Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Bakalářská práce

Online konferenční systém založený na HTML5

Vypracoval: Jiří Dušek

Vedoucí práce: PhDr. Milan Novák, Ph.D.

České Budějovice 2016

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů

V Českých Budějovicích dne 2016

Jiří Dušek

Abstrakt

HTML5 technologie umožňují v prostředí prohlížeče přistupovat k zařízením jako je webkamera či mikrofon. Bakalářská práce se zabývá využitím těchto prostředků pro jejich přenos napříč internetovou sítí v reálném čase v prostředí webového prohlížeče.

Základem práce je analýza současných webově založených online konferenčních systémů. V rámci této analýzy budou zjišťovány jejich parametry a použité technologie pro přenos multimediálních dat. Získané informace poslouží dále pro návrh vytvářeného konferenčního systému. V návaznosti na tuto problematiku následuje detailní analýza možností HTML5 technologií pro přenos audio a video mezi klientskými zařízeními v reálném čase.

V rámci praktické části práce bude sestaven na základě vytvořené analýzy prototyp online konferenčního systému založeného na HTML5 technologiích. Ten umožní vytváření konferenčních místností, které budou zajišťovat audiovizuální komunikaci a poskytnou online chat mezi účastníky místnosti. Vytvoření systému zahrnuje jeho návrh, implementaci, následné nasazení a testování. Bude vytvořen s ohledem na přístupnost z mobilních zařízení.

Abstract

HTML5 technologies allow access to data provided by a webcam or microphone within a web browser. This thesis deals with the use of such technical means for their transmission through the Internet in a web browser environment in real time.

The base of bachelor thesis is analysis of current web-based conference systems. In this analysis their parameters will be ascertained as well as the technologies used for the transmission of multimedia data. Obtained information will be used to create a design of conference system. In connection with this issue follows a detailed analysis of the possibilities of HTML5 technologies for transmitting audio and video between client devices in real time.

In the practical part of the thesis will be created on the basis of the analysis a prototype online conference system based on HTML5 technologies. The system will allow the creation of conference rooms that will provide audiovisual communication and an online chat between participants of the room. Creating the system includes the design, implementation, subsequent deployment and testing. It will be created with regard to the accessibility of mobile devices.

Klíčová slova

- Webově založený konferenční systém
- HTML5
- WebRTC
- Peer-to-peer síť
- Audio a video streamování
- RTCPeerConnection
- MediStream

Keywords

- Web-based conference system
- HTML5
- WebRTC
- Peer-to-peer network
- Audio and video streaming
- RTCPeerConnection
- MediStream

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří DUŠEK**
Osobní číslo:	**P13098**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Online konferenční systém založený na HTML5**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Bakalářská práce se bude zabývat možnostmi online streamování videa a zvuku prostřednictvím technologie HTML5. Autor provede analýzu možností uvedené technologie v oblasti audiovizuálního přenosu s ohledem na přístupnost prostřednictvím mobilních zařízení. Na základě analýzy vytvoří funkční internetovou aplikaci, která bude umožňovat správu uživatelů a vysílacích místností.

Rozsah grafických prací:	**CD ROM**
Rozsah pracovní zprávy:	**40**
Forma zpracování bakalářské práce:	**tištěná**

Seznam odborné literatury:

1. **WELLING, Luke a Laura THOMSON. PHP and MySQL Web development. 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2008, c2009. Developer's library. ISBN 0672329166.**
2. **OZER, By Jan a Laura THOMSON. Producing streaming video for multiple screen delivery. 4th ed. Galax, Va: Doceo Publishing, 2013. Developer's library. ISBN 0976259540.**
3. **AUSTERBERRY, David a Laura THOMSON. The technology of video and audio streaming. 2nd ed. Burlington, MA: Focal Press, 2004. Developer's library. ISBN 0240805801.**

Vedoucí bakalářské práce:	**PhDr. Milan Novák, Ph.D.** Ústav aplikované informatiky
Datum zadání bakalářské práce:	**15. dubna 2015**
Termín odevzdání bakalářské práce:	**29. dubna 2016**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 15. dubna 2015

Poděkování

Rád bych tímto poděkoval svému vedoucímu práce PhDr. Milanu Novákovi, Ph.D. za vstřícnost, rady a věcné připomínky při vedení mé bakalářské práce. Poděkovat bych také chtěl své úžasné rodině a přátelům, kteří mě podporovali a měli se mnou trpělivost.

Obsah

1 Úvod

Práce se zabývá průnikem využití HTML5 technologií pro účely audio a video streamování[1] v internetové síti mezi klientskými zařízeními a zároveň možnostmi současných online webově založených konferenčních systémů[2]. Výsledkem práce je vytvořený prototyp takového systému založeného na HTML5 technologiích.

Tyto konferenční systémy umožňují komunikaci účastníků ve virtuálním prostředí. Jsou využívány společnostmi, firmami či jinými organizacemi, ale nacházejí uplatnění i v privátním sektoru. V rámci práce jsou zjišťovány jejich vlastnosti a funkce, pojetí samotné konference, ale i použité technologie.

HTML5 technologie vyvíjené pod záštitou konsorcia W3C[3] přináší možnost přístupu k periferním zařízením jako je mikron či webkamera přímo v prostředí internetového prohlížeče. A to bez nutnosti instalace zásuvných modulů. Získané audio či video lze následně pomocí těchto technologií distribuovat k jinému zařízení v internetové síti. V rámci práce je detailně rozebrána principiální funkce takového spojení i jeho reálná implementace.

1.1 Cíle práce

Cílem bakalářské práce je analyzovat současnou oblast webově založených konferenčních systémů, které umožňují komunikaci v reálném čase. Zjistit jejich celkové možnosti, pojetí samotné konference, implementované funkce a použité technologie. Následuje jejich srovnání a zjištění, zda-li využívají HTML5 technologie.

Dalším cílem je detailně analyzovat problematiku HTML5 technologií z pohledu přenosu audia a videa internetovou sítí v reálném čase. Vytvoření multimediálního spojení probíhá v několika fázích. V rámci analýzy je rozebrán princip funkce a jednotlivé etapy nutné pro realizaci streamování audio či video obsahu s ohledem na současné možnosti nasazení těchto technologií. Součástí je také objasnění využití periferních zařízení z aktuálního pohledu v prostředí internetového prohlížeče.

Cílem praktické části je vytvoření funkčního prototypu online konferenčního systému založeného na HTML5 technologiích. Tento systém by měl odpovídat konceptu analyzovaných konferenčních systémů. Primárním cílem je realizace skupinových video hovorů. Důraz je kladen na jeho přístupnost a použitelnost z mobilních zařízení.

1.2 Metody práce

Teoretická část je zpracovávána metodou rešerše a celkového souhrnu poznatků z dané problematiky. Analyzované konferenční systémy jsou vybírány na základě zvoleného přehledu systémů, který je zpracován některým z portálů zabývajících se danou oblastí. Každý systém je analyzován. Podle společných vlastností těchto systémů jsou stanovena kritéria, jež se zanesou společně s odpovídajícími údaji do příslušných tabulek. Výsledkem je porovnání, které přehledným způsobem reprezentuje nejdůležitější parametry těchto

[1] Vymezení pojmu v kapitole 2.4

[2] Vymezení pojmu v kapitole 2.1

[3] Konsorcium W3C (World Wide Web) je mezinárodní komunita, ve které její členské organizace, zaměstnanci a veřejnost společně pracují na webových standardech. [1]

systémů. Následně jsou definovány obecné vlastnosti konferenčního systému vyplývající z provedené analýzy.

Další částí je analýza vztahující se k HTML5 technologiím a jejich možnostem audio či video streamování. Je zaměřena na nasazení této technologie v praxi, aby na jejím základě bylo možné realizovat praktickou část. Pro objasnění celé problematiky a současného stavu je též využíváno dokumentů, jež specifikují samotné standardy analyzovaných technologií, ale i testování dílčích funkcí.

V praktické části jsou dle provedené analýzy sestaveny funkční a technické požadavky pro vytvářený systém. Následuje zpracování samotného návrhu. Ten definuje strukturu systému, popisuje procesy, jež je nutné implementovat a obsahuje potřebné UML[4] diagramy. Implementace a nasazení zahrnují celkovou realizaci systému včetně využití potřebného softwaru a práce s podpůrnými prostředky či zařízeními. Poslední fáze představuje testování realizovaného systému.

[4]The Unified Modeling Language - je standard sjednocující způsob vizualizace modelování obchodních či jiných procesů, návrhů a sytémů. [2]

2 Vymezení pojmů

2.1 Webově založený konferenční systém

Webová konference je způsob komunikace v reálném čase, při kterém je propojeno více uživatelských zařízení skrze internetovou síť v prostředí prohlížeče. Její uživatelé vidí stejný obsah. Obsahují funkce jako je zasílání textových zpráv či poskytují možnost multimediálních hovorů. [3]

Webové konference uživatelům umožňují uskutečňovat online setkání a semináře, obchodní schůzky, prezentace, demonstrativní ukázky, výuku či nabízet zákazníkům online podporu. Systémové požadavky na jejich používání nebývají velké. Většina osobních počítačů má dostatečné vybavení pro používání webových konferencí. Pokud je vyžadována instalace dodatečného softwaru, bývá jednoduchá. Některá řešení nabízí služby za nominální měsíční částku. Nejefektivnější konferenční systémy vyžadují vysokorychlostní internetové připojení všech zúčastněných. Řízení samotné schůzky je v rukou moderátora, ale často lze přidělit práva uživatelům tak, že i oni vystupují jako moderátoři. [3] Konferenčním systémem tohoto pojetí se zabývá i tato práce.

2.2 Webcast

Webcasting představuje používání internetu k vysílání živého nebo opožděného audia, videa, stejně jako tradiční televizní a radiové vysílání. Uživatelé musí mít vhodnou multimediální aplikaci, aby mohli sledovat webcast. Například, univerzita může nabídnout online kurz, ve kterém učitel vysílá předtočenou nebo živou výuku. [6]

2.3 Webinář

Pojem "webinář"vznikl spojením slov web(ový) a seminář. Jak již z těchto dvou slov je patrné, jedná se o seminář, jenž je uskutečňován v reálném čase pomocí webových technologií. Webinář je doplňkem/alternativou e-learningového vzdělávání. [5]

2.4 Streamování

Streamování je technologie používaná obvykle k doručení audio či video obsahu. Streamování je nejrychlejší způsob přístupu k obsahu na internetu, ale není ovšem jedinou možností. Progresivní download je další používanou variantou. Již celá léta předtím než bylo streamování vůbec možné. [4]

3 Analyzované konferenční systémy

Práce se zabývá využitím HTML5 technologií pro distribuci multimediálních dat skrze internetovou síť. Proto jsou analyzované systémy přístupné z prostředí prohlížeče bez nutnosti instalace vlastního obslužného softwaru konference, jenž by byl nutně vyžadován pro chod jejích hlavních funkcí či streamování. Nicméně některé mohou vyžadovat instalaci softwaru pro podporu dílčí funkcionality či technologie, kterou daný systém využívá.

Analyzované systémy jsou vybírány z výčtu hodnocených systémů[5] za rok 2015 technicky zaměřeného amerického portálu PC Magazín (pcmag.com). Přičemž kritériem výběru pro analýzu je způsob implementace daného systému, tedy zda-li odpovídá konceptu webově založeného konferenčního systému.

Název systému	Webově založené
ClickMeeting	✓
Join me	✗
Adobe Connect	✓
Cisco WebEx Meeting Center	✗
Citrix GoToMeeting	✗
eVoice	✗
Skype for Business	✗
Onstream Meetings	✓
StartMeeting	✗
InterCall	✗

Tabulka 1: *Konferenční systémy v pořadí dle hodnocení pro rok 2015 v PC magazínu [8]*

3.1 Adobe Connect

Adobe Connect patří mezi pokročilé webově založené konferenční systémy. Uživatel má přiřazenou vlastní subdoménu, na níž jsou přístupné jeho jednotlivé konferenční místnosti. Je multijazyčný a nabízí změnu časového pásma. Český jazyk není podporován, nicméně je nabízeno 12 dalších.

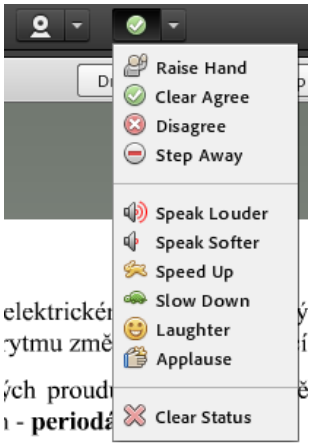
Místnosti jsou členěny takzvanými ”pody”neboli kapslemi, které reprezentují jednotlivé funkce. Jejich rozložení je zcela v rukách moderátora. Standardně je nabízeno několik předpřipravených šablon, které lze libovolně přepínat. Velikosti a pozice jsou přizpůsobitelné technikou drag & drop[6]. Šablony může uživatel též vytvářet vlastní tak, aby je v jiné místnosti mohl jednoduše použít.

Přehled připojených účastníků umožňuje v tomto systému zobrazovat i jejich stav. Každý účastník jej může libovolně měnit. Těchto stavů lze využít také například místo ankety, protože moderátor zná počty

[5] Přístupné na adrese http://www.pcmag.com/article2/0,2817,2388678,00.asp

[6] Jedná se o akci prováděnou v grafickém uživatelském rozhraní. Kurzorem je vybrán a poté tahem přemístěn určitý objekt na novou pozici. [10]

stavů uživatelů. Mezi nabízenými stavy jsou následující v mínění: souhlasím, nesouhlasím, zdržuji se, pomaleji, rychleji, mluvte hlasitěji, mluvte jemněji, smích, potlest, prázdný stav a zvednutí ruky například pro přihlášení se o slovo.



Obrázek 1: Adobe Connect - volba nastavení stavu

3.2 Click Meeting

Systém Click Meeting poskytuje propracované volby editace jednotlivých částí souvisejících s konferenční místností i s ní samotnou. Příchodu do místnosti předchází stránka konané události, ve které budoucí účastník zadává vstupní údaje. Správa zobrazení události umožňuje dynamické přidávání vlastních formulářových prvků. Následně je účastník přesunut do tzv. ”Waiting room”, tedy čekárny, kterou organizátor může též libovolně upravit stejně jako zobrazení samotné místnosti či ukončovací stránky.

Click Meeting nabízí kromě jiných i multimediální funkce jako například možnost vyhledávání a přehrávání videí ze serveru Youtube ve streamovaném prostoru konference. Do těchto videí také lze kreslit a zvýraznit určité oblasti v reálném čase.

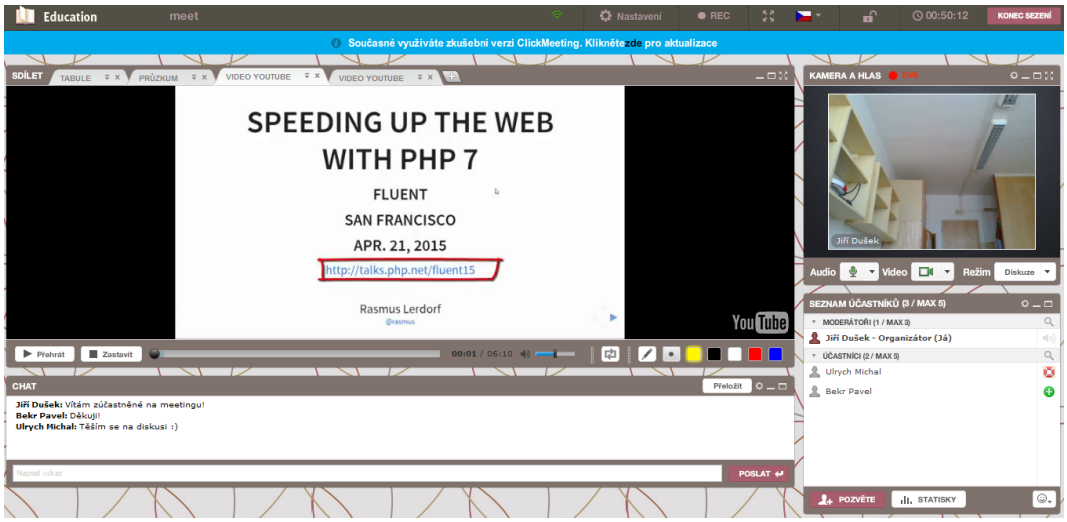
Systém disponuje speciální volbu nastavení vlastních režimů konference. Jednotlivé režimy: ”diskuse”- všichni mohou komunikovat se všemi, ”moderátoři”- pouze oni se slyší a vidí, ”pouze k poslechu”- všichni účastníci jsou ztlumeni, ”otázky a odpovědi”- účastníci kladou moderátorům otázky.

3.3 Onstream Meetings

Onstream Meetings představuje robustní webovou službou realizující pokročilé konference. Nabízí komplexní možnosti od online steamování, chatu či sdílení multimediálních souborů, až po sdílení pracovní plochy, záznam přenášeného videa či sdílenou kreslící tabuli a další funkce.

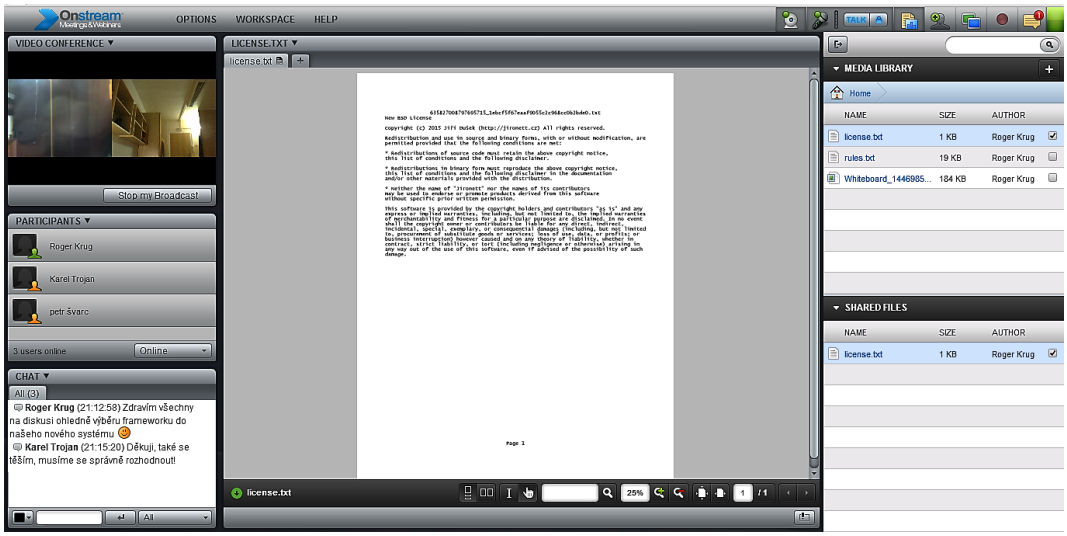
Rozvinutou funkcí tohoto systému je nahrávání konference, které neobnáší pouze audiovizuální záznam, ale také znovu zobrazení celé konverzace a sdílených souborů. Onstream Meetings umožňuje mimo pořádání videokonferencí také realizaci webinářů či webcastingů. Přístup k webinářům je umožněn dle specifikovaných parametrů až stovkám účastníků.

Samotný systém konference je postavený na technologii Adobe Flash. Verze instalovaného Flashe musí být aktuální, protože jak bylo testováno, samotná aplikace konference je náchylná na použití jeho starších



Obrázek 2: Click Meeting - komponenta zobrazující video na serveru Youtube

verzí. Kromě něho je vyžadována instalace software umožňujícího sdílení pracovní plochy počítače, pokud moderátor o využití této možnosti zvažuje.



Obrázek 3: Onstream Meetings - konferenční místnost

4 Srovnání vlastností analyzovaných systémů

Systémy se liší ve funkcích, které poskytují. Aby bylo možné je jednoduše porovnat, je nutné stanovit kritéria, která odpovídají stěžejním funkcím poskytovaných těmito systémy. Tato kritéria jsou pro názornost společně s analyzovanými systémy zaneseny do příslušných tabulek.

4.1 Základní funkce

V tabulce 2 je sestaven přehled základních zjišťovaných možností. Kromě video a audio konference jsou zkoumány možnosti textové komunikace, sdílení souborů a správy účastníků. Všechny systémy těmito možnostmi disponují.

Název systému	audio streamování	video streamování	chat	sdílení souborů	správa účastníků
Adobe Connect	✓	✓	✓	✓	✓
Click Meeting	✓	✓	✓	✓	✓
Onstream Meetings	✓	✓	✓	✓	✓

Tabulka 2: *Základní funkce konferenčních systémů*

4.2 Vlastnosti a rozšiřující funkce

Každý systém poskytuje celou paletu rozšiřujících funkcí. Mezi zkoumané patří možnosti sdílení plochy, ankety a další funkce. Všechny zmiňované systémy jsou multijazyčné a každý poskytuje podporu více jak 10 jazyků.

