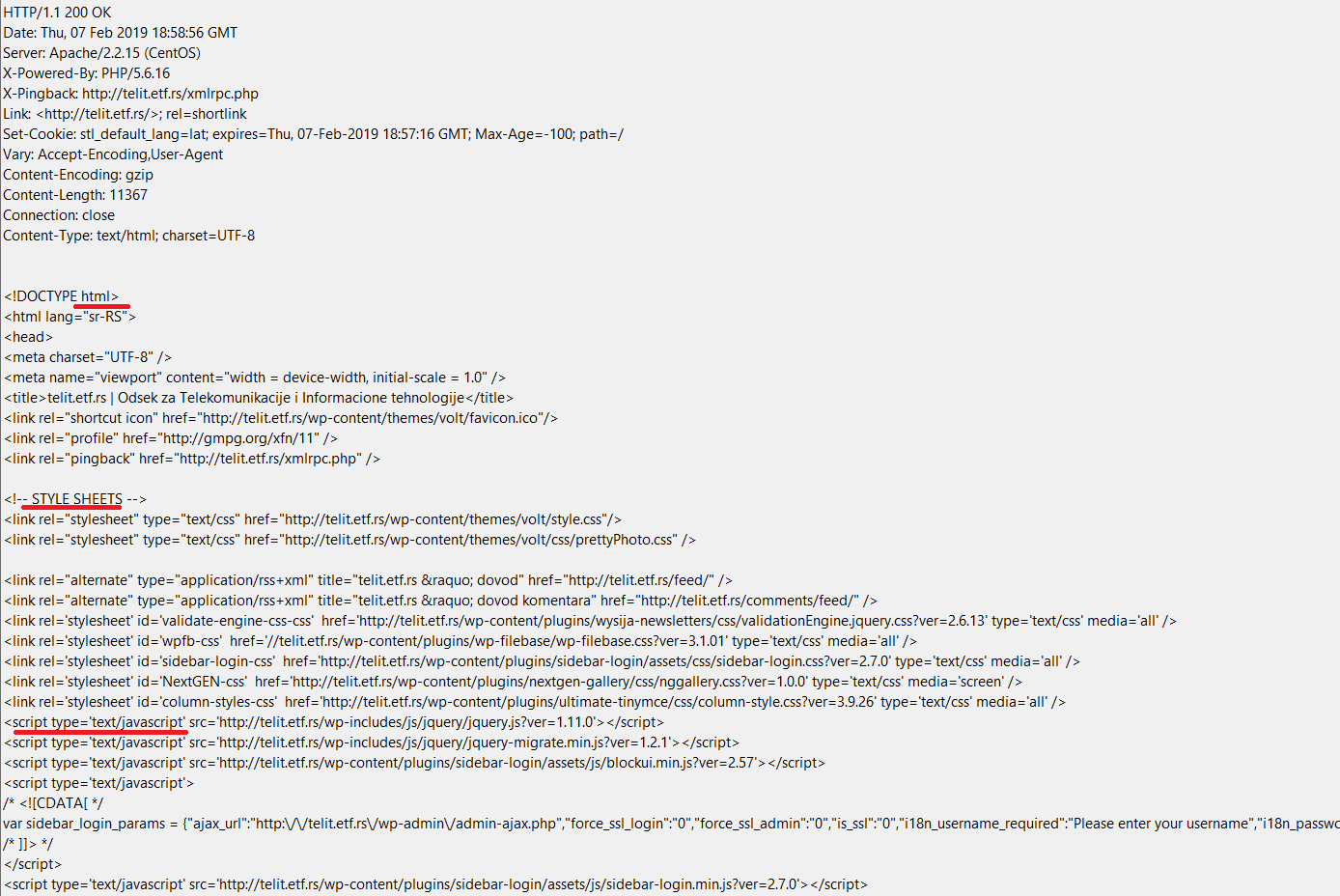


Slika 5.3 – Primer HTTP zahteva

Kao što možemo videti prva linija govori da se radi o GET zahtevu a da je web adresa sa koje želimo da preuzmemo zahtev adresa web stranice odseka za telekomunikacije. U nastavku se precizira da je u pitanju HTTP 1.1 standard.

Sledeća linija govori o tome da se ova konekcija ne održava otvorenom između klijenta i servera (Klijent je često ime za podnosioca web zahteva i koristićemo ga često), već je potrebno nakon isporuke web stranice konekciju zatvoriti.

Kao odgovor dobija se sledeći format poruke:



Slika 5.4 – Primer HTTP Odgovora

Kao što se na slici 5.4 može videti, u okviru odgovora web servera, u prvoj liniji je specificiran status kod odnosno broj koji označava uspešnost odgovora web servera. Ukoliko dođe do greške prilikom komunikacije, web server će vratiti odgovor sa nekim od status kodova, među kojima su najpoznatiji status kodovi dati u sledećoj tabeli.

|  |  |  |
| --- | --- | --- |
| Status kod | Skraćena poruka | Značenje |
| 200 | OK | Komunikacija je u redu |
| 404 | Not Found | Resurs nije pronađen na serveru |
| 500 | Internal Server Error | Interna greška – recimo server je u datom trenutku nedostupan |
| 401 | Unauthorized | Korisnik koji je isporučio zahtev nije autorizovan za određeni tip servisa i traženi odgovor mu ne može biti isporučen |

Tabela 5.1 – Neki od najpoznatijih status kodova

Takođe, na primeru slike 5.4 možemo uočiti da su crvenom bojom označene sekcije u telu odgovora i to:

* početak HTML poruke. HTML poruka ima jasno određenu strukturu i počinje sa oznakom koja govori pretraživaču da je u pitanju tip dokument HTML koji mu je prosleđen
* Mesto u okviru HTML dokumenta gde se precizira putanja ka stilizator fajlu (CSS fajl) koji u sebi nosi opis kako zahtevana web stranica (u ovom slučaju web stranica odseka za telekomunikacione i informacione tehnologije) treba da izgleda odnosno kako da se formatira
* Mesto u kojem se učitavaju *front-end* skripte (JavaScript) koje pružaju korisniku mogućnost interakcije sa web stranicom (unos korisničkog imena i šifre kao i logovanje, pregled dokumenata itd).

