POLITECHNIKA KRAKOWSKA

IM. TADEUSZA KOŚCIUSZKI

WYDZIAŁ FIZYKI MATEMATYKI I INFORMATYKI

KIERUNEK INFORMATYKA

BARTŁOMIEJ PACURA

**PORÓWNANIE ISTNIEJĄCYCH ROZWIĄZAŃ BAZUJĄCYCH NA TECHNOLOGII WEBRTC DO STWORZENIA CZATU TEKSTOWEGO/AUDIO/WIDEO.**

**THE COMPARISON OF WEBRTC TECHNOLOGIES FOR CREATING TEXT/AUDIO/VIDEO CHAT**

PRACA MAGISTERSKA

STUDIA NIESTACJONARNE

Promotor: Dr Agnieszka Krok

Kraków 2014

Spis treści

[**1. Wprowadzenie do Web Real-Time Communication** 5](#_Toc394440939)

[1.1 Komunikacja 5](#_Toc394440940)

[1.1.1 Definicja komunikacji 5](#_Toc394440941)

[1.1.2 Współczesne sposoby komunikacji 6](#_Toc394440942)

[1.1.3 Historia rozwoju sposobów komunikacji 7](#_Toc394440943)

[1.2 Wpływ aplikacji webowych na dzisiejsze życie 8](#_Toc394440944)

[1.3 Krótkie omówienie technologii WebRTC 9](#_Toc394440945)

[1.3.1 Pojęcie Real-Tome Communication 10](#_Toc394440946)

[1.3.2 Web Real-Time Communication 10](#_Toc394440947)

[1.4 Potrzeby, które zrodziły ten standard 11](#_Toc394440948)

[1.5 Cel pracy magisterskiej 13](#_Toc394440949)

[1.6 Historia standardu 13](#_Toc394440950)

[**2. Obecne możliwości i ograniczenia** 15](#_Toc394440951)

[2.1 Możliwości i założenia 15](#_Toc394440952)

[2.1.1 Słowo wstępu 15](#_Toc394440953)

[2.1.2 Bezpośrednia komunikacja przeglądarek i zmniejszenie roli serwera 15](#_Toc394440954)

[2.1.3 Wspólne i proste API 17](#_Toc394440955)

[2.1.4 Przechwytywanie strumienia audio/wideo 17](#_Toc394440956)

[2.1.5 Zarządzanie strumieniem danych 17](#_Toc394440957)

[2.1.6 Prowadzenie rozmów w trybie rzeczywistym 18](#_Toc394440958)

[2.1.7 Przeźroczystość oprogramowania i sprzętu 18](#_Toc394440959)

[2.1.8 Wieloosobowe konferencje 18](#_Toc394440960)

[2.1.9 Współdzielenie plików 18](#_Toc394440961)

[2.1.10 Zarządzanie jakością strumienia 18](#_Toc394440962)

[2.1.11 Negocjowanie mediów 19](#_Toc394440963)

[2.1.12 Wykorzystanie komunikacji w innych obszarach 19](#_Toc394440964)

[2.1.13 Ogromna ilość końcowych użytkowników 19](#_Toc394440965)

[2.2 Przykładowe aplikacje wykorzystujące technologię WebRTC 19](#_Toc394440966)

[2.2.1 Wideokonferencje 19](#_Toc394440967)

[2.2.2 Modyfikacja strumienia 20](#_Toc394440968)

[2.2.3 Użycie komunikacji rzeczywistej w grach 21](#_Toc394440969)

[2.2.4 Udostępnianie plików 22](#_Toc394440970)

[2.3 Obecne ograniczenia 23](#_Toc394440971)

[3. Omówienie składowych technologii 25](#_Toc394440972)

[3.1 Przechwytywanie strumieni poprzez MediaStream 25](#_Toc394440973)

[3.2 Komunikacja przesyłania danych na poziomie przeglądarki 26](#_Toc394440974)

[3.3 Sygnały i obsługa sesji 28](#_Toc394440975)

[3.5 Topologia sieci i obsługa połączeń przez Network Address Translation 29](#_Toc394440976)

[4. Bezpieczeństwo i prywatność 34](#_Toc394440977)

[4.1 Obecne zabezpieczenia wykorzystywane w WebRTC 34](#_Toc394440978)

[4.1.1 Przeglądarka internetowa 34](#_Toc394440979)

[4.1.2 Zezwolenie na dostęp do mikrofonu/kamerki 34](#_Toc394440980)

[4.1.3 Brak zewnętrznych wtyczek 35](#_Toc394440981)

[4.1.4 Szyfrowane połączenia 35](#_Toc394440982)

[4.2 Możliwe słabe punkty 36](#_Toc394440983)

[4.2.1 Wirusy i szkodliwe oprogramowanie 36](#_Toc394440984)

[4.2.2 Kradzież strumienia audio/wideo 36](#_Toc394440985)

[4.2.3 Atak DDoS na przeglądarkę 37](#_Toc394440986)

[4.2.4 Brak szyfrowania strumienia lub sygnałów 37](#_Toc394440987)

[4.3 Prywatność 37](#_Toc394440988)

[4.3.1 Kradzież strumień audio/wideo 37](#_Toc394440989)

[4.3.2 Znajomość adresu IP 38](#_Toc394440990)

[5. Porównanie istniejących technologii 39](#_Toc394440991)

[5.1 Projekt aplikacji 39](#_Toc394440992)

[5.1.1 Wymagania funkcjonalne 39](#_Toc394440993)

[5.1.2 Wymagania przy korzystaniu z WebRTC 40](#_Toc394440994)

[5.2 Kryteria porównawcze 40](#_Toc394440995)

[5.3 Zestawienie narzędzi 41](#_Toc394440996)

[5.3.1 SimpleWebRTC 41](#_Toc394440997)

[5.3.2 WebRTC jQuery plugin 43](#_Toc394440998)

[5.3.3 Peer.js 44](#_Toc394440999)

[5.3.4 EasyRTC 46](#_Toc394441000)

[5.3.5 WebRTC.io 48](#_Toc394441001)

[5.3.6 OpenTok 49](#_Toc394441002)

[5.3.7 WebRTCCopy 52](#_Toc394441003)

[5.4 Porównanie przedstawionych rozwiązań 53](#_Toc394441004)

[5.4.1 Zestawienie możliwości narzędzi 54](#_Toc394441005)

[5.4.2 Porównanie według kryteriów 54](#_Toc394441006)

[5.5 Podsumowanie 56](#_Toc394441007)

[6. Zakończenie 57](#_Toc394441008)

[Bibliografia 58](#_Toc394441009)

[Źródła rysunków 61](#_Toc394441010)

# **1. Wprowadzenie do** Web **Real-Time Communication**

Komunikacja jest jednym z najważniejszych elementów życia społecznego człowieka. Rozwój techniki ułatwił i przyspieszył możliwość przesyłania informacji pomiędzy ludźmi. Idea porozumiewania się została jednak bez zmian od dawien dawna. Technologia Web Real-Time-Communication jest jednym z rozwiązań, które ma być medium do komunikacji.

## 1.1 Komunikacja

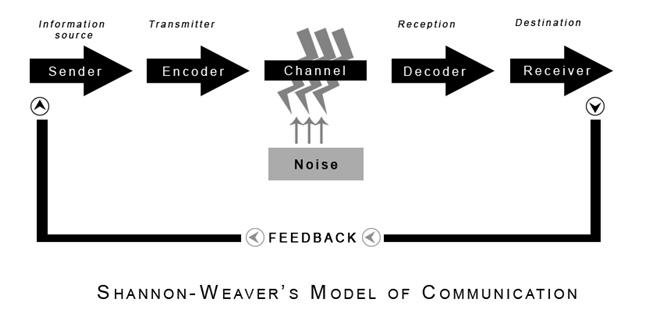
Cele postawione przed technologią WebRTC są ściśle związane z aspektem ludzkiego życia, jakim jest porozumiewanie się. By zrozumieć to zagadnienie, trzeba spojrzeć na nie od strony psychologicznej.

### 1.1.1 Definicja komunikacji

Słowo komunikacja pochodzi od łacińskiego czasownika ***communico, communicare*** (uczynić wspólnym, połączyć) i rzeczownika ***communio*** (wspólność, poczucie łączności) [1]. Celem komunikacji jest przekazanie drugiej osobie pewnej wiadomości (informacji) poprzez pewien kanał. Jest ona używana nie tylko przez ludzi, ale również przez zwierzęta (np. do ostrzeżenia o zbliżającym się niebezpieczeństwie) [1].

Komunikacja towarzyszy ludziom od początków ich istnienia i jest ona jednym z najważniejszych elementów ludzkiej egzystencji. Pozwala ona na przekazywanie nam myśli, wiedzy, emocji. Ludzie są stworzeniu do życia w grupie, co wymaga rozmowy, przekazania informacji. Brak lub ograniczenie komunikacji z innymi osobami wpływa negatywnie na naszą psychikę. Może to doprowadzić do samotności takiej osoby lub nawet depresję i zaburzenia psychologiczne [1].

Ogólny sposób komunikacji bardzo dobrze przedstawia model zaproponowany w 1948 roku przez amerykańskich naukowców Shannona i Weavera w artykule "A Mathematical Theory of Communication", który ukazał się w "Bell System Technical Journal":



Rysunek 1 Diagram przedstawiający sposób komunikacji według modelu Shannon’a-Weaver’sa [2].

Nadawcą jest osoba, która chcę przekazać wiadomość. Jest ona kodowana i przez kanał (medium) jest dostarczana do odbiorcy. Zanim ona do niego trafi jest dekodowana. Po zakończeniu przesyłania wiadomości jest odsyłana odpowiedź od odbiorcy do nadawcy. Dodatkowo pojawia się szum, którego celem jest utrudnienie przesłania wiadomości lub zniekształcenie jej [1].

Patrząc na ten diagram od strony komunikacji internetowej - wysyłania wiadomości e-mail - to kodowaniem i dekodowaniem jest proces zapisu wiadomości w odpowiednim standardzie, medium to połączenie TCP, a szumem może byś zakłócenie połączenia internetowego.

### 1.1.2 Współczesne sposoby komunikacji

Dzisiejszy rozwój technologii pozwala na bardzie wiele różnych sposób komunikacji między ludźmi i przekazywania sobie informacji. Poza takimi klasycznymi takimi jak rozmowa, pisanie listów, książek współczesny Internet i technologia pozwala m.in. na:

* rozmowy telefoniczne, VoIP
* popularne wiadomości SMS
* wiadomości e-mail
* fora, listy dyskusyjne
* rozmowy poprzez komunikatory tekstowe, np. polskie Gadu-Gadu, amerykański IRC
* rozmowy przez komunikatory wideo, np. popularny Skype
* serwisy społecznościowe, np. Facebook
* popularne kiedyś chaty, np. czateria.interia.pl

Wszystkie wymienione wyżej sposoby komunikacji są wynikiem wielu lat rozwoju technologii informatycznej. Ich celem jest jak najszybsze dostarczenie nam informacji niezależnie od miejsca, w którym przebywa użytkownik. Dzisiejszy świat pozwala na stały dostęp do Internetu i sieci komórkowej, co zmieniło sposoby porozumiewania się i zniwelowało takie pojęcia jak granice lub daleka fizyczna odległość.

### 1.1.3 Historia rozwoju sposobów komunikacji

Sposoby komunikacji zmieniały się na przestrzeni wieków. Każde nowe odkrycie rewolucjonizowało przesyłanie wiadomości i kontaktowanie się ludzi. Poniżej przedstawiono listę najważniejszych narzędzi i metod do porozumiewania się. Ma ona pokazać, że technologia przekazywania informacji cały czas się rozwija i będzie ciągle udoskonalana. Dodatkowo warto się zastanowić, jak ludzie będą się kontaktować ze sobą za kilkadziesiąt lat - przesyłanie myśli, hologramy?

Prehistoria to głównie etap kształtowania się mowy i języka. Jedynym sposobem komunikacji była rozmowa.

W starożytności najważniejsze jest stworzenie podstaw pisma. Wiadomości takie jednak były często przekazywane przez gońców, co na dużych odległościach było bardzo czasochłonne. W późniejszych czasach głównie był rozwijany druk i sposoby przesyłania wiadomości np. stworzenie poczt.

W 1794 roku Claude Chappe stworzył telegraf, co pozwoliło przesyłać krótkie wiadomości na bardzo długie odległości w dużo szybszym czasie niż przykładowo poczta.

W 1876 roku Elisha Gray i Alexander Bell zaprezentowali światu pierwszy telefon. Pozwolił on po raz pierwszy porozmawiać z drugą osobą. Można śmiało uznać, że jest to pierwszy wynalazek, który pozwala na komunikację w trybie rzeczywistym.

Kolejne ważne odkrycia pojawiły się dopiero w połowie XX-go wieku. Wideo konferencje pojawiły się już w latach 30-stych w Niemczech. Lata 50-te to pierwszy telefon komórkowy zaprezentowany przez firmę Motorola w 1956. Pod koniec lat 60-tych powstał zalążek Internetu - sieć ARPANET (1969) i poczta elektroniczna (1965). Była ona internetową alternatywą do klasycznych listów wysyłanych pocztą. W 1974 na uniwersytecie w Illinois zaczął działać pierwszy komunikator (chat online) PLATO. W 1992 została wysłana pierwsza wiadomość SMS (Short Message Service). W latach 90’tych zaczęły pojawiać się pierwsze serwisy społecznościowe np. theglobe.com (1995). Po 2000 roku popularne stały się darmowe rozmowy wideo w sieciach Skype i iChat.

W dzisiejszych czasach komunikacja wśród ludzi odbywa się głównie poprzez rozmowy telefoniczne, wiadomości SMS, e-mail, komunikatory w serwisach społecznościowych (np. Facebook, Google+).

## 1.2 Wpływ aplikacji webowych na dzisiejsze życie

Dzisiejsze przeglądarki nie służą tylko do wyświetlania stron internetowych, ale dzięki swojemu zaawansowaniu, są gotowymi środowiskami do uruchamiania rozbudowanych aplikacji. Wykorzystując takie technologie jak HTML5 (język znaczników do budowy struktury i zawartości strony www), CSS3 (odpowiada za wygląd strony), język skryptowy JavaScript (odpowiada za logikę działania strony i akcji wykonywanej na niej) można budować bardzo rozbudowane aplikacje zamknięte w oknie przeglądarki.

Główny wpływ na rozwój aplikacji webowych ma ułatwienie dostępu do Internetu, zwiększenie możliwości przeglądarek, ustandaryzowanie zasad działania technologii i wzrost szybkości działania naszych komputerów. Dodatkowym motywem jest zmiana sposobu komercjalizacji oprogramowaniu przez ich autorów. Przy aplikacjach desktopowych użytkownik płaci twórcy za licencję do danej wersji. Przy technologii webowej głównym zarobkiem są reklamy (np. udostępniane przez firmę Google). Dość często można spotkać model, w którym dana strona z podstawowymi możliwościami jest dostępna za darmo, a opłacając abonament ma się dostęp do pełnej wersji z dużo większą ilością elementów i funkcjonalności.

W ciągu kilku lat zmieniło się podejście do wykorzystania technologii webowej. W dzisiejszych czasach jest tendencja do migracji aplikacji uruchamianych na komputerze na rozwiązania bazujące na tzw. chmurze (rozwiązania po stronie serwerów i przeglądarki internetowej) - ogólnie technologii webowej. Nie zastąpi to specjalistycznych narzędzi (np. edytorów wideo), ale część rozwiązań użytkownych już jest coraz częściej wykorzystywanych z poziomu przeglądarki internetowej.