Název systému	sdílení plochy	anketa (hlasování)	poznámky	status uživatele	promítání dokumentů
Adobe Connect	✓	✓	✓	✓	✓
Click Meeting	✓	✓	✗	✓	✓
Onstream Meetings	✓	✓	✓	✗	✓

Tabulka 3: *Rozšiřující funkce analyzovaných systémů*

4.3 Oprávnění a role

Každý konferenční systém disponuje určitým rozdělením rolí a oprávnění. Základní rolí je běžný účastník a role, obecně nazývaná moderátor. V rámci analýzy je zjišťováno, jaké typy rolí systémy nabízí. V systému Adobe Connect jsou například role rozděleny na hostitele, který pořádá konferenci jako takovou, dále moderátory, kteří vedou její průběh, účastníky a pouhé pozorovatele.

Název systému	Definované role
Adobe Connect	hostitel, moderátor, účastník pozorovatel
Click Meeting	organizátor, moderátor, účastník pozorovatel
Onstream Meetings	moderátor, účastník

Tabulka 4: *Definované role uživatelů konferenčních místností*

4.4 Šablony zobrazení

Uspořádání jednotlivých prvků v místnosti souvisí s formou průběhu konference. Obsah setkání je různorodý, a proto jsou změny pozic a velikostí jednotlivých částí důležitým apektem systému. Přizpůsobení dané části je typicky řešeno technikou drag & drop, která poskytuje pohodlý způsob a umožňuje dynamicky upravovat velikosti jednotlivých komponent. K dispozici mohou být také předdefinované šablony či možnost jejich vytváření.

Název systému	Různé šablony	Vlastní šablony	Drag & Drop technika
Adobe Connect	✓	✓	✓
Click Meeting	✓	✓	✓
Onstream Meetings	✓	✓	✓

Tabulka 5: *Šablony zobrazení konferenční místnosti*

4.5 Použité technologie

Konferenční systémy mohou využívat pro realizaci komunikace různé technologie. Z tabulky 6 vyplývá, že všechny tyto systémy využívají technologie Adobe Flash, který typicky není podporován na mobilních zařízeních. Společnosti, jež je vyvíjí, musí vytvářet aplikace na míru pro mobilní platformy jako jsou Android a iOS. Každý z analyzovaných systémů takovou aplikaci nabízí. V žádném ze systémů není využito technologie WebRTC, která umožňuje audio a video streamování v prostředí prohlížeče bez nutnosti instalace dalšího software.

Název systému	Flash	Ajax	WebRTC	Jiné
Adobe Connect	✓			
Click Meeting	✓			
Onstream Meetings	✓			

Tabulka 6: *Technologie použité pro realizaci komunikace v analyzovaných systémech*

4.6 Koncept konferenčního systému

V rámci celého systému může existovat dělení uživatelských účtů podle poskytovaných funkcí a možností pro konferenční místnosti, kde se účastníci setkávají. Systém zajišťuje registraci a dle dané varianty je jeho používání zpravidla zpoplatněno. Jedním z omezení je počet účastníků, kteří mohou být do místnosti v jediném momentu připojeni. Systém poskytuje přehledy konaných konferencí, vytváření událostí konferenčních místností, rozesílání pozvánek k setkání a správu účtu.

Vytvoření nové místnosti může obnášet specifikování jejích parametrů jako začátek setkání, dobu trvání, název či jiné údaje. Dalšími funkcemi jsou možnosti vytvoření události konference či zasílání pozvánek účastníkům. Přístup je umožněn přes její URL[7] adresu a vstoupit mohou i neregistrovaní uživatelé. Samotná autorizace bývá ošetřena zadáním vstupního hesla či přístupem přes URL adresu, jež obsahuje speciální token či jiným mechanismem. Bezprostředně před vpuštěním účastníka do místnosti může být požadováno vyplnění některých údajů jako například již zmiňované přístupové heslo.

Připojenému členovi místnosti jsou automatizovaně poskytovány streamy vysílajících účastníků. Audio či velikost zobrazovaného videa lze přizpůsobit. Streamující může během hovoru svojí kameru či mikrofon ztlumit, připojit či odpojit.

V rámci místnosti je každému účastníkovi přidělena určitá role. Systém rolí je specifický pro každý systém. Nejvyšší oprávnění má její zakladatel, který je v roli jejího administrátora, ať už jakkoli nazývaného - typicky moderátor. Tato role zahrnuje řízení celé konference. Má kontrolu nad všemi účastníky a obsahy - může měnit role účastníků, přenastavovat parametry konference či posílat pozvánky ke konferenci apod. Kromě moderátora se typicky užívá role běžného účastníka nebo například role pouhého pozorovatele. Moderátor disponuje pokročilejšími volbami než běžný účastník. Disponuje možností vyloučení člena z místnosti či změny přístupového hesla do místnosti. Jednotlivým účastníkům může zakázat či povolit provádění používání určité funkce či zásahu do společně sdíleného obsahu.

Celý prostor konferenční místnosti je rozdělen do určitých sekcí. Jejich zobrazované rozložení lze přizpůsobit technikou drag & drop či vyměnit šablonu i v průběhu konference.

Místnost nabízí kromě videokonference a společného a odděleného chatu také celou paletu dalších funkcí jako jsou ankety, poznámky, sdílení a promítání souborů ale i kreslící plátno a další nástroje.

[7] Adresace umístění zdroje - webové stránky či souboru v internetové síti. [9]

5 WebRTC technologie

WebRTC je poskytované API prohlížeči a sada protokolů, na kterých pracují W3C a IETF standardizační organizace. S WebRTC mohou vývojáři rychle zabudovat peer-to-peer audio, video či datové sítě do jejich webových aplikací standardizovaným javascriptovým API. [11]

S WebRTC není třeba spoléhat na problematické plug-iny třetích stran v čele s Flash Playerem či pouštět se do vývoje vlastních plug-inů s omezenou podporou platforem, který je zbytečně nákladný na lidské i finanční zdroje investované nejen do jeho vývoje, ale i následného servisu. Plug-iny se dlouhodobě ukazují jako problém, a proto se postupně na různých frontách hledají způsoby, jak se závislosti na nich zbavit. Jde o dlouhodobý trend, protože plug-iny jsou obecně problematické z různých důvodů: bezpečnost, kompatibilita, spolehlivost, výkon apod. [12]

Tato technologie obsahuje 3 základní komponenty. První z nich je komponenta umožnující přístupovat k zařízením jako jsou mikrofon či webkamera z prostředí webového prohlížeče a poskytuje API pro práci se získanými médii. Tato média lze následně promítat, modifikovat či jinak zpracovávat. Hlavní předností této technologie, tedy další komponentou WebRTC, je však možnost již zmiňovaného vytváření tzv. peer-to-peer sítí, které umožňují realizovat přímé spojení dvou klientských zařízení. Tohoto spojení lze využít právě pro přenos audio či video v reálném čase. Také umožňuje vytvářet tzv. datové kanály, jež tvoří 3. komponentu WebRTC a slouží pro přenos textových řetězců nebo binárních dat.

WebRTC ale není jako takové doposud plně standardizovanou technologií. Nicméně současná implementace v internetových prohlížečích se odvíjí od návrhu[8] "WebRTC 1.0: Real-time Communication Between Browsers"vydaného organizací W3C. Nynější verze návrhu byla vydána k 28. lednu 2016.

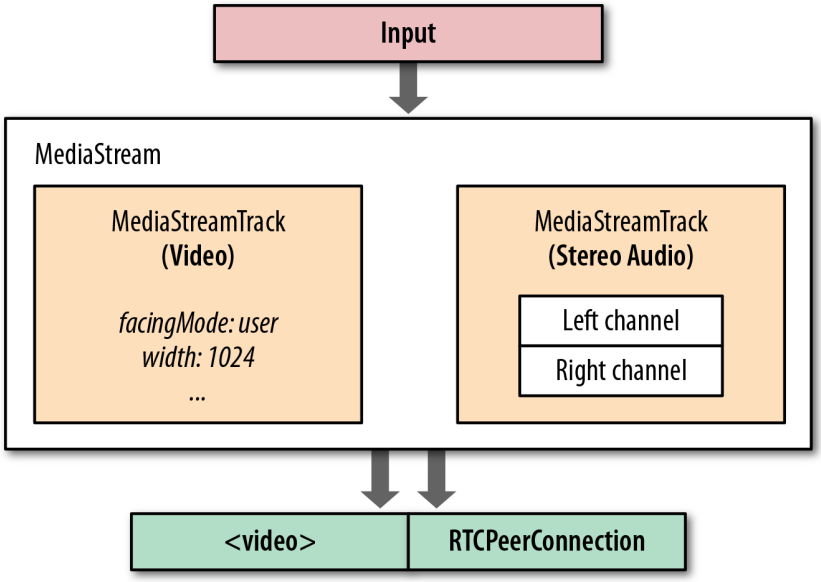
5.1 Uživatelská média

Uživatelská média reprezentovaná MediaStreamem jsou jednou ze součástí WebRTC, jež zaštiťují práci především s audio či video. Dvěmi hlavními komponentami v MediaStream API jsou rozhraní MediaStreamTrack a MediaStream. Objekt MediaStreamTrack reprezentuje médium jednoho typu, které pochází ze zdroje médií z user agenta[9], například video poskytované webkamerou. MediaStream seskupuje několik MediaStreamTracků do jediného bloku a ten může být nahráván nebo distribuován do elementu zpracovávajícího média. [17]

MediaStream představuje synchroní kontejner pro obsažené stopy - MediaStreamTracky - tak, aby se zamezilo nechtěnému časovému posunu mezi těmito stopami. Tedy například k posunu mezi reprodukovaným audio signálem a obrazem videa. Nelze jej však využít pro změnu vlastností či zastavení jednotlivých stop. Poskytuje ovšem API pro manipulaci se stopami jako takovými. [18]

[8] Návrh přístupný na adrese: https://www.w3.org/TR/2016/WD-webrtc-20160128

[9] User agent je software, který koncovému uživateli načítá, zobrazuje a usnadňuje interakci s webovým obsahem. Může se jednat o software, jehož uživatelské rozhraní je implementováno pomocí webových technologií. [20]



Obrázek 4: MediaStream shlukuje jednu nebo více synchronizovaných stop [21]

5.1.1 MediaStream a jeho API dle W3C návrhu

Konstruktor objektu MediaStream skládá nový stream z existujících stop, MediaStreamTracků. Volitelným parametrem konstruktoru je jiný MediaStream nebo sekvence objektů MediaStreamTrack. Po vytvoření objektu je inicializován jeho identifikátor. Následně je-li parametrem MediaStream, pro každou jeho stopou je volána metoda ”addTrack”. Parametrem může být i pole MediaStreamTracků. [17]

Při získání přístupu k uživatelským médiím je tento proces proveden automaticky a vytvořený MediaStream je poskytnut příslušnou metodou v případě jeho úspěšného získání viz. kapitola 5.1.3.

Je-li MediaStream beze stop či obsahuje stopy ukončené, je neaktivní. Disponuje atributem ”active”, který toto indikuje. Pokud se změní stav z aktivního na neaktivní či obráceně, jsou volány příslušné událostní metody v podobě callbacků[10] ”inactive”či ”active”. [17] Tyto callbacky dávají vývojáři možnosti obsluhovat vzniklé stavy specifickým způsobem. Jak bylo testováno na platformě Windows, aktuální podpora tohoto atributu v prohlížečích se však liší. Google Chrome (ve verzi 48.0) a Opera (ve verzi 35.0) jej narozdíl od Mozilly Firefox (ve verzi 44.0) podporují. Nastavení událostí by pak vypadalo následovně:

```
mediaStreamObject.onactive = function(){
    // kód prováděný na změnu stavu z active na inactive
};
mediaStreamObject.oninactive = function(){
    // kód prováděný na změnu stavu z inactive na active
};
```

[10] Callback je funkce, která je volána prostřednictvím ukazatele, paměťové adresy, funkce. Pokud je vložena adresa funkce jako argument jiné, je provedena. [24]

API metody objektu MediaStream: [17]

- addTrack - přidává stopu do MediaStreamu. V případě, že tato stopa je již obsažena, je přidávání stopy přerušeno.
- removeTrack - odebírá stopu z MediaStreamu. Parametrem je stopa, tedy objekt MediaStreamTrack, jež má být odebrán.
- getAudioTracks - návratovou hodnotou je sekvence MediaStreamTracků v podobě pole, které obsahuje média typu audio.
- getVideoTracks - návratovou hodnotou je sekvence MediaStreamTracků v podobě pole, které obsahuje média typu video.
- getTracks - návratovou hodnotou je sekvence všech MediaStreamTracků v podobě pole.
- clone - vytvoří nový objekt MediaStreamu a inicializuje jeho identifikátor na novou (jinou) hodnotu. Následně je nad každou stopou volána jeho vlastní metoda MediaStreamTrack.clone().

5.1.2 MediaStreamTrack a jeho API dle W3C návrhu

Objekt MediaStreamTrack, stopa, reprezentuje zdroj média poskytovaného internetovým prohlížečem. Několik MediaStreamTracků může reprezentovat stejný zdroj. Například pokud uživatel povolí stejnou kameru v uživatelském rozhraní prohlížeče dvakrát po sobě jdoucím volání ”getUserMedia”. Přistupovat ke stejnému zdroji z více prohlížečů v daný okamžik není možné. Objekt disponuje následujícími atributy: [17]

- enabled - atribut umožňuje indikovat a povolit stopu do aktivního a pasivního stavu.
- id - identifikátor dané stopy.
- kind - určuje typ daného média. Nabývá hodnot ”audio”nebo ”video”.
- label - popis použitého zdroje.
- muted - určuje, zda-li byl zdroj ztišen či nikoli.
- readonly - lokální MediaStreamTrack je sdílen s neměnným nastavením stopy. [17]
- readyState - atribut reprezentuje stav, v jakém se objekt nachází. Nabývá hodnot ”live”a ”ended”.
- remote - určuje zdroj stopy.

MediaStreamTrack také disponuje možností nastavení událostí zpracovávaných callbacky. Jedná se o události ”onmute”a ”unmute”, které jsou vyvolány v případě, že zdroj přestane či opět obnoví poskytování dat, dále ”onended”, která je volaná příznačně při ukončení stopy a ”onoverconstrained”, která je volána ve chvíli, kdy náhle již není možné poskytovat takový stream dat, který by odpovídal zadaným parametrům konfiguračního objektu ”Constraints”. [17]

API metod MediaStreamTracku: [17]

- applyConstraints - umožňuje změnu nastavených parametrů objektu ”Constraints”daného MediaStreamTracku. Její reálná implemementace je aktuálně však dostupná pouze v prohlížeči Mozilla Firefox.
- getCapabilities - návratovou hodnotou jsou možnosti konfigurace média.
- getConstraints - návratovou hodnotou je konfigurace objektu ”Constraints”pro danou stopu.
- getSettings - návratovou hodnotou je možná konfigurace stopy na daném zařízení.
- stop - nastaví atribut ”readyState”na hodnotu ”ended”a reálně odpojí stopu od zdroje.
- clone - bezparametrická metoda, jež klonuje daný MediaStreamTrack. Po volání této metody dojde k vytvoření objektu MediaStreamTrack.

5.1.3 Získání MediaStreamu

Média lze získat prostřednictvím nativního objektu Navigator[11]. Aby bylo vůbec možné získat nějaký typ média, je nutné specifikovat objekt, jež popisuje, o jaký typ média a s jakými parametry je v daném okamžiku žádano.

Objekt ”Constraints”stanovující omezení je obecným předpisem umožňující aplikacím, jak vybrat správný zdroj médií, tak i ovlivnit, jak bude daná stopa pracovat. [17] Pokud nejsou specifikovány parametry pro dané médium či je hodnota příslušného klíče objektu nastavena jako ”false”, není o tento typ média žádáno. Pakliže obsahuje hodnotu ”true”nebo klíč obsahuje objekt, jež blíže specifikuje parametry žádaného média, následuje pokus o jeho získání. Není-li žádáno o žádný typ média či je objekt constraints prázdný, žádost o udělení MediaSteamu je vyhodnocena jako chybná. Základní objekt specifikující média vypadá následovně:

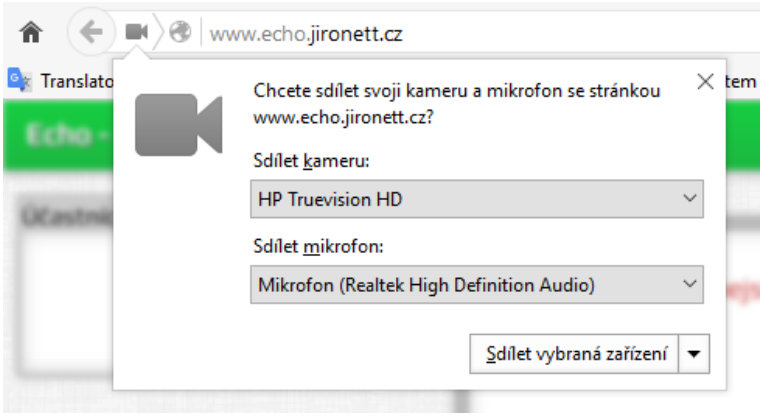
```
var constraints = {
    audio: true,
    video: true
};
```

Kterákoli položka objektu constraints poskytovaná tímto API bude zohledněna pouze, pokud je podporována daným prohlížečem. Neznámá či nepodporovaná vlastnost, jež je vyžadována, bude tiše vynechána. V javascriptovém kódu je nejprve očekávána kontrola podpory dané vlastnosti prostřednictvím metody ”getSupportedConstraints”, která vrací názvy všech podporovaných vlastností. [17]

```
// Ukázka kontroly podporované vlastnosti
var supports = navigator.mediaDevices.getSupportedConstraints();
if(!supports["facingMode"]) {
    // chybějící podpora vlastnosti facingMode (přední kamera)
}
```

[11] Navigator je rozhraní reprezentující stav a identitu user agenta. Skriptům umožňuje získání těchto informací a jeho metod. [23]

Uživatel je při pokusu o získání přístupu k nějakému médiu prohlížečem vyzván k udělení povolení. Dle implementačního doporučení[12] návrhu W3C o získání médií v rámci konkrétní webové adresy může být toto povolení uchováno v paměti příslušného prohlížeče. Při jeho příští návštěvě prohlížeč toto povolení udělí automaticky. Toto doporučení je implementováno v prohlížčích Google Chrome a Opera.



Obrázek 5: Povolení využívání webkamery a mikrofonu v Mozilla Firefox

V praxi využitelný příklad by mohl vypadat následovně: povolíme zvuk a nastavíme požadovanou velikost videa. Minimální velikost 1024x1280 px, ideálně 1280x720 px a maximálně do 1920x1080 px.

```
var constraints = {
    audio: true,
    video: {
        width: { min: 1024, ideal: 1280, max: 1920 },
        height: { min: 576, ideal: 720, max: 1080 }
    }
}
```

Vzhledem k tomu, že WebRTC API není organizací W3C doposud označeno jako finální, došlo také ke změnám v pracovního návrhu WebRTC ve způsobu získání MediaStreamu. Běžné získání média javascriptovým API je v současné době možné dvěma přístupy. Původně a stále široce rozšířený způsob získání MediaStreamu využívá callbacků a metody volané přímo nad samotným objektem Navigator. Vzhledem k neustálenému API této technologie, prohlížeče využívají pro jeho objekty tzv. prefixů jako například ”moz”či ”webkit”. Parametry metody ”getUserMedia”jsou již zmiňovaný objekt ”Constraints”, dále callback volaný po úspěšném získání MediaStreamu a callback volaný při neúspěšném pokusu o jeho získání.

Novým způsobem je získání MediaStreamu prostřednictvím objektu MediaDevices, jež je součástí Navigátoru. Mění se i samotný způsob práce s metodou ”getUserMedia”, která dále nepoužívá ke zpracování callbacků, ale objekt typu Promise[13]. Mozilla Firefox tento nový přístup již implementovala.

