Agnieszka CHODOREK, Robert R. CHODOREK

TRANSMISJA DANYCH NIEMEDIALNYCH Z WYKORZYSTANIEM WEBRTC

Architektura WebRTC umożliwia budowę połączeń konferencyjnych, którymi przesyłane są dane medialne (dźwięk z mikrofonu, obraz z kamery). Umożliwia ona również przesyłanie danych niemedialnych (pliki, obrazy, wykresy, ale również dane pomiarowe - np. pochodzące z Internetu Rzeczy). Dane niemedialne w systemach zgodnych z architekturą WebRTC przesyłane są równocześnie z danymi medialnymi, a transmisja odbywa się osobnym kanałem danych. W artykule przedstawione zostały istotne elementy techniki WebRTC wykorzystywane do transmisji danych niemedialnych. Przedstawiono sposób zarządzania połączeniem dla transmisji danych oraz procedury stosowane do wymiany danych. Opisane zostały sposoby reprezentacji danych niemedialnych dostępne w technice WebRTC. Przedstawione zostały również podstawowe elementy sygnalizacji, wykorzystywane podczas tworzenia sesji WebRTC dla transmisji danych niemedialnych.

WSTĘP

WebRTC (ang. *Web Real-Time Communications*) [1][2] to zarówno architektura systemów konferencyjnych, konkurencyjna do aktualnie stosowanej architektury opartej na protokole SIP (ang. *Session Initiation Protocol*), jak również technika umożliwiająca budowę systemów zgodnych z tą architekturą. Technika WebRTC wykorzystuje język skryptowy JavaScript oraz język opisu stron WWW HTML5. W systemach wykorzystujących WebRTC przeglądarka internetowa pełni rolę interfejsu użytkownika.

W ogólnym ujęciu, architektura WebRTC wspiera systemy telekonferencyjne o charakterze zdecentralizowanym, gdzie konferencja jest realizowana za pomocą bezpośrednich połączeń p2p (ang. *peer-to-peer*) pomiędzy terminalami WebRTC (przeglądarkami). Z użyciem tej architektury można jednak realizować również systemy scentralizowane, korzystające z mostka konferencyjnego lub z serwerów mediów pełniących funkcję mostka (np. [3]).

Systemy konferencyjne przesyłają głównie strumienie medialne (audio i wideo), pochodzące z lokalnych przetworników dźwięku i obrazu (mikrofon, kamera internetowa). W wielu przypadkach pożądane byłoby, aby system konferencyjny oprócz transmisji audio i wideo mógł realizować transmisję dodatkowych danych, jak np. pliki graficzne, dokumenty, podawane na bieżąco wyniki pomiarów. W technice WebRTC istnieje możliwość zestawiania połączeń, nie tylko do transmisji informacji multimedialnych w czasie rzeczywistym, ale również do transmisji danych pochodzących z innych źródeł niż kamera i mikrofon. Dane te przesyłane są równocześnie z nie wymagają zazwyczaj rygorów czasu rzeczywistego lub rygory te są mocno rozluźnione w porównaniu z transmisją audio/wideo.

Takie rozwiązanie można wykorzystać np. do realizacji chatu skojarzonego z aplikacją ale również do przesyłania danych z urządzeń diagnostycznych wspierających Internet Rzeczy (ang. *Internet of Things*, IoT). Dyskusję na temat zastosowania w samochodach Internetu Rzeczy i komunikacji realizowanej w czasie rzeczywistym można znaleźć np. w [4].

Artykuł koncentruje się na sposobie przekazywania danych niemedialnych (transmisji prostych przekazów tekstowych, plików jak i danych z Internetu Rzeczy) pomiędzy systemami zbudowanymi z wykorzystaniem WebRTC. Rozdział pierwszy omawia funkcję `RTCPeerConnection()`, zastosowaną w kontekście transmisji danych niemedialnych oraz sposób przekazywania danych niemedialnych pomiędzy przeglądarkami. Rozdział drugi, na przykładzie systemu zdecentralizowanego i scentralizowanego (korzystającego z serwera mediów Kurento jako mostka konferencyjnego), omawia sygnalizację realizowaną podczas ustanawiania sesji WebRTC na potrzeby transmisji danych niemedialnych.