Dawniej użytkownik musiał instalować klienta poczty e-mail na swoim komputerze, a potem go skonfigurować podając adresy serwerów, porty, dane autoryzacyjne. Wymagało to pewnej wiedzy. Obecnie loguje się on tylko na stronę www i ma wszystko w jednym miejscu. Jak chce dodać załączniki do treści e-mail to przesuwa pliki z pulpitu w odpowiednie miejsce na stronie. Aplikacje webowe bardzo często są rozbudowane (np. poczta Gmail, Facebook) i nie ustępują pod względem funkcjonalnym swoim odpowiednikom w wersji desktopowej.

Korzystanie z rozwiązań internetowych jest coraz bardziej popularne i wykorzystywane. Użytkownik ma dostęp do danej usługi w każdym miejscu na świecie gdzie jest dostępna sieć Internet, na dowolnym urządzeniu (komputer, tablet, telefon komórkowy, telewizor, itd.). Media te mają dzisiaj zainstalowaną nawet prostą przeglądarkę internetową. Wystarczy zapamiętać adres www strony i dane do logowania. Kolejnym atutem tego rozwiązania jest coraz większa integracja usług. Można dzisiaj połączyć konto z serwisu Facebook z pocztą internetową Gmail poprzez kilka kliknięć i już po chwili mieć dostępne na liście kontaktów dane ludzi, którzy są znajomi w serwisie społecznościowym. Z dnia na dzień usługi internetowe będą coraz bardziej wykorzystywane, głównie przez łatwy do nich dostęp oraz prostotę ich obsługi. Bardzo łatwo też jest poprawiać błędy i wprowadzać nowe funkcjonalności w tego typu oprogramowania, ponieważ są dostępne od razu dla wszystkich użytkowników i nie wymagają przygotowywania i instalacji aktualizacji.

Niestety technologie webowe mają też swoje zagrożenia. Najważniejszym z nich jest bezpieczeństwo danych i dostępu do usług. Jeśli ktoś się włamie do konta pocztowego, to ma dostęp do wszystkich kont w serwisach internetowych. Kolejnym ważnym elementem jest to, że korzystając aplikacji webowych trzeba udostępnić prywatne dane (osobowe, zdjęcia, notatki) obcym firmom i zaufać im, że nie będą ich przeglądać ani wykorzystywać.

## 1.3 Krótkie omówienie technologii WebRTC

Technologia WebRTC jest wykorzystywana do komunikacji w trybie rzeczywistym. Jest to bardzo ważna informacja, ponieważ definiuje ona wymagania i sposób jej działania.

### 1.3.1 Pojęcie Real-Tome Communication

Technologia Real-Time Communication można przetłumaczyć jako komunikacja w czasie rzeczywistych. Sama nazwa określa warunki jakie mają być spełnione:

* ma służyć do komunikacji
* ma ona się odbywać w czasie rzeczywistym

Jest to bardzo ogólne pojęcie. Wykorzystywane jest ona dzisiaj w każdym komunikatorze, takim jak Jabber, Skype, czy polskie Gadu-Gadu.

### 1.3.2 Web Real-Time Communication

WebRTC (Web Real-Time Communication) jest komunikacją w trybie rzeczywistym z wykorzystaniem technologii webowej. Jej zadaniem jest dostarczenie użytkownikom możliwości komunikacji z poziomu przeglądarki internetowej bez instalacji dodatkowych wtyczek. Wymaga ona jednak posiadania jej nowszej wersji. Ważną informacją jest to, że ta technologia jest jeszcze na etapie projektowania i niektóre elementy mogą się nieznacznie zmienić. Obecnie wspierają ją Chrome, Firefox i częściowo w nowej wersji Opera. Internet Explorer nie posiada jeszcze jej obsługi. WebRTC jest rozwinięciem technologii HTML5, która jest dopiero wprowadzana i jej standard jest jeszcze rozwijany.

Dlaczego, pomimo że jest to jeszcze rozwijający się standard, jest z dnia na dzień coraz chętniej poznawana i wykorzystywana? Po wprowadzeniu w piątej wersji HTML’a znaczników audio i video oraz API strumieni i metody getUserMedia (obsługiwanej po stronie JavaScript) strona ma dostęp do naszego mikrofonu i kamerki. Może ona przechwycić w bardzo łatwy sposób strumień, który jest generowany przez kamerkę lub/i mikrofon, i dowolnie potem na nim pracować. Możliwe jest zrobienie zdjęcia z poziomu strony www (przeglądarki), przesłanie danych z kamerki w dowolne miejsce lub zmodyfikowanie ich. Daje to olbrzymie możliwości. Głównym celem tej technologii jest możliwość stworzenia chatu, w którym dwie lub więcej osób będą mogły rozmawiać lub prowadzić wideokonferencje. Dodatkowo jest możliwość prowadzenia w prosty sposób czatu tekstowy.

Docelowo ma technologia ta ma być dostępna na bardzo wielu mediach takich jak komputery, smartfony, tablety, telefony, telewizory. Obecnie trwają już prace wykorzystania jej w komunikacji VoIP. Pozwoli to na prowadzenie rozmowy pomiędzy użytkownikami, z których jeden będzie korzystał z telefonu, a drugi z przeglądarki internetowej.

Rysunek Pierwsza rozmowa wideo developerów chrome i firefox poprzez przeglądarki wykorzystując WebRTC [3].

Po stronie interfejsu WebRTC ma się charakteryzować bardzo prostą obsługą. Po wejściu na odpowiednią stronę użytkownik ma pozwolić jej na dostęp do mikrofonu/kamerki internetowej, co poskutkuje tym, że po chwili będzie miał już połączenie z drugą osobą.

W założeniu tej technologii przeglądarki mają mieć bezpośrednie połączenia między sobą i przesyłać dane pomiędzy siebie. Rola serwerów została zmarginalizowana do generowania i obsługi połączeń i sesji.

Standard ten jest tak przygotowany, by dać końcowym programistom, którzy będą tworzyć aplikacje dla przeglądarek, gotowe i bardzo proste API. Jednak by to osiągnąć trzeba było przygotować od podstaw i wykorzystać mnóstwo technologii i rozwiązań. Jest to bardzo rozbudowany system z wieloma elementami.

## 1.4 Potrzeby, które zrodziły ten standard

Każda wprowadzona nowa technologia wynikała z za potrzebowania na coś. Na powstanie standardu WebRTC miało wpływ kilka elementów.

Rozwój technologii internetowych i możliwości przeglądarek powodują, że coraz więcej usług jest pisanych z wykorzystaniem możliwości jakie dają webowe narzędzia. HTML w wersji 5 posiada bardzo dużo możliwości. Ma on docelowo wyeliminować takie technologie jak Adobe Flash, Microsoft Silverlight i applety java’owe. Obecnie bardzo dużo komunikatorów internetowych jest napisana z wykorzystaniem tych środowisk. Wymaga to instalowanie odpowiedniego oprogramowania lub bibliotek. Microsoft oficjalnie już zrezygnował z technologii Silverlight na rzecz HTML5. Celem numer jeden jest wyeliminowanie wtyczek i obsługa komunikacji tylko poprzez standard HTML5 i WebRTC. Jedną z prób rozmów wideo z poziomu przeglądarki było stworzenie przez firmę Microsoft wtyczki do Internet Explorer, która pozwala na prowadzenie rozmowy w sieci Skype. Niestety wymagało to instalowanie dodatkowego oprogramowania i było dostępne tylko dla wcześniej wymienionej przeglądarki.

Drugą potrzebą było stworzenie jednego uniwersalnego standardu, który by opisywał w jaki sposób mają się odbywać rozmowy audio/wideo. Obecnie większość dostępnych komunikatorów ma swoje sposoby działania i są one zazwyczaj zamknięte.

Obecnie większość komunikatorów, czatów do swojej pracy wymaga zewnętrznych serwerów. Strumień audio/wideo lub wiadomość tekstowa jest najpierw wysyłana do serwera, a stamtąd dopiero do drugiego odbiorcy. Powoduje to większy ruch sieciowy i dodatkowe obciążenie hostów. Standard WebRTC pozwala na przesyłanie danych bezpośrednio pomiędzy przeglądarkami.

W ciągu kilku lat zwiększył się zasób urządzeń, na których można prowadzić komunikację. Dawniej były to komputery i to najczęściej z systemem Microsoft Windows. Obecnie są to urządzenia z system MS Windows, MacOs, Linux; tablety, smartfony z systemami takimi jak Windows Phone, Symbian, iOS; dedykowane przeglądarki internetowe, zarówno te w wersji desktopowej jak Mozilla Firefox, Google Chrome, Opera czy Internet Explorer; do tego dodatkowo telewizory, może w przyszłości inteligentne zegarki. Przy dużej ilości końcowych urządzeń i systemów pojawia się problem z tworzeniem dedykowanych komunikatorów. Trzeba pisać osobne wersje na każdą platformę, co zwiększa koszty i utrudnia utrzymywanie jakości i wprowadzanie nowych funkcjonalności. Jeden system WebRTC ma skutkować tym, że programista końcowego produktu ma pisać tylko jeden kod, który będzie docelowo działał na wszystkich platformach i urządzeniach. Samym zaimplementowaniem technologii WebRTC zajmą się twórcy przeglądarek i systemów operacyjnych.

## 1.5 Cel pracy magisterskiej

Tematem pracy jest porównanie istniejących rozwiązań bazujących na technologii WebRTC. Narzędzia te mają przyspieszyć i pomóc w tworzeniu gotowych aplikacji. Do porównania zostały wybrane kilka najciekawszych. Standard ten dopiero się rozwija, ale widać już w sieci duże zainteresowanie tym rozwiązaniem. Jest bardzo duże zaangażowanie programistów Google’a, fundacji Mozilla w rozwój tej technologii i jej promowanie. Może ona w najbliższej przyszłości wprowadzić wiele zamieszania na rynku webowym. Gdy wszystkie przeglądarki będą już poprawnie i całościowo implementować te rozwiązanie, to jest duża szansa, że będzie można prowadzić rozmowy wideo w przeźroczysty sposób, niezależnie od platformy sprzętowej i systemowej.

WebRTC to nie tylko proste API po stronie przeglądarki, ale również bardzo dużo ciekawych technologii i rozwiązań, które się pojawiły podczas tworzenia tego standardu.

By móc zgodnie z tematem porównać istniejące już biblioteki, skrypty trzeba mieć podstawową wiedzę o tym zagadnieniu. W kolejnych rozdziałach przedstawione zostaną elementy, jakie składają się na tę technologię. Obejmuje ona nie tylko interfejs dostępny po stronie przeglądarki, ale również rozwiązania po stronie serwerowej i warstwie sieciowej, która służy do komunikacji pomiędzy tymi elementami.

Tak zebrana wiedza pozwoli na odpowiednie przygotowanie kryteriów podczas wyboru elementów do porównania.

Celem poniższej pracy magisterskiej jest przygotowanie podłoża do późniejszego wykorzystania tego zagadnienia. Znając zasadę działania składowych elementów tego systemu, jak i gotowych już rozwiązań, łatwo będzie przygotować gotową aplikacje.

## 1.6 Historia standardu

Pierwsze wzmianki pojawiły się w 2009, kiedy to twórcy przeglądarki Chrome zaczęli eksperymentować z rozwiązaniem dla komunikacji w czasie rzeczywistym. W 2010 roku firma Google kupiła firmę On2 [4] odpowiedzialną za kodeki wideo z linii VP i firmę Global IP Solutions (GIPS) [5], która rozwijała rozmowy audio/wideo.

W maju 2011 firma Google oficjalnie rozpoczęła rozwój technologii WebRTC przy współpracy programistów odpowiedzialnych za przeglądarki Firefox i Opera [6]. 23 sierpnia 2011 roku został udostępniony pierwszy szkic standardu WebRTC zaproponowanego przez organizację W3C [7]. W kolejnych latach standard ten jest rozwijany.

W lipcu 2012 na konferencji Google I/O została zaprezentowana technologia WebRTC. 4 stycznia 2013 roku obywał się pierwsza słynna rozmowa programistów przeglądarek Chrome i Firefox poprzez technologię WebRTC [3].

Kolejne przeglądarki otrzymywały wparcie dla tej technologii: Google Chrome w wersji 23 (listopad 2012), Mozilla Firefox w wersji 22 (czerwiec 2013), Opera w wersji 18 (listopad 2013).

# **2. Obecne możliwości i ograniczenia**

Technologia WebRTC jest ciągle rozwijana i udoskonalana. By móc przygotować porównanie istniejących rozwiązań, które pozwolą na przygotowanie projektu aplikacji, trzeba najpierw poznać jakie funkcjonalności wspiera obecnie ten standard.

## 2.1 Możliwości i założenia

Podczas tworzenia standardu Web Real Time Communication zostały określone funkcjonalności, które mają być przez niego udostępniane. Poniższy rozdział przedstawia te najważniejsze i najbardziej przydatne.

### 2.1.1 Słowo wstępu

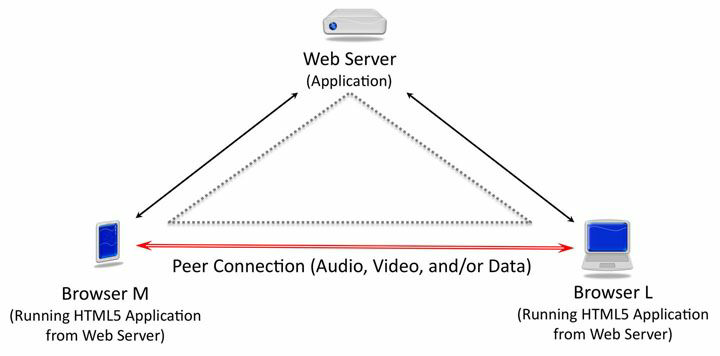
Podczas tworzenia standardu WebRTC wzięto pod uwagę wiele różnych założeń, które mają ułatwić późniejsze jego wykorzystanie i zapotrzebowań, które trzeba rozwiązać. Standard ten nadal jest rozwijany i jeszcze wiele z jego rozwiązań nie zostało do końca sprecyzowanych. Obecnie przeglądarki desktopowe nie do końca wspierają niektóre rozwiązania lub jeszcze nie mają go zaimplementowanego.

Wstępne założenia, jakie określono do zrobienia, są bardzo obiecujące. W poniższych podrozdziałach zostaną opisane najważniejsze funkcjonalności tego standardu, jakie mają być wspierane przez wszystkie przeglądarki i korzyści wynikające z tego. Podrozdział 2.3 zawiera informacje i obecnych ograniczeniach w dostępnie do tej technologii, oraz w tym, co zostało jeszcze do zrobienia.

### 2.1.2 Bezpośrednia komunikacja przeglądarek i zmniejszenie roli serwera

Projektantom tej technologii przyświecał cel, by przeglądarki mogły bezpośrednio ze sobą się komunikować [8, rozdział 3.1.2]. Rola serwerów została zmarginalizowana do nawiązywania połączeń między użytkownikami (poprzez np. współdzielony adres www), zarządzania sesjami. Dzisiaj bardzo często strumienie są wysyłane najpierw na serwer, a dopiero potem do użytkownika docelowego [8, rozdział 3.1.1].