[12]Doporučení dostupné na adrese https://www.w3.org/TR/2015/WD-mediacapture-streams-20150414/#implementation-suggestions

[13]Objekt Promise je použit pro odložené či asynchronní výpočty. Promis představuje operaci, která ještě nebyla provedena, ale její zpracování je v budoucnu očekáváno. [25]

Ukázka již zastaralého řešení fungující napříč prohlížeči podporující MediaStream API:

```
var getUserMedia = navigator.getUserMedia || navigator.webkitGetUserMedia
|| navigator.mozGetUserMedia;
var constraints = {audio: true, video: true};
getUserMedia(constraints, function(mediaStream){
    // kód obsluhující úspěšné získání MediaStreamu
}, function(error){
    // kód obsluhující neúspěšné získání MediaStreamu
});
```

Ukázka současného návrhu:

```
var constraints = {audio: true, video: true};
navigator.mediaDevices.getUserMedia(constraints).then(function(mediaStream){
    // kód obsluhující úspěšné získání MediaStreamu
}).catch(function(error){
    // kód obsluhující neúspěšné získání MediaStreamu
});
```

5.1.4 Testování použitelnosti v internetových prohlížečích

V rámci testování této technologie na platformě Windows ve třech nejpoužívanějších prohlížečích (dle statistik ze serveru W3schools.com), které tuto technologii podporují, bylo zjištěno, že po zastavení jedné stopy z MediaStreamu metodou ”stop”, dojde v prohlížečích Google Chrome (ve verzi 48.0) a Opera (ve verzi 35.0) k jejímu faktickému zastavení, přičemž distribuce ostatních stop beze změny pokračuje. V prohlížeči Mozilla Firefox (ve verzi 44.0) však dojde k zastavení všech stop daného MediaStreamu. Při zastavení video stopy dojde dokonce k neočekávanému pádu tohoto prohlížeče.

Dalším rozdílem mezi prohlížeči je způsob implementace odebrání stopy z MediaStreamu metodou ”removeTrack”. V testové situaci byl získán MediaStream, který obsahoval, jak audio, tak i video složku. Nicméně chování je shodné i při využití pouze jednoho, libovolného média. Následně byl injektován do RTCPeerConnection objektu, který jej vysílal jinému klientskému zařízení. Dále byl i injektován do lokálního video elementu. Google Chrome a Opera odeberou stopu z instance objektu MediaStream a ukončí poskytování této stopy přes RTCPerrConnection, ale aplikace streamu v lokálním video elementu stále pokračuje. Mozilla Firefox též odebere stopu z instance MediaStreamu, nicméně její odesílání přes RTCPeerConnection dále pokračuje v nezměněné podobě a naopak dojde k zastavení stopy v lokálním video elementu.

V důsledku aktuální nestejnorodosti podporovaných API mezi prohlížeči je tedy při využívání MediaStream API nutné zohlednit i daný prohlížeč a jeho verzi, respektive podporu konkrétní funkce.

5.2 Peer-to-peer komunikace

Instance objektu RTCPeerConnection umožňuje zřídit peer-to-peer komunikaci. Ta je řízena nějakým signálním mechanizmem, obecně skriptem ve stránce přes server za použití XMLHttpRequest [14] nebo webových socketů apod. [13] Aktivní spojení lze realizovat jak mezi 2 zařízeními, tak též v rámci jediného zařízení.

Objekt RTCPeerConnection je určen k přenosu dat a multimediálního obsahu. Z tohoto důvodu mu jsou v procesu propojování nastaveny parametry definující přenos. Pokud tedy samotný přenos obsahuje nějaký typ média, je nejprve nutné získat samotný MediaStream. V rámci jednoho spojení lze používat jak datové kanály, tak i současně streamování médií. Nelze však zahájit peer-to-peer spojení, jež je prázdné, tedy takové, které nepřenáší žádný multimediální obsah a neobsahuje žádný datový kanál.

WebRTC aplikace vyžaduje provedení několika následujících kroků: získání streamu audio či video, získání síťových informací jako jsou IP adresy a porty a vyměnit je s druhým WebRTC klientem pro umožnění vytvoření spojení, koordinovat komunikaci pro evidování chyb, inicializaci či ukončení spojení, vyměnit informace o mediích jako například informace o kodecích či rozlišení. [15] Pro udržování těchto dat je používán protokol Session Description Protocol (SDP).

5.2.1 Session Description Protocol - SDP

Aby bylo možné realizovat přenos dat mezi jednotlivými klienty, je nejprve nutné specifikovat lokální popis. Ten je definován SDP standardem, jehož původní verze[15] pochází z roku 1998 vydaná organizací IETF. Je určený právě pro popsání multimediální relace.

SDP je formát pro popis parametrů streamovaných médií, a to pro inicializaci relace, nabídku relace či její obnovení. SDP jako takové média nepřenáší, ale je využit pro shromáždění typů streamovaných médií, jejich formátu a dalších potřebných parametrů. SDP je navržen tak, aby byl rozšiřitelný pro podporu nových typů médií a formátů. [26] Je využit také dalšími protokoly jako je Session Initiation Protocol - SIP. [33]

SDP, popis multimediální relace, má zcela textovou podobu využívající znakové sady ISO 10646 v kódování UTF-8. Názvy atributů a hodnot využívají pouze znaků z US-ASCII podmnožiny UTF-8, ale textové hodnoty mohou nabývat všech znaků z ISO 10646 znakové sady. [26]

SDP relační popis obsahuje řádky textu, jež každý obsahuje typ parametru a příslušnou hodnotu: ⟨type⟩=⟨value⟩. Typ ⟨type⟩ je vždy přesně jeden znak, u nějž záleží na jeho velikosti. Hodnota ⟨value⟩ je strukturovaný textový řetězec, jehož formát záleží na daném typu ⟨type⟩. [26]

SDP zahrnuje: [26]

- Typ média (audio, video, ...)
- Transportní protokol (RTP, UDP, ...)
- Formát média a kodeky (H.261 video, MPEG video, ...)

[14] XMLHttpRequest je API, které poskytuje klientské rozhraní pro přenos dat mezi klientem a serverem, což představuje jednoduchý způsob získání dat bez nutnosti znovu načítat celou stránku. [16]

[15] Dokument přístupný na adrese https://www.ietf.org/rfc/rfc2327

- Pro multicast[16] relaci: adresu a port multicastu
- Pro unicast[17] relaci: vzdálená adresa a kontaktní port pro přenášená média

```
Session description
     v= (protocol version)
     o= (originator and session identifier)
     s= (session name)
     i=* (session information)
     u=* (URI of description)
     e=* (email address)
     p=* (phone number)
     c=* (connection information - not required if included in all media)
     b=* (zero or more bandwidth information lines)
          One or more time descriptions ("t=" and "r=" lines; see below)
     z=* (time zone adjustments)
     k=* (encryption key)
     a=* (zero or more session attribute lines)
          Zero or more media descriptions

Time description
     t= (time the session is active)
     r=* (zero or more repeat times)

Media description, if present
     m= (media name and transport address)
     i=* (media title)
     c=* (connection information -- optional if included at session level)
     b=* (zero or more bandwidth information lines)
     k=* (encryption key)
     a=* (zero or more media attribute lines)"
```

Obrázek 6: Struktura SDP [35]

5.2.2 Session Initiation Protocol - SIP

V odvětví tradičních telekomunikací a síťového inženýrství byly vždy striktně odděleny 2 odlišné fáze při realizaci (hlasových) hovorů. První fází, kterou zajišťuje SIP, je tzv. ”call setup”, nastavení hovoru, které zahrnuje nastavení všech potřebných detailů pro uskutečnění hovoru mezi 2 zařízeními. Po nakonfigurování hovoru zařízení se spustí vlastní datový přenos, který využívá zcela odlišné skupiny protokolů, které realizují samotný přenos hlasových paketů. VoIP[18] je SIP protokolem, který patří přímo do aplikační vrstvy. [31] Tento protokol[19] je standardizovaný od roku 2002.

Technologie VoIP společně s SIP je pak aplikována například v aplikaci jako je Skype apod. [44] SIP standard využívá ke své práci SDP standardu. WebRTC technologie též přejímají SDP standard, nicméně samotné SIP již nikoli. [45]

5.2.3 Javascript Session Establishment Protocol - JSEP

5.2.3.1 Formát popisu multimediální relace

Ve specifikaci WebRTC jsou samotné popisy multimediálních relací formátovány jako SDP zprávy. I přes to, že manipulace s tímto formátem není v prostředí Javascriptu optimální, je široce rozšířený a dochází

[16] Multicast je metoda přenosu dat, jejíž data jsou odesílána z jednoho zdroje více síťovým adresám současně. Tato metoda je často využívána pro streamování videa či audia. [34]

[17] Unicastový přenos zasílá zprávy jediné síťové adrese. [34]

[18] VoIP (Voice over IP), je technika pro přenos audio komunikace a multimédií v rámci internetové sítě. [38]

[19] Dokument přístupný na adrese https://www.ietf.org/rfc/rfc3261

k vývoji jeho nových funkcí. Výsledkem je využívání SDP pouze pro jeho vnitřní reprezentaci. Ke zjednodušení jeho zpracování a zajištění flexibility je v Javascriptu definován objekt RTCSessionDescription, který jej zapouzdřuje. [36]

Pro tento objekt existují dle W3C návrhu následující typy: [36]

- Offer - popis relace zahrnuje mnoho možných konfigurací médií.
- Answer - relační popis je použit jako odpověď na Offer či je zaslána jako finální odpověď po tzv. "Praanswer". Posléze je výměna považována za kompletní.
- Praanswer - Není konečnou odpovědí. Využívá se zejména k inicializaci spojení (i opakovaně), později je však zasláná konečná "Answer", která dokončí signální proces modelu Offer/Answer.
- Rollback - Jedná se o speciální typ popisu, který vrací změny SDP do předchozího stavu. Objekt tohoto typu nedisponuje žádným obsahem.

5.2.3.2 Model Offer/Answer

WebRTC využívá pro realizaci komunikace spojení založené na modelu Offer/Answer[20] s využitím SDP, který je specifikován RFC[21] dokumentem[22] z roku 2006, jež je JSE protokolem využíván.

Jedná se o model, kde jeden z klientů vytvoří tzv. Offer, tedy SDP nabídku. Ta obsahuje eventuální realizovatelné konfigurace budoucího multimediálního přenosu dle možností prvního klienta, odesílatele. Offer může být generována v jakýkoli okamžik v průběhu života aplikace. [28] Vytvořená nabídka je následně zaslána druhé straně. Ta ji přijme a na jejím základě posléze vytvoří odpověď - Answer, kterou zašle zpět. Po jejím obdržení začne první strana streamovat média dohodnutých parametrů.

JSEP poskytuje mechanizmus vytvoření Offers a Answers včetně jejich aplikace. Nicméně realizaci vlastního přenosu vygenerovaného SDP ke vzdálené straně je zcela ponechána na samotné aplikaci, která má nad SDP nabídkami a odpověď mi plnou kontrolu. [36]

RTCPeerConnection disponuje v rámci svého API metodami pro jejich vytvoření. Po volání metod "createOffer"a "createAnswer"jsou vytvořeny již zmiňované RTCSessionDescription objekty. [13] Každá z propojovaných stran, tedy každý z obou objektů RTCPeerConnection, musí disponovat vlastním relačním multimediálním popisem, jež se vztahuje k němu samotnému a zároveň popisem druhé strany, který je přenesen zvoleným signálním mechanizmem.

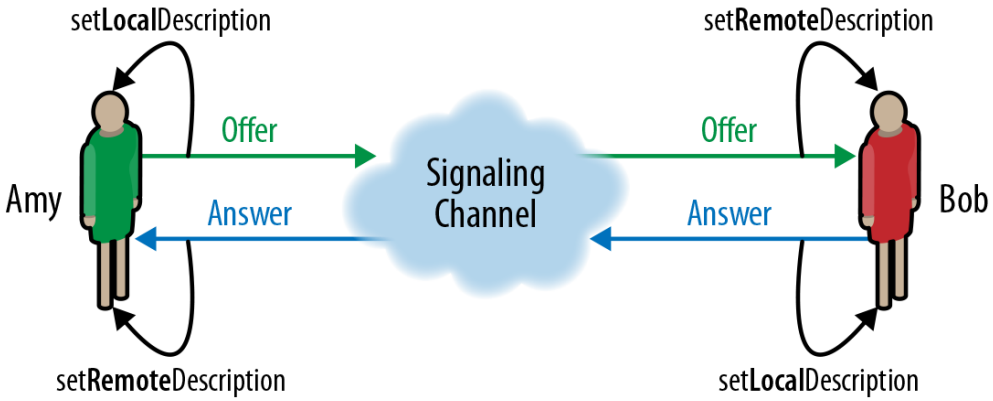
Při prvotním nastavování Answer je zasílána tzv. Praanswer, která není konečná, ale umožní zahájení komunikace a později je nahrazena finálním popisem typu Answer. Pro nastavení popisu je použito metod "setLocalDescription"a "setRemoteDescription". [13]

Metoda "createOffer"a "createAnswer"přijímají jako parametry callback úspěšného a neúspěšného vytvoření SDP. Vstupním parametrem úspěšného callbacku je vygenerovaný objekt typu RTCSessionDescription. V rámci tohoto callbacku by měl být nastaven lokální popis metodou "setLocalDescription"a zaslán

[20] Model Offer/Answer je definován RFC dokumentem 3264 přístupným na adrese http://tools.ietf.org/html/rfc3264

[21] RFC představuje formální dokument vydaný organizací Internet Engineering Task Force (IETF). Je výsledkem schválení redakčního výboru a následné kontroly zúčastněných stran. Některé z nich jsou pouze informativní, jiné jsou posléze standardizovány. [37]

[22] Dokument přístupný na adrese http://tools.ietf.org/html/rfc4566



Obrázek 7: Model Offer/Answer [21]

druhému klientovi. Toto je současná implementace, která funguje napříč prohlížeči, jež tuto technologii podporují.

Metody, využívající callbacků, jsou zachovány pouze za účelem zachování zpětné kompability. [13] Mozilla Firefox (testováno ve verzi 44.0) podporuje dle současného návrhu W3C i nově popsaný přístup, kde je místo volání callbacků použito objektů typu Promise. Stejně jako je tomu v případě získání MediaStreamu.

Ukázka vytvoření nabídky a její zaslání druhému klientovi:

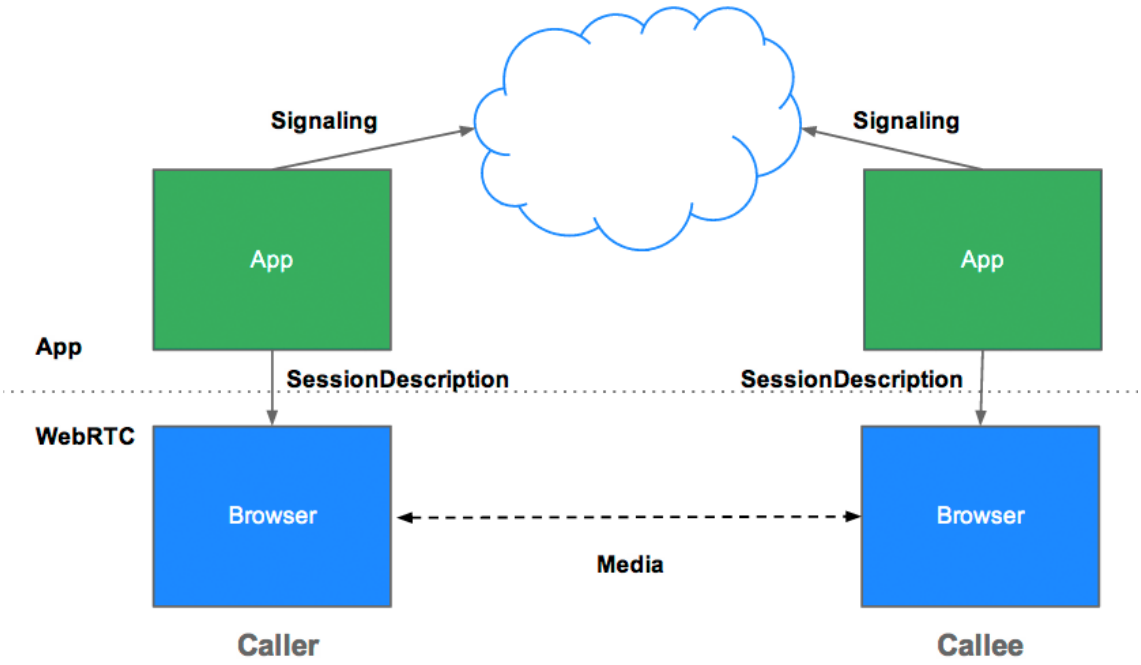
```
peer.createOffer(function(sessionDescription){
    peer.setLocalDescription(sessionDescription, function(){
        // odeslání popisu druhému klientovi zvoleným signálním mechanizmem
        signalChannel.send(peer.localDescription);
    }, function(error){
        console.log("Chyba nastavování lokálního SDP", error);
    });
}, function(error){
    console.log("Došlo k chybě při vytváření Offer", error);
});
```

Pro detekci v jakém signálním stavu se RTCPeerConnection v daný okamžik nachází, byly definovány následující stavy. Aktuální stav je uchováván atributem ”signalingState”. Signální stavy RTCPeerConnection iniciované danou akcí - volající: [13]

- stable - new RTCPeerConnection()
- have-local-offer - ”setLocalDescription”- Offer
- have-remote-pranswer - ”setRemoteDescription”- Praanswer
- stable - ”setRemoteDescription”- Answer
- closed - ”close”

Signální stavy RTCPeerConnection - volaný: [13]

- stable - new RTCPeerConnection()
- have-remote-offer - ”setRemoteDescription”- Offer
- have-local-pranswer - ”setLocalDescription”- Praanswer
- stable - ”setLocalDescription”- Answer
- closed - ”close”



Obrázek 8: JSEP architektura [15]

Answer je generována dle příchozí Offer. Tedy vytváří SDP zprávu tak, že má-li k dispozici MediaStreamTracky odovídající nabídce, vytvoří pro ně popis v odpovědi. Může tedy nastat následující situace: první z klientů zasílající Offer nabízí pouze audio a druhý klient disponuje navíc ještě video stopou, kterou chce streamovat. Reálné peer-to-peer spojení bude však odesílat pouze audio signál. V případě, že každý z klientů disponuje pouze odlišnými typy médií, spojení bude vždy obsahovat jeden jednosměrný stream. Přenášený obsah tedy není streamován na základě připojených MediaStremů či MediaStreamTracků k danému objektu RTCPeerConnection, ale na základě samotné SPD nabídky.

Fakt vyplývající z této skutečnosti však lze v praxi obejít další konfigurací při vytváření nabídky. Konfigurační objekt musí disponovat atributy ”offerToReceiveAudio”a ”offerToReceiveVideo”, jež jsou nastaveny na hodnotu ”true”. [22]

Dle specifikace by měl objekt RTCPeerConnection zahrnovat také atributy ”currentLocalDescription”a ”pendingLocalDescription”. První vrací již sjednaný a aktuálně platný popis, druhý je používán v průběhu

procesu nastavování nové SDP zprávy a ICE kandidátů. Stejný koncept je uveden i pro atribut ”remoteDescription”. RTCPeerConnection v prohlížečích Opera, Mozilla Firefox ani Google Chrome v současné době těmito atributy nedisponují.

Jak již bylo částečně zmíněno, metody ”createOffer”, ”createAnswer”, ale i ”addIceCandidate”, ”setLocalDescription”, ”setRemoteDescription”a ”getStats”jsou dle aktuální verze WebRTC specifikace sice zachovány, ale počítá se s využitím jejich nových variant, které jsou založené na objektech typu Promise. Google Chrome (ve verzi 48.0) doposud nový přístup neimplementuje.

5.2.3.3 Interactive Connectivity Establishment

Zřízení připojení je též součástí JSEP architektury. Na začátku procesu propojování klienti neznají topologii sítě, v níž se nacházejí. V praxi to znamená, že mohou či nemusí být za jedním či více překladači adres tzv. NAT[23]. Interactive Connectivity Establishment - ICE umožňuje klientům najít potencionální cesty, skrze které mohou komunikovat. Každý klient má různé kandidátní transportní adresy - kombinaci IP adres a portů pro konkrétní přenosové protokoly. Kombinace adres: [32]

- host candidate - adresa přímo připojeného rozhraní
- reflexive - adresa přeložená na veřejné straně pomocí STUN serveru za překladačem adres
- relay -adresa přiděleného TURN serveru

V praxi všechny kombinace nebudou funkční. Například pokud budou oba klienti za překladači adres, jejich přímo přidělené IP adresy jim neumožní vzájemně komunikovat. Účelem ICE je najít takový pár adres, který lze použít. ICE systematicky zkouší všechny možné páry, dokud nenalezne jeden či více, které budou fungovat. [32] ICE kandidát tedy reprezenzentuje informace o konkrétním potencionálním způsobu propojení. V rámci RTCPeerConnection existuje kandidát jako RTCIceCandidate objekt, který tyto informace udržuje. Stejně jako multimediální relační popis je přenášen signálním mechanizmem druhému klientovi. Přičemž první testovanou konfigurací ICE pro propojení klientů odpovídá lokálnímu rozhraní daného zařízení.