1. ŁĄCZENIE TERMINALI WEBRTC

Technika WebRTC umożliwia realizację bezpośredniego połączenia p2p pomiędzy terminalami WebRTC. Bardzo wiele terminali WebRTC, pracujących w sieciach stacjonarnych (firmowych lub domowych) lub też korzystających z terminali mobilnych dołączonych do Internetu np. za pomocą techniki LTE, jest dołączonych do sieci Internet poprzez NAT (ang. *Network Address Translation*) lub/i firewall. Dlatego też integralną częścią techniki WebRTC są funkcje umożliwiające zestawienie połączenia dla terminali znajdujących się za NAT-em lub/i firewallem. Wymaga to wbudowania w terminale klienta protokołu ICE (ang. *Interactive Connectivity Establishment*) [5]. Protokół ICE wykorzystuje protokół STUN (ang. *Session Traversal Utilities for NAT*) [6] i protokół TURN (ang. *Traversal Using Relay NAT*) [7].

Połączenie konferencyjne w modelu zdecentralizowanym wymaga utworzenia połączeń każdy-z-każdym pomiędzy terminalami. Połączenie pomiędzy dwoma terminalami WebRTC jest zestawiane przy wykorzystaniu klasy `RTCPeerConnection`. Klasa ta posiada zdefiniowane metody do nawiązywania połączenia, zarządzania połączeniem w trakcie jego trwania i kończenia połączenia.

Klasa `RTCPeerConnection` tworząc nowe połączenie przygotowuje transmisję dla terminali znajdujących się za NAT-em lub/i firewallem. Podczas ustanawiania nowego połączenia, do konstruktora klasy `RTCPeerConnection` podawane są parametry konfiguracyjne. Jednym z istotniejszych parametrów jest wskazanie serwerów ICE dla agenta ICE. Serwery ICE to serwery STUN i/lub TURN. Jeżeli nie zostanie określona lista serwerów, to agent ICE kontaktuje się z predefiniowaną listą serwerów (obecnie lista ta dla przeglądarek w wersjach stabilnych nie posiadają predefiniowanych serwerów ICE i zawsze trzeba podać choć jeden serwer w konfiguracji).

WebRTC wymaga, aby połączenie WebRTC było szyfrowane. Dotyczy to zarówno transmisji mediów, danych niemedialnych, jak i sygnalizacji. Certyfikaty do uwierzytelniana tworzone są podczas ustanawiania połączenia pomiędzy terminalami WebRTC.

Podstawowe metody klasy `RTCPeerConnection` przedstawiono w tabeli 1.

Tab. 1. *Podstawowe metody klasy* `RTCPeerConnection`

Nazwa metody	Opis
`addIceCandidate()`	metoda wywoływana gdy dodawany jest nowy kandydat do ICE
`addTrack()`	metoda dodaje nową ścieżkę (w strumieniu medialnym jest to np. nowy kanał audio) do połączenia
`close()`	metoda zamyka bieżące połączenie
`createAnswer()`	metoda tworzy odpowiedź SDP
`createOffer()`	metoda tworzy ofertę SDP
`generateCertificate()`	metoda tworzy certyfikat X.509
`getLocalStreams()`	metoda zwraca tablicę mediów skojarzonych z lokalnym systemem WebRTC
`getRemoteStreams()`	metoda zwraca tablicę mediów skojarzonych ze zdalnym systemem WebRTC
`getStreamById()`	metoda zwraca strumień (lokalny lub zdalny) o określonym identyfikatorze
`removeStream()`	metoda usuwa strumień mediów z oferty; gdy negocjacje są zakończone konieczna jest renegocjacja
`removeTrack()`	metoda usuwa ścieżkę z aktualnie nadawanych
`setConfiguration()`	metoda ustawia konfigurację na nowo zdefiniowaną
`setLocalDescription()`	metoda zmienia lokalny opis sesji skojarzony z połączeniem
`setRemoteDescription()`	metoda zmienia zdalny opis sesji skojarzony z połączeniem

W technice WebRTC połączenie jest w jednym ze stanów, zdefiniowanych przez typ `RTCPeerConnectionState`. Rozpoznawane są następujące stany: `new` (stan nowości), `connecting` (stan łączenia), `connected` (stan połączenia), `disconnected` (stan rozłączenia), `failed` (stan niezdatności), `close` (stan zamknięcia). W klasie `RTCPeerConnection` zdefiniowane są stany dla: połączenia sygnalizacyjnego, połączenia wykorzystywanego do transmisji mediów i danych, pozyskiwanych serwerów ICE oraz stan połączenia ICE.