Z serwerów pobierana jest gotowa aplikacja webowa (strona internetowa), która przejmuje obsługę i zarządza potem rozmową. Gdy prowadzi się z drugą osobą konferencję, to strumień wideo i audio jest przesyłany do niej bezpośrednio. Bardzo dobrze przedstawia to rysunek 3 [8, rozdział 1.1.4].



Rysunek Komunikacja pomiędzy przeglądarkami, które pobierają aplikację z serwera www.

Pokazuje on w bardzo uproszczony sposób jak wygląda komunikacja między dwoma przeglądarkami. Po pobraniu aplikacji (strony HTML5) z serwera przeglądarki same zajmują się przesyłaniem danych pomiędzy sobą. Podejście to zmniejsza obciążenie serwera. Nie musi się on zajmować wysyłaniem danych do odbiorców. Zmniejsza to również wykorzystanie łącza internetowego, ponieważ dane są przesyłane tylko raz (do odbiorcy), a nie duplikowane przez serwer, który musi otrzymać te dane, a potem przesłać ponownie.

Serwery mają odpowiadać za przesłanie aplikacji (strony www), nawiązywanie i utrzymywanie sesji, wymiany przez klientów informacji o komunikacji (tzw. sygnały), ustawień [8, rozdział 4.1]. Dodatkowo ich zadaniem jest obejście ograniczeń komunikacji Peer-to-Peer, które nakłada NAT (Network Address Translators - translacja adresów prywatnych sieci na publiczne) i Firewall (filtrowanie ruchu wejściowego/wyjściowego przez klienta) [8, wstęp do rozdziału 3].

### 2.1.3 Wspólne i proste API

W ramach prac nad tym standardem przygotowano API, które jest dostępne dla wszystkich przeglądarek. Docelowo ma ono posiadać wspólny interfejs i być niezależne od wersji przeglądarki (patrz rozdział 3.1). Ułatwi to pisanie aplikacji, gdyż przygotowany kod pod instancją Google Chrome będzie poprawnie działał również pod Mozilla Firefox czy inną przeglądarką. Interfejs dla programisty ma być jak najbardziej prosty i intuicyjny. Został on przystosowany do założeń i składni języka JavaScript, w którym będzie wykorzystywany.

WebRTC jest technologią rozwijającą się, przez co funkcje mają obecnie prefiksy, które przypisali im twórcy przeglądarek (patrz rozdział 3.1). Docelowo będą miały one jednakową nazwę.

### 2.1.4 Przechwytywanie strumienia audio/wideo

Najważniejszym elementem technologii WebRTC jest możliwość przechwytywania strumienia audio i wideo [9, wstęp do rozdziału 1; 18, rozdział 2]. Aplikacja zgłasza, że chce mieć dostęp do obrazu z kamerki lub/i dźwięku z mikrofonu.

Zabezpieczeniem przed podsłuchiwaniem lub szpiegowaniem użytkownika jest to, że musi on świadomie zezwolić aplikacji na dostęp do tych urządzeń. Po przechwyceniu strumienia audio/wideo aplikacja może już dowolnie nim zarządzać i na nim operować (tj. modyfikować go).

### 2.1.5 Zarządzanie strumieniem danych

Gdy aplikacja ma dostęp do strumienia audio/wideo, to programista ma całkowitą dowolność, co z nim może zrobić. Podczas rozmowy z drugą osobą można taki strumień przesłać do innej przeglądarki. Można go zapisać na dysk [10, rozdział 8.10], zmodyfikować [11]. Kolejny podrozdział 2.2.2 przedstawia bardzo ciekawy przykład, w którym obraz z kamerki jest zamieniamy na znaki ASCII. Istnieje również możliwość przesłania drugiej osobie zawartości pliku z naszego dysku twardego jako strumienia, lub takiego, który wcześniej został zmodyfikowany [8, rozdział 5.3.2.5]. Jest bardzo dużo możliwości pracy nad danymi, jakie dostajemy z mikrofonu i kamerki wideo. Gotowy strumień można wysłać do dowolnej ilości punktów docelowych [8, rozdział 1.3] - przykładowo prowadzić wieloosobową konferencję.

### 2.1.6 Prowadzenie rozmów w trybie rzeczywistym

Głównym celem stworzenia technologii WebRTC jest możliwość komunikacji w trybie rzeczywistym. Gdy prowadzona jest rozmowa wideo z drugą osobą, to dostanie ona od razu obraz z kamerki. Nie za godzinę, nie jutro - tylko teraz.

Ograniczeniem jest tylko przepustowość łącza, ale wtedy można zmniejszyć jakość wideo, co poprawi komfort rozmowy.

### 2.1.7 Przeźroczystość oprogramowania i sprzętu

Bardzo ważnym założeniem jest to, by to jaka jest używana przez użytkownika przeglądarka i na jakim pracuje ona systemie i sprzęcie, było całkowicie przeźroczyste (w założeniu, że wspierają ona technologię WebRTC).

Dzięki jednemu wspólnemu standardowi nie ma znaczenia, między jakimi przeglądarkami jest prowadzona rozmowa. Jeden rozmówca może korzystać z Google Chrome pod systemem Android, a drugi z Mozilla Firefox’a pod systemem Linux.

### 2.1.8 Wieloosobowe konferencje

Technologia WebRTC pozwala nam na prowadzenie rozmów tekstowych/audio/wideo z wieloma osobami na raz (np. rozmowa wideo w OpenTok, rozdział 5.3.6).

### 2.1.9 Współdzielenie plików

Jednym ze strumieni, z jakim możemy pracować może być dowolny plik. Można bezpośrednio wysłać jego zawartość bezpośrednio do drugiej przeglądarki (patrz przykładowa aplikacja, rozdział 2.2.4). Plik taki jest przesyłany z ominięciem serwera w tzw. sposobie P2P (Peer-to-Peer).

### 2.1.10 Zarządzanie jakością strumienia

Podczas przechwytywania strumienia wideo programista może określić, jaką chce obsługiwać jakość strumienia [8, rozdział 5.3.2.2; 9, rozdział 11]. Można ustawić wymiary domyślne, opcjonalne strumienia, np. ilość klatek na sekundę.

### 2.1.11 Negocjowanie mediów

Podczas połączenia przeglądarka potrafi negocjować dostępne formaty audio i wideo. Dostosuje ona wysyłany strumień do możliwości odbiorcy [8, rozdział 4.1.2].

### 2.1.12 Wykorzystanie komunikacji w innych obszarach

Fakt, że przeglądarki mogą się łączyć bezpośrednio i mogą w trybie rzeczywistym przesyłać sobie dane można wykorzystać nie tylko w komunikacji audio/wideo, ale również w innych obszarach. Jednym z nich mogą być gry komputerowe, które użytkownik uruchamia w oknie przeglądarki. Dobrym przykładem może być gra online ping-pong (patrz rozdział 2.2.3).

Pomiędzy graczami wymieniane są informacje o pozycji piłeczek, pałeczek. Serwer uczestniczy tylko w nawiązaniu połączenia, sprawdzenia statusów użytkowników i zapisaniu wyników pojedynków.

### 2.1.13 Ogromna ilość końcowych użytkowników

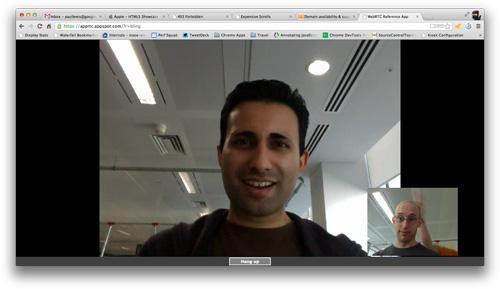
Na świecie są niezliczone ilości urządzeń, które mają dostęp do Internetu i posiadają przeglądarkę internetową. Po zaimplementowaniu przez ich twórców technologii WebRTC i ich aktualizacji dostajemy bardzo dużą ilość końcowych użytkowników, którzy mogą wykorzystać tę funkcjonalność.

## 2.2 Przykładowe aplikacje wykorzystujące technologię WebRTC

Pomimo, że WebRTC jest cały czas rozwijana i jest jeszcze wspierana przez przeglądarki w bardzo małym stopniu, w sieci pojawiają się kolejne strony, które wykorzystują tę technologię. Pokazuje to, że jest bardzo duże zainteresowanie tą nowością. Poniższe podrozdziały przedstawią kilka z najciekawszych. Oczywiście można znaleźć inne zastosowania dla tej technologii, które nie zostały wymienione poniżej.

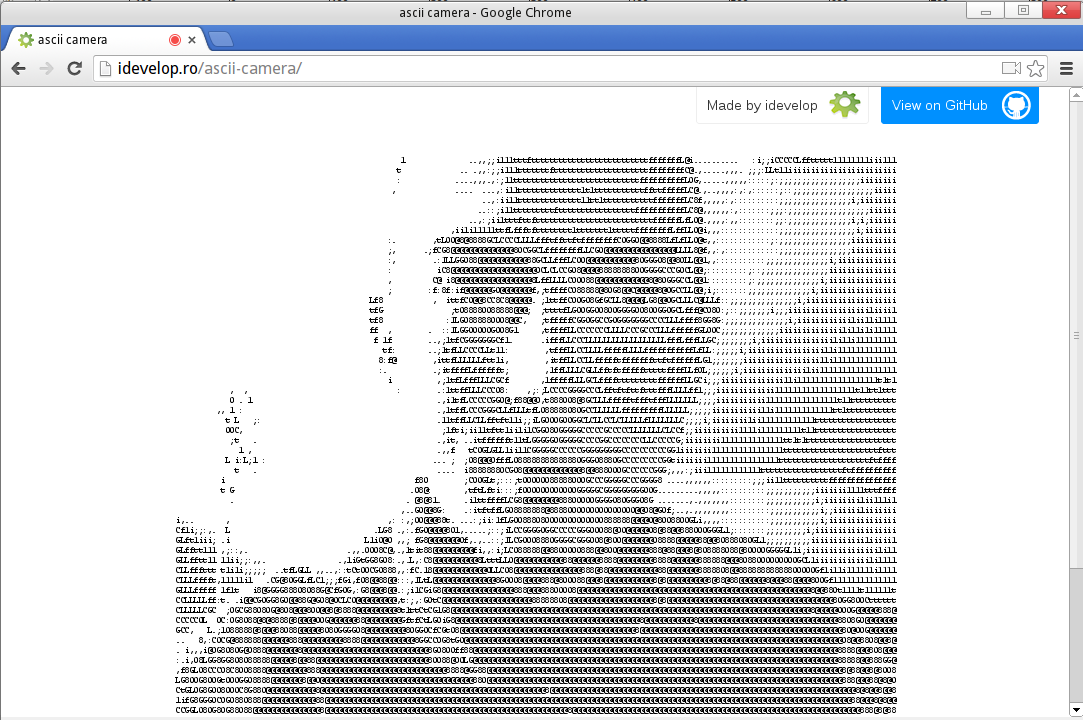
### 2.2.1 Wideokonferencje

Witryna apprtc.appspot.com (http://apprtc.appspot.com/) jest oficjalną stroną twórców standardu, dzięki której można przeprowadzić rozmowę wideo w technologii WebRTC.

Wystarczy, że zostanie udostępniony drugiej osobie adres url, który został wygenerowany i może ona wtedy dołączyć się do naszej konwersacji.

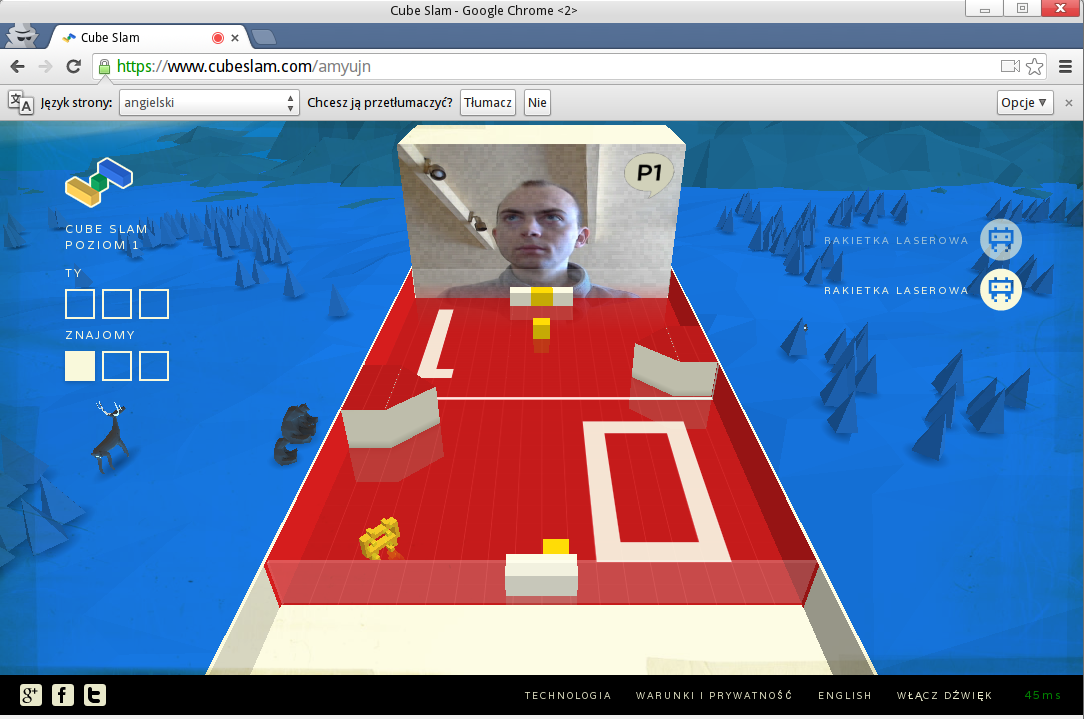
Rysunek Przykład działania aplikacji apprtc.appstop.com

### 2.2.2 Modyfikacja strumienia

ASCII Camera (http://idevelop.ro/ascii-camera/) to bardzo ciekawy przykład, który pokazuje możliwości modyfikacji strumienia wideo. Zamienia ona obraz rejestrowany przez kamerę i zamienia go na znaki ASCII.

Rysunek Przykład działania aplikacji ASCII Camera

### 2.2.3 Użycie komunikacji rzeczywistej w grach

Bardzo dobrym obszarem do wykorzystania WebRTC są gry online. CubeSlam (https://www.cubeslam.com/) to bardzo prosta gra typu ping-pong, która pokazuje możliwości tej technologii. Gracze odbijają piłeczkę pałeczkami. Na dużych monitorach, zaraz za boiskiem, wyświetlany jest obraz z kamerki przeciwnika. Jeśli użytkownik chce z kimś zagrać, to wystarczy wysyłać przeciwnikowi adres url, który jest przypisany do danej gry.

Rysunek . Zrzut z gry CubeSlam

### 2.2.4 Udostępnianie plików

ShareFest (https://www.sharefest.me/) to bardzo prostą aplikacją webową, która pozwala na udostępnianie i przesyłanie plików drugiej osobie. Po wybraniu pliku do przesłania generowany jest unikalny adres. Gdy ten adres otworzy druga osoba, to zostanie poinformowana o możliwości zapisaniu tego pliku na dysk. Ważne jest to, by nadawca nie zamykał tej strony, ponieważ dane są wysyłane bezpośrednio do odbiorcy, a rolą serwera jest tylko zadbanie o utrzymanie sesji.