JSEP shromažďuje ICE kandidáty podle potřeby aplikace. Tato fáze je nazývána ”gathering”(neboli shromažďování) a v objektu RTCPeerConnection je uchovávána atributem ”iceGatheringState”. Proces shromažďování kandidátů je spuštěn na základě specifického parametru multimediálního relačního popisu či jiného, který indikuje restart ICE. Před prvotním zasláním SDP zprávy nemohou být ICE kandidáti generováni. Restartování ICE může být iniciováno aplikací nebo samotným prohlížečem v reakci na požadavek změn ICE konfigurace. [36] V ukázce je nastíněn způsob restartováni ICE. Mozila Firefox (ve verzi 44.0) restartování ICE nepodporuje.

Využito je ICE Trickling techniky, s níž volající postupně poskytytuje kandidáty. Tento proces umožňuje volanému zahájit okamžité zřizování ICE propojení na základě zaslaných kandidátů, aniž by byly zaslány kompletně všechny. Výsledkem je rychlejší nastavení médií. [36]

[23] NAT - překládá IP adresu z veřejné, globální adresy, na adresu privátní sítě a naopak. Na jedné veřejné adrese může být celá soukromá podsíť dosažitelná přes jedinou IP adresu. [29]

```
// původní verze vytvoření nabídky s využitím callbacků
peer.createOffer(successCallback, errorCallback, {iceRestart:true});

// použití dle aktuální specifikace
var offer = peer.createOffer({iceRestart:true});
offer.then(successCallback).catch(errorCallback);
```

Z pohledu shromažďování ICE kandidátů jsou definovány následující stavy, jež jsou uchovávané v atributu ”iceGatheringState”RTCPeerConnection: [13]

- new - objekt byl právě vytvořen a zatím nebylo iniciováno shromažďování.
- gathering - probíhá proces shromažďování možných ICE kandidátů.
- complete - shromažďování ICE kandidátů již skončilo. Pokud dojde k přidání dalších možných rozhraní, stav se vrátí opět na ”gathering”.

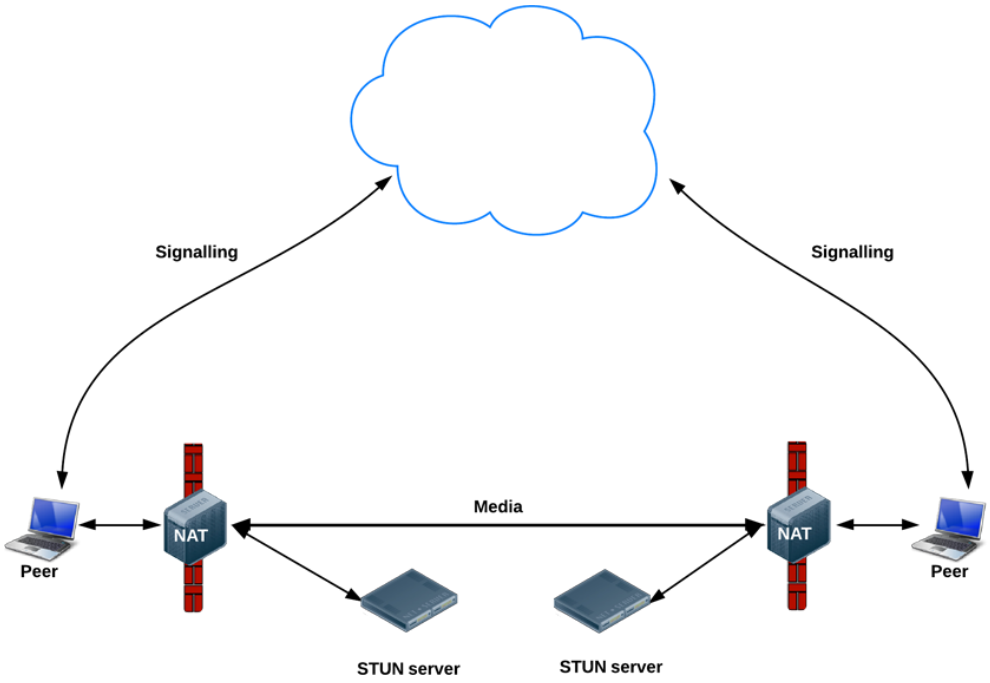
Vzhledem k tomu, že shromážďování výsledků možného použití STUN a TURN serverů může trvat potencionálně dlouhou dobu, lze využít tzv. ”ICE pool”, který umožňuje zahájit proces získávání kandidátů ještě před iniciováním spojení a snížit tak potřebnou dobu pro jeho vytvoření. [36] Aktivováno je atributem nastavovaným v konfiguračním objektu konstruktoru RTCPeerConnection a to atributem ”iceCandidatePoolSize”.

V průběhu životního cyklu RTCPeerConnection může ICE nabývat různých stavů připojení: [13]

- new - ICE shromažďuje adresy, a nebo čeká na vzdálené kandidáty, které mají být dodány.
- checking - Kontrol kandidátských párů pro vytvoření spojení
- connected - Bylo nalezeno použitelné připojení pro všechny komponenty, ale dále je zjišťováno, zdali existuje lepší spojení.
- completed - Shromažďování a ověřování kandidátů skončilo
- failed - ICE nedokázalo najít spojení nejméně pro jednu komponentu.
- disconnected - Kontrola živosti alespoň jedné komponenty selhala.
- closed - spojení ukončeno

5.2.3.4 STUN server - Session Traversal Utilities for NAT

STUN servery jsou veřejně přístupné v prostředí internetu a mají jeden jednoduchý úkol. Zkontrolovat a odeslat IP adresu a port příchozího dotazu aplikace, která běží za službou NAT a odeslat je zpět. Aplikace používá STUN server pro zjištění její veřejné IP adresy a portu. Tento proces umožňuje WebRTC klientovi rychle získat veřejně přístupnou adresu a poté ji zaslat signálním mechanizmem (v podobě ICE kandidáta) druhému klientovi pro zřízení propojení. [14]



Obrázek 9: Využití STUN serveru [14]

5.2.3.5 TURN server - Traversal Using Relays around NAT

RTCPeerConnection nejprve zkouší vytvořit přímé spojení mezi klienty přes UDP nebo TCP protokol. Pokud není přímá komunikace možná, RTCPeerConnection zřídí komunikaci přes tzv. TURN server, který vystupuje jako prostředník a data přeposílá mezi koncovými body. Jsou veřejně dosažitelné, a tak jsou přístupné i ze sítí, které se nacházejí za firewally či proxy servery. TURN server má konceptuálně jednoduchý úkol - přeposílat stream, ale oproti STUN serveru je z hlediska výkonu náročnější. [14]

Specifikace a konkrétní vlastnosti TURN serveru jsou standardizovány. Implementace takového serveru by měla být realizována na základě RFC5766[24]. Z open-source řešení, tedy těch s volně distribuovaným kódem, existují například projekty zvané Coturn či ReTurn iniciované společnostmi Google a Plantronics.

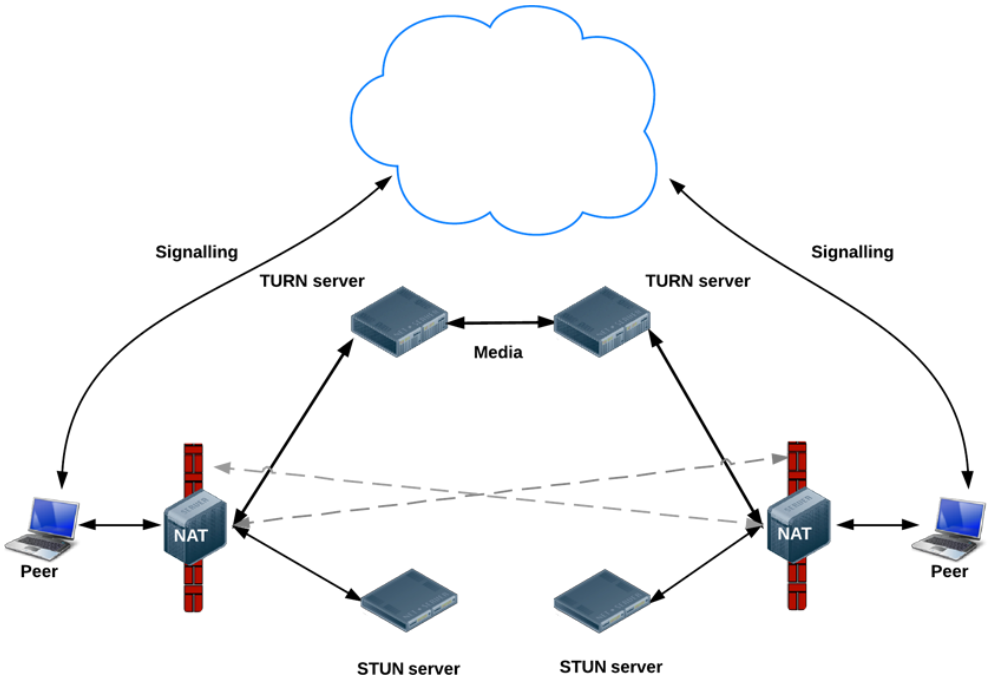
5.2.4 Vytvoření a nastavení objektu RTCPeerConnection

Konstruktor objektu RTCPeerConnection přijímá jako volitelný parametr konfigurační objekt. Nejdůležitějším atributem je ”iceServers”, který specifikuje STUN a TURN servery potřebné pro realizaci propojení dvou klientských zařízení mimo pouhou lokální síť.

Jedním z volitelných atributu je ”iceTransportPolicy”, který poskytuje volbu určení typu používaných ICE kandidátů: [13]

- all - Může být použit libovolný kandidát. Je výchozí hodnotou.
- public - ICE nepoužije kandidáty s privátními adresami.
- relay - Jsou použiti pouze kandidáti odkazující na TURN server.

[24]RFC přístupné na adrese https://tools.ietf.org/html/rfc5766



Obrázek 10: Využití TURN serveru [14]

Ukázka vytvoření nové instance:

```
var configuration = {
    "iceServers": [
        {"urls": "stun:example.org:3478"},
        {"urls": "turn:example.org:3479",
        "username": "userName",
         "credential": "password"}]
};
var peer = new RTCPeerConnection(configuration);
```

Metoda ”setConfiguration”může být volána v libovolný moment, pokud ”signalState”RTCPeerConnection není ve stavu ”closed”. Nastaví novou konfiguraci z přijatého parametru. I v případě, že jsou nastaveny nové ICE servery, nedojde k ICE restartovanání. To musí být provedeno aplikací následně. [13]

5.2.5 Média v RTCPeerConnection

Aby mohlo být započato streamování multimediálního obsahu, musí RTCPeerConnection disponovat odpovídajícími médii. Za dobu vývoje návrhu WebRTC došlo k změně API a pojetí v připojení a práci s těmito médii.

Návrh[25] WebRTC předcházející aktuálnímu vydaný k 10. únoru 2015 stále pojímá práci s médii jako manipulaci s celými MediaStream objekty, jež jsou spravovány metodami ”addStream”, ”removeStream”a dalšími. [19]

[25]Návrh přístupný na adrese https://www.w3.org/TR/2015/WD-webrtc-20150210

Ukázka původního přidání streamu:

```
var peer = new RTCPeerConnection(options);
peer.addStream(localMediaStream);
```

Novým přístupem je přidávání samotných MediaStreamTracků. Jsou přidávány metodou "addTrack"a odebírány metodou "removeTrack". Návratovou hodnotou metody "addTrack"je objekt "RTCRtpSender", který umožňuje aplikaci řídit samotný přenos. [13] Na základě testování na platformě Windows bylo zjištěno, že aktuální návrh je implementován prohlížečem Mozilla Firefox (testováno ve verzi 44.0). Původní verze je podporována i ostatními prohlížeči.

Návrh umožňuje využít možnosti dynamického generování nového SDP popisu pro již existující spojení. Změnu médií obsluhuje událostní metoda "onnegotiationneeded", která je volána na základě prováděných změn s těmito médii. [19]

5.2.6 DataChannel

Rozhraní RTCDataChannel reprezentuje obousměrný datový kanál mezi dvěmi vzájemně propojenými zařízeními technikou peer-to-peer. Objekt tohoto typu je vytvořen nad instancí RTCPeerConnection a to metodou "createDataChannel"nebo po iniciaci metody "RTCDataChannelEvent". [46] V rámci instance RTCPeerConnection spojení lze využívat více datových kanálů.

RTCDataChannel API podporuje flexibilní sadu datových typů. Jeho API je navrženo tak, aby přesně napodobovalo webové sockety. Podporovanými datovými typy pro přenos jsou textové řetězce stejně jako některé z binárních typů v javaScriptu jako je Blob, ArrayBuffer a ArrayBufferView. Tyto typy mohou být užitečné při práci s přenosem souborů a multiplayerových her. [47] Metody volané při události: [13]

- onopen - volaná při otevření datového kanálu
- onmessage - volaná při obdržení zprávy od druhého koncového bodu
- onerror - volána při výskytu chyby v průběhu spojení.
- onbufferedamountlow - volána v momentě, kdy počet bytů čekajích na odeslání k druhé straně je nižší nebo rovno nastavené hranici
- onclose - volaná při zavření datového kanálu

Metoda vytvářející datový kanál "createDataChannel"přijímá 2 parametry. Prvním je popis tzv. "label"a druhým je volitelný konfigurační objekt. Každý datový kanál disponuje identifikátorem, který je automaticky přidělován prohlížečem, pakliže není specifikován konfiguračním objektem. V rámci konfigurace lze dále nastavit maximální dobu nebo počet opakování přerušeného spojení, zda-li zprávy chodí v takovém pořadí, v jakém byly odeslány či nastavit vlastní identifikátor. [13]

Pro šifrování využívají datové kanály Datagram Transport Layer Security (DTSL) protokol. DTSL je odvozen od SSL[26], z čehož vyplývá, že bezpečnost spojení je na úrovni zabezpečení poskytovaného konceptem SSL. DTLS je standardizován a podporován všemi prohlížeči, které podporují WebRTC. [47]

[26]Secure Socket Layer (SSL) používá kombinaci veřejného a symetrického klíče k šifrování zabezpečeného spojení

Ukázka použití RTCDataChannel API:

```
var peer = new RTCPeerConnection();
var options = { maxRetransmitTime: 3500};
var channel = peer.createDataChannel("data-channel", options);
channel.onopen = function (){
    // při otevření datového kanálu
};
channel.onmessage = function (event){
    // při přijetí zprávy
    var receivedMessage = event.data;
    channel.send("Opověd'"); // zaslání odpovědi
};
channel.onclose = function (){
    // při zavření datového kanálu
};
channel.onerror = function (error){
    // zpracování chybového stavu
};
```

5.2.7 Podpora v internetových prohlížečích

Podpora v jednotlivých prohlížečích se liší. Lze ji porovnávat, jak z pohledu vůbec samotné možnosti využít WebRTC, tak i z pohledu samotné implementace API a jejich chování v rámci jednotlivých prohlížečů, které se v zásadě v současné době značně liší, což vývojářům webových aplikací znesnadňuje vytvářet WebRTC aplikace.

Dle testování a záznamů vedených na serveru Can i use[27], který se problematikou podpory webových technologií zabývá, vyplývá, že v současné době lze využívat API WebRTC v prohlížečích Mozilla Firefox, Google Chrome a Opera. Dále pak též v jejich mobilních verzích kromě prohlížeče Opera Mini. Není však podporováno prohlížeči Safari, Internet Explorerem ani prohlížečem Edge. Kompletní přehled je k dispozici v tabulce 7. Společnost Apple ve svých produktech přímo tuto technologii doposud nepodporuje. Prohlížeč Safari je postaven na funkčním jádře Webkit. Ten je v současné podobě používán i dalšími OS X, iOS a linuxovými aplikacemi. Dle pracovního statusu přímo z webových stránek[28] jádra Webkit je však WebRTC technologie aktuálně ve vývoji.

Na základě testování této technologie byla zjišťována nutnost použití šifrovaného spojení. Aby mohlo být WebRTC reálně nasazeno, v prohlížeči Google Chrome a Opera je šifrované spojení k načtené stránce

mezi dvěma zařízeními typicky webového nebo emailového serveru a klientského zařízení v internetové či privátní síti.[48]

[27] Přehled přístupný na adrese http://caniuse.com/#feat=rtcpeerconnection a dále http://caniuse.com/#feat=stream

[28] Přístupné na adrese https://webkit.org/status/#specification-webrtc

vyžadováno. Bez něho prohlížeče neudělí oprávnění ani pro přístup k uživatelským zařízením. Šifrované spojení není nutně vyžadováno prohlížečem Mozilla Firefox a to jak pro získání uživatelských médií, tak pro jejich streamování pomocí RTCPeerConnection.

Název prohlížeče	Stream API	RTCPeerConnection API	Obě API od verze
Google Chrome (desktop)	✓	✓	23
Mozilla Firefox (desktop)	✓	✓	22
Opera (desktop)	✓	✓	18
Google Chrome (Android)	✓	✓	47
Mozilla Firefox (Android)	✓	✓	44
Prohlížeč Android	✓	✓	47
Opera (Android)	✓	✓	33
Prohlížeč UC (Android)	✓	✓	33
Edge	✓ (12+)	✗	
Internet Explorer	✗	✗	
Internet Explorer Mobile	✗	✗	
Opera Mini	✗	✗	
Safari	✗	✗	
Google Chrome (iOS)	✗	✗	
Mozilla Firefox (iOS)	✗	✗	

Tabulka 7: *Podpora Stream API a RTCPeerConnection API v internetových prohlížečích*

6 Realizace konferenčního systému

Vzhledem k rozsahu nabízených funkcí analyzovaných konferenčích systémů je v rámci realizace takového prototypu vybráno pro implementaci několik stěžejních komponent. Jedná se o evidenci uživatelů, konferenční místnosti, jejich řízení a společný chat. Implementace těchto komponent směřuje k realizaci nejdůležitější části, kterou je možnost pořádání skupinových audio či video hovorů pomocí WebRTC technologie.

6.1 Požadavky a očekávané chování systému

Důležitým požadavkem je nutnost uzpůsobit zobrazení všech vytvářených částí tak, aby systém byl plně přístupný též i na mobilních zařízeních. Sestavené požadavky jsou přehledně reprezentovány Use case diagramem na obrázku 19. Celý systém umožňuje konání libovolného počtu konferencí souběžně.

6.1.1 Uživatelé systému

Aby bylo možné pořádat uzavřená setkání, ke kterým mají přístup pouze její zvaní členové, je nutné zajistit určitý systém identit. V konferenční místnosti jsou vyžadovány role bežného účastníka a moderátora, jež setkání inicioval. Pro přihlášení moderátora je vyžadováno uživatelského jména a hesla. Běžný účastník se do systému neregistruje a přihlašuje se uvedením svého (libovolného) jména a hesla konferenční místnost. Systém disponuje volně přístupnou registrací uživatelů - moderátorů.

6.1.2 Konferenční místnost

Rámec pojetí konferenční místnosti by měl odpovídat obecnému konceptu, jež byl analyzován v teoretické části práce. Moderátor může vytvářet konferenční místnosti. Každá z nich disponuje vlastním názvem a přístupovým heslem. Místnost je přístupná pomocí specifické URL adresy. Ta je tvořena kombinací uživatelského jména moderátora a jejím názvem.

Při prvotním přístupu do místnosti je běžný účastník vyzván k zadání výše uvedených údajů. Pakliže není konference moderátorem již ukončena, je vpuštěn. Moderátor přihlášený do systému je vpuštěn do místnosti automaticky.

Připojenému členovi se v místnosti automatizovaně zobrazí video a reprodukuje audio sdílené všemi vysílajícími v rámci místnosti. Po příchodu ostatním neposkytuje žádný multimediální obsah. Pokud je vybaven příslušnými zařízeními, disponuje možností své audio či video kdykoli připojit či odpojit. Každý streamující včetně přihlášeného člena jsou reprezentováni samostaným viditelným blokem, jež zahrnuje i ovládání jeho médií. Velikost okna příchozích streamů, vlastního streamu a lokálně zobrazovaného videa jsou přizpůsobitelné. U příchozích streamů lze též povolit či zakázat audio. Účastník také neomezeně využívá společného chatu a vidí seznam aktuálně připojených členů. Dle uvážení může v jakýkoli okamžik ukončit svojí přítomnost v místnosti.

Moderátor má kontrolu nad všemi možnostmi místnosti, které oproti účastníkovi zahrnují i právo ukončení konference, vyloučení člena z místnosti či změnu přístupového hesla místnosti.

Pokud moderátor či účastník nedisponuje vyhovujícím prohlížečem, který by podporoval potřebné technologie, je upozorněn.