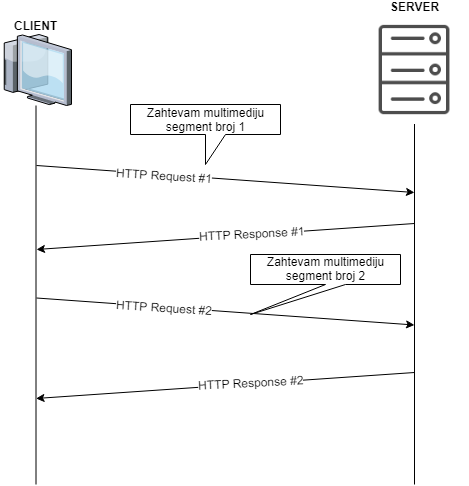
Na ovaj način uspešno je izvršena web komunikacija između web pregledača (*browser-*a) i web servera. U daljem tekstu ćemo ova dva pojma nazivati **klijent** i **server** respektivno, jer se u modernoj nomenklaturi na ove pojmove najčešće referira na ovaj način.

* 1. Web komunikacija pomoću standardnih AJAX zahteva

AJAX (*Asynchronous Javascript And XML*) predstavlja prvi sledeći iskorak ka bržim i responsivnijim web stranicama. Naime, pošto se u ovom slučaju zahteva interaktivnost postiže na asinhroni način odnosno skripta/skripte na web stranici se izvršavaju čak i pre nego odgovor sa servera stigne, ovaj metod je zapravo otvorio pitanje prenosa multimedijalnih sadržaja preko web-a. Princip je imao svoja ograničenja i nije zaživeo zbog kompleksnosti samih zahteva i njihovih limitacija.

Ovaj princip je jako brzo zamenjen novom doktrinom da multimedijalne sadržaje krajnjim korisnicima treba dostavljati korišćenjem UDP načina komunikacije, koji ne zahteva potvrdu primljenog signala i njegovu proveru već se signal šalje bez ikakve provere. To je uzrokovalo uvođenje novih mehanizama za komuniciranje.

Ovaj mehanizam nije se mogao koristiti za potrebe RTC komunikacije, jer se od korisnika zahteva da istovremeno šalje i prima sadržaje što iziskuje slanje GET i POST zahteva simultano, što dodatno komplikuje komunikaciju.



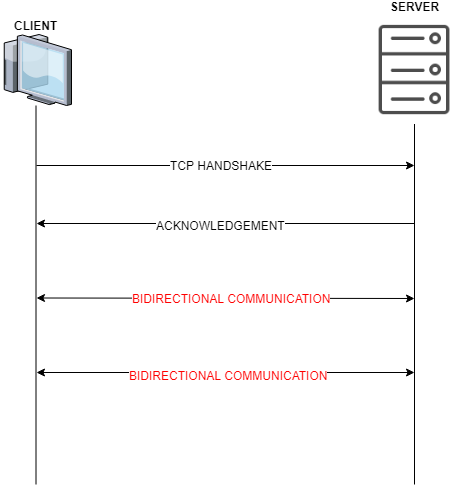
Slika 5.5 – HTTP Komunikacija klijent server i zahtev za multimedijalnim sadržajem

* 1. Pojam Web Socketa u web komunikacijama

Razmatranjem kako bi se mogao rešiti problem asinhrone komunikacije uvodi se novi pojam Web Socketa i novi način komunikacije. On je sam po sebi novi protokol koji je odmah stekao veliku popularnost zbog svoje jednostavnosti u domenu implementacije. Ono što karakteriše ovaj koncept komunikacije jeste to da inicijalna komunicija započinje rukovanjem (*Handshake*) a zatim se između klijenta i servera ostvaruje asinhrona bidirekcionalna komunikacija.

Bitno je napomenuti da Web Socket-i koriste *event driven* mehanizam komuniciranja, odnosno i klijent i server ostaju u nepromenjenom (idle) stanju sve dok ih neki događaj odnosno event ne natera da započnu komunikaciju ili izvrše neku operaciju. Na ovom principu razvijene su mnoge chat applikacije i chat room-ovi.

Jednom uspostavljena konekcija između klijenta i servera se održava otvorenom i prilikom izvršavanja nekog događaja odnosno reagovanja na isti, komunikacija je znatno brža u odnosu na standardne HTTP sinhrone zahteve.



Slika 5.6 – Princip komunikacije pomoću Web Socket-a

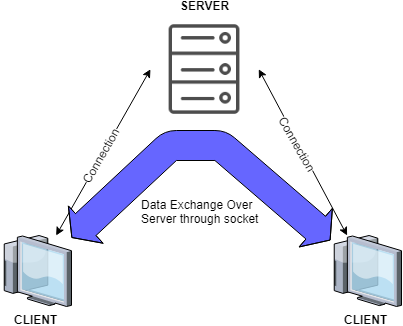
Prilikom implementacije demonstracionog rešenja jednim većim delom korišćeni su i Web Socket-i tako da je funkcionalnost Web Socket-a prikazana i na praktičnom primeru.

Kako bi se ostvario prenos slike, zvuka ili video stream-a preko web-a, ono što je potrebno jeste da web pretraživač ume efikasno da manipuliše sa resursima koji su mu na raspolaganju. Drugim rečima, web pretraživač bi trebalo da ume da efikasno pristupi perifernom uređaju (kameri, mikrofonu) i da sa njega pokupi podatke koje će proslediti drugom uređaju. Prvobitno, ovaj mehanizam akvizicije podataka rešavao se instalacijom eksternih (*3rd Party*) dodataka koji su pružali mogućnost browser-u da uspešno prikuplja podatke sa perifernih jedinica. Mana ovakvog pristupa bila je to da je svaki komunikacioni servis zahtevao instalaciju svog dodatka zarad komunikacije. Sledeći korak u razvoju Real Time Web servisa je webRTC!