W ramach klasy `RTCPeerConnection` dostępne są interfejsy dla połączenia multimedialnego (w skład połączenia wchodzą wszystkie zdefiniowane ścieżki audio i wideo) i definicja połączenia dla danych niemedialnych [8]. Połączenia dla danych niemedialnych tworzone są za pomocą metody `createDataChannel()` klasy `RTCPeerConnection`. Metoda ta posiada obowiązkowy parametr – nazwę kanału transmisyjnego (w postaci łańcucha znaków). Do metody tej podawane są również opcje służące do określenia zachowania się kanału transmisji danych. Możliwe jest zdefiniowanie czy protokół do transmisji danych niemedialnych ma dostarczać dane w kolejności nadawania, czy też nie (zmiennej logicznej `ordered`). Określane są także: maksymalna liczba retransmisji (`maxRetransmits`), maksymalny czas transmisji pakietu w trybie bez zapewnienia niezawodności (`maxPacketLifeTime`), nazwa protokołu wykorzystywanego w ramach kanału do transmisji danych (`protocol`), czy odbywa się negocjacja (`negotiated`) i identyfikator połączenia (`id` – gdy jest on nieustawiony, to jest przydzielany automatycznie).

Na rys. 1 przedstawiony został skrypt definiujący połączenie WebRTC realizujący transmisję danych niemedialnych (na skrypcie pominięto definicję funkcji obsługujących błędy). Do konstruktora klasy `RTCPeerConnection` podano identyfikator serwera ICE (jest to serwer STUN) (rys. 1a). W terminalu WebRTC inicjującym połączenie (rys. 1a), w ramach już zdefiniowanego połączenia reprezentowanego przez zmienną `pol`, zdefiniowano kanał transmisji danych o nazwie „*mojkanal*". Innych parametrów konfiguracyjnych nie określono.

a)

```
var serwery = {'iceServers': [{'url':
'stun:stun.l.google.com:19302'}]};
var konfiguracja = null;
var pol = new RTCPeerConnection(serwery,
              konfiguracja);
var dane = pol.createDataChannel("mojkanal");
dane.onopen = function () {
  console.log("kanał danych otwarty"); };
dane.onmessage = function (event) {
  console.log("odebrane: " + event.data); };
dane.onclose = function () {
  console.log("kanał danych zamknięty"); };
```

b)

```
var serwery = {'iceServers': [{'url':
'stun:stun.l.google.com:19302'}]};
var konfiguracja = null;
var polodb = new RTCPeerConnection(serwery,
                 konfiguracja);
polodb.ondatachannel = odbior;
function odbior(event) {
    kanalOdb = event.channel;
    kanalOdb.onmessage = odbior;
    kanalOdb.onopen  = otwarty;
    kanalOdb.onclose = zamkniety; }
function otwarty() {
  console.log("kanał danych otwarty"); };
function odbior(event) {
  console.log("odebrane: " + event.data); };
function zamkniety() {
  console.log("kanał danych zamknięty"); };
```

Rys. 1. *Przykładowy skrypt definiujący kanał do transmisji danych niemedialnych: a) skrypt uruchamiany w terminalu WebRTC inicjującym połącznie, b) skrypt uruchamiany w terminalu WebRTC dołączającym się do już utworzonego połączenia.*

Dla utworzonego połączenia danych należy określić funkcje obsługujące trzy podstawowe zdarzenia: nawiązanie połączenia (`onopen`), przesłanie wiadomości (`onmessage`) i zakończenia połączenia (`onclose`). W przykładzie przedstawionym na rys. 1a funkcje te wypisują na konsoli przeglądarki komunikaty o otwarciu bądź zamknięciu połączenia transmisji danych niemedialnych. Dane niemedialne, otrzymane w ramach transmisji, również wyświetlane będą na konsoli (typowo są one jeszcze gdzieś przekazywane).