6.2 Návrh

Systém vyžaduje použití 2 základních rolí. První rolí je moderátor a druhou je účastník s pracovními názvy ”presenter”a ”participant”. V rámci systému jsou navržena jednotlivá zobrazení, která umožňují používat dílčí části systému. Pro navštívení konference účastník využívá zobrazení přihlášení do místnosti, samotnou místnost a její ukončení. K ostatním nemá přístup kromě úvodní stránky a registrace uživatele, které jsou přístupné bez nutnosti autorizace stejně jako chybová stránka. Všechna navržená zobrazení:

- Úvodní strana s přihlášením pro uživatele systému
- Registrace uživatele
- Správa konferenčních místností
- Přihlášení do místnosti
- Konferenční místnost
- Zobrazení skončení konference
- Zobrazení vyloučení účastníka
- Chybové zobrazení

V rámci systému je používáno jednotného označení klient, který představuje člena konference - účastníka nebo moderátora. Každý z nich vyžaduje jiný způsob autorizace. Pokud se neautorizovaný klient pokusí přistoupit do místnosti, je mu zobrazeno přihlášení do místnosti pro účastníka. Neautorizovaný přístup do sekce pro moderátora odkáže klienta na úvodní stránku.

Navrhovaný systém představuje komplexní aplikaci, a proto je vhodné přizpůsobit její architekturu tak, aby oddělovala funkční části. Takovou představuje MVC[29] či podobně založená architektura, jež odděluje řízení činností, provedení požadovaných procedur a zobrazení aplikace.

Ke každé z navržených entit viz. kapitola 6.2.3 bude přiřazena třída, jež zapouzdřuje procedury a funkce k ní se vztahující. Ponesou jednotné označení ”Manager”tedy ”RoomManager”, ”UserManager”a ”ParticipantManager”.

6.2.1 Konferenční místnost

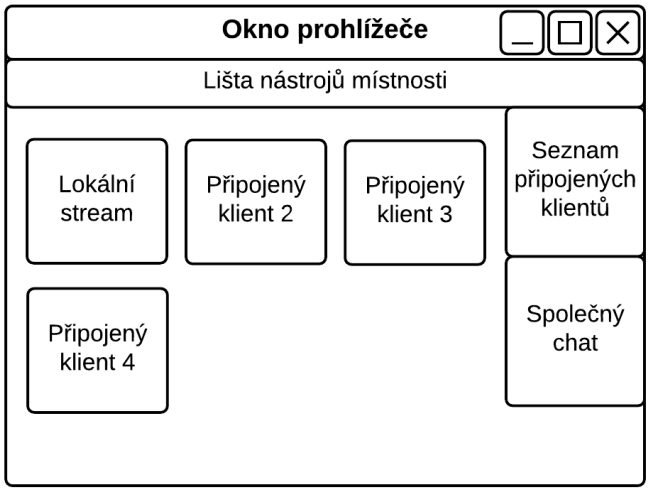
V průběhu načítání zobrazení místnosti je nutné ověřit, zda-li požadována místnost existuje, nebyla-li již konference skončena, je-li klient autorizován a patří do dané místnosti. Jedná-li se o bežného účastníka, je kontrolováno, není-li z místnosti vyloučen. Moderátorovi jsou načteny i další nástroje.

V průběhu konference se do místnosti připojují a odpojují její účastníci či moderátor. Aby aplikace mohla umožnit jejich vzájemnou komunikaci, musí pružně reagovat na tyto změny a evidovat klienty připojené v danou chvíli včetně místností, ve kterých se nacházejí. Základním předpokladem je zřízení

[29] Popis architektury softwarové aplikace přístupný na adrese https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

komunikace mezi dvěma klienty v reálném čase. Toho bude využito nejen pro společný chat, ale zvláště pro předávání aplikačních signálů, jež iniciují provedení požadovaných procedur. Důležitým aspektem je z pohledu návrhu také volba implementace jejich přenosového mechanizmu. Pro signální komunikaci byla zvolena technologie webových socketů pro účelnost jejich použití a jednoduchost implementace. Umožňují obousměrné zasílání zpráv mezi serverem a klientem v reálném čase. Pro přemostění komunikace mezi samotnými klienty je nutné implementovat program, který toto umožní. Touto částí návrhu se podrodněji zabývá kapitola 6.2.2.

Příchod do místnosti klientem je komplexní proces zahrnující celou řadu procedur, které jsou sestaveny na základě vytvořené analýzy a návrhu. Příchod je znázorněn diagramy aktivit počínaje obrázkem 20. Návrh realizace peer-to-peer spojení popisuje diagram aktivit na obrázku 23.



Obrázek 11: Návrh rozmístění prvků konferenční místnosti

Po zobrazení místnosti dojde k inicializaci prostředí a následně automatickému zaslání signálu, jež registruje klienta na signálním serveru. Posléze dostává od signálního serveru zprávu o přijetí, která obsahuje seznam klientů připojených v místnosti. Současně klienti obdrží signál o vstoupení nového člena do místnosti. Pokud streamují nějaký multimediální obsah, zahájí proces zřizování peer-to-peer komunikace.

Jsou navrženy třídy ”View”a ”Echo”. Třída ”View”bude využita pro manipulaci s elementy a provádění grafických procedur. Třída ”Echo”bude provádět logiku aplikace od její inicializace až po realizaci peer-to-peer spojení. V plném rozsahu využije zmiňované třídy ”View”, kterou zapouzdří.

Analýza technologie WebRTC a návrh funkčních požadavků umožňují definovat signály, jež bude nutné zasílat v rámci komunikace mezi klienty:

- Žádost o zařazení klienta do místnosti
- Odpověď klientovi o zařazení do místnosti
- Odpojení klienta z místnosti
- Informace o zařazení nového klienta do místnosti
- Zaslání zprávy chatu

- Vyloučení účastníka
- Ukončení místnosti

Dále jsou vyžadovány signály nutné pro zřízení peer-to-peer komunikace a streamování médií:

- Zaslání SDP nabídky
- Zaslání SDP odpovědi
- Zaslání kandidáta - informace nutné pro zřízení připojení
- Ukončení peer-to-peer komunikace
- Žádost druhého klienta o zřízení peer-to-peer komunikace

6.2.2 Signální server

Signální server slouží k přenosu zpráv různých účelů v reálném čase mezi jednotlivými účastníky každé konference a zároveň umožní udržovat aktuální přehled o probíhajících setkáních v podobě evidovaných konferenčních místností a jejich členů. Sestavené diagramy aktivit počínaje obrázkem 22 zahrnují jak samotné zpracování příchozího požadavku, tak i další aktivity na signálním serveru.

Webové sockety jsou zvoleným přenosovým mechanizmem. Poskytují jednoduchost implementace a zároveň splňují požadavek na potřebu komunikace v reálném čase. Při zasílání požadavků neumožňují relevantně ověřit pravou indentitu odesílatele. Pro její zajištění odesílatel používá při komunikaci se signálním serverem osobní klíč, k němuž nemá přístup nikdo jiný. Tento klíč je kombinací náhodně vygenerovaného řetězce a identifikátoru, aby byl vždy unikátní. Klient může se serverem komunikovat tehdy, odpovídají-li k záznamu vedenému v databázi též i zaslané údaje: identifikátor místnosti, klientský klíč a identifikátor účastníka, aby se zamezilo neoprávněnému vstupu do konference. Moderátor může oproti účastníkovi zasílat také signály, které ukončí místnost či odpojí účastníky.

Je navržena třída ”EchoSignalServer”. Ta uchovává seznam aktivních místností, které se průběžně evidují či ruší. Do každé místnosti lze kdykoli připojit či odpojit klienta. Pro evidování - připojení - člena či místnosti je používáno pojmu registrace. Dále spouští webový server[30] a obsluhuje interakci mezi uživateli místnosti v podobě příchozích požadavků. Po jejich zpracování bude ve většině případů vyžadováno odeslání zprávy některému či všem účastníkům, a proto jsou navrženy metody ”unicast”a ”roomBroadcast”. Přičemž metoda ”unicast”zašle zprávu jedinému konkrétnímu příjemci a metoda ”roomBroadcast”zašle zprávu všem účastníkům místnosti kromě samotného odesílatele.

Z pohledu autorizace se účastník může vyskytovat ve stavu, kdy není registrován v probíhající konferenci a tedy může pouze žádat o přidání do místnosti. Následně je ve stavu, kdy již registrován byl a může zasílat další signály.

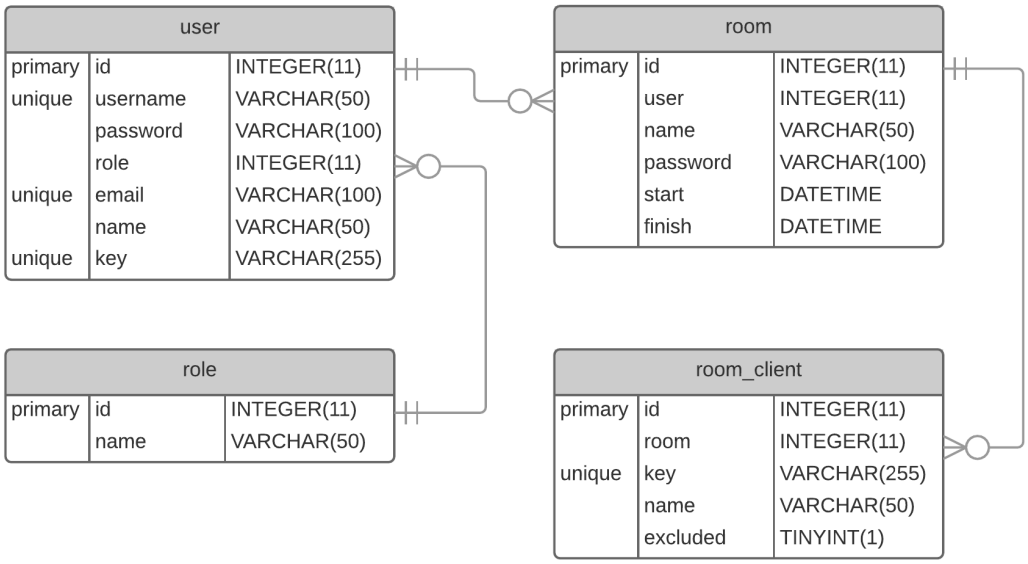
[30] Webový server je program, který využívá HTTP (Hypertext Transfer Protocol), pro poskytování souborů v reakci na požadavky uživatelů nějaké služby, které jsou předány prostřednictvím jejich HTTP klientů. [50]

6.2.3 Datový model

S ohledem na všechny dílčí části systému byl sestaven návrh datového modelu. Rozvržení zahrnuje 4 entity. První je ”user”tedy uživatel, který reprezentuje uživatele tohoto systému, v jehož rámci vystupuje v roli moderátora a disponuje odpovídajícími nástroji jako je vytváření místností a další. Dále entity definující samotnou místnost - ”room”a připojovaného klienta do místnosti - ”participant”, u nějž není vyžadována žádná předchozí registrace v systému.

Každý klient musí disponovat soukromým klíčem, aby mohla být provedena jeho identifikace na signálním serveru, a proto uživatel systému stejně jako obyčejný účastník místnosti disponují atributem klíče - ”key”.

Při prvotním vstupu do místnosti je účastník vyzván k zadání jejího hesla, které je jedním z atributů místnosti. Aby bylo možné rozeznat začátek a konec konané konference, zahrnuje též odpovídající atributy. Běžný účastník disponuje atributem ”excluded”, který indikuje, zda-li byl z místnosti vyřazen či nikoli.



Obrázek 12: Datový model konferenčního systému

6.3 Implementace

6.3.1 Webový server

Pro realizaci serverové části je využito hostingových služeb s vlastní doménou. Vzhledem k tomu, že prohlížeče Opera a Google Chrome nepodporují využití WebRTC bez zabezpečeného přenosu, je nutné zajistit šifrované spojení k serveru, které je nastavováno v administraci hostingu.

6.3.1.1 Vytvoření základního skeletonu aplikace

Vlastní program je psán v jazyce PHP. Pro usnadnění práce je využito nástroje Composer[31], který umožňuje dynamické stahování a aktualiaci potřebných programových knihoven a jejich závislostí. Celá serverová

[31] Přístupný na adrese https://getcomposer.org

část je postavena na Nette frameworku[32], který poskytuje robustní zázemí pro realizaci webových projektů založených na MVP[33] architektuře. Vytvářený konferenční systém dostal pracovní název Echo. Následujícím konzolovým příkazem jsou do složky Echo načteny všechny balíčky jež, Nette framework zapouzdřuje v rámci běžného webového projektu. Zdrojové soubory aplikace zpracovávané serverem se nacházejí v adresáři "app". Dále je využito projektu Dibi[34], což je knihovna představující databázovou vrstvu.

- `composer create-project nette/web-project=1.7 Echo`
- `composer require dibi/dibi`

6.3.1.2 Zpracování požadavků

Příchozí http požadavek je frameworkem zpracován a přeložen na tzv. aplikační požadavek. Na základě směrovače zvaného "Router"je předán odpovídající řídící třídě. Ty nesou v Nette frameworku pojmenování "Presenter". Obsloužení aplikačního požadavku - například zobrazení konferenční místnosti - je provedeno odpovídající metodou "action"a vykreslení pomocí příslušné metody "render"daného presenteru. Pro vygenerování výsledného HTML je použito šablonovacího systému Latte. Jednotlivá zobrazení definovaná v návrhu jsou zpracovávána následujícími objekty:

- FrontPresenter - úvodní stránka
- FrontPresenter - registrace uživatele
- RoomPersenter - přihlášení do místnosti
- RoomPersenter - konferenční místnost
- RoomPersenter - vyloučení účastníka
- RoomPersenter - ukončení místnosti
- BackendPresenter - správa místností
- ErrorPresenter - chybové hlášení

6.3.1.3 Práce s MySQL databází

V konferenčním systému je vyžadována práce s daty uchovávanými v databázi. Použitou databází je MySQL. Pro komunikaci s ní je využito knihovny Dibi. Ta zajišťuje určitou abstrakci zapouzdřením SQL dotazů do tzv. "Fluent"objektů, které umožňují postupné skládání a rozšiřování dotazů. V požadovaném momentu je volána příslušná metoda, jež provede danou operaci a vrátí výsledek.

Pro každou databázovou tabulku je vytvořena třída představující repozitář metod - operací, které jsou při práci s danou tabulkou využívány. Třída "RoomRepository"odpovídá tabulce "room". Analogicky jsou

[32] Přístupný na adrese https://nette.org

[33] Popis MVP architektury přístupný v dokumentu umístěném na adrese http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

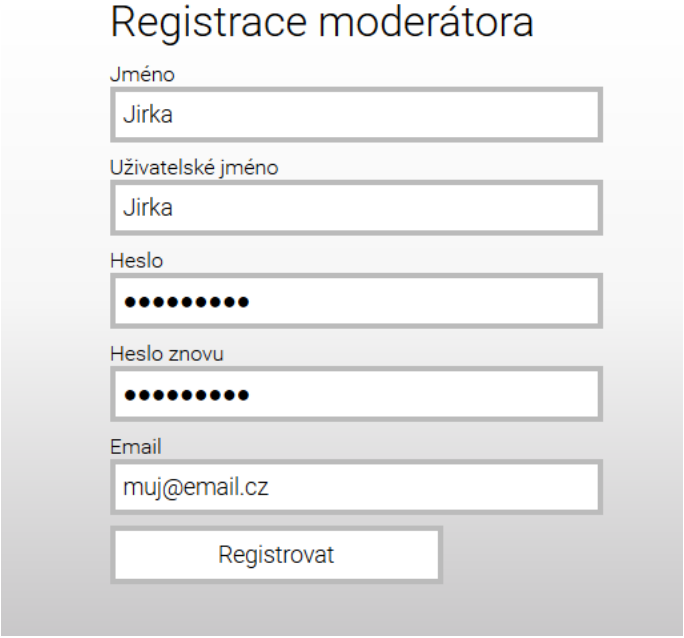
[34] Knihovna přístupná na adrese https://dibiphp.com

vytvořeny i další. Každý z těchto repozitářů je zapouzdřen třídou ”Manager”. Každý ”Manager”tak využívá repozitáře pro komunikaci s databází. Ukázka vložení záznamu nové místnosti v RoomRepository:

```
public function insert($userID, $roomName, $hash, $slug)
{
    return $this->db->insert('room', ['user' => $userID,
        'name' => $roomName, 'password' => $hash,
        'slug' => $slug, 'start' => new DateTime()])
        ->execute();
}
```

6.3.1.4 Registrace moderátora

Samotný moderátor se registruje ve volně přístupném zobrazení ”registration”presenteru ”FrontPresenter”, které nevyžaduje žádnou autorizaci. Uživatelské jméno a email jsou unikátní v rámci celého systému. Formulář obsahuje antispamovou ochranu v podobě skrytého pole pojmenovaného ”email”, které musí být pro úspěšnou registraci prázdné. Generování formuláře a zpracování registrace je řešeno vlastní Nette komponentou.



Obrázek 13: Registrace moderátora

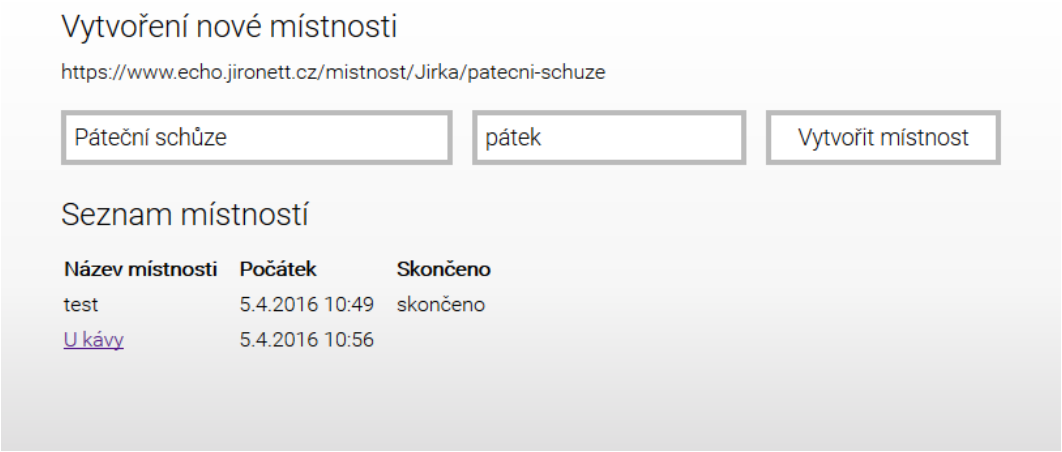
6.3.1.5 Vytvoření místnosti

Moderátor má v přehledu místností k dispozici jednak seznam konaných konferecí, tak může vytvářet i místnosti nové. Ty, které nebyly ukončeny, lze kdykoli využívat pro konferenci.

Přístup na toto zobrazení kontroluje autorizaci uživatele jako moderátora. Následně jsou načteny z databáze všechny jeho místnosti. Novou vytváří moderátor prostřednictvím formuláře, kde vyplní název a

přístupové heslo místnosti. Zadávaný název místnosti je v reálném čase převáděn javascriptem na budoucí tvar URL, pod nímž bude místnost přístupná. Formulář a jeho zpracování je zapouzdřen do Nette komponenty. O provedení samotné procedury se stará metoda RoomManagera ”createRoom”. Při jejím vytváření je ověřováno, zda-li již neexistuje místnost daného moderátora, jejíž přísptupový URL tvar tzv. slug by byl již použit. Ukázka vytvoření místnosti v RoomManageru:

```
public function createRoom($userID, $roomName, $password, $slug)
{
    if($this->loadRoom($userID, $roomName)){
        return false;
    }
    $this->roomRepository
    ->insert($userID, $roomName, Passwords::hash($password), $slug);
    return true;
}
```



Obrázek 14: Vytvoření nové místnosti

6.3.1.6 Načtení místnosti

První fází pro načtení místnosti je kontrola přístupu přes danou URL adresu. Celou proceduru zajišťuje metoda ”actionDefault”třídy RoomPresenter. Z databáze je načten záznam dané místnosti. Existuje-li, následně je ověřeno, je-li klient přihlášen a patří-li do dané místnosti. Je také kontrolováno vyloučení účastníka a konec konference. Neodpovídá-li některý z údajů požadavkům, je klient přesměrován na jiné zobrazení. Údaje načtené z databáze jsou předány do šablony a podle role je zobrazena celá místnost. Následuje inicializace klientským javascriptem.