* 1. WebRTC komunikacija

Ukoliko posmatramo prethodni scenario u kome se za komunikaciju između dva klijentska uređaja i servera koriste Web Socketi za komunikaciju, ono što se može zaključiti da se nakon rukovanja dva uređaja sa serverom komunikacija između njih može uspostaviti mehanizmom eventova.

Ono što je problematično jeste činjenica da se ovakva komunikacija odvija **kroz server** i da je sa sigurnosnog aspekta veoma veliki rizik komunikacija kroz ovakve kanale. Neretko se dešava da neovlašćenim upadima u farme servera hakeri pokupe veliku količinu poverljivih podataka koji se prenose preko tih servera. Ono što je bitno reći jeste da je sa WebRTC-om kao i sa Socketima prvo razvijan koncept prenosa podataka (*data traffic*) a tek kasnije su razvijane podrške za govorne i video sisteme. Zbog problema instaliranja dodataka i sigurnosnih rizika, koncept Socket-a je polako napušten za RT komunikaciju, iako se danas Socketi veoma mnogo koriste u druge svrhe, RT komunikacja se po novom standardu vrši pomoću webRTC principa. Glavna ideja oko koje je nastala ova tehnologija jeste da bi iz procesa komunikacije tačnije iz procesa razmene podataka trebalo izbaciti posrednika (socket server) i da bi se komunikacija trebala izvršavati direktno između dva web pretraživača.

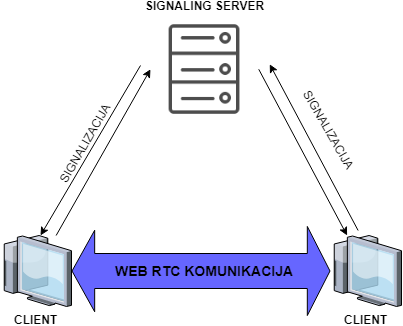


Slika 5.6 – Princip komunikacije dva klijenta pomoću Web Socket-a

Kada pričamo o komunikaciji u realnom vremenu između dva web pretraživača koji komuniciraju direktno to može delovati pomalo zbunjujuće, ali pod tim pojmom se zapravo misli na komunikaciju u sa što manjim RTT-om koja se vrši na poseban način.

Kako bi se napustio koncept *3rd* *party* plugin-ova koji su potrebni da se instaliraju kako bi pretraživač mogao pristupati perifernim jedinicama, sa novom verzijom HTML standarda (HTML 5) kao i sa novim ECMA6 JavaScript standardom uveden je set API-ja(Aplikativni Programski Interfejsi – možemo reći i set funkcionalnosti) koji omogućava pregledačima da direktno pristupaju multimedijalnim periferijama pomoću globalnog objekta koji se zove Media Devices odnosno globalne funkcije.

Na ovaj način umnogome je olakšana akvizicija podataka i kreiranje audio/video strimova direktno u pregledaču web stranica bez potrebe za dodatnim instalacijama. Ova funkcionalnost svakako je jedna od mnogih koja je zaslužna za popularnost webRTC standarda.



Slika 5.7 – Princip Web RTC komunikacije

Za ovaj vid komunikacije potrebno je uspostaviti signalizacioni server, koji je zadužen da prenese komunikacione tokene između dva krajnja korisnika, odnosno da prenese parametre buduće komunikacije. Kada se izvrši zamena parametara, i kada korisnici znaju na koji će način i preko kog kanala da komuniciraju, komunikacija se ostvaruje a signalizacioni server ni na koji način ne učestvuje u toj komunikaciji. Ovime smo praktično realizovali jedan na jedan komunikaciju između dva uređaja.

A da li je to baš 1-na-1 direktna komunikacija?

Ta komunikacija jeste veza 1 na 1 u pogledu toga da između dva klijentska uređaja nemamo logičkih jedinica koje će biti zadužene za transport multimedijalnih sadržaja kroz mrežu. Sa transportne i sa networking strane gledišta, svakako da se između ova dva uređaja nalazi još veliki broj mrežnih uređaja rutera, svičeva, bridževa i firewall-ova kroz koje se multimedijalni sadržaj mora propagirati ka krajnjem korisniku. Parametri komunikacije koji se inicijalno razmenjuju su striktno vezani za mrežu odnosno za način kojim će se poruka propagirati kroz mrežu.

Jako bitno je napomenuti da zarad jednostavnosti, webRTC nema preciziran standard za signalizaciju tako da na nama ostaje odluka o opredeljenju za korišćenje signalizacionog standarda, a nekako najpogodnijim čini se mehanizam Web Socketa.

Mrežni segment komunikacije između dva klijenta (koji se često nazivaju i *Peers*) komplikuje se kada se u obzir uzmu realne lokalne mreže (Slika 5.1 ) koje su međusobno zaštićene pomoću firewall-ova i koje moraju međusobno da komuniciraju pomoću javnih IP adresa koje će Peer-ovi međusobno razmeniti.

* + 1. STUN Serveri

STUN serveri su poseban tip servera koji se koristi za potrebe Web RT Komunikacije a njihov zadatak jeste da lokalnim Peerovima, tačnije Peerovima u različitim lokalnim mrežama distribuira njihove globalne IP adrese kako bi ti uređaji mogli međusobno da se usaglase oko načina komunikacije.

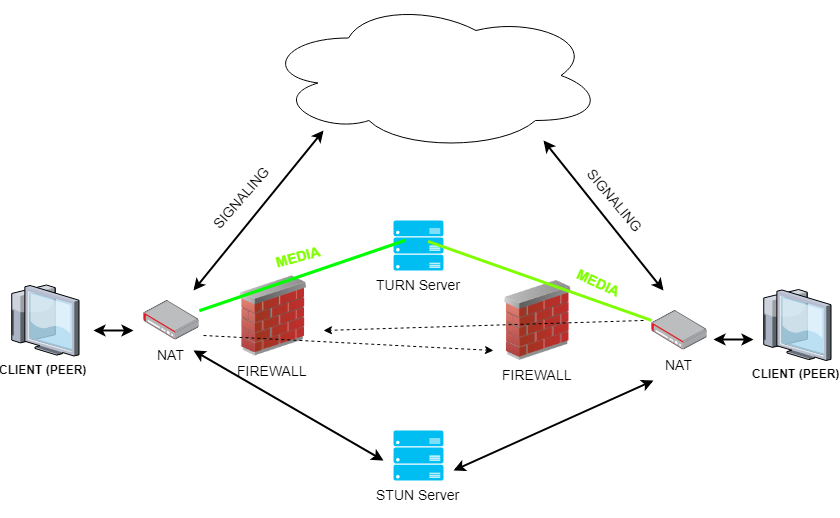
Ovo rešenje je jedino tehnički prihvatljivo rešenje koje se nametnulo iz jednog prostog razloga, a to je taj da se javna IP adresa nekog uređaja jedino može nabaviti VAN te mreže odnosno onog momenta kada NAT uređaj uradi promenu lokalne adrese u javnu adresu.