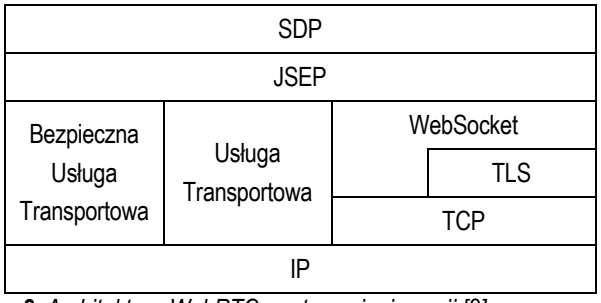
Dla terminala WebRTC dołączającego się do już zainicjowanego połączenia (rys. 1b), po zainicjowaniu klasy `RTCPeerConnection` reprezentowanej przez zmienną `polodb`, w odróżnieniu od przykładu z rys. 1a, nie jest tworzony jawnie kanał transmisji danych, lecz pobierany jest jego identyfikator z połączenia sesji WebRTC. Dla transmitowanych danych niemedialnych określany jest typ obiektu języka JavaScript (zmienna `binaryType`), który służy do reprezentacji odbieranych danych binarnych. Standardowo jest to obiekt `blob` [9], reprezentujący dane binarne. Jest on stosowany do transmisji plików. Drugim zdefiniowanym typem reprezentacji danych binarnych jest `arraybuffer` (bufor w pamięci). Dostęp do bufora `arraybuffer` może być bardzo uproszczone poprzez zastosowanie widoku (ang. *view*) danych języka JavaScript. Dla bufora `arraybuffer` należy utworzyć klasę `ArrayBufferView`, która pozwala na dostęp do bufora `arraybuffer` zgodnie z predefiniowanym przez użytkownika typem danych (np. typu integer o rozmiarze 32 bitów).

Możliwy jest również odczyt rozmiaru bufora wykorzystywanego do transmisji danych (zmienna `bufferedAmount`) i określanie wartości (zmienna `bufferedAmoun-`

tLowThreshold), przy której generowane jest ostrzeżenie o zbyt małej wielkości bufora.

2. USTANAWIANIE SESJI WEBRTC NA POTRZEBY TRANSMISJI DANYCH NIEMEDIALNYCH

Ustanawianie sesji WebRTC realizowane jest zgodnie z architekturą zobrazowaną na rysunku 2. Opis sesji, sporządzony przez system końcowy w formacie protokołu SDP (ang. *Session Description Protocol*) [10] jest przesyłany za pomocą protokołu sygnalizacyjnego JSEP (ang. *JavaScript Session Establishment Protocol*), korzystającego (poprzez interfejs WebSockets) w warstwie transportowej z protokołu TCP (ang. *Transmission Control Protocol*). Jak widać na rysunku, protokół JSEP do transmisji komunikatów SDP może użyć także "bezpiecznego" interfejsu gniazd WebSockets (ang. *WebSockets Security*, WSS), gdzie transmisja TCP podlega ochronie kryptograficznej z wykorzystaniem protokołu TLS (ang. *Transport Layer Security*). JSEP może również skorzystać z dowolnej innej usługi transportowej lub bezpiecznej usługi transportowej. Wymiana opisu sesji pomiędzy przeglądarkami realizowana jest na zasadzie oferta-odpowiedź. Przeglądarka, która jest dysponentem strumienia danych, wysyła do drugiego systemu końcowego informacje o swojej ofercie medialnej. Drugi system końcowy analizuje ofertę i przesyła komunikat zwrotny (odpowiedź SDP).