6.3.1.7 API pro signální server

Signální server musí kontrolovat existenci a oprávnění připojovaných klientů do místností, a proto je v

aplikaci vytvořeno API o 2 metodách, které mu na základě speficické URL adresy a obsažených parametrů umožní získat data načtená z databáze konferenčního systému. Toto API realizuje třída "SignalServerAPIPresenter". Komunikace probíhá na základě předem dohodnutého klíče, aby nebylo možné získat přístup k datům klientů bez oprávnění. Výsledkem zpracování příchozího dotazu je objekt JSON, který je odeslán jako odpověď. Ta je následně signálním serverem vyhodnocena a připojovaný klient je připojen do místnosti.

API umožňuje načíst jak uživatele systému, tak účastníka. Obě metody kontrolují oprávnění komunikace a následně načítají informace o místnosti a daném klientovi pomocí dotazu, jež zahrnuje příslušnou tabulku klienta a místnosti. Není-li takový záznam k dispozici či místnost je již skončena, nebo účastník je z místnosti vyloučen, data neposkytnou. Ukázka implementace metody obsluhující načtení účastníka:

```
public function loadRoomParticipant($participantID)
{
    $roomParticipant = $this->participantRepository
    ->getRoomParticipant($participantID)->fetch();
    if($roomParticipant && $roomParticipant->finish !== null){
        return false;
    }
    if($roomParticipant && $roomParticipant->excluded === 1){
        return false;
    }
    return $roomParticipant;
}
```

6.3.2 Klientská část

6.3.2.1 Příprava prostředí

Při realizaci kaskádových stylů a klientského javascriptu je využíváno Node[35]. Node s NPM[36] správcem jeho javascriptových balíčků je nejprve instalováno. Použitým balíčkem je Gulp[37], který slouží k automatizaci procesů. Společně s dalšími balíčky[38] je využit pro generování a kompresi kaskádových stylů z preprocesoru Less[39] a klientského javascriptu. Soubor s automatizačním programem je umístěn v kořenovém adresáři celé aplikace pod názvem "gulpfile.js". Soubory s vyvýjeným javascriptem a Less kódem jsou odděleny od finálně generovaných. Umístěny jsou v příslušných podadresářích adresáře "dev". Instalace Node balíčků jsou v adresáři aplikace stahovány prostředictvím NPM:

- `npm init`

[35] Node je přístupné na adrese http://nodejs.org

[36] NPM je přístupné na adrese https://www.npmjs.com

[37] Web balíčku Gulp je přístupný na adrese http://gulpjs.com

[38] Balíčky gulp, gulp-concat, gulp-uglify, gulp-less, less-plugin-clean-css jsou skrze svůj název přístupné na stránkách správce balíčků NPM https://www.npmjs.com

[39] Less je přístupný na adrese http://lesscss.org

- `npm install --save gulp`
- `npm install --save gulp-less`
- `npm install --save less-plugin-clean-css`
- `npm install --save gulp-concat`
- `npm install --save gulp-uglify`

6.3.2.2 Javascriptový rámec klientské části

Objekt ”Echo”disponuje metodou ”fire”, kterou je běh celé aplikace v klientském prostředí spuštěn. Jako vstupní parametr přijímá objekt definující údaje klienta. Nejprve je provedena inicializace a následně vytvořeno websocketové spojení. Posléze se zaregistrují událostní metody obsluhující interakci s klientem a automatizovaně se zasílá signál žádající o registraci na signálním serveru.

Signál je definován objektem, který disponuje atributy ”roomID”pro identifikaci místnosti, do které uživatel patří, ”clientID”jako identifikátor účastníka, ”clientKey”představuje bezpečnostní klíč, podle kterého je ověřena pravost identity odesílatele, dále volitelný atribut ”data”a atribut ”type”, jež informuje server o jakou akci se jedná.

Po kontrole technologií a inicializaci aplikace je odeslán signál o žádost registrace na signální server. Následně je klientovi jako odpověď zaslán signál se seznamem aktuálně připojených účastníků. Zpracování příchozích signálů obsluhuje metoda ”processSignal”, která je implementována řídící strukturou switch. Signál je vyhodnocen akcí v podobě odpovídající metody.

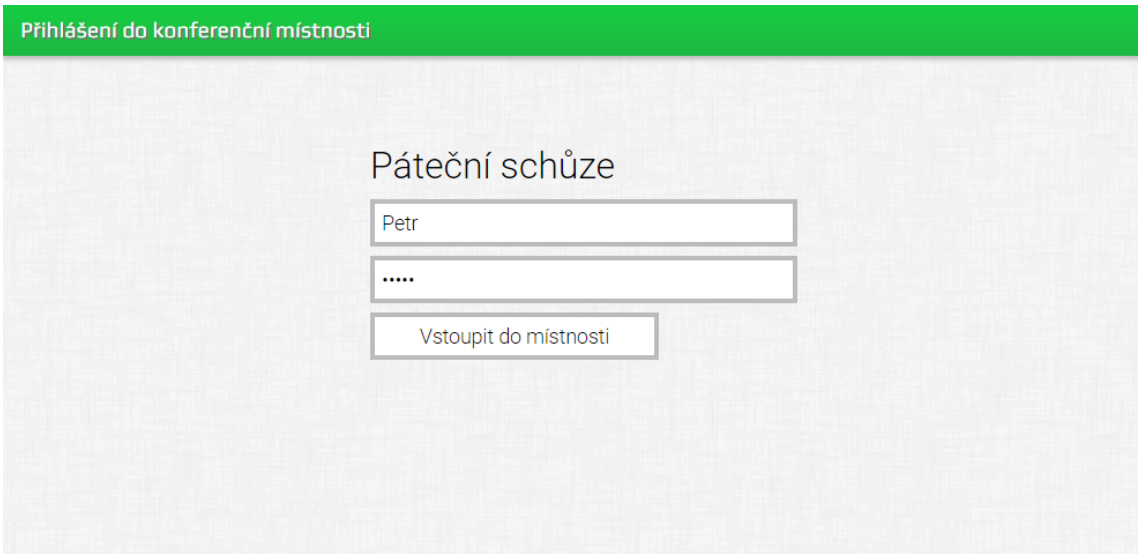
Nejdůležitější zprávy místnost sděluje klientovi prostřednictvím dialogů. Každý dialog je definován vlastní třídou ”Dialog”, která je javascriptem ovládána. Disponuje metodami ”show”a ”close”, kterými je dialog otevřen či zavřen. Každý umožňuje nasazení specifické implementace těchto metod pomocí callbacků.

6.3.2.3 Implementace WebRTC v klientské části

Implementace WebRTC je provedena na základě provedené analýzy této technologie. Zřízení samotného spojení je realizováno dle vytvořeného návrhu na obrázku 23. Místnost musí evidovat jednotlivá spojení pro jednotlivé členy konference, a proto připojení klienti jsou uchováváni v objektu ”clients”. Přístupní jsou přes přidělený identifikátor, který je v případě moderátora modifikován kvůli případné duplicitě s účastníkovým. Pro každého klienta, který streamuje audio či video, nebo přijímá od daného klienta stream, je v objektu ”peers”pod téže klíčem jako v případě objektu klientů vytvořena instance RTCPeerConnection.

Vytvoření nového RTCPeerConnection zahrnuje iniciaci nové instance objektu, nastavení událostních metod a pokud daný klient disponuje streamem, je k němu připojen. Událost, při níž je přidán stream druhého klienta provede kontrolu, jedná-li se pouze o audio. V tom případě je v streamu daného klienta zobrazen informativní text o zvoleném médiu. Je inicializována metoda pro příjem vzdálených ICE kandidátů a nastavena metoda ukončující spojení.

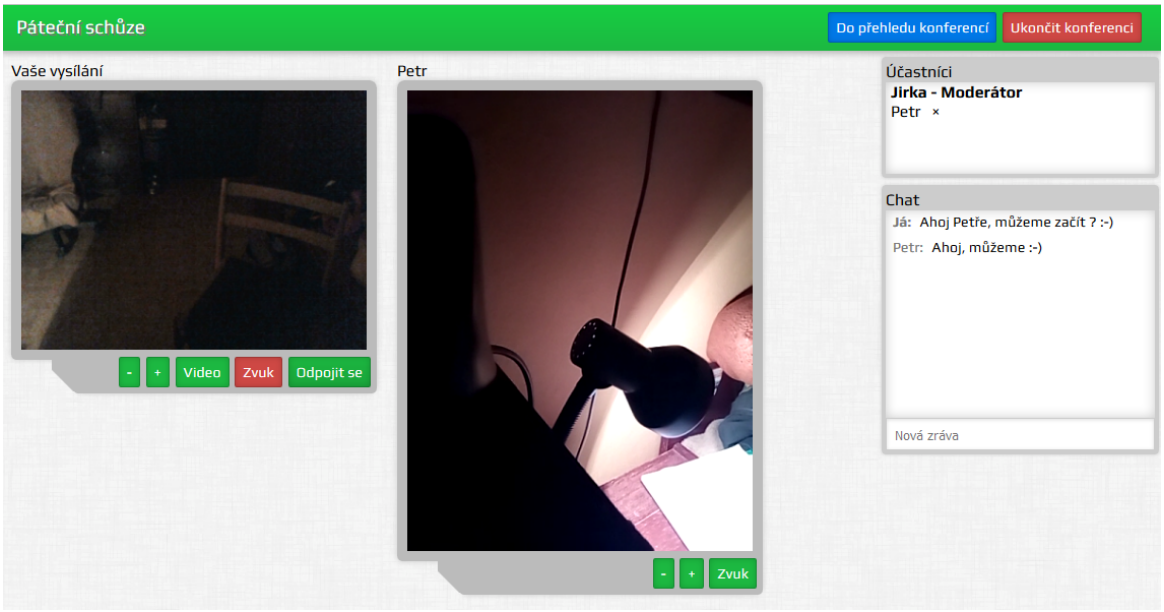
Stream každého klienta je reprezentován vizuálním blokem s ovládacími nástroji. Tato okna příchozích streamů jsou vytvořena již při přijetí SDP nabídky, potažmo odpovědi metodou ”newClientFrame”objektu



Obrázek 15: Přihlášení účastníků do konferenční místnosti

View, který obsluhuje požadavky spojené s grafickými změnami a manipulací s elementy místnosti. Ve struktuře html místnosti je vytvořené skryté klientské okno. To je při volání této metody naklonováno, jsou změněny jeho identifikátory podle připojovaného klienta, nastaví se klientovo jméno a následně je zobrazeno. Všechny klientské streamy se vyskytují v divisionu s identifikátorem ”client-frames”.

V rámci průběhu konference může nastat situace, kdy 2 klienti začnou zahajovat komunikaci v téměř stejný okamžik. Proto vždy později nabízející nové spojení ukončí předešlé RTCPeerConnection druhého klienta a v callbacku tento klient zahájí novou komunikaci, aby nedocházelo ke kolizím. Odpojí-li klient vlastní stream, je pro zřízení spojení využito broadcastového signálu ”call-me”, který vyzve klienty místnosti, aby mají-li k dispozici nějaký stream, zahájili alespoň jednosměrnou komunikaci a klient, jež odpojil vlastní média, byl pouhým příjemcem ostatních.



Obrázek 16: Konferenční místnost

Tlačítky zvuku a videa lze u připojeného klienta povolit či zakázat sdílení daného média. Jsou tak přenastavovány parametry konfiguračního objektu pro získání médií streamu. Je-li zakázáno streamování média v průběhu spojení s některým z dalších klientů, stream je stále aktivní, ale přenáší data, která již nenesou informaci daného média. WebRTC umožňuje v rámci konfigurace uživatelských médií nastavení velikosti streamu videa. To je v systému implementováno volbou 4 variant velikostí. Výchozí je šířka 320 px. Zařízení nastavují nejbližší možnou velikost uvedenou v zadané konfiguraci. Ukázka metody:

```
echo.joinMyMediaIntoConference = function()
{
    echo.closeRemotePeers();
    echo.closeLocalStream();
    view.openDialog("allowing-media");
    echo.setLocalStream(function () {
        echo.connectPeers();
        view.closeDialog('allowing-media');
        view.userMediaBtn.text('Odpojit se');
    }, function () {
        alert("Nebyl nastaven mikrofon ani kamera.");
        echo.callMe();
        view.closeDialog('allowing-media');
        echo.config.usedMediaConstraints.video = false;
        echo.config.usedMediaConstraints.audio = false;
    });
};
```

Obrázek 17: Nastavení parametrů streamovaného videa

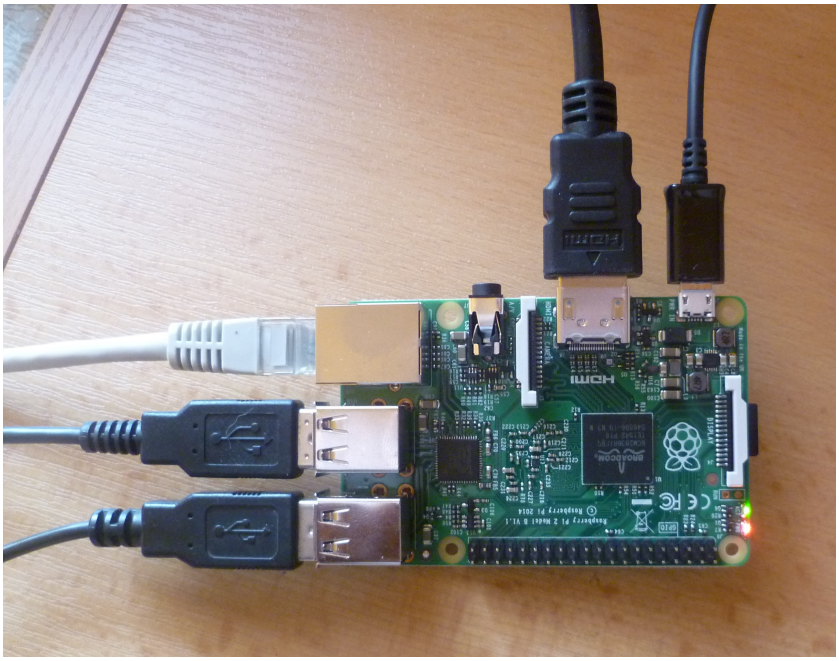
Při připojení vlastních médií do konference je volána metoda ”getStream”, která získá MediaStream. Ten je následně nastaven metodou ”setLocalStream”do okna lokálního streamu klienta. Jedná-li se pouze o médium typu audio, je okno doplněno informativním textem. Následuje proces, který zahájí generování SDP popisu a jeho rozesílání klientům v místnosti, která metodou ”connectPeers”provede tento proces pro všechny připojené klienty. Posléze RTCPeerConnection generuje událostní metodou nad tímto objektem ”onicecandidate”ICE kandidáty pro výměnu síťových informací a skrze signální server zasílá kandidáty klientům. Ti je nastavují v metodě ”onCandidate”danému RTCPeerConnection. Podmínkou je, aby stav

daného spojení nebyl nastaven již na ”closed”. Ve chvíli, kdy jsou vzájemně nalezeny vhodné parametry spojení, je streamování automaticky započato.

6.3.3 Signální server

6.3.3.1 Využitý Hardware

Webové sockety vyžadují běh serverové aplikace, jež bude obsluhovat příchozí signální požadavky. S ohledem na možnosti přímého testování celé aplikace je využito vlastního zařízení, které je připojeno k internetové síti prostřednictvím veřejné IP adresy. Použit je miniaturizovaný počítač Raspberry Pi 2 s operačním systémem Raspbian[40], jež je založen na známé linuxové distribuci Debian.



Obrázek 18: Raspberry Pi 2

6.3.3.2 Webový server a šifrovaná komunikace pomocí SSL

Realizaci webového serveru, který umožní přijímat požadavky pro signální aplikaci, zajišťuje balíček ”Apache2”, který není součástí předinstalovaných programů po instalaci systému. Zmiňované příkazy terminálového okna vyžadují administrátorská oprávnění.

- `apt-get install apache2`

Šifrovaná komunikace je vyžadována pro chod WebRTC komponent v prohlížečích Google Chrome a Opera. Aby mohla být webovými sockety realizována komunikace mezi signálním a webovým serverem s aplikací konferenčního systému, je též nutné zajistit šifrovanou komunikaci prostřednictvím SSL certifikátu. Ty je možné zakoupit či vytvořit vlastní tzv. self-signed certifikáty. Self-signed certifikát zajišťuje šifrovanou komunikaci, nicméně neumožňuje ověřit identitu serveru, k němuž se klient připojuje. Pro vytvoření vlastního certifikátu bylo využito balíčku ”openssl”, který je na této distribuci již předinstalován.

[40] Přístupný na adrese https://www.raspbian.org

Pro korektní práci s tímto balíčkem je provedena jeho aktualizace. Po jejím provedení je nutné aktivovat ssl modul a umožnit jeho konfiguraci v balíčku Apache.

- `apt-get upgrade openssl`
- `a2enmod ssl`
- `a2ensite default-ssl`

Pro příkaz vytvářející certifikát jsou specifikovány potřebné parametry. Důležitým parametrem je přepínač ”-x509”, který definuje výstup jako self-signed certifikát. Umístěn je do adresáře ”/etc/apache2/ssl”. Během vytváření samotného certifikátu jsou dále vyplněny samotné údaje o certifikátu. Následně je zamezeno neoprávněnému přístupu k certifikátu změnou oprávnění.

- `mkdir /etc/apache2/ssl`
- `openssl req -x509 -nodes -days 365 -newkey rsa:2048`
  `-keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt`
- `chmod 600 /etc/apache2/ssl/*`

Další částí je konfigurace Apache, tak aby používal šifrované spojení. Ta se nachází v souboru ”/etc/apache2/sites-enabled/default-ssl.conf”, kde byly nastaveny údaje pro vygenerovaný ssl certfikát, tedy port a adresa zařízení, na kterém bude aplikace spuštěna. Minimální konfigurace obnáší v rámci sekce Virtual Host nastavení portu: ⟨VirtualHost _default_:3000⟩ a dále její parametr ”ServerName”s adresou a portem ve tvaru ”adresa:port”. Výchozí nastavení portu je 443, ale není nutné jej využít. V rámci práce je používáno portu 3000. K přiřazení cesty pro SSL klíče a certifikátu slouží konfigurační parametry ”SSLCerticate-File”a ”SSLCertificateKeyFile”. Nedílnou součástí je i zajištění konfigurace routeru tak, aby umožňoval zpracovávání požadavků příchozích na port 3000 zařízením Raspebbry Pi. K tomu je využito nastavení ”Forwarding”společně s rezervací IP adresy v rámci privátní sítě.

6.3.3.3 Instalace potřebného software

Samotná signální aplikace je realizována na platformě Node zracovávající javascript na straně serveru. Balíček Node je na této linuxové distribuci již obsažen, ale pro získání některé implementace webových socketů je instalován i správce jeho balíčků - npm.

- `apt-get install npm`

Celá část signalizačního serveru je umístěna do pracovního adresáře ”echo-signal-server”. V tomto adresáři je inicializováno vytvoření projektu pomocí příkazu ”npm init”, který posléze vyžaduje vyplnění údajů o projektu, autorovi a verzi vyvíjené aplikace. Vzniká soubor ”package.json”, který kromě těchto informací uchovává též informace o použitých závislostech projektu - tedy programových knihovnách stahovaných ze vzdáleného repozitáře. Projekt vyžaduje balíček implementující webové sockety. Ten je uložen do adresáře ”node_modules”, kde jsou jednotlivé balíčky uchovávány. Kromě webových socketů je využito také balíčku pro práci s odesílání HTTP respektive HTTPS požadavků, jež je využit pro komunikaci s aplikací konferenčního systému.

- `npm init`
- `npm install --save websocket`
- `npm install --save request`

6.3.3.4 Program signálního serveru

Program signálního serveru je psán v jazyce javascript a je zpracováván již zmiňovanou platformou Node. Nejprve jsou prostřednictvím Node načteny potřebné knihovny: websocketový server, http (https) server, request a knihovnu umožňující práci se soubory, s jejíž pomocí je zajištěn přístup k vygenerovanému ssl certifikátu.

Prvotně je vytvořena instance třídy "EchoSignalServer". Nad tímto objektem je zavolána jeho metoda "fire". Následně probíhá inicializace websocketového serveru a https serveru, jehož parametrem je načtený ssl certifikát. Tento server neslouží pro generování grafického výstupu, ale pro řízení websocketové komunikace, a proto neposkytuje zobrazitelné rozhraní. Následně jsou nastaveny událostní metody pro obsloužení příchozích požadavků. Posledním krokem je samotné zahájení jejich příjmu.

Proces zpracování požadavku začíná kontrolou jeho původu. Tím je doména, z níž byl požadavek zaslán. Dále je kontrolován dohodnutý vysílací protokol webových socketů a ověření typu zprávy. Formát zasíláné zprávy je vždy výhradně text v podobě serializovaného objektu, jež je definován jednotným rozhraním "Signal"v klientské části.