* + 1. TURN Serveri

Korišćenje TURN servera zamišljeno je kao pomoćni mehanizam, mada se pokazuje da se bez ovih servera ozbiljna web RTC komunikacija ne može koristiti. Naime ovi serveri zaduženi su za prenos enktiptovanog multimedijalnog sadržaja u slučaju da nije moguće ostvariti dogovorenu komunikaciju preko unapred dogovorenih kanala. Tada se notifikuje TURN server i od njega se zahteva da izvrši prenos multimedijalnog sadržaja (zvuka, video strima ili podataka).

Korišćenje TURN servera možemo posmatrati i kao povratak unazad jedan stepenik ka sistemima koji koriste Web Sockete, iako to baš nije najilustrativniji slučaj.

Postoji veliki broj STUN i TURN servera koji se mogu zakupiti od velikih provajdera, mada postoje i besplatni STUN i TURN serveri koji imaju limitirane kapacitete, ali su pogodni za testiranja webRTC komunikacionih software-a.



Slika 5.8 – Princip korišćenja STUN i TURN servera kod webRTC-a

1. Implementacija jednog web RTC servisa

Prilikom implementacije jednog webRTC sistema/servisa, nakon upoznavanaj sa mehanizmom rada potrebno je odabrati tehnologiju u kojoj ćemo razvijati naše rešenje. U daljem tekstu je prikazan razvoj jednog rešenja pomoću NodeJS programskog jezika.

Ceo princip razvoja rešenja može se razdvojiti na dve velike celine i to:

* Razvoj *Back-End* sistema za signalizaciju i za opsluživanje HTTP zahteva (Web Server)
* Razvoj *Front-End* komponente koja je zadužena za Peer to Peer komunikaciju nakon uspešne uspostave veze.
  1. Razvoj Back-End Sistema

Pod razvojem *Back-End*  sistema podrazumevamo implementaciju signalizacionog servera i HTTP serving server-a. Kako je ovaj deo pisan u NodeJS programskom jeziku, za signalizaciju korišćena je **Socket.io** biblioteka za rad sa Web Socket-ima, a za posluživanje HTTP web stranica korišćen je modul koji je veoma popularan i koji se zove **Express.**

U sledećem segmentu prikazan je kod koji je relativno jednostavan i zauzima svega 60 linija!



**else**{

//slucaj kada predjemo na 3+ konekcije, bitno je novonakacenom socketu reci da nije pozeljan na serveru!!!

socket.emit('connRedirect');

console.**log**(`Socket.io server: A user is redirected: ${connectionNumber}`);

connectionNumber--;

}

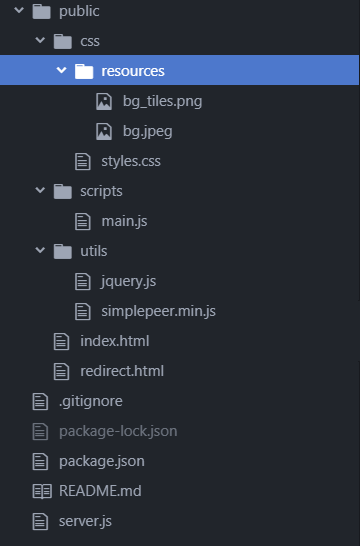
});

server.listen(port, **function**() {

console.**log**(`Server is listening at http://localhost:${port}`);

});

Kao što možemo primetiti server se pokreće na portu 4000 i prilikom HTTP Get zahteva dostavlja mu se ceo „public“ folder. Cela struktura projekta prikazana je na sledećoj slici gde se uočava public folder i njegova sadržina.



Slika 6.1 – fajlovi i folderi koji se koriste za potrebe projekta

Kao što možemo videti sadržaj projektnog foldera, fajl *server.js* koristi se za serversku stranu, a svi fajlovi koji se nalaze unutar foldera public se na HTTP zahtev dostavljaju korisniku a on zatim pokreće dostavljene fajlove u svom browser-u.

Folder **css** sadrži *Cascading Style Sheets* fajlove i resursne fajlove kao što su slike koje će biti prikazane na web stranici, kao i opis načina prikaza hiper-teksta (boja, pozicija, veličina, ...)

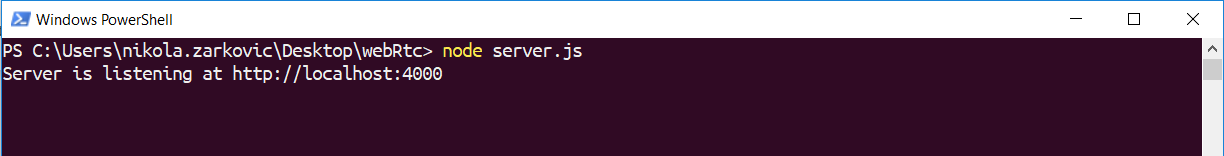
Folder **scripts** sadrži *Front End*  skriptu – **main.js**koja se dostavlja korisniku u okviru public foldera i koju web pregledač pokreće prilikom učitavanja web stranice.

Folder **utils** takođe sadrži *Front End*  JavaScript fajlove koji se učitavaju, ali to su utilizacione skripte koje nam omogućavaju webRTC komunikaciju (moduli koji su potrebni za webRTC).