Rysunek . Strona główna projektu ShareFest

## 2.3 Obecne ograniczenia

Standard WebRTC jest cały czas rozwijany. Pomimo swoich możliwości jest on traktowany obecnie jako ciekawostka. Nie jest on do końca wspierany przez wszystkie przeglądarki, co jest głównym ograniczeniem jego popularności. Przeglądarki Microsoft Internet Explorer i Apple Safari nie wspierają tej technologii w żadnej swojej wersji. Przeglądarki Google Chrome, Mozilla Firefox i Opera obsługują metodę getUserMedia, strumienie (MediaStream), połączenia do serwerów TURN, przesyłanie danych Peer-to-Peer. Elementy takie jak dostęp do pulpitu, przesyłanie “dalej” strumienia od jednego użytkownika do drugiego (przy np. konferencji wieloosobowej) jest dopiero na etapie implementacji w przeglądarkach.

Najnowsze wersje mobilne przeglądarek Chrome, Firefox i Opera dla systemu android posiadają wsparcie dla rozmów wideo.

Przeglądarka Internet Explorer dla systemu Windows Phone i Chrome dla systemu iOS nie posiadają wsparcia technologii WebRTC.

W sieci istnieje projekt “Is WebRTC Ready Yet” (http://iswebrtcreadyyet.com/), którego celem jest monitorowanie stanu wsparcia w przeglądarkach danego rozwiązania. Innym ciekawym projektem jest strona CatIUse (http://caniuse.com/) , która wyświetla możliwości użycia wybranej technologii webowej w danej przeglądarce. Można sprawdzić status implementacji obsługi metody getUserMedia pod adresem http://caniuse.com/#search=getUserMedia/Stream%20API .

Podczas tworzenia tego standardu pojawił się problem sieciowy z NAT’em (Network Address Translators), który przy wyjściu ruchu poza sieć lokalną zamienia adres prywatny na publiczny. Gdy jest potrzeba przesłać dane z jednej przeglądarki do drugiej, to trzeba znać jej bezpośredni adres IP. W celu obejścia tego problemu do obsługi WebRTC są potrzebne serwery, które obsługują technologie STUN (Session Traversal Utilities for NAT) i TURN (Traversal Using Relays around NAT), których celem jest obsłużenie sesji i połączeń dla klientów, którzy są poza NAT’em (patrz rozdział 3.5).

Innym dużym problemem z rozpowszechnieniem tej technologii jest aktualizacja przeglądarek do nowszych wersji. Dużo użytkowników nie zna się na komputerach i posiada ich stare wersje. Najbardziej problematyczną aplikacją będzie pod tym względem Internet Explorer. Jego wersje są przypisane do konkretnej wersji systemu operacyjnego Windows [12]. Firmie Microsoft zależy na tym, by użytkownicy przesiadali się na nowsze wersje systemu Windows, co może sugerować że starsze wersje ich przeglądarek nie dostaną obsługi technologii WebRTC.

Kolejnym problemem jest niedostateczna implementacja tej technologii w systemach mobilnych.

# 3. Omówienie składowych technologii

Technologia WebRTC składa się z wielu elementów. Podczas projektowania tej technologii zostały stworzone przydatne narzędzia i standardy. W tym rozdziale zostaną omówione najważniejsze i najciekawsze z nich. Znając składowe technologii WebRTC będzie łatwiej omówić i przeanalizować gotowe już rozwiązania, służące do zbudowania prostej aplikacji.

Elementy te można podzielić na trzy główne kategorie, które mocno ze sobą współpracują:

* elementy frontowe - technologie, które są użyte po stronie przeglądarki użytkownika i z którymi ma on styczność.
* komunikacja - protokoły i inne elementy, które są wykorzystywane do komunikacji z serwerami i innymi użytkownikami
* elementy serwerowe - aplikacje i rozwiązania używane po stronie serwerowej

## 3.1 Przechwytywanie strumieni poprzez MediaStream

Głównym elementem po stronie przeglądarki, który jest wykorzystywana przez WebRTC, jest przechwytywanie strumienia. Może nim być strumień audio/wideo czy nawet sama zawartość pliku. Po przechwyceniu go, programista może go w dowolny sposób modyfikować czy przesyłać dalej.

Przeglądarki udostępniają metodę *navigator.getUserMedia*, która pozwala aplikacji webowej na dostęp do strumienia [9, rozdział 10.1; 8, rozdział 5.3.2.1; 18, rozdział 2]. Jest to element API (Application Programming Interfaces) języka JavaScript [13].

navigator.**getUserMedia** ( *constraints*, *successCallback*, *errorCallback* );

gdzie:

* *constraints* - to stałe opisujące strumień, jaki chcemy uzyskać
* *successCallback* - funkcja, która ma być wykonana, gdy uda się przechwycić strumień
* *errorCallback* - funkcja wykonana przy niepowodzeniu

Gdy aplikacja chce uzyskać dostęp do strumienia poprzez powyższą metodę, przeglądarka prosi użytkownika o uprawnienia (patrz rozdział 4.1.2).

WebRTC jest jeszcze technologią rozwijającą się, przez co przeglądarki używają tej funkcji pod własnymi nazwami [13]. Przedstawia to poniższa tabelka:

|  |  |
| --- | --- |
| Nazwa funkcji | Przeglądarka |
| getUserMedia | ma to być docelowa nazwa funkcji |
| webkitGetUserMedia | Chrome, Opera |
| mozGetUserMedia | Firefox |
| msGetUserMedia | Internet Explorer (przydzielona, nie obsługiwana) |

Tabela 1. Różnice w nazw funkcji getUserMedia w kolejnych przeglądarkach.

Metoda getUserMedia pozwala na określenie parametrów strumienia, który chcemy używać [8, rozdział 5.3.2.2]. Dla wideo możemy określić jakość i wymiary otrzymywanego obrazu. W parametrze constraints możemy wstawić tablicę z wymiarami dla strumienia wideo. Określa się sekcje mandatory i optional, w których podajemy oczekiwaną przez nas jakoś i opcjonalną, którą dopuszczamy.

Po uzyskaniu strumienia poprzez metodę getUserMedia aplikacja może wydobyć z niego daną ścieżkę audio (getAudioTracks()) lub wideo (poprzez getVideoTracks()) [8, rozdział 5.3.2.5].

### 3.2 Komunikacja przesyłania danych na poziomie przeglądarki

Bardzo ważnym elementem technologii WebRTC jest API RTCPeerConnection. Służy ono do nawiązywania połączeń pomiędzy dwoma odbiorcami (przy komunikacji webowej są to przeglądarki internetowe). Pozwala ono na pominięcie roli serwerów podczas przesyłania danych [8, rozdział 2.1.2; 18, rozdział 3]. Przy komunikacji wieloosobowej obsługa wielu odbiorców spoczywa na programiście i jego aplikacji.

API to korzysta z protokołu ICE (Interactive Connectivity Establishment) [18, rozdział 5], którego celem jest umożliwienie dostępu do mediów dla urządzeń, których ruch przechodzi przez NAT (Network Address Translation) [8, wstęp rozdziału 6.2.7]; Jego zadaniem jest również weryfikacja poprawności komunikacji (dane zostaną wysłane do odbiorcy, który ich oczekuje).

RTCPeerConnection korzysta z RTCSessionDescription do negocjacji obsługi mediów i obsługi sesji multimediów [8, rozdział 2.1.3].

Podczas komunikacji w trybie rzeczywistym powinny występować jak najmniejsze opóźnienia w dostarczaniu danych do odbiorcy. Wszelkie zakłócenia mogą wpływać na pogorszenie się jakości przesyłanych danych, np. przycinanie się rozmowy wideo [8, rozdział 3.1.2]. Ważne jest zadbanie o rozwiązanie takich problemów jak:

* chwilowe zmniejszenie się przepustowości łącza
* utrata części przesyłanych pakietów internetowych
* opóźnienia w dostarczaniu pakietów

Można je rozwiązać korzystając z:

* buforowania przesyłanych danych
* chwilowe pogorszenie jakości transmisji - przykładowo pogorszenie jakości wideo, ale utrzymanie rzeczywistego przesyłanego obrazu bez przycięć - za to jest odpowiedzialny kodek wideo

WebRTC pozwala na przesyłanie danych w trybie rzeczywistym. RTCDataChannel jest kanałem do bezpośredniej komunikacji pomiędzy odbiorcami [8, rozdział 5.3.1.2]. Podczas konferencji wideo będzie to przesyłanie strumienia audio i wideo pomiędzy przeglądarki internetowe.

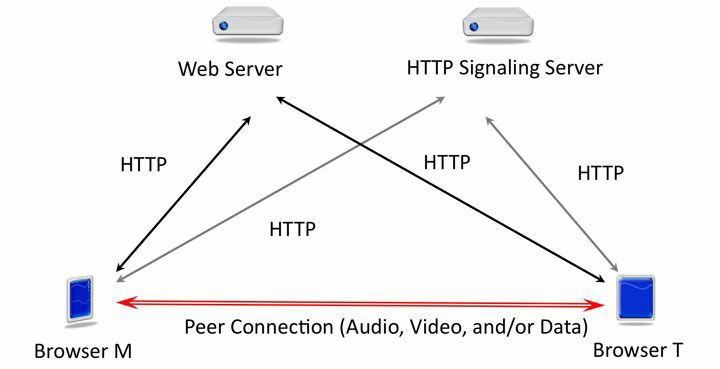
Celem powyższego API jest przesyłanie danych z jak największą transmisją i jak najmniejszym opóźnieniem. Dodatkowo pozwala na równoległą obsługę wielu kanałów danych i piorytetowanie ich. Może ono być używane również do przesyłania komunikatów w grach internetowych czy przesyłania plików.

## 3.3 Sygnały i obsługa sesji

Sygnały służą do koordynacji komunikacji i przesyłania wiadomości pomiędzy przeglądarkami. Głównymi ich zadaniami są [8, wstęp do rozdziału 4.1; 18, rozdział 1 i 4]:

* Negocjowanie typów mediów, dostępnych kodeków i ich ustawień
* Autoryzacja użytkowników
* Kontrola sesji, np. obsługa błędów, wiadomości

Obsługa sygnałów nie jest określona w żadnym standardzie. Programista sam musi określić ich konwencję i sposób zachowania. Określona jest tylko warstwa transportowa (HTTP), sposób zapisu (HTML) i medium (WebRTC) [8, rozdział 4.1.1].

Obsługa sygnałów wymaga serwera jako pośrednika pomiędzy przeglądarkami. Same wiadomości mogą być przesyłane poprzez żądanie HTTP [8, rozdział 4.2.1], WebSocker [8, rozdział 4.2.2] lub Data Channel [8, rozdział 4.2.3]. W przypadku korzystania z HTTP dane mogą być wysyłane wykorzystując XHR, AJAX lub REST’owe API. [8, rozdział 4.2.1]. Poniższy rysunek numer 8 przedstawia przykład wykorzystania sygnałów poprzez powyższą metodę [8, rozdział 4.2.1]:

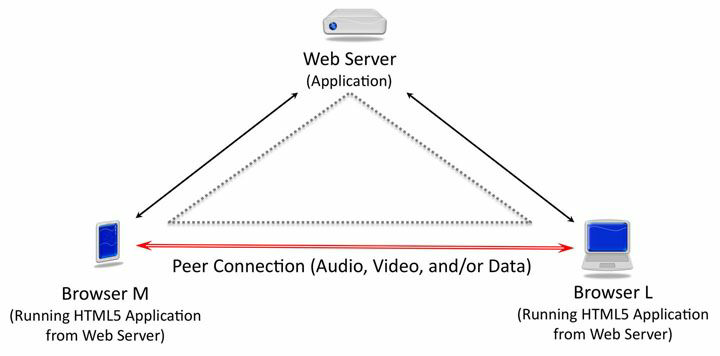
Rysunek . Przesyłanie sygnałów pomiędzy przeglądarkami wykorzystując protokół HTTP.

Programista może samemu zaprojektować protokół do obsługi sygnałów, skorzystać z dostępnych, takich jak SIP czy Jungle lub wykorzystać inną dostępną bibliotekę [8, rozdział 4.3].

## 3.5 Topologia sieci i obsługa połączeń przez Network Address Translation

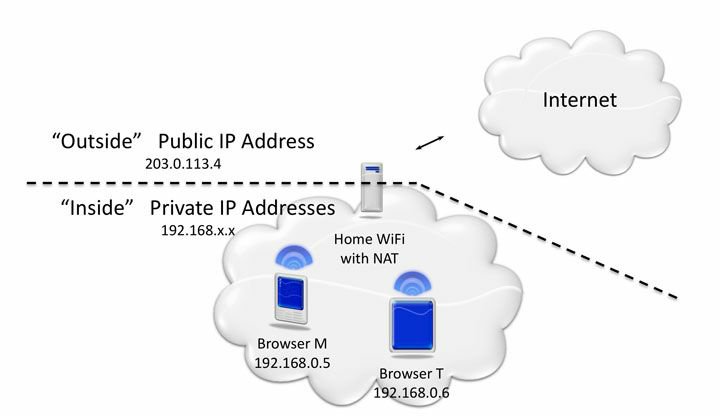
Celem WebRTC jest zmniejszenie roli serwerów podczas komunikacji. Dane mają być przesyłane bezpośrednio odbiorcami. Zmniejsza to transfer danych i niepotrzebnie nie obciąża serwerów.

Poniższy rysunek numer 9 przedstawia prosta sieć (wszystkie urządzenia są w tej samej sieci).

Serwer www udostępnia tylko kod aplikacji webowej. Przeglądarki pobierają go, a następnie nawiązują połączenie pomiędzy sobą. Dane (audio/wideo) są wysyłane bezpośrednio pomiędzy nimi [8, rozdział 1.1.4] ponieważ znają one swoje prywatne adresy IP.

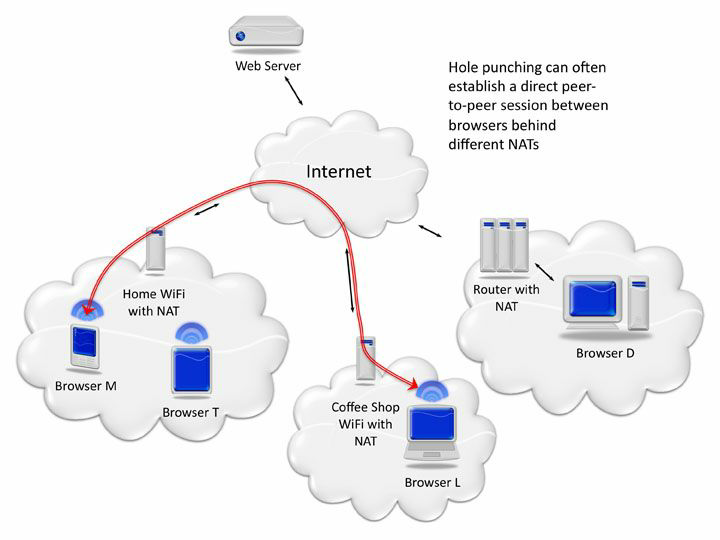
Rysunek . Bezpośrednie przesyłanie danych pomiędzy odbiorcami. Serwer służy tylko jako dostawca aplikacji www.