Rys. 2. *Architektura WebRTC - ustanawianie sesji [2].*

Opis SDP pozwala przeglądarkom uzgodnić zarówno parametry sesji jako całości, jak i parametry przesyłanych danych (multimedialnych i niemedialnych). Jest on opisem dwupoziomowym - osobno opisywany jest poziom sesji, osobno poziom mediów [10]. Poziom sesji jest poziomem ogólnym, poziom mediów poziomem szczegółowym. Niektóre informacje poziomu mediów, jeżeli dotyczą całej sesji (wszystkich przesyłanych strumieni), mogą zostać zamieszczone na poziomie sesji. W architekturze WebRTC, opis SDP został formalnie podzielony na pięć komponentów semantycznych [11][12]: komponent metadanych sesyjnych, komponent opisu sieci, komponent opisu strumienia, komponent opisu bezpieczeństwa, komponent opisów QoS (ang. *Quality of Service*) i grupowania. Pierwszy komponent odpowiada poziomowi sesji, kolejne są wykorzystywane na poziom mediów.

Rysunek 3 przedstawia wymianę komunikatów oferty SDP i odpowiedzi SDP pomiędzy dwiema przeglądarkami Google Chrome (rys. 3a,b) oraz pomiędzy przeglądarką Google Chrome a serwerem mediów Kurento (rys. 3a,c) pełniącym rolę mostka konferencyjnego. Ponieważ w obu sesjach WebRTC użyto tej samej przeglądarki (tego samego systemu komputerowego), oferta SDP nie zmieniła się. Zaprezentowane na rysunku 3 komunikaty oferty i odpowiedzi zostały ograniczone tylko do poziomu sesji i fragmentu poziomu mediów odpowiadającego za opis strumieni danych niemedialnych. Opis strumieni danych medialnych (audio, wideo) nie został pokazany na rysunku 3.

Opis SDP składa się z linii tekstu, zawierające pola komunikatu. Linię rozpoczyna nazwa pola. Separatorem pomiędzy nazwą pola a wartością pola komunikatu SDP jest znak równości. Pola opisu sesji znajdują się na początku komunikatu. Kończą się one wraz z pierwszym polem opisu mediów (polem m).

a)

```
v=0
o=- 402520935775111953 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE data
a=msid-semantic: WMS
m=application 9 DTLS/SCTP 5000
c=IN IP4 0.0.0.0
a=ice-ufrag:JL1x
a=ice-pwd:gciTnjEDDKPvfeudI9Q79E7w
 a=fingerprint:sha-256
65:26:F0:60:3F:32:59:44:E0:D2:5A:21:3B:78:09:47:
EE:5F:09:06:E2:E0:7B:70:9E:65:85:ED:5F:04:93:6C
a=setup:actpass
a=mid:data
a=sctpmap:5000 webrtc-datachannel 1024
```

b)

```
v=0
o=- 699559329071777764 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE data
a=msid-semantic: WMS
m=application 9 DTLS/SCTP 5000
c=IN IP4 0.0.0.0
b=AS:30
a=ice-ufrag:cSGz
a=ice-pwd:+Gi9tybY5hAdR6UMWS4wYm3s
 a=fingerprint:sha-256
BF:8B:BC:B0:80:22:53:31:AE:BF:68:91:DF:DF:FC:5E:
D3:A7:75:4A:EF:33:BB:71:CF:8C:5A:47:28:7D:EB:80
a=setup:active
a=mid:data
a=sctpmap:5000 webrtc-datachannel 1024
```

c)

```
v=0
o=- 3699787296 3699787296 IN IP4 0.0.0.0
s=Kurento Media Server
c=IN IP4 0.0.0.0
t=0 0
a=msid-semantic: WMS
ZvftCX3GcYTKSR9ZR272u9OhYzj8y8J44PYb
a=group:BUNDLE data
m=application 1 DTLS/SCTP 5000
a=mid:data
a=sctpmap:5000 webrtc-datachannel 1024
a=setup:active
a=ice-ufrag:Dkny
a=ice-pwd:V+PYazg2ZvN2xCVqhKU7KC
a=fingerprint:sha-256
DF:6B:FD:D6:30:E3:F2:10:39:DA:E2:07:E0:08:43:52:
BD:F4:88:67:DC:AF:6C:82:ED:3B:FB:34:46:DE:84:BB
```

Rys. 3. *Opis SDP: a) oferta wysłana przez przeglądarkę, b) odpowiedź na ofertę, wysłana przez przeglądarkę, c) odpowiedź na ofertę, wysłana przez serwer Kurento.*