Projekat je moguće naći na sledećem GitHub repozitorijumu, sa pripadajućom projektnom dokumentacijum(ovaj .docx fajl): <http://github.com/dichtung/webRTC>.

* 1. Izgled korisničkog interfejsa

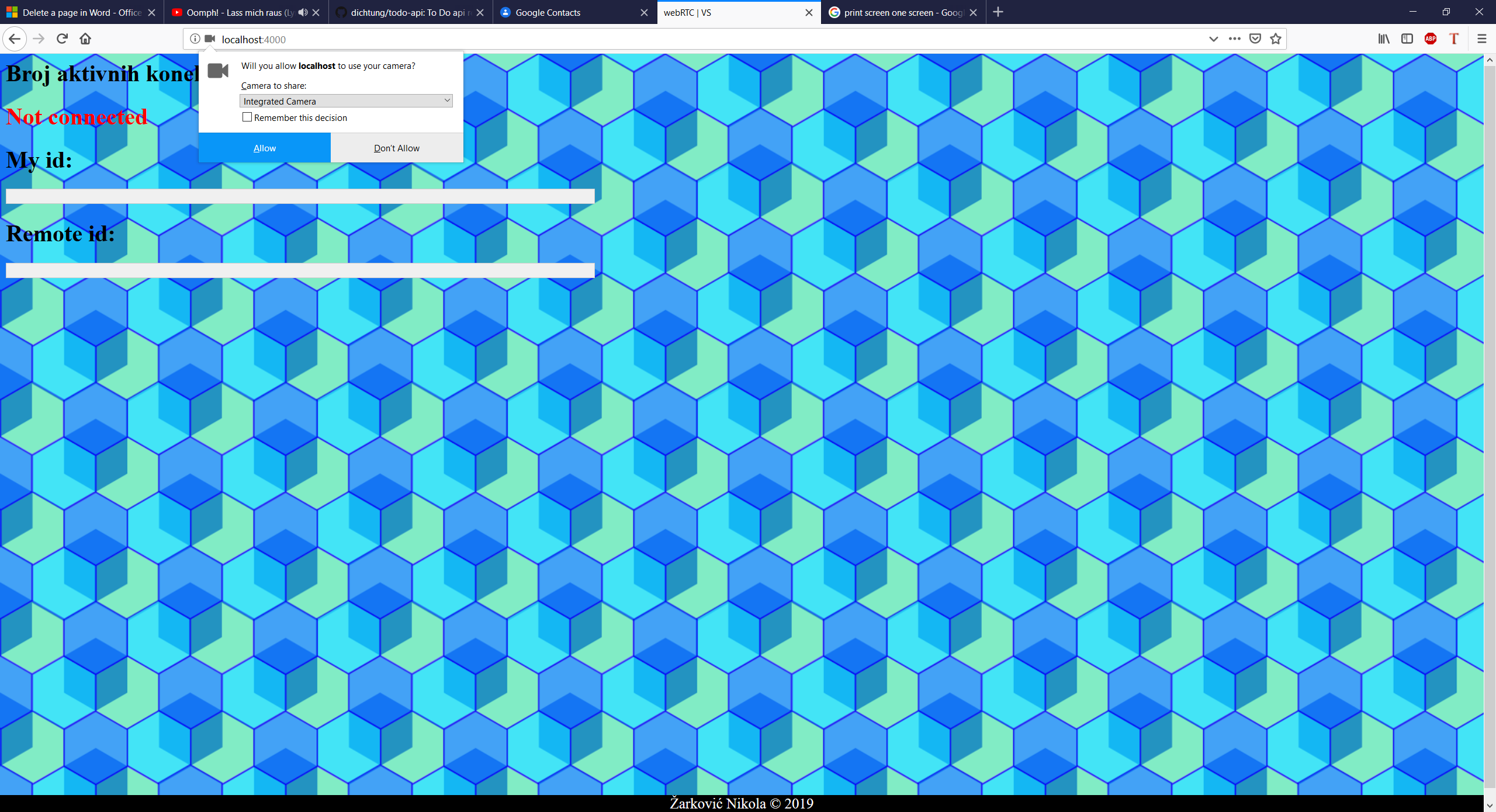
Prilikom pokretanja servera, izdavanjem komande *node server.js* dočekaće nas pozdravna konzolna poruka da je server uspešno pokrenut kao i to na kom portu je dostupan za pristup.



Slika 6.2 – Pozdravna korisnička poruka da je server aktivan na portu 4000 i da osluškuje zahteve