Metoda zpracovávající signál "processSignal"je řešena stejným způsobem jako v klientské části implementací řídící struktury switch. Na základě typu signálu je zvolena odpovídající akce. Ukázka implementace této metody:

```
self.processSignal = function(signal, connection){
    if(signal.type == 'registerClient'){
        self.onRegisterClient(signal, connection);
        return;
    }
    var success;
    switch (signal.type) {
        case 'chat-msg' : success = self.onChatMessage; break;
        case 'sdp-offer': success = self.onSDPOffer; break;
        // ...
    }
    if(success){
        self.processVerified(signal, connection, success);
    } else {
        connection.close(4000);
    }
};
```

Všechny požadavky, jež vyžadují účastníkovu předešlou autorizaci, využívají metody ”processVerified”. Jako parametr této funkce je vkládán i callback, jež je proveden po úspěšném ověření registrace klienta v místnosti. Kromě zaevidování účastníka do místnosti je této switche využito ve všech ostatních případech. Pokud klient není registrován, ale zasílá signál, jež vyžaduje předešlou autorizaci, komunikace je ukončena. Při používání webových socketů je pro ukončování spojení využíváno také vlastních chybových kódů v rozsahu 4000-4999, jak uvádí jejich specifikace dle RFC6455[41]. Ukázka metody ”processVerified”:

```
self.processVerified = function(signal, connection, successCallback)
{
    if(!rooms.isClientRegistered(signal.roomID, signal.clientID, signal.clientKey)){
        connection.close(4001);
    } else {
        successCallback(signal);
    }
};
```

Každá místnost je reprezentovaná objektem, který disponuje atributy ”roomID”a ”clients”. Jednotliví klienti jsou definováni samostatným objektem. Ten je generován a evidován metodou ”registerClient”. Jednou z uchovávaných hodnot každého klienta je i objekt ”WebsocketConnection”reprezentující propojení s daným účastníkem. Tohoto objektu je využíváno pro zaslání signálu danému klientovi.

Pro ověření existence klienta v konferenčním systému při websoketovém požadavku na jeho registraci do místnosti je využito API konferenčního systému, které zajišťuje třída ”SignalServerAPIPresenter”. Na základě dohodnutého klíče a dalších vstupních parametrů je signální aplikaci vrácen objekt JSON načteného moderátora či účastníka. Pokud se připojovaný klient v databázi nenachází či je z místnosti vyloučen, nebo je místnost ukončena, je i websoketová komunikace zastavena.

6.3.4 Nasazení STUN a TURN serveru

Pokud se nepodaří navázat přímé spojení, TURN server je posledním mechanizmem, který je využit při zřizování multimediální komunikace pomocí WebRTC. V konferenčním systému je využito implementace TURN serveru zvaného Coturn[42]. Jedná se o open-source řešení, jehož vývoj byl iniciován společností Google. Implementace STUN serveru je součástí balíčku Coturn.

Aby bylo možné testovat WebRTC technologii v rámci propojování mezi veřejnými sítěmi a privátní sítí, ve které je připojeno i Raspberry Pi se spuštěným TURN serverem, je využito veřejného STUN serveru, který je poskytován společností Google. Adresa a port použitého STUN serveru je ”stun2.l.google.com:19302”.

- `apt-get install coturn`

Konfigurace Coturn serveru je nastavována v souboru ”/etc/turnserver.conf”. Pro šifrovanou komunikaci je třeba vyplnit parametry pro ssl certifikát a klíč: ”cert”a ”pkey”příslušnými soubory ”apache.crt”a

[41] RFC6455 dostupné z https://tools.ietf.org/html/rfc6455

[42] projekt přístupný na adrese https://github.com/coturn/coturn

”apache.key”. Výchozím portem šifrované komunikace je port 5349. Pro nešifrovanou komunikaci je výchozím portem 3478. Zvolený port je opět nutné v routeru zpřístupnit. Dalším parametrem, který je nutné nastavit, je ”realm”, který obsahuje doménu, z níž je požadavek odesílán.

Výchozí stav umožňuje přeposílání multimediálního obsahu bez nutnosti autorizace. Jednou z možností jejího nasazení je definice uživatelů, která se nachází v souboru ”/etc/turnuserdb.conf”. Tento soubor je dynamicky načítán při příchozím dotazu tak, aby bylo možné jej měnit i při spuštěném TURN serveru. Každý uživatel je uveden na samostatné řádce a to ve tvaru ”uzivatelskeJmeno:heslo”. V rámci implementace prototypu konferenčního systému je využito společného uživatelského účtu. Přístup z klientského javascriptu probíhá prostřednictví RTCPeerConnection, kde jedním z parametrů konfigurace jeho konstruktoru je i TURN server s přístupovými údaji viz. kapitola 5.2.4. Spuštění serveru je provedeno příkazem ”turnserver”.

6.4 Testování WebRTC

Kvalita přenosu je závislá na dané síti, skrz níž jsou jednotliví klienti připojováni. Dalším faktorem, který ovlivňuje kvalitu samotného streamování, jsou koncová zařízení, jež streamy distribuují či přijímají a to v závislosti na jejich výkonu, typu síťového připojení, využívaného prohlížeče či instalovaného operačního systému. Vzhledem k rozsahu této problematiky je testování WebRTC technologií prováděno na základě vjemu pozorováním streamovaných médií připojenými klienty.

6.4.1 Streamování mezi 2 klientskými zařízeními

Toto testování obnášelo propojení vždy 2 klientských zařízení napříč internetovou sítí. V těchto testových situacích náleželo každé ze zařízení do jiné sítě - bylo za jedním či více překladači IP adres. Plošně se jednalo o vzdálenosti vždy delší než 20 km. Bylo provedeno 10 testových situací - 10 různých sítí a míst, přičemž každý test byl uskutečněn 3x. V prvním případě se jednalo o propojení běžných počítačů, ve druhém o propojení jednoho mobilního zařízení a počítače. Ve třetím případě byla obě zařízení mobilní. Používanými platformami byly na běžném počítači Windows a na mobilních zařízeních Android.

Ve 4 testových situacích probíhalo streamování zcela plynule. Nevykazovalo známky zpoždění či významnější zhoršení kvality obrazu či zvuku. To platilo i s použitím mobilních zařízení.

V dalších 3 testových situacích, sítích, docházelo naopak ke značnému vizuálnímu posunu streamovaného audia a videa v odhadovaném zpoždění v řádech 3-6 sekund. Vyjímkou nebylo vynechávání částí přenášeného videa. Audio složka při delší prodlevě nad 3 sekundy přestala být reprodukována. V jedné z těchto situací docházelo též ke značnému a opakovanému rozpadu obrazu a trhanému přenosu se změnou snímku až po 10 sekundách při testování všech kombinacích výše uvedených zařízení.

Ve 3 dalších situacích docházelo k občasnému vynechání části videa či zpoždění streamu do 3 sekund, nejednalo se však o tak markantní projevy jako u předchozích 3 situací. Audio bylo konstantně přenášeno, ovšem s občasnými chybami.

V testovaných případech byla využívána mobilní zařízení, jež disponovala 4 jádrovými procesory a operační pamětí odpovídající či vyšší než 1 GB. Ta dosahovala ve všech případech shodné kvality přenosu jako na běžném počítači. V rámci testování docházelo k přerušovanému a trhanému streamování či příjmu

streamu bez reprodukovaného audio signálu na všech mobilních zařízeních, jež používala jedno jádrový procesor s operační pamětí v rozsahu 512 MB - 1 GB. Nízká kvalita se projevovala i přes to, že komunikace mezi počítači na téže síti nevykazovala značné poruchy.

V případech, kdy bylo streamování současně audio i video velmi nekvalitní, byla provedena zkouška pouze audio hovoru. V rámci těchto testování bylo reprodukování zvuku plynulé a docházelo k minimálním pozorovatelným chybám.

6.4.2 Streamování mezi více zařízeními

Prvotním testem bylo pouze streamování audia 5 zařízeními. Během této konference byl přenášený zvuk plynulý a docházelo k občasným chybám jen některých zúčastněných. Posléze bylo připojeno též i video. Jeho kvalita se nejevila být ovlivněna počtem účastníků, přičemž docházelo k občasnému zpoždění některých klientů stejně jako v předešlých testech s pouhými dvěma členy konference. Testování probíhalo jak s běžnými počítači, tak s mobilními zařízeními.

Posléze bylo do konferenční místnosti připojeno v jeden okamžik celkem 10 klientů. Jednalo se o 5 počítačů a 5 mobilních zařízení. I testování přenosu samotného audia probíhalo s opožděním některých streamů a výpadky dílčích úseků většiny připojených klientů. Při následném připojení video streamu každý z klientů vykazoval zcela chybějící audio z několika jiných zařízení. Oproti předchozímu testu byla zaznamenána vyšší doba pro zřizování samotného spojení od 4 sekund a více. Všichni klienti vykazovali, že v alespoň polovině přijímaných video streamech docházelo ke značným chybám a zpoždění streamů v řádu sekund. Ani jeden z obou testových případů nedovoloval plynulou konferenci. Nicméně zkolabování streamového spojení jako takového nebylo indikováno žádným z provedených testů. Dvě klientská zařízení byla v průběhu konference odpojena od signálního serveru v důsledku výpadku websocketového spojení.

7 Závěr

V práci bylo dosaženo všech cílů, které byly vytyčeny. Analýza konferenčních systémů ukázala, že pouze 3 z vybraných jsou webově založené. Ty využívají pro přenos audio a video technologii Flash a pro jejich přístupnost z mobilních zařízení do konference používají vlastní aplikace. Na základě realizované analýzy bylo vytvořeno jejich srovnání a sestaven obecný koncept konferenčního systému.

Nasazení technologie WebRTC v konferenčním systému bylo úspěšně realizováno. Umožňuje přístup k uživatelským zařízením jako je mikrofon či webkamera a zároveň umožňuje zřizovat peer-to-peer sítě pro streamování získaných médií nebo dat napříč internetovou sítí z prostředí prohlížeče. Fáze a technologie nutné pro zřízení spojení byly popsány a implementovány. Nicméně peer-to-peer komunikaci není možné realizovat ve všech případech. Jak bylo zjištěno, tato technologie je schopna takový stav detekovat a použít mechanizmus, který provede přemostění této komunikace přes tzv. TURN server, jehož implementace byla též úspěšně realizována. Systém lze využívat i na mobilních zařízeních bez nutnosti instalace speciálního software.

Analýza WebRTC také ukázala, že tato technologie není podporována všemi rozšířenými prohlížeči v závislosti na použité platformě. V rámci těch podporovaných se její samotná implementace v určitých aspektech navzájem odlišuje.

Primárním cílem bylo vytvořit systém, který umožní streamovat audio a video pomocí WebRTC technologie, čehož bylo úspěšně dosaženo. Systém je schopen pořádat paralelně běžící konferenční setkání. Do systému je možné se registrovat a pořádat vlastní konference, nebo se připojit jako běžný účastník. V konferenční místnosti je možné být pouhým příjemcem multimediálního obsahu, streamovat audio, video či obě média zároveň. Streamování lze v průběhu konference kdykoli přerušit, ztlumit a nebo se opět připojit. Místnost disponuje také společným online chatem. Moderátor má kontrolu nad a všemi členy.

Testování streamování představuje rozsáhlou problematiku jak z pohledu sítí, tak koncových zařízení či použitého softwaru, a proto by tato oblast mohla být předmětem další práce. Nicméně na základě pozorování bylo zjištěno, že streamování médií pomocí této technologie v současné době nedosahuje ve všech případech kvality, která by umožnila nepřerušovanou konferenci. V rámci testování technologie docházelo v nezanedbatelném množství případů k opakovanému zkreslení či zpoždění přenášeného streamu v řádech sekund. Naopak při přenosu samotné audio složky docházelo ve všech testovaných případech kromě hromadného připojení 10 klientů pouze k minimálnímu posunu či chybám. Mobilní zařízení nižších technických parametrů streamovala ve značně snížené kvalitě, která se projevovala úplnou ztrátou zvuku a trhaným přenosem videa oproti výkonějším zařízením.

8 Literatura a použité zdroje

[1] About W3C. *W3C* [online]. [cit. 2016-03-11]. Dostupné z: https://www.w3.org/Consortium/

[2] The Unified Modeling Language. *UML diagrams* [online]. [cit. 2016-03-12]. Dostupné z: http://www.uml-diagrams.org/

[3] ROUSE, Margaret. Web conferencing. In: *TechTarget* [online]. 2008 [cit. 2016-03-08]. Dostupné z: http://searchunifiedcommunications.techtarget.com/definition/Web-conferencing

[4] Internet Streaming: What It Is and How It Works *About tech* [online]. 2015, 14.10.2015 [cit. 2015-11-20]. Dostupné z: http://ipod.about.com/od/glossary/g/streaming_def.htm

[5] ŠUBRT, Filip. Co je to webinář? In: Inflow [online]. 2011 [cit. 2016-03-15]. Dostupné z: http://www.inflow.cz/co-je-webinar

[6] ROUSE, Margaret. Webcast. In: *TechTarget* [online]. 2005 [cit. 2016-03-15]. Dostupné z: http://searchnetworking.techtarget.com/definition/Webcast

[7] MILLER, Michael J. Living History: Retracing the Evolution of the PC and PC Magazine. In: *PC Magazine* [online]. 2002 [cit. 2016-03-16]. Dostupné z: http://www.pcmag.com/article2/0,2817,7388,00.asp

[8] MCLAUGHLIN, Molly K. The Best Video Conferencing Services of 2015. In: *PC Magazine* [online]. 2015 [cit. 2016-03-16]. Dostupné z: http://www.pcmag.com/article2/0,2817,2388678,00.asp

[9] MASINTER, L., H. ALVESTRAND, D. ZIGMOND a Z. PETKE (eds.). Guidelines for new URL Schemes: RFC2718. In: *The Internet Engineering Task Force* [online]. 1999 [cit. 2016-03-08]. Dostupné z: http://www.ietf.org/rfc/rfc2718

[10] Drag and Drop. In: *The Tech Terms Computer Dictionary* [online]. 2011 [cit. 2016-03-08]. Dostupné z: http://techterms.com/definition/drag_and_drop

[11] GOUAILLARD, Alexandre. *Webrtc in Safari?* [online]. 2015 [cit. 2016-02-23]. Dostupné z: http://webrtcbydralex.com/index.php/2015/08/29/webrtc-in-safari

[12] MACICH, Jiří. WebRTC: online komunikace bez plug-inů v prohlížeči. *Root.cz: Informace nejen ze světa linuxu* [online]. 2012 [cit. 2015-11-24]. ISSN 1212-8309. Dostupné z: http://www.root.cz/clanky/webrtc-online-komunikace-bez-plug-inu-v-prohlizeci

[13] BERGKVIST, Adam, Daniel C. BURNETT, Cullen JENNINGS, Anant NARAYANAN a Bernard ABOBA (eds.). WebRTC 1.0: Real-time Communication Between Browsers: W3C Working Draft 28 January 2016. In: *W3C* [online]. 2016 [cit. 2016-03-02]. Dostupné z: https://www.w3.org/TR/2016/WD-webrtc-20160128

[14] DUTTON, Sam. WebRTC in the real world: STUN, TURN and signaling. In: *HTML5 Rocks* [online]. 2013 [cit. 2016-02-22]. Dostupné z: http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

[15] DUTTON, Sam. Getting Started with WebRTC. In: *HTML5 Rocks* [online]. 2013 [cit. 2016-02-22]. Dostupné z: http://www.html5rocks.com/en/tutorials/webrtc/basics/

[16] XMLHttpRequest. *Mozilla developer network* [online]. [cit. 2016-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

[17] Media Capture and Streams: W3C Last Call Working Draft 14 April 2015. *W3C* [online]. 2015 [cit. 2016-02-25]. Dostupné z: https://www.w3.org/TR/2015/WD-mediacapture-streams-20150414/

[18] KHAN, Muaz. WebRTC Tips & Tricks [online]. In: . 2014 [cit. 2016-03-16]. Dostupné z: http://muaz-khan.blogspot.cz/2014/05/webrtc-tips-tricks.html

[19] BERGKVIST, Adam, Daniel C. BURNETT, Cullen JENNINGS, Anant NARAYANAN a Bernard ABOBA (eds.). WebRTC 1.0: Real-time Communication Between Browsers: W3C Working Draft 10 February 2015. In: *W3C* [online]. 2016 [cit. 2016-03-02]. Dostupné z: https://www.w3.org/TR/2015/WD-webrtc-20150210

[20] Definition of User Agent. *W3C* [online]. [cit. 2016-03-15]. Dostupné z: https://www.w3.org/WAI/UA/work/wiki/Definition_of_User_Agent

[21] GRIGORIK, Ilya. *High Performance Browser Networking* [online]. 2013. O'Reilly Media, 2013 [cit. 2016-02-26]. ISBN 978-1-4493-4471-9. Dostupné z: http://chimera.labs.oreilly.com/books/1230000000545/ch18.html

[22] KHAN, Muaz. RTCDataChannel for Beginners. In: *WebRTC Experiments & Demos* [online]. [cit. 2016-04-12]. Dostupné z: https://www.webrtc-experiment.com/docs/rtc-datachannel-for-beginners.html

[23] Navigator. *Mozilla developer network* [online]. [cit. 2016-02-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator

[24] LAZARIS, Louis. Callback Functions in JavaScript. In: *Impressive Webs* [online]. 2012 [cit. 2016-03-15]. Dostupné z: http://www.impressivewebs.com/callback-functions-javascript

[25] Promise. *Mozilla developer network* [online]. [cit. 2016-02-26]. Dostupné z: https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise

[26] HANDLEY, M. a V. JACOBSON. SDP: Session Description Protocol: RFC2327. In: *The Internet Engineering Task Force* [online]. 1998 [cit. 2016-03-01]. Dostupné z: https://tools.ietf.org/html/rfc2327

[27] HANDLEY, M., V. JACOBSON a C. PERKINS. SDP: Session Description Protocol: RFC 4566. In: *The Internet Engineering Task Force* [online]. 2006 [cit. 2016-04-12]. Dostupné z: http://tools.ietf.org/html/rfc4566

[28] ROSENBERG, J. a H. SCHULZRINNE. An Offer/Answer Model with the Session Description Protocol (SDP): RFC 3264. In: *The Internet Engineering Task Force* [online]. 2002 [cit. 2016-04-12]. Dostupné z: http://tools.ietf.org/html/rfc3264

[29] ROUSE, Margaret. Network Address Translation (NAT). In: *TechTarget* [online]. 2015 [cit. 2016-03-05]. Dostupné z: http://searchenterprisewan.techtarget.com/definition/Network-Address-Translation

[30] SCHULZRINNE, H., S. CASNER, R. FREDERICK a V. JACOBSON (eds.). RTP: A Transport Protocol for Real-Time Applications. In: *The Internet Engineering Task Force* [online]. 2003 [cit. 2016-03-15]. Dostupné z: https://tools.ietf.org/html/rfc3550

[31] What is SIP?. *Network world* [online]. 2004 [cit. 2016-03-01]. Dostupné z: http://www.networkworld.com/article/2332980/lan-wan/what-is-sip-.html

[32] ROSENBERG, J. (ed.). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. In: *The Internet Engineering Task Force* [online]. 2010 [cit. 2016-03-03]. Dostupné z: https://tools.ietf.org/html/rfc5245

[33] ROSENBERG, J., H. SCHULZRINNE, G. CAMARILLO, A. JOHNSTON, J. PETERSON, R. SPARKS, M. HANDLEY a E. SCHOOLER. SIP: Session Initiation Protocol: RFC3261. In: *The Internet Engineering Task Force* [online]. 1998 [cit. 2016-03-01]. Dostupné z: https://tools.ietf.org/html/rfc3261

[34] Multicast-Unicast Conversion. In: *Cisco Meraki* [online]. [cit. 2016-03-15]. Dostupné z: https://documentation.meraki.com/MR/Other_Topics/Multicast-Unicast_Conversion

[35] IP Packet switching in Telecom - Part 4. *TelecomHall* [online]. 2012 [cit. 2016-03-01]. Dostupné z: http://www.telecomhall.com/ip-packet-switching-in-telecom-part-4.aspx

[36] Javascript Session Establishment Protocol: draft-ietf-rtcweb-jsep-12. *The Internet Engineering Task Force* [online]. 2015 [cit. 2016-02-29]. Dostupné z: https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-12

[37] ROUSE, Margaret. Request for Comments (RFC). In: *TechTarget* [online]. 2011 [cit. 2016-03-15]. Dostupné z: http://whatis.techtarget.com/definition/Request-for-Comments-RFC

[38] ROUSE, Margaret. VoIP (voice over IP). In: *TechTarget* [online]. 2008 [cit. 2016-03-15]. Dostupné z: http://searchunifiedcommunications.techtarget.com/definition/VoIP