Opis poziomu sesji zawiera pola v, o, s, t i a. Zarówno w komunikacie oferty (rys. 3a), jak i w obu komunikatach odpowiedzi (rys. 3b,c) pole wersji SDP (v) przyjmuje wartość zero, co jest zgodne ze specyfikacją protokołu SDP (wersja zerowa protokołu). W przypadku pól o i s opisu sesji można zaobserwować pewne odstępstwa od ustalonego formatu. Format pola o (informującego o pochodzeniu sesji) jest częściowo ignorowany. Zarówno przeglądarka Google Chrome, jak i serwer Kurento nie przedstawiają użytkownika inicjującego sesję (myślnik). Ignorują również unikastowy adres IP komputera inicjującego sesję (przeglądarka Chrome wstawiła adres interfejsu loopback, serwer Kurento wstawił adres zawie-

rający same zera). Pole `s`, zgodnie ze specyfikacją protokołu SDP, zawiera nazwę sesji. Zostało ono niewypełnione przez przeglądarkę (myślnik), a serwer Kurento w tym polu przedstawił siebie.

Kolejnym polem poziomu sesji jest pole `t`, definiujące ograniczenia czasowe sesji. Wszystkie analizowane systemy końcowe ustawiły nieograniczony czas trwania sesji (pole `t`), zgodnie z zaleceniem podanym w RFC 3264 [13].

Dwa ostatnie pola to pola `a`, przenoszące dodatkowe atrybuty (dodatkową informację sesyjną). Na poziomie sesji przenoszona jest informacja na temat grupowania strumieni mediów z wykorzystaniem grupowania BUNDLE [14]. Na tym poziomie przenoszone są również identyfikatory strumieni mediów WMS (ang. *WebRTC Media Stream*). Przeglądarka Google Chrome nie podała identyfikatora (rys. 3a,b). Serwer Kurento – (rys. 3c) nadał identyfikator.

Pole c poziomu opisu mediów zostało przez serwer Kurento umieszczone w poziomie opisu sesji, jednak jego zawartość jest taka sama w przypadku serwera i przeglądarki Google Chrome. Pole `c` zawiera informacje dotyczące połączenia, tj. typ sieci (tu: `IN` - Internet), używana wersja protokołu IP (tu: `IP4` - IPv4) oraz adres połączenia. Analizowane systemy końcowe nie podały adresu połączenia (wyzerowały bity adresu IPv4).

Opis strumienia mediów zaczyna się od pola `m`. Fragmenty komunikatów SDP z rysunku 3 zawierają tylko informację o transmisji danych aplikacji niemedialnej - wartość pola `m` zaczyna się od słowa kluczowego `application`. Pole `m` wskazuje, że do transmisji danych użyty zostanie protokół SCTP, a do ochrony kryptograficznej danych - protokół TLS. Użytkownik wysyłający ofertę inicjuje połączenie SCTP (atrybut setup pola `a`) w sposób pasywny, a odbierający ofertę (i wysyłający odpowiedź) - w sposób aktywny. Atrybut `mid` pola a definiuje identyfikator strumienia w grupie (tu: `data`). Atrybut `sctpmap` pola a definiuje odwzorowanie (ang. *map*) numeru portu (zdefiniowanego w polu `m`) na nazwę aplikacji przekazującej dane i jednocześnie wskazującą na format zawartości strumienia (tu: `webrtc-datachannel`). Ostatni parametr (tu: `1024`) to rozmiar wiadomości. Pole `b` atrybutem `AS` jest używane do wskazania przepływności danego strumienia. Wartością domyślną atrybutu `AS` jest 30. Atrybuty `ice-ufrag`, `ice-pwd` oraz `fingerprint` pola `a` należą do komponentu opisu bezpieczeństwa. Dwa pierwsze dotyczą uwierzytelniania podczas współpracy z protokołem STUN. Atrybut `fingerprint`, z kolei, przenosi informację o użytej funkcji skrótu (funkcji haszującej) oraz skrót powstały w wyniku działania tej funkcji

PODSUMOWANIE

Technika WebRTC umożliwia stosunkowo prostą realizację usługi konferencyjnej przy zastosowaniu komputerów PC (stacjonarnych jak i notebooków) czy smartfonów jako terminali. Oznacza to znaczne ograniczenie kosztów budowy sytemu konferencyjnego, w tym również systemu na potrzeby logistyki.