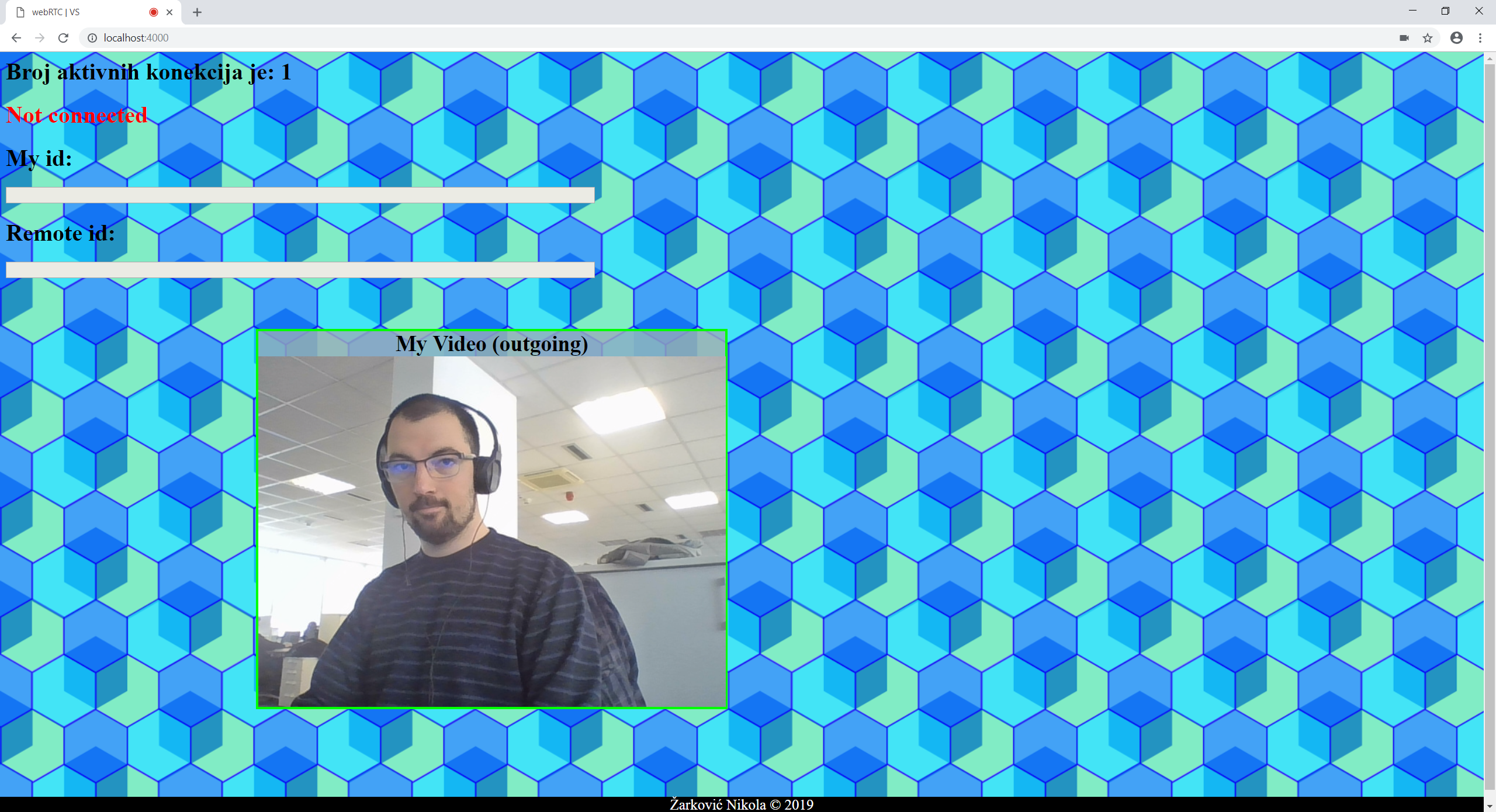
Drugim rečima konfigurisali smo web server koji osluškuje zahteve na pomenutom portu i kada na njega pošaljemo HTTP GET request na adresu ***http://<adresa uredjaja na kom je servis>:4000*** kao odgovor dobićemo web stranicu

Prva stvar koja se dešava nakon učitavanja stranice jeste ta da se korisnik pita za permisije odnosno za dozvolu korišćenja perifernih uređaja (Web kamere i/ili mikrofona) u zavisnosti od multimedijalnih servisa koje želimo da prenosimo.



Slika 6.3 – Izgled Web stranice prilikom prvog pokretanja

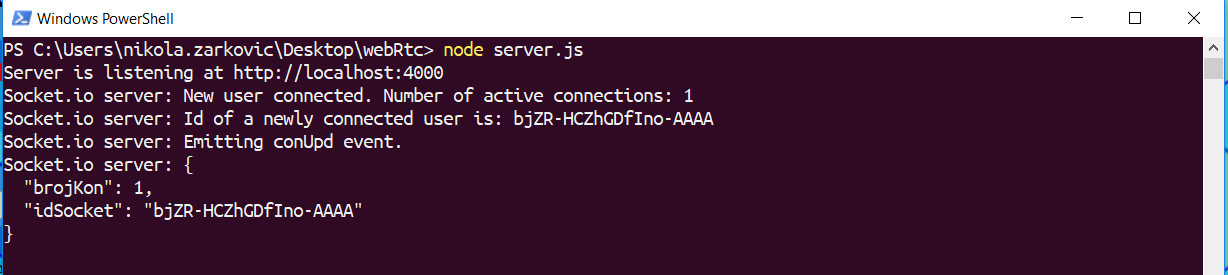
Kada se dozvoli pristup audio/video periferijama (konkretno kameri u slučaju sa slike 6.3), video strim sa korisničke slike se prikazuje korisniku u formatu kontrolnog video strima.



Slika 6.4 – Prikaz korisničkog kontrolnog strima i čekanje na drugog Peer-a

Kada sa drugog uređaja pokušamo da pristupimo web serveru, i kada dozvolimo web pregledaču da pristupa informacijama sa naših perifernih uređaja, inicira se proces signalizacije. Treba imati u vidu da je proces signalizacije i razmene parametara komunikacije implementiran pomoću Web Socket-a, kao i to da posmatrani Web server ima dvojaku ulogu (signalizacija između dva klijenta kao i prosleđivanje web sadržaja).

Ono što je potrebno uočiti jeste da je nakon svake ostvarene konekcije web server konfigurisan da prikaže poruku u okviru svog terminala.