Bardzo często ruch części przeglądarek połączonych do Internetu przechodzi przez NAT. Network Address Translation (tłumaczenie adresów sieciowych) jest usługą, która zamienia adresy IP źródłowe i docelowe. NAT staje się taką bramą pomiędzy sieciami internetowymi - wewnętrzną i zewnętrzną [8, rozdział 3.2.3; 18, rozdział 3].

Adres prywatny IP jest ukrywany przed światem zewnętrznym adresem publicznym. Router podczas zamiany adresów często też zmienia numer portu. Dzięki temu może obsłużyć wiele żądań z różnych urządzeń korzystających z tej samej sieci internetowej.

Rysunek . Sposób działania usługi NAT pomiędzy siecią lokalną a globalną.

Przesyłanie danych pomiędzy przeglądarkami, które “są za NAT’em” przedstawia poniższy rysunek numer 11 [8, rozdział 3.2.1]:



Rysunek . Przesyłanie danych pomiędzy przeglądarkami w sieci, która zawiera usługę NAT.

Jak wcześniej zostało wspomniane ruch internetowy, który “przechodzi przez NAT” ma zmieniony adres IP. Gdy przeglądarki przesyłają dane pomiędzy sobą, to muszą one znać swoje prywatne adresy. Pojawia się problem - jak przesłać dane do odbiorcy, skoro nie zna się jego adresu?

ICE jest protokołem, którego zadaniem jest rozwiązywanie problemów z połączeniami pomiędzy komputerami, których sieci są za NAT’em [8, rozdział 6.2.7]. Różnymi sposobami stara się on nawiązać najlepsze połączenie typu Peer-to-Peer pomiędzy urządzeniami. W sytuacji, gdy nie uda się nawiązać połączenia wykorzystywane są kolejno serwery STUN lub TURN [18, rozdział 1].

Serwer STUN (Session Traversal Utilities for NAT) działa na zasadzie klient - serwer. Jego zadaniem jest rozpoznanie adresu IP i portu urządzenia klienta. Odczytuje on adres IP i port z przychodzącego żądania (urządzenia będącego poza NAT’em) odsyła je z powrotem jako odpowiedź [8, rozdział 6.2.5]. Dzięki temu urządzenie klienckie poznaje swój adres w sieci i może tę informację wykorzystać do bezpośredniego połączenia z innych użytkownikiem.

Jeśli próba połączenia P2P (Peer-to-Peer) za pomocą serwera STUN się nie powiedzie, to jest wykorzystywany serwer TURN [8, rozdział 6.2.6]. Jest on rozszerzeniem funkcjonalności STUN’a. Poza rozpoznawaniem publicznego adresu IP potrafi przesyłać cały strumień od jednego użytkownika do drugiego. Jest to ostateczne rozwiązanie służące do komunikacji. Wymaga on jednak do tego zadania dużo większych zasobów sprzętowych i sieciowych niż swój poprzednik.

# 4. Bezpieczeństwo i prywatność

Temat prywatności i bezpieczeństwa od dłuższego czasu jest jednym z ważniejszych punktów, podczas wprowadzania każdej nowej technologii. Po aferze PRISM z podsłuchiwaniem użytkowników przez rząd amerykański temat prywatności stał się jednym z najważniejszych i najczęściej poruszanych [14, 15].

Każda technologia może chronić system tylko w ograniczonym stopniu. Każdy użytkownik musi podjąć pewne kroki, by zapewnić sobie bezpieczeństwo w Internecie.

## 4.1 Obecne zabezpieczenia wykorzystywane w WebRTC

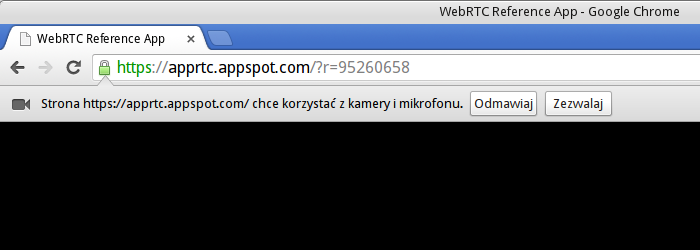
Podczas projektowania systemu WebRTC próbowano w jak największym obszarze zabezpieczyć transmisję i działanie. Poniżej przedstawiono kilka z najważniejszych zabezpieczeń.

### 4.1.1 Przeglądarka internetowa

Pierwszym ogniwem, który jest najbliżej użytkownika jest przeglądarka internetowa. Częsty cykl wydawniczy pozwala na szybsze naprawianie błędów i wykrytych luk bezpieczeństwa. Nie uchroni to jednak komputera, gdy jest wykorzystywana do przeglądania niebezpiecznych lub podejrzanych stron. Użytkownik musi świadomie podawać swoje dane tylko tym witrynom, które są zaufane i znajome.

## 4.1.2 Zezwolenie na dostęp do mikrofonu/kamerki

Gdy aplikacja webowa prosi o dostęp do urządzenia takiego jak mikrofon lub kamerka internetowa, to użytkownik musi świadomie dać jej dostęp do niego [18, rozdział 2].



Rysunek . Komunikat, jaki się wyświetla, gdy aplikacja www prosi o dostęp do kamerki internetowe.

Prośba o dostęp do takich uprawnień jest pierwszym etapem zabezpieczenia przed niechcianym podglądaniem i podsłuchiwaniem użytkownika. Informacja o zgodzie powinna być ustawiona dla każdej sesji aplikacji webowej [8, rozdział 10.2]. Dodatkowo przeglądarki w różny sposób starają się informować użytkownika, że dana aplikacja ma dostęp do mikrofonu lub kamery internetowej. W przeglądarce Chrome jest to czerwona kropka widoczna na zakładce.

### 4.1.3 Brak zewnętrznych wtyczek

Standardowi WebRTC przyświeca cel, by użytkownik nie potrzebował instalować jakichkolwiek wtyczek i rozszerzeń. Niweluje to potencjalne miejsce ataku poprzez dziury i błędy, które znajdują się w dodatkowych bibliotekach.

### 4.1.4 Szyfrowane połączenia

Standard WebRTC zaleca korzystać podczas połączenia z serwerem z szyfrowanego łącza wykorzystującego protokół HTTPS. Jest to ważne również na etapie pobierania kodu aplikacji www na lokalny komputer, jak i podczas uruchamiania go w oknie przeglądarki.

Strumienie przesyłane pomiędzy przeglądarkami korzystają z protokołów DTLS (Datagram Transport Layer Security) i SRTP (Secure Real-time Transport Protocol). SRTP korzysta z symetrycznego klucza AES [8, rozdział 10.3.1].

Przy połączeniu z serwerem STUN używany jest HTTPS z TLS (Transport Layer Security) [8, rozdział 6.2.5]. Można wtedy sprawdzić poprawność certyfikatu witryny i użytkownika w zaufanym systemie (Urząd Certyfikacji) [16, rozdział 3.1.1]. Niestety jest to drogie rozwiązanie (koszt certyfikatu), dlatego często używa się do weryfikacji użytkownika loginu, hasła i odpowiednio ustawionego ciasteczka [8, rozdział 10.1.3].

Sam etap wymiany kluczy do szyfrowania połączeń i sygnałów powinien również być zabezpieczony.

## 4.2 Możliwe słabe punkty

W poniższym podrozdziale przedstawiono tylko kilka z możliwych błędów lub prób ataków. Przy odpowiednio zabezpieczonym systemie, którego użytkownicy są świadomi bezpieczeństwa i zagrożeń, możliwość ataku lub błędu jest bardzo mała.

### 4.2.1 Wirusy i szkodliwe oprogramowanie

Nawet najlepsze zabezpieczenia przeglądarki nie uchronią użytkownika przed niebezpiecznym oprogramowaniem. Wszelkie wirusy, aplikacje typu malware z łatwością mogą udostępnić wrażliwe dane lub w nieświadomy dla użytkownika sposób zamienić zawartość strony lub jej działania. Każdy komputer powinien być zaopatrzony w odpowiednie oprogramowanie, które zapewni mu bezpieczeństwo przed niechcianymi aplikacjami.

### 4.2.2 Kradzież strumienia audio/wideo

Zgoda na udostępnienie aplikacji strumienia audio/wideo przez użytkownika nie chroni go przed tym, co się potem z tymi danymi stanie. Powinien on dawać zgodę tylko zaufanym witrynom.

Zła aplikacja może zmodyfikować strumień, który dostaje z zewnętrznego urządzenia, a potem wysłać go na serwer bez wiedzy użytkownika o tym. Strony nie muszą informować o tym, co robią z danymi, które dostają. Jest to szczególnie ważne, przy prywatnych i poufnych rozmowach czy wideokonferencjach.

Taka sama sytuacja może wystąpić, gdy działanie serwera jest zagrożone przez włamanie. Pomimo, że całe połączenie jest szyfrowane haker może przejąć klucze lub dane o transmisji i je potem wykorzystać.

### 4.2.3 Atak DoS na przeglądarkę

Strumień z danymi może generować znaczny ruch internetowy. Odbiorca nie wie, ile danych dostaje i w pewnym momencie może dojść do ataku DoS (Denial of Service - odmowa dostępu) [17, dodatek C], podczas którego haker może wykorzystać błędy w przeglądarce.

### 4.2.4 Brak szyfrowania strumienia lub sygnałów

W sytuacji, gdy przesyłane dane nie są szyfrowane, to można podsłuchać (więc i przejąć) ruch sieciowy i wykorzystać go w różnych zamiarach. Włamywacz może również zmienić dane o nadawcy, odbiorcy lub serwerze, czy też zmniejszyć poziom zabezpieczeń jakie są używane podczas transmisji.

## 4.3 Prywatność

Technologia WebRTC stawia sobie bardzo wysoko punkt związany z bezpieczeństwem, ale dużo czynników zależy od działań człowieka. Użytkownik musi być świadomy tego, co umieszcza w Internecie.

Istnieje przekonanie o wolności i anonimowości w Internecie ale przykładowo osoba, która zamieszcza pornografię dziecięcą może być namierzona przez adres IP oraz dodatkowe informacje (login, adres e-mail) i pociągnięta do odpowiedzialności karnej za swoje czyny.

### 4.3.1 Kradzież strumień audio/wideo

Użytkownik, który korzysta z wideokonferencji czy przesyła jakiekolwiek dane (słowne, pliki, audio), musi być świadomy tego, że ktoś może podsłuchać czy przejąć ruch sieciowy. Uruchamianie nieznanych aplikacji webowych, którym dajemy dostęp do urządzeń przechwytujących dźwięk lub obraz, mogą w niecny sposób wykorzystać (przejąć, przesłać dalej, udostępnić bez zgody) takie dane. Ważne jest, by korzystać tylko z wiarygodnych stron www.

### 4.3.2 Znajomość adresu IP

Adres IP jest tym elementem, który pozwoli namierzyć lokalizację użytkownika w sieci internetowej. Jest on używany przez serwer, z którego korzystamy.

Technologia WebRTC do przesłania strumienia potrzebuje znać adres prywatny użytkownika. Znając adres prywatny i publiczny można dokładnie określić, z jakiego komputera są przesyłane dane [8, rozdział 10.6.2].

# 5. Porównanie istniejących technologii

Rozdział ten zawiera prosty projekt funkcjonalny aplikacji, dla której będzie wybrane gotowe narzędzie. Wymaga to przygotowania kryteriów według których będzie ono ocenianie.

## 5.1 Projekt aplikacji

Projekt aplikacji jest pierwszym etap przed przystąpieniem do implementacji. W poniższym rozdziale określone zostaną podstawowe wymagania, jakie ma spełniać program.

Zgodnie z tematem pracy aplikacja ma być czatem tekstowy/audio/wideo dla użytkowników. O ile konferencja audio - wideo jest ze sobą ściśle powiązana, to czat tekstowy będzie dodatkowym niezależnym modułem. Na poniższym projekcie skupiono się na wymaganiach funkcjonalnych, które pozwolą na wybór odpowiednich narzędzi.

Kwestia GUI (graficzny interfejs użytkownika) zostaje tutaj pominięta. Nie jest on najważniejszym elementem systemu. Można go szybko zrealizować wykorzystując taką bibliotekę, jak np. Twitter Bootstrap (http://getbootstrap.com/).

Technologia WebRTC nakłada na projekt pewne wymagania, którym projektant musi sprostać. Jest to spowodowane samą architekturą i budową tego systemu.

### 5.1.1 Wymagania funkcjonalne

Przykładowa prosta aplikacja będzie się składać z kilku elementów. Określenie ich na samym początku pozwoli dobrze zaprojektować końcowy produkt. Technologia WebRTC w założeniu ma działać na przeglądarce internetowej, dlatego ma być to jedyne wymagane oprogramowanie i docelowe środowisko uruchomieniowe.

Aplikacja ma być czatem użytkowników. Ma ona pozwalać na przechwytywanie z urządzenia typu kamera obrazu i dźwięku. Te dane wraz z tekstem mają być przesyłane do odbiorcy. Dla uproszczenia w pierwszej wersji może być pominięty aspekt autoryzacji i logowania użytkowników do systemu. Mają oni porozumiewać się w tzw. pokojach - grupie użytkowników, którzy prowadzą konwersację. W internetowych produktach jest dość popularne generowanie unikalnego klucza, który identyfikuje określony pokój. Adres www wraz z tym kluczem jest wysyłany do kolejnych odbiorców. Po otwarciu go trafiają oni od razu do odpowiedniej grupy.

Otrzymane dane (tekstowe, strumień audio/wideo) aplikacja powinna przesłać do odbiorcy. Najlepiej jakby były one transportowane bezpośrednio (P2P), na co pozwala technologia WebRTC. Przy tym etapie projektowania można założyć, że dane będą transportowane w ten sposób - wszelkie błędy powinny rozwiązać serwery STUN i TURN (patrz rozdział 3.5).

Kolejnym ważnym wymaganiem jest to, by skorzystać z istniejących już rozwiązań. Przy obecnym postępie technologicznym warto wykorzystywać gotowe narzędzia. Zaoszczędza to czas i pracuje się wtedy na przetestowanym i używanym przez wielu użytkowników rozwiązaniu.

### 5.1.2 Wymagania przy korzystaniu z WebRTC

Każda wykorzystana technologia opiera się o pewnych wymaganiach, które zdefiniowali jej twórcy. Nie inaczej jest z WebRTC. Przy wyborze tej techniki trzeba wziąć pod uwagę jej ograniczenia. Aplikacja powinna się składać z kilku następujących elementów.

W pierwszym etapie (patrząc od strony użytkownika - przeglądarki internetowej) potrzebna jest biblioteka, która pozwoli na przechwycenie treści audio/wideo. Dane te następnie mają być przesłane do odbiorcy.

Przydatny jest serwer z kodem aplikacji, lecz nie jest to konieczne wymaganie. Można wykorzystać darmowy serwer HTTP - apache.

Kolejnym ważnym punktem będzie serwer służący do wymiany komunikatów i obsługi sesji. Nie ma potrzeby od razu definiować interfejsu służącego do komunikacji. Warto wykorzystać takie, jakie udostępniają istniejące już rozwiązania.

## 5.2 Kryteria porównawcze

Przed przystąpieniem do rozpoznania narzędzi i wybrania z nich odpowiednich warto przygotować kryteria porównawcze. Pozwoli to określić cechy (elementy), jakie mają być brane pod uwagę podczas analizy.

Pierwszym i najważniejszym elementem, na który trzeba uwzględnić, są możliwości danego produktu. Narzędzie jest przydatne, gdy spełnia choć jeden warunek postawiony w rozdziale 5.1.2. Im więcej punktów spełnia dane rozwiązanie, tym bardzie jest przydatne.