[39] Browser Statistics. *W3schools* [online]. 2015 [cit. 2016-02-25]. Dostupné z: www.w3schools.com/browsers/browsers_stats.asp

[40] GetUserMedia/Stream API. *Can i use* [online]. [cit. 2016-02-24]. Dostupné z: http://caniuse.com/#feat=stream

[41] WebRTC Peer-to-peer connections. *Can i use* [online]. [cit. 2016-02-24]. Dostupné z: http://caniuse.com/#feat=rtcpeerconnection

[42] MediaDevices. *Mozilla developer network* [online]. [cit. 2016-02-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices

[43] Navigator.getUserMedia(). *Mozilla developer network* [online]. [cit. 2016-02-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia

[44] Is Skype a VoIP Service or VoIP App? *About.com: Voice Over IP* [online]. 2014 [cit. 2016-03-01]. Dostupné z: http://voip.about.com/od/voipsoftware/f/Is-Skype-A-Voip-Service-Or-Voip-App.htm

[45] The Future of SIP in WebRTC: SIP is Dead! Long Live SIP!. *Sight Call* [online]. [cit. 2016-03-01]. Dostupné z: http://www.sightcall.com/future-of-sip-in-webrtc/

[46] RTCDataChannel. In: Mozilla developer network [online]. [cit. 2016-03-15]. Dostupné z: https://developer.mozilla.org/en/docs/Web/API/RTCDataChannel

[47] RISTIC, Dan. WebRTC data channels. In: *HTML5 Rocks* [online]. 2014 [cit. 2016-03-15]. Dostupné z: http://www.html5rocks.com/en/tutorials/webrtc/datachannels/

[48] ROUSE, Margaret, Brendan CUSACK a Michael COBB. Secure Sockets Layer (SSL). In: *TechTarget* [online]. 2014 [cit. 2016-03-15]. Dostupné z: http://searchsecurity.techtarget.com/definition/Secure-Sockets-Layer-SSL

[49] Public and Private Addresses. *Microsoft TechNet* [online]. [cit. 2016-03-20]. Dostupné z: https://technet.microsoft.com/en-us/library/cc958825.aspx

[50] ROUSE, Margaret. Web server. In: *TechTarget* [online]. [cit. 2016-03-20]. Dostupné z: http://whatis.techtarget.com/definition/Web-server

[51] FETTE, I. a A. MELNIKOV (eds.). The WebSocket Protocol. In: *Internet Engineering Task Force* [online]. 2011 [cit. 2016-03-22]. Dostupné z: https://tools.ietf.org/html/rfc6455

[52] Noobs. *Raspberry Pi* [online]. [cit. 2016-03-22]. Dostupné z: https://www.raspberrypi.org/downloads/noobs/

[53] Websocket. *NPM* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/websocket

[54] Request. *NPM* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/request

[55] WAN, Alvin. How To Create a SSL Certificate on Apache for Debian 8. In: *DigitalOcean* [online]. 2015 [cit. 2016-03-20]. Dostupné z: https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-debian-8

[56] Free open source implementation of TURN and STUN Server. *Github* [online]. 2014 [cit. 2016-03-22]. Dostupné z: https://github.com/coturn/coturn

[57] *Apache: HTTP server project* [online]. [cit. 2016-04-15]. Dostupné z: https://httpd.apache.org

[58] *ClickMeeting* [online]. [cit. 2016-04-12]. Dostupné z: http://www.clickmeeting.com

[59] *Join me* [online]. [cit. 2016-04-12]. Dostupné z: https://www.join.me

[60] *Adobe Connect* [online]. [cit. 2016-04-12]. Dostupné z: http://www.adobe.com/products/adobeconnect.html

[61] *Cisco WebEx* [online]. [cit. 2016-04-12]. Dostupné z: https://www.webex.com

[62] *Citrix GoToMeeting* [online]. [cit. 2016-04-12]. Dostupné z: http://www.gotomeeting.com

[63] *eVoice* [online]. [cit. 2016-04-12]. Dostupné z: https://www.evoice.com

[64] *Skype for bussines* [online]. [cit. 2016-04-12]. Dostupné z: https://www.skype.com/en/business/skype-for-business

[65] *Onstream Meetings* [online]. [cit. 2016-04-12]. Dostupné z: http://www.onstreammedia.com

[66] *StartMeeting* [online]. [cit. 2016-04-12]. Dostupné z: https://www.startmeeting.com

[67] *InterCall* [online]. [cit. 2016-04-12]. Dostupné z: http://www.intercall.com

[68] *Node* [online]. [cit. 2016-03-22]. Dostupné z: https://nodejs.org

[69] *Less: Getting started* [online]. [cit. 2016-04-16]. Dostupné z: http://lesscss.org/

[70] *Gulp: Automate and enhance your workflow* [online]. [cit. 2016-04-16]. Dostupné z: http://gulpjs.com

[71] *gulp-less* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/gulp-less

[72] *less-plugin-clean-css* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/less-plugin-clean-css

[73] *gulp-uglify* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/gulp-uglify

[74] *gulp-concat* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com/package/gulp-concat

[75] *NPM* [online]. [cit. 2016-03-22]. Dostupné z: https://www.npmjs.com

9 Seznam obrázků

10 Seznam tabulek

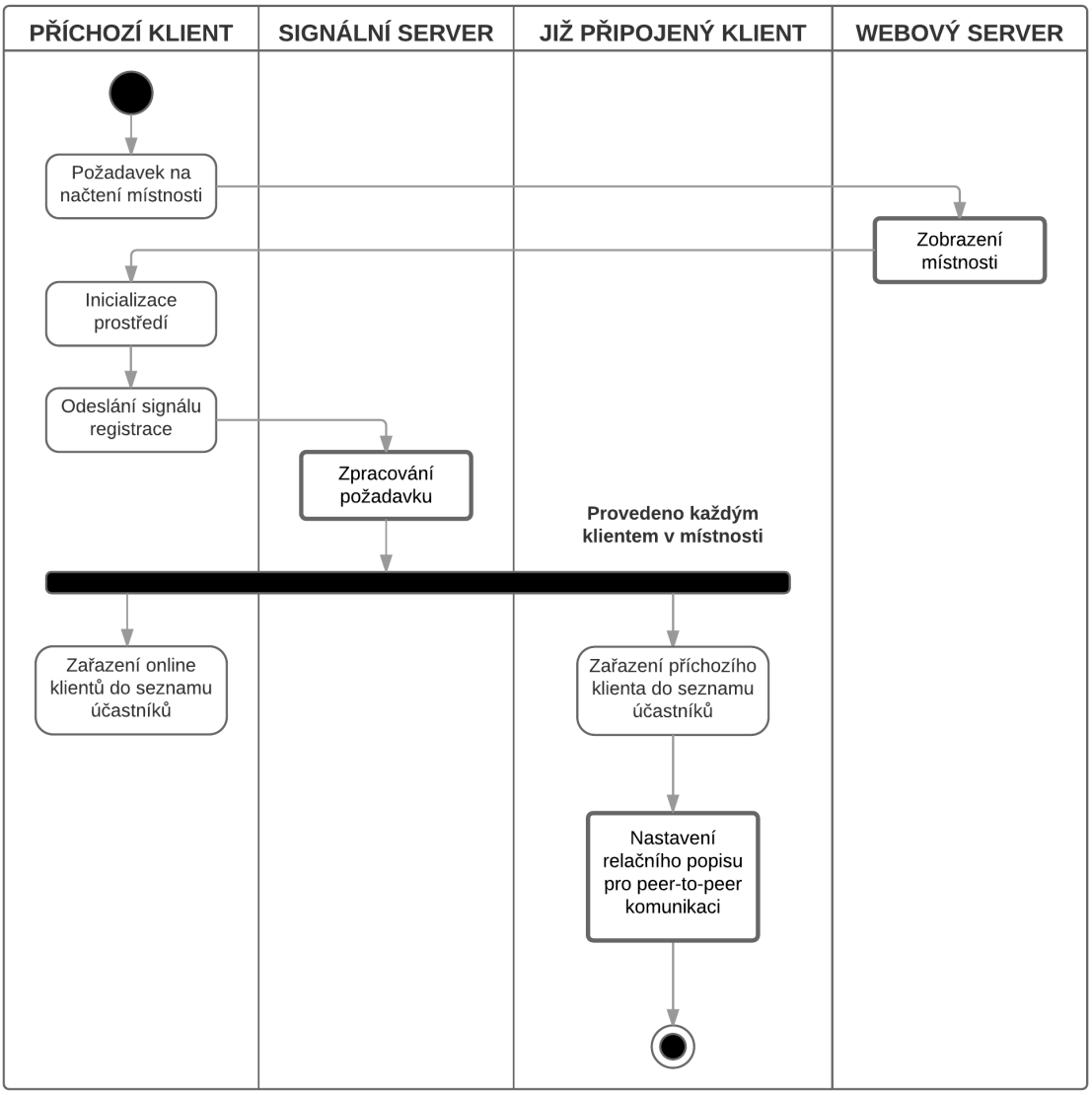
11 Přílohy

11.1 Přiložené CD

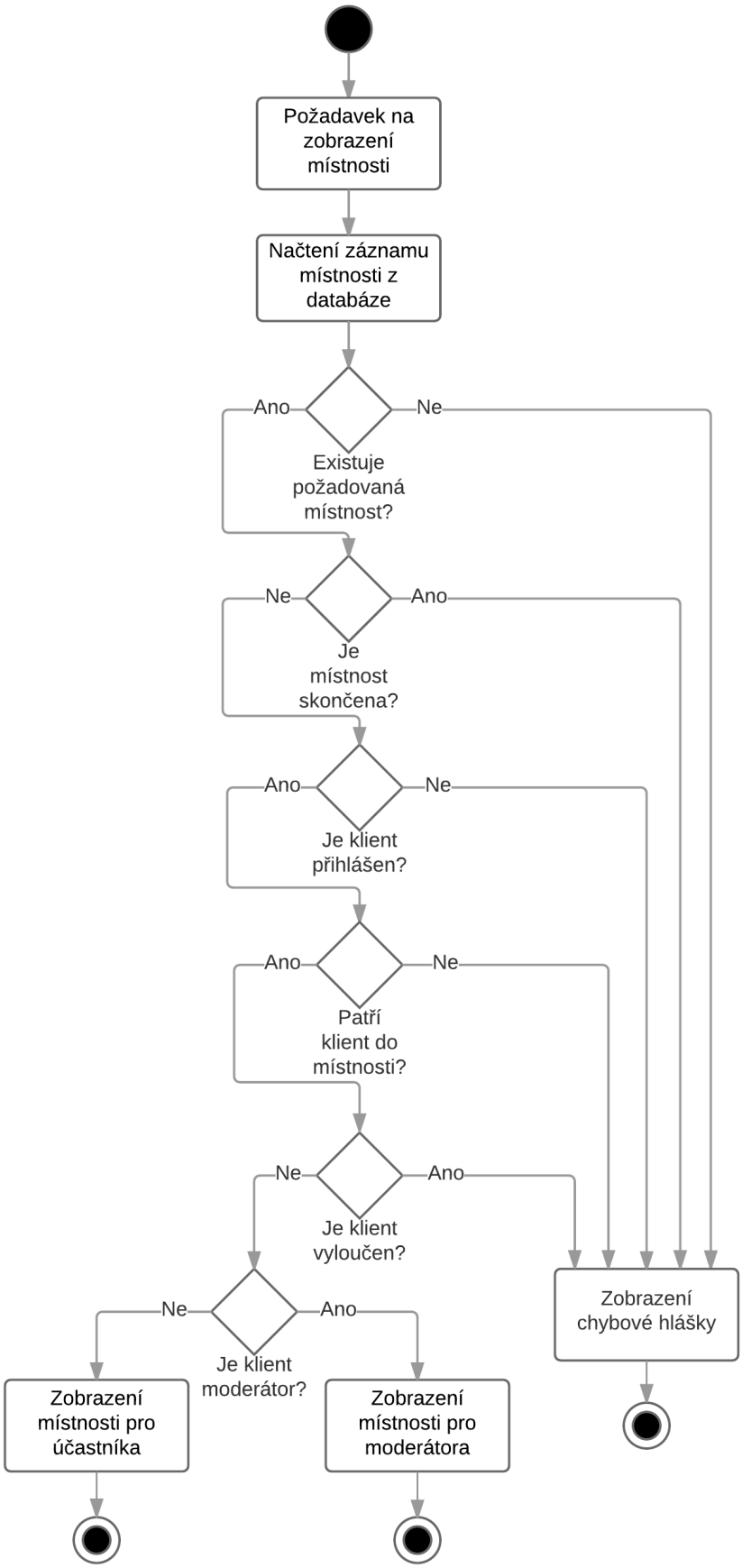
Na přiloženém CD se nachází plné znění bakalářské práce pod názvem ”dusek_jiri_bp.pdf”, dále se na CD nachází kompletní zdrojové soubory ke všem částem vytvořeného konferenčního systému.

11.2 Obrázkové přílohy

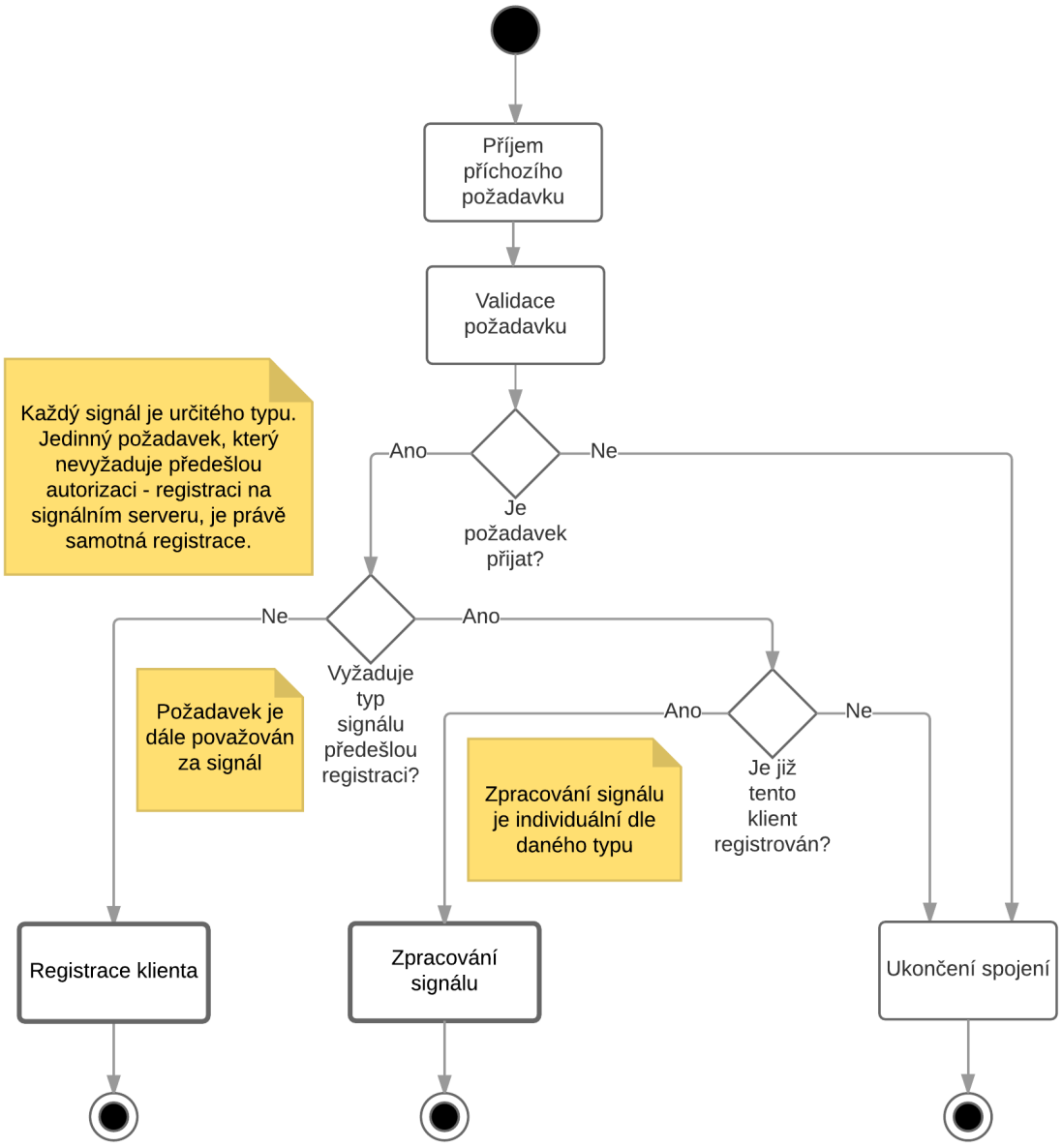
Obrázek 19: Use case - konferenční systém



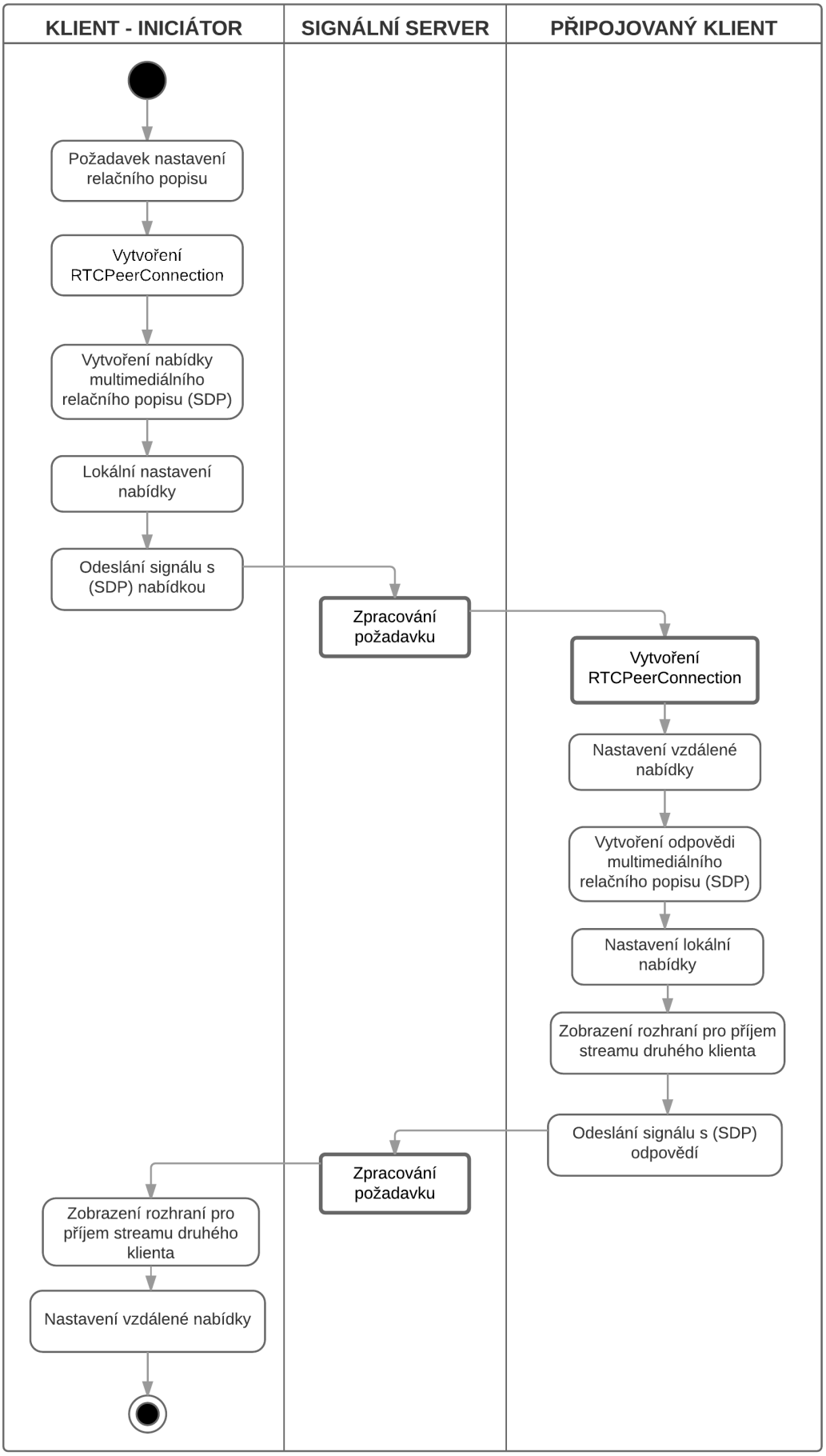
Obrázek 20: Příchod klienta do místnosti



Obrázek 21: Webový server - zobrazení místnosti



Obrázek 22: Zpracování signálního požadavku



Obrázek 23: Proces nastavení relačního popisu