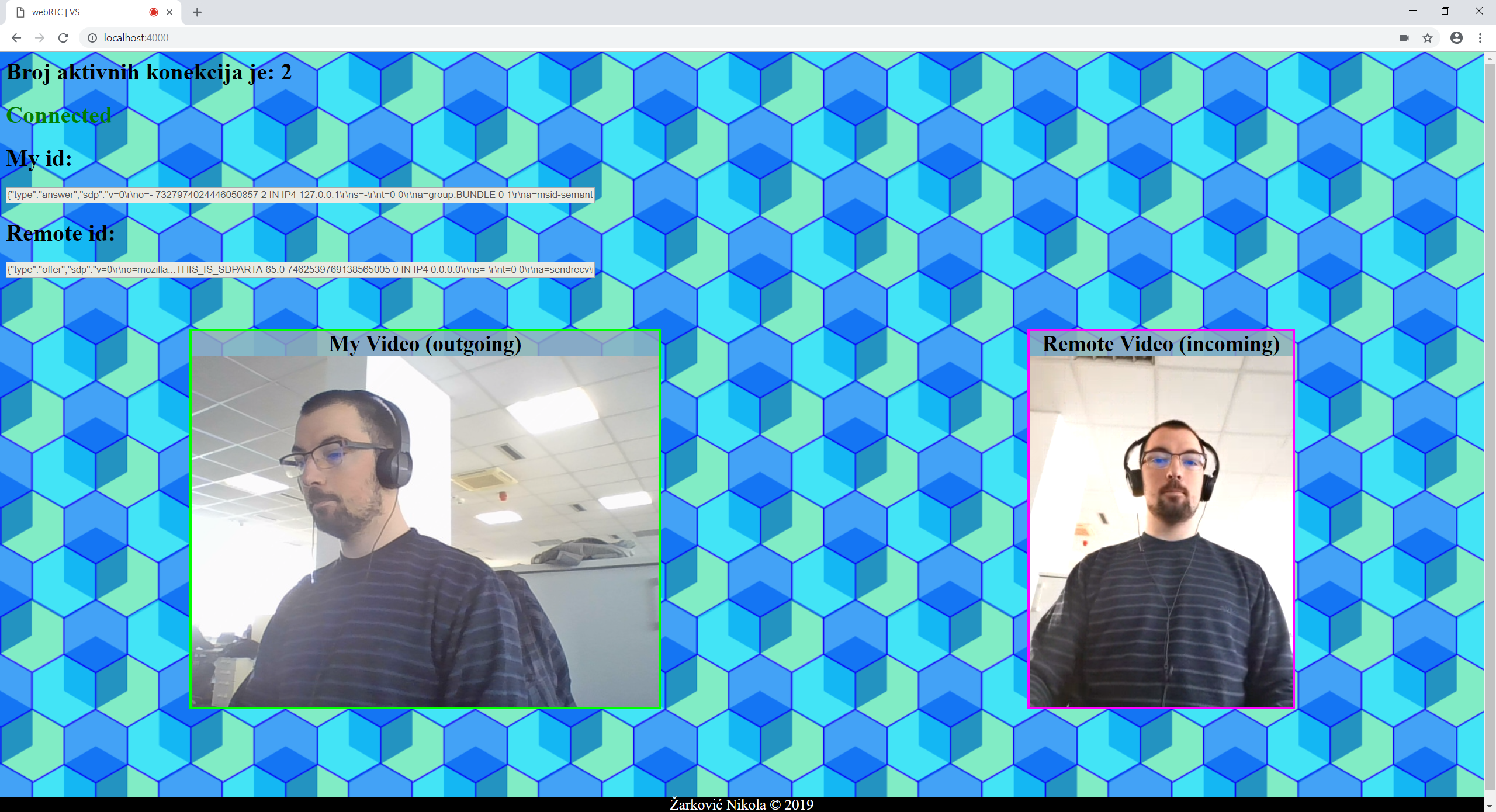
Slika 6.5 – Notifikacija na serverskoj strani da je korisnik ostvario konekciju

Kada započne proces signalizacije, u poljima My Id i Remote Id generišu se tokeni, odnosno JSON poruke koje u sebi sadrže informaciju o načinu komunikacije i mehanizmu rutiranja poruka. Kada se signalizacija uspešno završi, započinje prenos media stream-a pomoću webRTC protokola nezavisno od signalizacionog (Socket) servera.



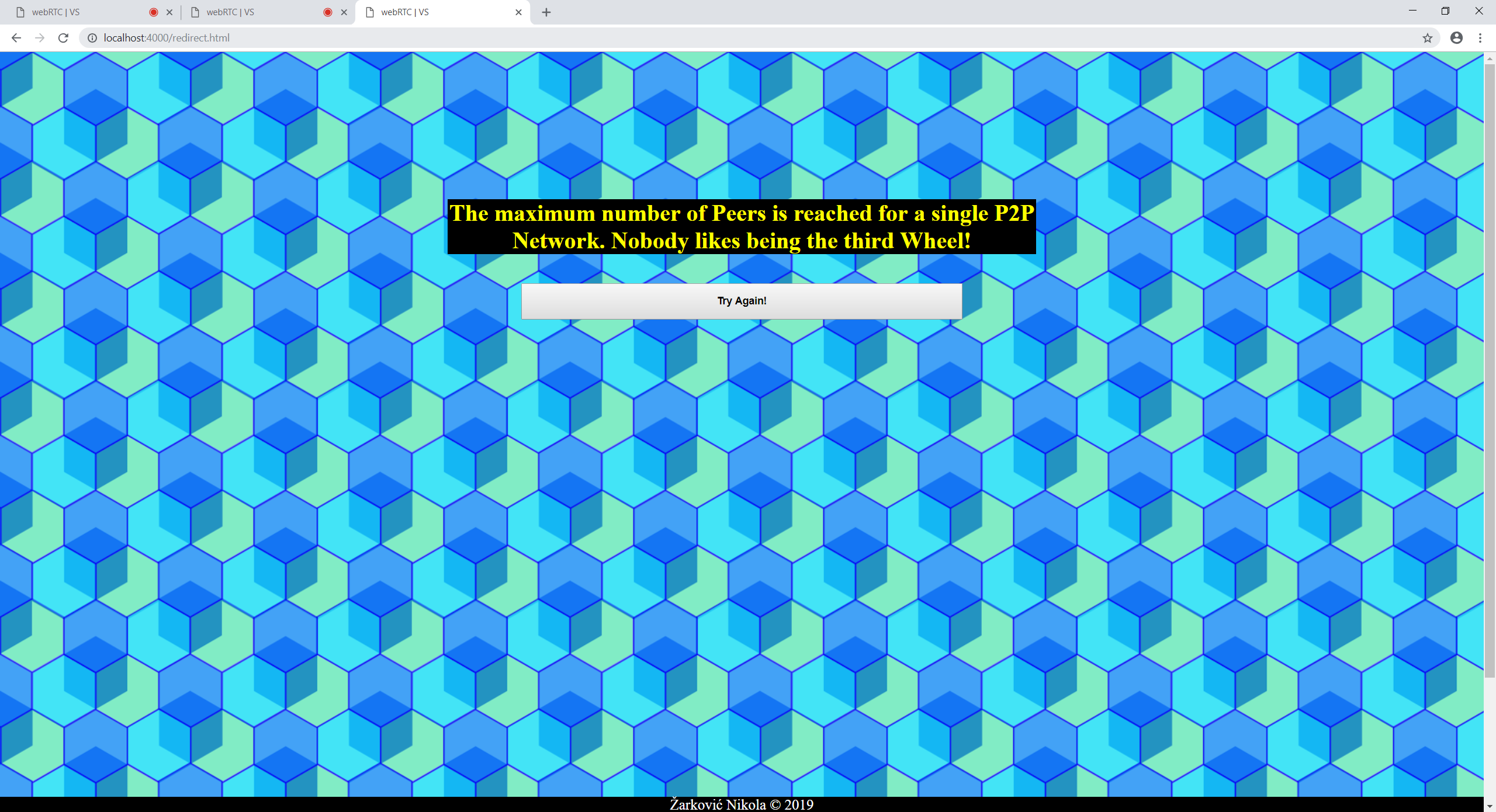
Slika 6.6 – Stanje nakon konekcije drugog korisnika na server