Dokumentacja, opis interfejsu, stanowią kolejny ważny element. Dobrze opisany produkt pozwoli na szybkie jego wykorzystanie, poznanie możliwości. Przykłady zamieszczone przez autorów umożliwiają na sprawdzenie manualnie funkcjonalności ich produktu przed przystąpieniem do implementacji własnej aplikacji. Przydają się również, gdy tworzymy na ich bazie gotowy system. Bardzo cenną rzeczą są tzw. tutoriale, które opisują instalacje i budowę aplikacji krok po kroku.

Kolejną punktem jest łatwość instalacji i użycia danego produktu. Pomimo dobrej dokumentacji dana biblioteka może być skomplikowana w wykorzystaniu, co znacznie zmniejsza użyteczność narzędzia.

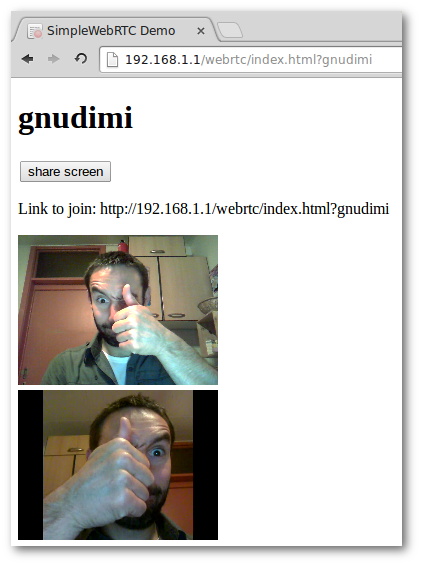
Ważnym czynnikiem decydującym o wyborze technologii jest fakt, czy jest ona cały czas rozwijana i utrzymywana. Wykorzystanie dobrego, ale już starego narzędzia może w przyszłości wygenerować duże problemy z utrzymaniem aplikacji. Może to skutkować potrzebą przepisania naszego produktu na inną bibliotekę.

Dodatkowym składnikiem, który może decydować o wyborze jest to, czy dane rozwiązanie jest komercyjne czy darmowe. Przy prostych aplikacjach warto korzystać z darmowego narzędzia, ale jeśli chcemy zbudować zaawansowany system przewidziany dla wielu użytkowników, to warto się zainteresować komercyjnym rozwiązaniem. Ostatnimi czasy popularny się staje model, w którym twórcy udostępniają za darmo w ograniczonym zakresie (funkcjonalnym lub czasowym) dany produkt. Pozwala ono przy małych projektach przetestować go i w przyszłości zdecydować się na płatne i bardziej rozbudowane wersje.

## 5.3 Zestawienie narzędzi

Wśród wykorzystanych w porównaniu narzędzi widać ciekawy trend. Wszystkie projekty są dostępne za darmo w serwisie GitHub. Bardzo często wykorzystywany jest Node.js, który pozwala na tworzenie aplikacji serwerowych w języku JavaScript i zarządzanie nimi [18, rozdział 4]. Każdy produkt został opisany według podanych w rozdziale 5.2 kryteriów

### 5.3.1 SimpleWebRTC

SimpleWebRTC jest darmową biblioteką, której celem jest szybkie przygotowanie prostej aplikacji po stronie przeglądarki. Strona projektu to http://simplewebrtc.com/, natomiast kod tego narzędzia jest rozwijany na portalu GitHub pod adresem https://github.com/HenrikJoreteg/SimpleWebRTC.

Rysunek . Przykład działania aplikacji SimpleWebRTC.

Pierwszym elementem, który się rzuca w oczy na stronie tego rozwiązania jest prostota jego użycia, która był celem jej powstania. Biblioteka ta korzysta domyślnie z serwera STUN Google’a (stun.l.google.com:19302) oraz z serwera sygnałów dostępnego pod adresem signaling.simplewebrtc.com:8888. Istnieje możliwość nadpisania tych wartości własnymi.

Po sklonowaniu repozytorium, a następnie instalacji odpowiednich modułów Node.js’a i uruchomieniu pliku server.js dostajemy bardzo prostą aplikację. Pozwala ona na stworzenie pokoju i przeprowadzenie konwersacji audio/wideo oraz udostępnienie pulpitu (tej funkcjonalności niestety nie udało się uruchomić).

Biblioteka ta jest reklamowana jako bardzo proste narzędzie, co ukazuje załączona do kodu przykładowa aplikacja. Niestety bardzo dużym minusem tego narzędzia jest dokumentacja techniczna. Brakuje informacji o jego możliwościach, dostępnym interfejsie. Jedyną drogą, by poznać funkcjonalność tej biblioteki jest przeglądanie jej kodu, co znacznie utrudnia z nią pracę i wymaga znajomości języka JavaScript. Dobrze, że jest on skomentowany przez twórców.

WebRTC odpowiada za obsługę audio/wideo, przesyłania danych pomiędzy użytkownikami, obsługę serwerów STUN i sygnałów. Programista musi jedynie skonfigurować tę bibliotekę i dorobić jej obsługę po stronie przeglądarki, np. osadzenie elementu wideo czy możliwość wpisania nazwy nowego pokoju.

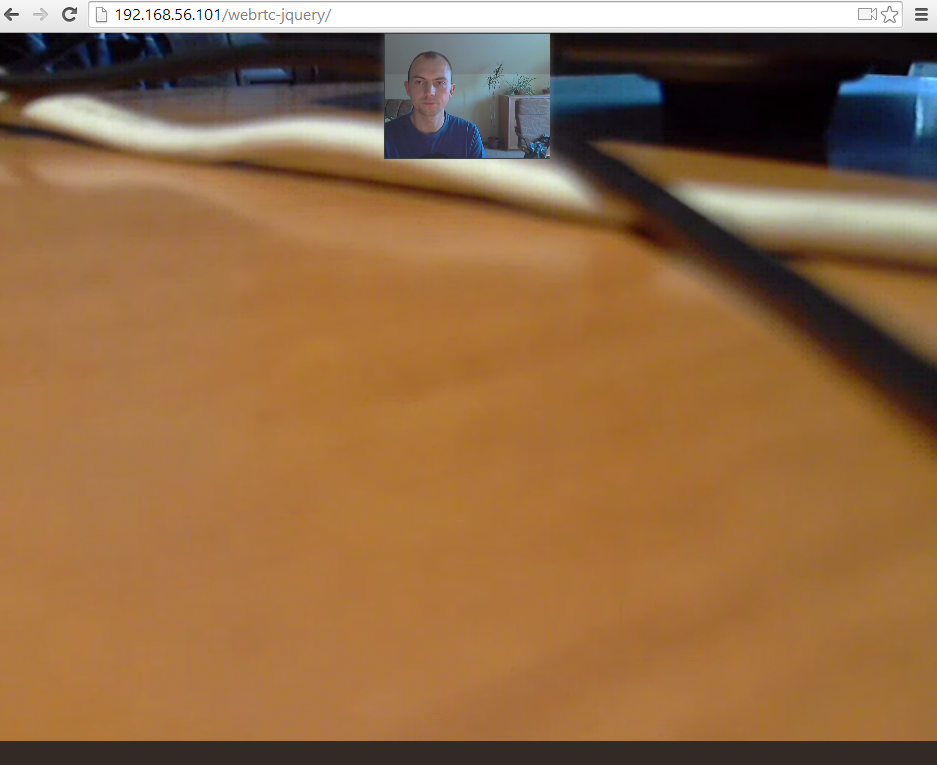
Strona projektu zawiera odnośniki do innych narzędzi, które możemy wykorzystać w założonej aplikacji, np. SignalMaster (https://github.com/andyet/signalmaster). Może posłużyć za własny serwer sygnałów. Autor zaznacza, że jego biblioteka SimpleWebRTC jest wykorzystywana przez projekt talky.io (https://talky.io/), który pozwala na darmową konwersacje audio/wideo.

### 5.3.2 WebRTC jQuery plugin

WebRTC jQuery plugin jest darmową wtyczką, której celem jest prosta obsługa WebRTC. Wymaga ona do działania biblioteki jQuery. Kod tego rozszerzenia znajduje się w serwisie GitHub (https://github.com/freshtilledsoil/webrtc-jquery-plugin).

Opisywane narzędzie to bardzo prosta biblioteka, która nie posiada dużych możliwości. Pozwala ona na przechwytywanie strumienia audio/wideo z przeglądarki, stworzenie pokoju, przesyłanie w sieci lokalnej danych pomiędzy użytkownikami. Konferencja może być prowadzona tylko pomiędzy dwoma uczestnikami. Funkcjonalności te można zobaczyć w wersji online pod adresem http://www.freshtilledsoil.com/the-future-of-web/webrtc-video/.

W kodzie repozytorium git znajduje się plik z biblioteką w języku JavaScript i przykładowa aplikacja www. Jej możliwości zostały opisane w powyższym akapicie. Niestety udało się ją uruchomić tylko pod przeglądarką Google Chrome.

Strona projektu zawiera krótką dokumentację jej możliwości. Zamieszczono tam opis jak zainstalować i przygotować prostą aplikację (tę przykładową). Dodatkowo twórca zamieścił informację, na jakich przeglądarkach będzie działać jego biblioteka.

Rysunek . Konferencja audio wideo z wykorzystaniem WebRTC jQuery plugin.

Niestety aplikacja nie jest już rozwijana. Ostatnie zmiany w repozytorium są datowane na ponad rok czasu. Jednak dzięki możliwościom serwisu GitHub jest duże prawdopodobieństwo, że ktoś przejmie ten projekt i będzie go dalej rozwijał.

### 5.3.3 Peer.js

 System Peer.js (http://peerjs.com/) jest darmową aplikacją rozwijaną na GitHub’ie. Składa się ona z dwóch powiązanych ze sobą członów - biblioteki do obsługi technologii WebRTC po stronie klienckiej (https://github.com/peers/peerjs) i gotowego serwera (https://github.com/peers/peerjs-server), którego celem jest nawiązywanie połączeń przy wykorzystaniu wcześniej wspomnianego narzędzia. Są one napisane w języku JavaScript. Serwer opiera się o technologię Node.js.

Rysunek . Konferencja audio/wideo przy wykorzystaniu Peer.js.

Twórcy aplikacji nie muszą instalować dedykowanego serwera do nawiązywania połączeń. Można wykorzystać przygotowany przez nich hosting. W wersji darmowej pozwala on na obsługę do 50 użytkowników. Rozwiązanie takie pozwala na szybkie przetestowanie aplikacji w sieci. Zmniejsza również ilość systemów, które ma do utrzymania twórca aplikacji we wstępnym etapie prac.

Na stronie projektu (http://peerjs.com/) znajdują się działające przykłady, które ukazują możliwości tego narzędzia. Dostarcza ona też dobrze przygotowaną dokumentację i instrukcję, jak krok po kroku stworzyć gotowy projekt.

Peer.js pozwala na stworzenie konferencji audio/wideo i czata tekstowego. Można dzięki niej też przesyłać dane (pliki). Dodatkowo jeden z załączonych przykładów przedstawia, jak użyć tę technologię do stworzenia webowej wieloosobowej gry internetowej - Tetris.

Cały kod wraz z przykładami można sklonować z GitHub’a i uruchomić z poziomu lokalnej przeglądarki. Domyślnie jako serwer do nawiązywania połączeń jest ustawiony hosting udostępniony przez twórców. Najpierw trzeba pobrać repozytorium git’a, zainstalować niezbędne zależności po stronie Node.js, a następnie wykonać skrypt znajdujący się w katalogu bin podając własną domenę, port i ścieżkę path. Uruchomi to gotowy serwer, dzięki któremu będzie można uzyskać połączenia przez przeglądarki. Można go wykorzystać jako lokalny serwer, ustawiając w konstruktorze elementu Peer odpowiednio jego serwera, port i adres path.

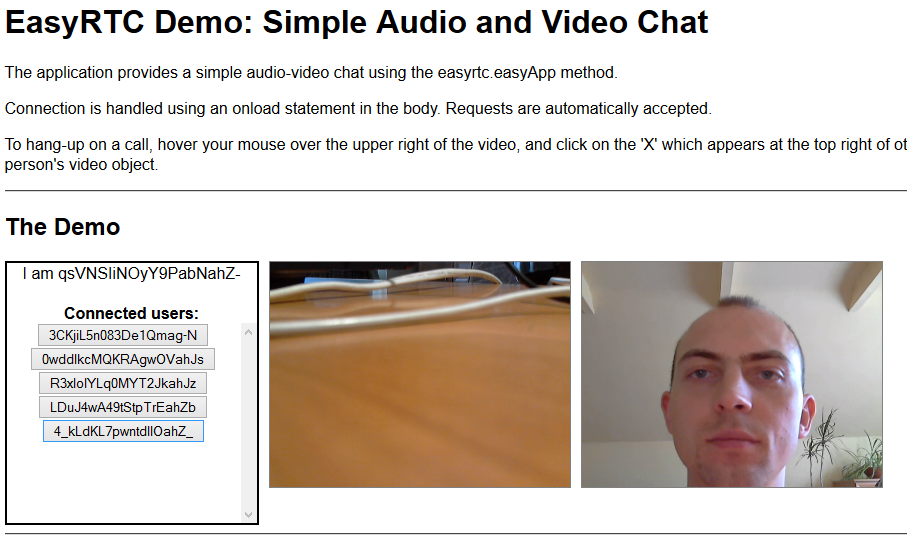
Biblioteka kliencka w domyślnej konfiguracji korzysta z serwera STUN Google’a. Można jednak ten adres nadpisać w konstruktorze obiektu Peer i skorzystać z innego, np. własnego serwera. Dodatkowo jest również możliwość ustawienia adresu serwera TURN (patrz rozdział 3.5).

Kod zarówno biblioteki klienckiej jak i serwerowej jest dość dobrze skomentowany, co pozwoli na szersze poznanie jego możliwości. Autorzy dodatkowo na swojej stronie zamieścili tabelę, w której można sprawdzić pomiędzy jakimi wersjami przeglądarek jest możliwe przesyłanie danych P2P (http://peerjs.com/status/).

Dzięki dobrej dokumentacji z tego narzędzia korzysta się łatwo i przyjemnie. Wykorzystanie technologii Node.js po stronie serwera pozwala osobie, która zajmuje się frontendem łatwo napisać odpowiedni kod aplikacji.

### 

### 5.3.4 EasyRTC

EasyRTC jest darmowym narzędziem do obsługi WebRTC. Na stronie projektu (http://easyrtc.com/) znajduje się informacja o wersji płatnej. Udostępnia ona odpowiednie API dla systemów takich jak Android i iOs. Kod projektu (w wersji darmowej) jest dostępny po portalu GitHub (https://github.com/priologic/easyrtc).

Rysunek . Przykład działania systemu EasyRTC.

Projekt ten składa się on z dwóch części - serwerowej i klienckiej. Umieszczone zostały one w jednym repozytorium. Pierwsza część opiera się o technologię Node.js. Znajduje się tam prosty serwer, służący do połączeń pomiędzy przeglądarkami. Zapewnia on również obsługę przykładowych aplikacji, które mają przedstawić możliwości tego narzędzia. Ich wersja online dostępna jest również pod adresem http://demo.easyrtc.com/demos/index.html. Przedstawiają one takie funkcjonalności jak czat tekstowy, konferencja audio/wideo czy przesyłanie plików pomiędzy użytkownikami. Drugi element to biblioteka JavaScript’owa, która zapewnia powyższe elementy po stronie przeglądarki internetowej.