Wykorzystanie kanału danych niemedialnych, zdefiniowanego w standardzie WebRTC, pozwala na przesyłanie danych istotnych dla wielu procesów logistycznych. Mogą to być dane zarówno w postaci plików, jak i dane pochodzące z pomiarów czy monitoringu, przeprowadzanych w czasie rzeczywistym.

Przedstawione w artykule elementy techniki WebRTC wykorzystywane do transmisji danych niemedialnych są łatwe do wdrożenia. Wystarczy w tym celu dodać do kodu strony WWW odpowiednio przygotowany skrypt w języku JavaScript i uruchomić go w przeglądarce WWW. Pełne wsparcie dla transmisji danych posiadają popularne przeglądarki jak Chrome, Firefox czy Opera. W przypadku bardziej złożonych rozwiązań można system konferencyjny uzupełnić o bezpłatny serwer Kurento, który będzie spełniał rolę mostka konferencyjnego.

BIBLIOGRAFIA

1. Loreto S., Romano S.P., Real-Time Communication with WebRTC: Peer-to-Peer in the Browser, O'Reilly Media, Inc. 2014.
2. Chodorek A., Chodorek R. R., Model warstwowy ustana-wiania sesji WebRTC. Studia Informatica, 37 (2) : 117-126, 2016.
3. http://doc-kurento.readthedocs.io/en/stable/introducing_kurento.html
4. Viscusi S., Connected Cars & RTC Making Travel Safer. http://www.realtimecommunicationsworld.com/topics/realtimecommunicationsworld/articles/408237-connected-cars-rtc-making-travel-safer.htm
5. Rosenberg J., Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, 2010.
6. Rosenberg J., Mahy R., Matthews P., Wing D., Session Traversal Utilities for NAT (STUN). RFC 5389, 2008.
7. Mahy R., Matthews P., Rosenberg J., Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)., RFC 5766, 2010.
8. WebRTC 1.0: Real-time Communication Between Browsers. W3C Working Draft 13 March 2017 http://www.w3.org/TR/webrtc/
9. https://developer.mozilla.org/en-US/docs/Web/API/Blob
10. Handley M., Jacobson V., Perkins C., SDP: Session Description Protocol. RFC 4566, July 2006.
11. Nandakumar S., Jennings C., SDP for the WebRTC. Inter-net-Draft draft-ietf-rtcweb-sdp-04, 26 marca 2017.
12. Chodorek R. R., Chodorek A., Rzym G., Wajda K., Zastosowanie protokołu SDP w systemach multimedialnych realizowanych w technice WebRTC. Przegląd Telekomunikacyjny, Wiadomości Telekomunikacyjne, 6/2016 : 564-567, 2016.
13. Rosenberg J., Schulzrinne H., An Offer/Answer Model with the Session Description Protocol (SDP). RFC 3264, 2002.
14. Holmberg C., Alvestrand H., Jennings C., Negotiating Media Multiplexing Using the Session Description Protocol (SDP). Internet-Draft, draft-ietf-mmusic-sdp-bundle-negotiation-36.txt, 27 października 2016.

Transmission of the non-media data with the use of the WebRTC

This paper presents the most important elements of the WebRTC technology that are used for transmission of non-media data. Non media data (as files, pictures or results of measurements) are transmitted simultaneously with the media data (real-time transmission of multimedia information), although real-time constraints haven't to be preserved. The paper shows connection management and session setup for the purposes of transmission of non-media data.

Autorzy:
dr inż. **Agnieszka Chodorek** – Politechnika Świętokrzyska, Wydział Elektrotechniki, Automatyki i Informatyki, Katedra Systemów Informatycznych; 25-314 Kielce; al. Tysiąclecia Państwa Polskiego 7. E-mail: a.chodorek@tu.kielce.pl
dr inż. **Robert R. Chodorek** – AGH Akademia Górniczo-Hutnicza, Wydział Informatyki, Elektroniki i Telekomunikacji, Katedra Telekomunikacji; 30-059 Kraków; Al. A. Mickiewicza 30. E-mail: chodorek@agh.edu.pl