Napokon, media strimovi bivaju razmenjeni iono što se dešava jeste protok informacija u oba smera komunikacije čime ostvarujemo komunikaciju u realnom vremenu.



Slika 6.7 – Ostvarena komunikacija između dva uređaja (PC levo i Pametni telefon desno)

Ukoliko se pojavi i treći klijent koji želi da uđe na komunikacioni kanal, signaling server o tome vodi računa i ovaj vid komunikacije će zabraniti odnosno izvršiće preusmeravanje na stranicu *redirect.html* koja korisniku pokazuje sledeći



Slika 6.8 – Izgled strane za redirekciju

Kako bismo dokazali da se komunikacija izvršava nezavisno od signaling servera, njega možemo isključiti izdavanjem komande *Ctrl + C* u komandnoj liniji, čime će se njegov rad onesposobiti. Ono što primećujemo jeste da se media strimovi oba Peer-a tj. klijenta i dalje strimuju na web stranicama

Što se kvaliteta video signala tiče, HTML 5 standardom odnosno native API-jem koji pruža JavaScript dozvoljena je mogućnost modifikovanja prostorne rezolucije odnosno zadavanja minimalne i maksimalne željene prostorne rezolucije kao i modifikacija odnosa dimenzija (*aspect ratio*) ali realno kvalitet prenošenog strima zavisi od 2 faktora:

* Fizičke osobine kamere – ovaj faktor je veoma limitirajući, posebno uzimajući u vidu da su proizvođači računarske opreme uglavnom stavljali kamere koje imaju mogućnost akvizicije video signala sa rezolucijom od 0.3Mpix. Sa ovakvim aparatom nije moguće na pogodan način slati signal HD rezolucije, ali je kvalitet slike za potrebe video conferencing-a veoma zadovoljavajući
* Kvalitet veze između Peerova – Kvalitet veze je takođe limitirajući faktor, i kao takav drastično utiče na kvalitet video signala.

Kako se za prenos video signala koristi MPEG stream kvalitet strima se drastično menja u zavisnosti od mrežnih parametara. To se konkretno u ovom slučaju ogleda u promeni prostorne rezolucije pošto ona nije ograničena na web stranici.

1. Zaključak

Na razvoju web RTC sistema se uveliko radi, kao i na njihovoj standardizaciji te ovaj način komunikacije pretenduje da postane novi standard u web komunikaciji posredstvom interneta. Ova promena sa obom svakako donosi i probleme veoma nestabilnog kvaliteta signala i nepostojanja kontrole reprodukovanog sadržaja a koji je direktno vezan sa binarnim protokom prenošenog signala na fizičkom nivou, te taj problem treba premostiti i naći najoptimalnije rešenje.

Kada sagledamo celu sliku uviđamo da mesta za napredak i poboljšanje ovih sistema i te kako ima, a jedan od mnogih jeste donošenje standarda koji se tiče signalizacije i razmene parametara konekcije. Kad bi se sa jedne strane sagledao problem, ja kao i većina autora ozbiljne literature na temu WebRTC-a delimo mišljenje da bi najisplativije i najefektnije bilo združiti funkcionalnosti TURN servera i u njega ugraditi signalizacioni mehanizam čime bi se znatno pojednostavila arhitektura sistema.

Nadamo se da će kroz par godina implementacija WebRTC-a dostići još veću popularnost te da ćemo servise za prenos multimedijalnih sadržaja na koje smo navikli i koje zahtevaju instalaciju posebnih aplikacija i plugin-ova moći da koristimo kroz još prostije i jednostavnije interfejse u okviru naših web pretraživača.

Takođe, spekuliše se i o mogućnosti korišćenja webRTC arhitekture i u okviru hibridizacije sa CDN mrežama u okviru OTT servisa za brzu komunikaciju sa drugim korisnicima, tako da bi bilo zanimljivo videti pomake i na tom polju tehnologija koje se veoma brzo razvijaju.

LITERATURA

1. **Vogt, Christian, Max Jonas Werner, and Thomas C. Schmidt. "Leveraging WebRTC for P2P content distribution in web browsers." Network Protocols (ICNP), 2013 21st IEEE International Conference on. IEEE, 2013.** S.Loreto, S.P. Romano “Real Time Communication with WebRTC”, May 2014
2. Loreto, Salvatore, and Simon Pietro Romano. Real-Time Communication with WebRTC: Peer-to-Peer in the Browser. " O'Reilly Media, Inc.", 2014.
3. Nurminen, Jukka K., et al. "P2P media streaming with HTML5 and WebRTC." Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on. IEEE, 2013
4. Johnston, Alan B., and Daniel C. Burnett. WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web. Digital Codex LLC, 2012.
5. Loeser, Jork, and Hermann Haertig. "Low-latency hard real-time communication over switched Ethernet." Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.. IEEE, 2004.
6. Loeser, Jork, and Hermann Haertig. "Low-latency hard real-time communication over switched Ethernet." Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.. IEEE, 2004.