W repozytorium Git (https://github.com/xirsys/easyrtc#documentation) znajduje się dobra dokumentacja. Instruuje ona jak krok po kroku, jak zainstalować to narzędzie, jak je uruchomić i jak zbudować aplikację w kolejnych etapach. Sam interfejs API też jest dobrze opisany i zaopatrzony odpowiednimi przykładami. Sam kod bibliotek jest dobrze udokumentowany.

EasyRTC ma duże możliwości. Pozwala ono na stworzenie czatu tekstowego, konferencji audio/wideo, przesyłania danych, udostępnianie pulpitu. Domyślnie jest ustawione kilka darmowych serwerów STUN, ale można użyć własny. Autorzy na stronie dokumentacji pokazują jak wykorzystać w ich projekcie autoryzację użytkowników, szyfrowanie danych oraz tworzyć pokoje, w których będą oni obsługiwani.

Narzędzie to może spokojnie do zbudowania dużej aplikacji i czatu audio/wideo. Dużym jest plusem jest wachlarz możliwości, które zapewnia oraz dobra dokumentacja i przykłady.

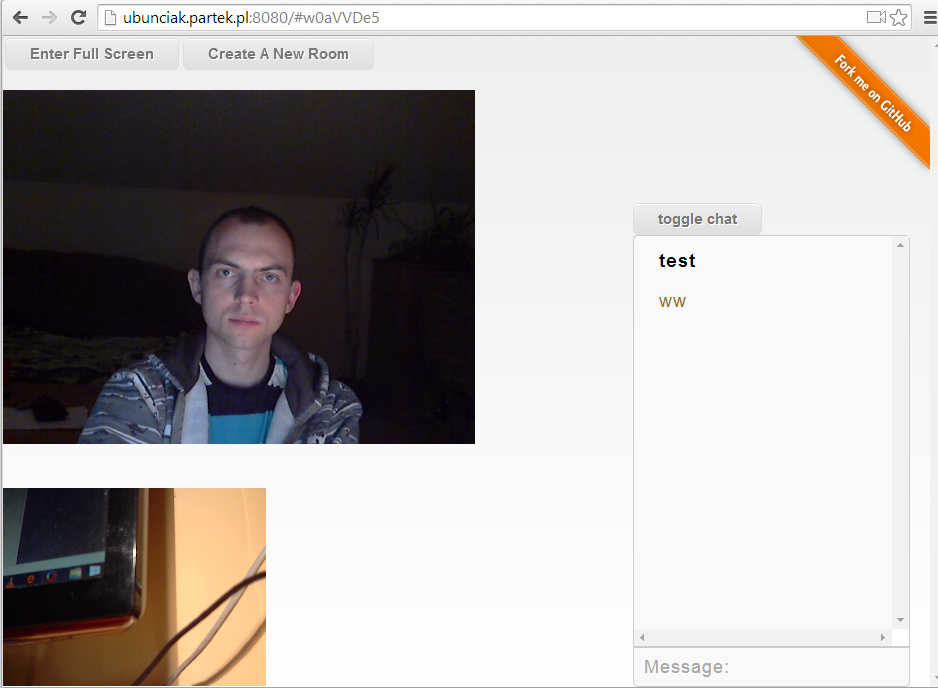
### 5.3.5 WebRTC.io

WebRTC.io to darmowy projekt na GitHub’e (https://github.com/webRTC). Jego celem jest stworzenie abstrakcyjnej warstwy do komunikacji przy użyciu WebRTC. Składa się on z dwóch elementów, które są w osobnych repozytoriach. Webrtc.io-client (https://github.com/webRTC/webrtc.io-client) jest biblioteką stworzoną w języku JavaScript, która służy do obsługi połączeń po stronie przeglądarki. Drugie repozytorium - webRTC.io (https://github.com/webRTC/webRTC.io) - dostarcza serwer sygnałów, który można zaimplementować w technologii Node.js.

Powyższe rozwiązania to zestaw gotowych narzędzi, które można wykorzystać do zbudowania czatu użytkownika. Domyślnie korzystają one z serwerów STUN Google’a, ale można je zmienić na dowolne inne. Niestety kod tego projektu nie jest utrzymywany. Ostatnie zmiany odbyły się ponad rok temu. Jest on jednak na tyle zaawansowany, że warto go umieścić w poniższym zestawieniu.

Pierwszym elementem, który rzuca się w tym projekcie jest bardzo słaba dokumentacja. Jest informacja jak zainstalować moduł odpowiedzialny za serwer sygnałów, oraz jak przygotować bardzo prostą działającą aplikację, która będzie się na nim opierać. Niestety autorzy nie zamieścili dokładnych informacji o jego możliwościach. Jest tylko notka, że ma on służyć jako abstrakcyjna warstwa do WebRTC. Projekt nie posiada dokumentacji interfejsu dostępnego API.

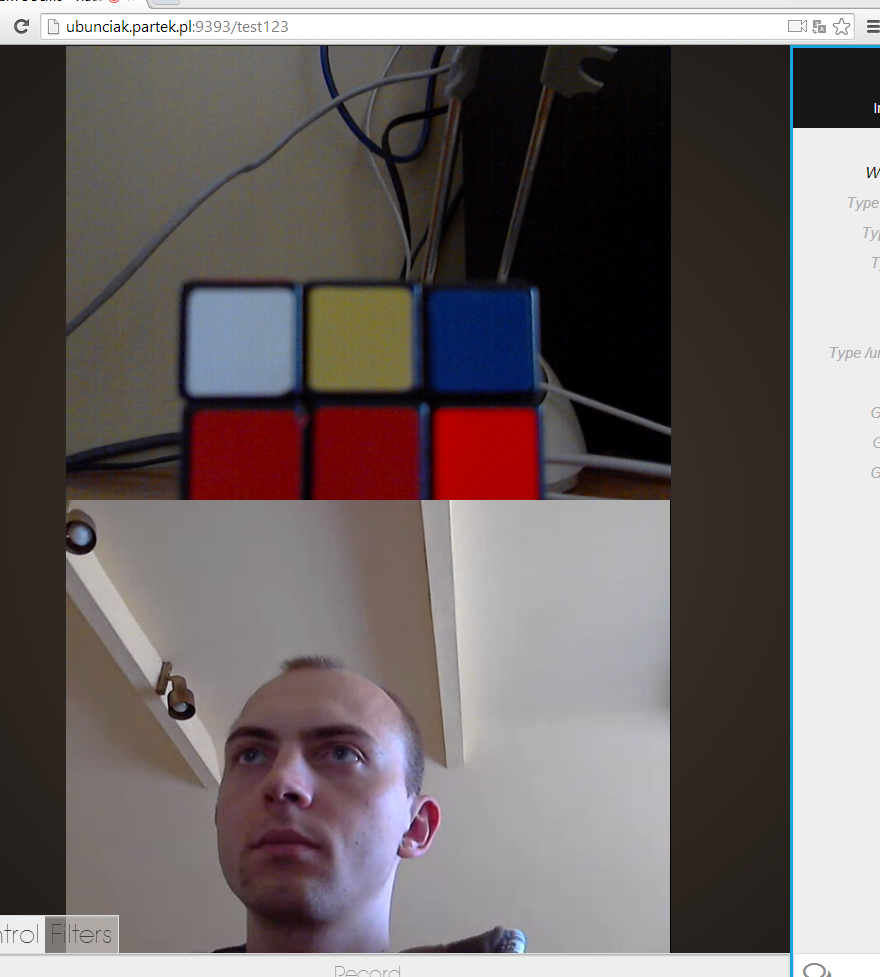
Autorzy w ramach tego projektu dostarczyli na szczęście trzecie repozytorium (https://github.com/webRTC/webrtc.io-demo) z prostą przykładową aplikacją. Dostarcza ona gotowy serwera. Po wpisaniu adresu w przeglądarce wyświetli się gotowa aplikacja. Informacja jak zainstalować ją krok po kroku jest opisana na stronie projektu. Demo składa się z prostego serwera sygnałów, który ma posłużyć do nawiązywania połączeń i aplikacji, która pozwala na kontaktowanie się użytkownikom. Komunikacja odbywa się w podobny sposób jak działają biblioteki socket.io i WebSocket.

Przykładowa aplikacja przedstawia możliwości omawianego narzędzia. Biblioteka JavaScript pozwala na przechwycenie strumienia audio/wideo z przeglądarki i przesłanie go do odbiorcy. Poza danymi multimedialnymi można również przesyłać dane tekstowe - czat testowy. Konferencje odbywają się w ramach pokoi, których identyfikatory są generowane. Gotowy serwer sygnałów poza przechowywaniem i zarządzaniem informacji o pokojach służy też jako serwer www aplikacji. Dane pomiędzy przeglądarkami są wysyłane bezpośrednio.

Rysunek . Przykładowa aplikacja z wykorzystaniem WebRTC.io.

### 

### 5.3.6 OpenTok

OpenTok (http://tokbox.com/) jest komercyjnym rozwiązaniem, które ma dostarczać gotowy czat tekstowy i audio/wideo. Jest on dedykowany projektom, które przewidują dużą ilość użytkowników i długi czas ich rozmów. Użytkownik na start dostaje 10 tysięcy minut w czasie pierwszego darmowego miesiąca. Kolejne miesiące są rozliczane wg ilości wykorzystanych przez użytkowników minut.

Rysunek . Aplikacja demo OpenTok.

Projekt dostarcza gotowe API, dzięki któremu można przygotować gotową aplikację czatu tekstowego i audio/wideo. Po zarejestrowaniu się na stronie użytkownik dostaje identyfikator API i token dla aplikacji, które są generowane dla każdej usługi osobno. Klucze te są niezbędne do uruchomienia aplikacji napisanych w tym systemie.

OpenTok dostarcza gotowe rozwiązania w technologii WebRTC, które udostępniają możliwość stworzenia czasu dla użytkowników. Głównym celem jest udostępnienie możliwości wieloosobowej konferencji audio/wideo i czatu tekstowego. Twórcy chcą przygotować platformę do komunikacji dla wszystkich systemów. Poza przeglądarkami internetowymi autorzy przygotowaniu również adapter pozwalający na obsługę WebRTC dla Internet Explorer’a, gotowe środowisko programistyczne dla urządzeń mobilnych z systemem Android i iOS. Projekt dostarcza również obsługę serwerów STUN i TURN. Twórcy chwalą się również inteligentną obsługę jakości rozmów w zależności od możliwości i prędkości łącza internetowego. Dodatkowo jest możliwość archiwizacji konferencji audio/wideo - nagrania ich i zapisania na później.

Twórcy w ramach projektu dostarczają gotowe rozwiązania dla programistów. Po stronie przeglądarki internetowej - tzw. frontowej - udostępniają bibliotekę JavaScript, której celem jest obsługa WebRTC po stronie użytkownika systemu. Po stronie backendowej można przygotować serwer z aplikacją i obsługą sygnałów i połączeń. Na serwisie GitHub znajduje się użytkownik opentok (https://github.com/opentok), który zawiera w sobie repozytoria związane z tym projektem. Kody są cały czas aktualizowane - widać, że jest to komercyjne oprogramowanie, które twórcy cały czas utrzymują i rozwijają. Znajdują się tutaj gotowe biblioteki w takich językach jak Ruby, Node.js, .Net, PHP, Java; dla środowisk Android, iOS; wtyczka dla PhoneGap’a.

Czat ma być aplikacją webową, dlatego skupimy się na tej technologii. Po stronie frontowej do obsługi aplikacji potrzebne jest dołączenie w kodzie biblioteki https://swww.tokbox.com/webrtc/v2.0/js/TB.min.js zgodnie z opisem na stronie https://dashboard.tokbox.com/quickstart. Strona ta zawiera instrukcję, jak krok po kroku przygotować i sprawdzić, czy nasz token i klucz API są poprawne.

Do obsługi, po stronie serwerowej, sesji i połączeń, zostało przygotowane repozytorium https://github.com/opentok/opentok-node. Dostarcza ono gotowy kod, który pozwoli na stworzenie usługi w technologii Node.js. Znajduje się tam dobra dokumentacja, która opisuje jak obsługiwać sesje, generować tokeny potrzebne do stworzenia pokojów i jak archiwizować rozmowy.

Na serwisie GitHub znajduje się również repozytorium z przykładową aplikacją - https://github.com/opentok/OpenTokRT. Do jej uruchomienia również potrzebny jest wygenerowany token i klucz API. Na tej stronie znajduje się instrukcja krok po kroku jak zainstalować i uruchomić ją w lokalnym środowisku. Serwer, wykorzystujący technologię Node.js, obsługuje tworzenie pokoi, zarządzanie sesjami i potrafi zwrócić aplikację www. Wspomniany w tym podrozdziale przykład spełnia wszystkie wymagania, jakie zostały określone w rozdziale 5.1.1, to jest czat tekstowy i możliwość konferencji audio/wideo. Jest on również dostępny w wersji online pod adresem https://opentokrtc.com/.

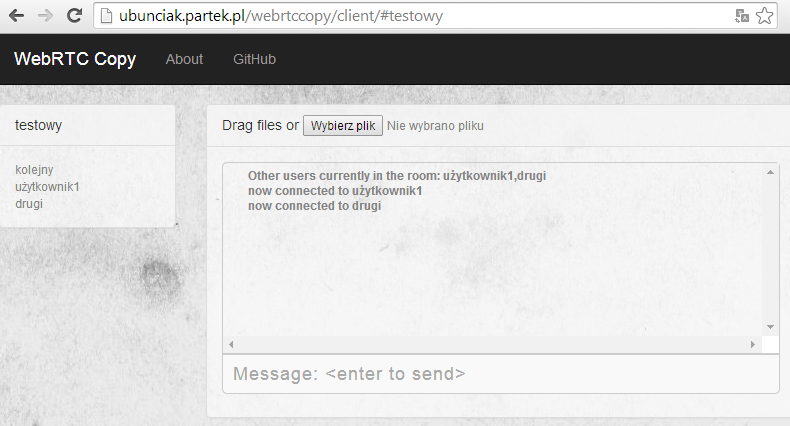
Dokumentacja projektu jest dostępna na stronie twórców OpenTok http://tokbox.com/opentok/quick-start/. Zawiera ona opis podstawowych funkcjonalności tego systemu, zarówno dla biblioteki JavaScript dla przeglądarki jak i dla SDK dla różnych języków. Dostępne są przykłady ich użycia. Twórcy udostępnili również opis interfejsów API funkcji i modułów.

### 5.3.7 WebRTCCopy

WebRTCCopy jest darmowym projektem dostępnym na serwisie GitHub (https://github.com/erbbysam/webRTCCopy). Jest to gotowa implementacja czatu tekstowego dla użytkowników z możliwością przesyłania plików. Dodatkowo twórcy zapewniają warstwę szyfrowania danych.

Repozytorium git’owe składa się z dwóch elementów - części klienckiej, która jest przewidziana do obsługi po stronie przeglądarki internetowej i części backendowej, która udostępnia serwer aplikacji i serwer sygnałów i przechowywania sesji. Jest ona oparta o technologię Node.js.

Aplikacja udostępnia prosty czat tekstowy dla użytkowników, który odbywa się w wydzielonych pokojach. Dodatkowo jest możliwość przesyłania plików w sieci P2P pomiędzy przeglądarkami uczestników. System ten może posłużyć jako gotowa wstępna implementacja komunikatora, który można w ramach projektu rozwinąć, dodając obsługę konferencji audio/wideo. Projekt ten jest darmowy, przez co można go użyć jako bazowego systemu dla naszej aplikacji.

Na stronie WebRTCCopy znajduje się prosta dokumentacja, która opisuje możliwości i wymagania tego systemu. By zainstalować WebRTCCopy na naszym komputerze wystarczy pobrać repozytorium, zainstalować zależności Node.js i uruchomić serwer, którego wywołanie znajduje się w opisie.

Rysunek . Czat tekstowy w aplikacji WebRTCCopy.

Aplikacja ta zawiera możliwość stworzenia pokoju, w ramach którego będzie odbywał się czat tekstowy. Możną ją przetestować w wersji online pod adresem https://www.rtccopy.com. Ostatni raz była on uaktualniana kilka miesięcy temu.

## 5.4 Porównanie przedstawionych rozwiązań

Przygotowanie przedstawienie rozwiązań w rozdziale 5.3 pozwoliło zapoznać się z narzędziami i ich możliwościami. Do wyboru odpowiednich potrzebne będą dwa porównania. Jednym z nich jest zestawienie dostępnych funkcjonalności. Drugim jest konfrontacja według kryteriów przedstawionych w podrozdziale 5.2

### 5.4.1 Zestawienie możliwości narzędzi

Podrozdział ten przedstawia zbiorcze porównanie możliwości narzędzi zgodnie z założeniami z rozdziału 5.1.1. Zbiera to tabela numer 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SimpleWebRTC | WebRTC jQuery plugin | Peer.js | EasyRTC | WebRTC.io | OpenTok | WebRTCCopy |
| Obsługa audio/wideo | tak | tak | tak | tak | tak | tak | nie |
| Czat tekstowy | nie | nie | tak | tak | tak | tak | tak |
| Pokoje | tak | tak | tak | tak | tak | tak | tak |
| Połączenie P2P | tak | tak | tak | tak | tak | tak | tak |
| Obsługa NAT | tak | nie | tak | tak | tak | tak | tak |
| Serwer STUN/TURN | nie | nie | nie | nie | nie | tak | tak |
| Serwer sygnałów | nie | nie | tak | tak | tak | tak | tak |
| Serwer aplikacji | tak | tak | tak | tak | tak | tak | tak |

Tabela 2. Porównanie narzędzi według funkcjonalności.

Analizując powyższą tabelę można zauważyć, że pod względem funkcjonalności najlepszym rozwiązaniem będzie OpenTok. Spełnia on wszystkie założenia. Kolejnymi wartymi uwagi projektami są EasyRTC, WebRTC.io i Peer.js. Wymagają one jednak zewnętrznych serwerów STUN i TURN do obsługi.

### 5.4.2 Porównanie według kryteriów

Po analizie funkcjonalnej nadszedł czas na zestawienie według kryteriów z rozdziału 5.2. Pozwoli ono na wybór najciekawszych narzędzi do zaimplementowania aplikacji WebRTC.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SimpleWebRTC | WebRTC jQuery Plugin | Peer.js | EasyRTC | WebRTC.io | OpenTok | WebRTCCopy |
| Możliwości | 5 | 4 | 7 | 7 | 7 | 8 | 7 |
| Dokumentacja | Słaba | Słaba | Dobra | Dobra | Słaba | B. dobra | Słaba |
| Tutoriale | Brak | Są | Są | Są | Brak | Są | Brak |
| Przykłady | Jest | Jest | Są | Są | Jest | Jest | Jest |
| Czy płatny | Nie | Nie | Tak | Tak | Nie | Nie | Tak |
| Czy rozwijany | Tak | Nie | Tak | Tak | Nie | Tak | Nie |
| Łatwość instalacji i użycie | Trudna | Trudna | Łatwa | Łatwa | Trudna | Łatwa | Trudna |

Tabela 3. Porównanie narzędzi według kryteriów z rozdziału 5.2.

W celu uproszczenia porównania przyjęto następuje kryteria i wartości:

* Możliwości - jakie funkcjonalności dostarcza dane rozwiązanie. Wartość to ilość elementów, które spełnia (patrz tabela 2)
* Dokumentacja - dokumentacja techniczna, opis interfejsu, możliwości ocenione w skali słaba, dobra, bardzo dobra
* Tutoriale - czy są instrukcje krok po kroku
* Przykłady - czy są przykładowe aplikacje, które ukazują możliwości narzędzia
* Czy płatny - czy projekt jest darmowy czy komercyjny (wszystkie narzędzia obecne w porównaniu posiadają wersję płatną)
* Czy rozwijany - czy projekt jest nadal rozwijany. Przyjęto czas maksymalnie 3 miesięcy od ostatniej zmiany w repozytorium
* Łatwość instalacji i użycie - czy narzędzie się łatwo instaluje i z niego korzysta. Ma na to wpływ dokumentacja

## 5.5 Podsumowanie

Podczas analizy tabel 2 i 3 można zauważyć kilka interesujących wniosków. Projekty komercyjne mają dobrą dokumentację i ich twórcy dbają o ich rozwój. Autorzy projektów darmowych nie przywiązują wagi do dokumentacji. Udostępniają oni najczęściej prostą przykładową aplikację. Wszystkie porównywane narzędzia mają dostępny swój kod na serwisie GitHub, nawet te komercyjne.

Zestawienie w podrozdziale 5.4 pokazuje, że postawione na samym początku rozdziału kryteria najlepiej spełnia komercyjne narzędzie OpenTok. Dostarcza ono wszystkie założone funkcjonalności. Dokumentacja, strona i przykłady projektu są na dobrym poziomie. Kolejnymi wartymi uwagi narzędziami są EasyRTC oraz Peer.js. Wśród tych trzech wymienionych produktów warto wybrać jeden, na bazie którego przygotuje się gotową aplikację. Inne pozycje można potraktować jako ciekawostki. Można je użyć i zbudować na ich podstawie aplikacje, ale będzie to wymagało nakładu czasu i pracy, by spełniły one wszystkie postawione w rozdziale 5.1.1 kryteria.

# 

# 

# 6. Zakończenie

Tematem powyższej pracy magisterskiej jest “Porównanie istniejących rozwiązań bazujących na technologii WebRTC do stworzenia czatu tekstowego/audio/wideo”. Jak ona pokazała, standard Web Real Time Communication ma być docelowym rozwiązaniem, które pozwoli na łatwą komunikację, niezależnie od używanej platformy sprzętowej i systemowej. Czat tekstowy, czy konferencja audio/wideo są najłatwiejszymi sposobami przekazywania informacji. Służą one zwykłym ludziom, do utrzymywania przyjaźni i znajomości, niezależnie od wieku czy dzielących ich odległości.

W pracy skupiono się zgodnie z tematem na porównaniu istniejących rozwiązań. Tempo rozwoju nowych technologii jest dzisiaj bardzo duże. Ciągle powstają nowe projekty, które mają ułatwić człowiekowi codzienne życie. Wybór gotowego narzędzia, zamiast tworzenie go od podstaw, pozwala szybko stworzyć aplikację i wypuścić ją na rynek jako nowy produkt. Nie wymaga to dogłębnej znajomości standardu WebRTC, ale wystarczy do tego podstawowa wiedza o nim. Jednak przy takim modelu tworzeniu oprogramowania trzeba poświęcić czas na przygotowanie kryteriów zgodnych z projektem, a następnie analizę i wybór odpowiedniego narzędzia, który będzie je spełniał. W rozdziale numer 5 przedstawiono porównanie siedmiu najciekawszych według autora produktów. Jest to najważniejszy element pracy, który został zrealizowany z powodzeniem. Na końcu zestawienia wybrano rozwiązanie, które spełnia ustalone na początku projektu założenia.

Podczas tworzenia pracy magisterskiej napotkano kilka trudności. Jednym z nich była podstawowa znajomość języka JavaScript i sposobu działania sieci internetowych. Jednak czas poświęcony na jej zdobywanie pozwolił na poznanie tych elementów i poszerzenie wiedzy związanej z nimi. Analizowanie funkcjonalności narzędzi, które następnie porównano, wzbogaciło autora o odkrycie technologii Node.js stającej się z dnia na dzień coraz bardziej popularną i wykorzystywaną w rozwiązaniach serwerowych.

Pomimo olbrzymich możliwości technologii WebRTC rozdział trzeci ujawnił, że standard ten nie jest jeszcze dojrzały i gotowy. Niepełne wsparcie przez Google Chrome, Mozilla Firefox czy Opera, a nawet całkowite jego ignorowanie w systemowych przeglądarkach Internet Explorer i Safari, nie pozwalają na użycie go jako uniwersalnego rozwiązania do komunikacji. Przed jego twórcami stoi jeszcze dużo pracy w popularyzacji tego rozwiązania wśród producentów oprogramowania dla komputerów stacjonarnych i urządzeń mobilnych.

Układ rozdziałów w pracy ukazuje tok pracy autora. Na samym początku zastanowiono się, czym jest komunikacja, jak ona wygląda teraz, a jak wyglądała kiedyś. Następnie zdobyto podstawową wiedzę na temat technologii WebRTC. Pozwoliło to na określenie celu, zakresu i tematu pracy. Kolejny drugi rozdział to zbiór możliwości, na które pozwala omawiany standard. Autor po tym rozdziale ocenił, czy jest możliwość zrealizowania postawionego celu w temacie pracy. Rozdział trzeci to etap poznawania sposobu działania WebRTC, użytych technologii i standardów. Pozwoliło to na określenie, w jaki sposób trzeba zaprojektować przykładową aplikację (rozdział 5.1), jakie ona może mieć funkcjonalności i jak przygotować kryteria dla porównania narzędzi. Kolejny czwarty element to krótkie omówienie tematu prywatności. Ostatnim etapem pracy było przygotowanie projektu aplikacji i wybór odpowiedniego narzędzia. Była to baza dla kolejnych prac, w tym określeniem kryteriów porównawczych. Następnie przygotowano listę kilku najciekawszych rozwiązań. Każde z nim było rozpoznawane, instalowane i sprawdzane pod kątem możliwości spełniania zdefiniowanych wcześniej założeń. Skutkowało to ich porównaniem i wyłonieniem najlepszych dla projektu narzędzi.

Powyższą pracę magisterską można określić jako pierwszy etap w przygotowaniu czatu dla użytkowników. Po określeniu rozwiązań, na bazie których może być on stworzony, można przystąpić do kolejnego etapu prac, jakim jest implementacja gotowego produktu. Będąc w tym miejscu można się jeszcze zdecydować, czy zbudować projekt na bazie istniejących rozwiązań, czy zdecydować się na samodzielne tworzenie całego systemu (w przypadku, gdy nie udało się wyłonić odpowiedniego narzędzia). Dzięki rozdziałowi trzeciemu posiada się wymaganą wiedzę, jaka jest niezbędna do stworzenia gotowego produktu niezależnie od wybranej metody.

# Bibliografia

1. “Podstawy Komunikowania Społecznego”, Bogusława Dobek-Ostrowska, Wrocław 1999, Wydawnictwo ASTRUM
2. Artykuł online “Shannon and Weaver Model of Communication” (stan na 2014-07-21) http://communicationtheory.org/shannon-and-weaver-model-of-communication/
3. Artykuł online “Chrome and Firefox now talking to each other through WebRTC chats” (stan na 2014-06-22) http://www.engadget.com/2013/02/04/chrome-and-firefox-now-talking-to-each-other-through-webrtc-chat/
4. Artykuł online “Google Closes On2 Technologies Acquisition” (stan na 2014-06-26) http://investor.google.com/releases/2010/0219.html
5. Artykuł online “Google buys GIPS for $68 million - to take on Skype? Apple? Microsoft?” (stan na 2014-07-22) http://www.disruptivetelephony.com/2010/05/google-buys-gips-for-68-million---to-take-on-skype-apple-microsoft.html
6. Nota na stronie projektu WebRTC (stan na 2014-05-22) http://www.webrtc.org/blog/introducingwebrtc-anopenreal-timecommunicationsproject
7. Szkic standardu WebRTC na stronie organizacji W3C (stan na dzień 2014-07-13) http://dev.w3.org/2011/webrtc/editor/webrtc-20110823.html
8. “WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web Second”, Edition, Johnston, Alan B. & Burnett, Daniel C., Digital Codex LLC, Czerwiec 2013
9. Specyfikacja przechwytywania strumieni w serwisie W3C (stan na 2014-05-22) http://dev.w3.org/2011/webrtc/editor/getusermedia.html
10. Scenariusze przechwytywania strumieni w W3C (stan na 2014-07-02) https://dvcs.w3.org/hg/dap/raw-file/tip/media-stream-capture/scenarios.html#capturing-a-media-stream
11. Przykłady modyfikacji strumieni w W3C (stan na 2014-06-25) https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/webrtc-integration.html
12. Wykaz przeglądarek Internet Explorer w danych wersjach systemu Microsoft Windows (stan na 2014-06-25) http://windows.microsoft.com/en-us/internet-explorer/which-version-am-i-using#ie=other
13. Opis interfejsy metody getUserMedia na portalu dokumentacji Mozilli (stan na 2014-05-22) https://developer.mozilla.org/en-US/docs/Web/API/Navigator.getUserMedia
14. Artykuł “U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program” (stan na 2014-05-22) http://www.washingtonpost.com/www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497\_story.html
15. Artykuł “PRISM scandal: tech giants flatly deny allowing NSA direct access to servers” (stan na 2014-06-13) http://www.theguardian.com/world/2013/jun/07/prism-tech-giants-shock-nsa-data-mining
16. “Network Security with OpenSSL”, Pravir Chandra, Matt Messier, John Viega; 2002, O'Reilly
17. “Hacking Exposed 6: Network Security Secrets & Solutions”, Stuart McClure, Joel Scambray George Kurtz, 2009, The McGraw-Hill Companies
18. “Real-Time Communication with WebRTC”, Salvatore Loreto, Simon Pietro Romano, 2014, O'Reilly Media

# 

# 

# Źródła rysunków

1. Z http://communicationtheory.org/shannon-and-weaver-model-of-communication/
2. Z http://www.engadget.com/2013/02/04/chrome-and-firefox-now-talking-to-each-other-through-webrtc-chat/
3. Z bibliografia pozycja nr 8, rozdział 1.1.4
4. Zrzut pobrany z http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/
5. Zrzut własny z aplikacji ASCII Camera
6. Zrzut własny z gry CubeSlam
7. Zrzut własny z aplikacji ShareFest
8. Z bibliografii, pozycja nr 8, rozdział 4.2.1
9. Z bibliografii, pozycja nr 8, rozdział 1.1.4
10. Z bibliografii, pozycja nr 8, rozdział 3.2.3
11. Z bibliografii, pozycja nr 8, rozdział 3.2.1
12. Zrzut własny aplikacji AppSpot
13. Z http://www.sourcetrunk.com/node/165
14. Zrzut własny z aplikacji WebRTC jQuery plugin
15. Zrzut własny z aplikacji Peer.js
16. Zrzut własny z aplikacji demo EasyRTC
17. Zrzut własny z przykładowej aplikacji WebRTC.io
18. Zrzut własny z aplikacji demo OpenTok
19. Zrzut własny z aplikacji WebRTCCopy