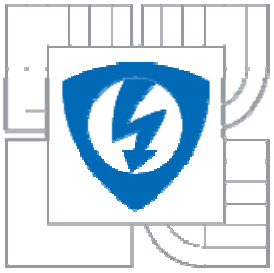
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

TVORBA REAL-TIME APLIKACE PRO PLATFORMU IMS

CREATING REAL-TIME IMS APPLICATION

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE
AUTHOR

BC. FILIP NOVOTNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ĽUBOŠ NAGY

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Filip Novotný	***ID:***	83415
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Tvorba real-time aplikace pro platformu IMS

POKYNY PRO VYPRACOVÁNÍ:

Cílem projektu je prostudovat a popsat architekturu IMS (IP Multimedia Subsystem) se zaměřením na vývoj real-time aplikací pro tuto technologii prostřednictvím vývojového nástroje IMS pro tvorbu aplikací - SDS Ericsson. Na základě získaných poznatků navrhněte systém pro poskytování real-time služeb pro koncové uživatele. Informace o poskytované službě jednotlivým klientům (např. čas uskutečněné služby, délka trvání služby, atd.) budou zpřístupněny klientům pomocí webovského rozhraní.

DOPORUČENÁ LITERATURA:

[1] POIKSELKA, M., MAYER, G. The IMS: IP Multimedia Concepts and Services. V. Británie: WILEY, 2009. 560 s. Třetí vydání. ISBN 978-0-470-72196-4.

[2] AL-BEGAIN, K., BALAKRIHNA, C., GALINHO, L.A., FERNANADEZ, D.M.: IMS: A Development and Deployment Perspective. V. Británie: WILEY, 2009. 316 s. ISBN 978-0-470-74034-7.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Ľuboš Nagy

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Tato práce se věnuje problematice IMS (IP Multimedia Subsystem). IMS je standardizovaná architektura sítí nové generace, která nabízí pokročilé služby v mobilních i pevných sítích.

První kapitola popisuje čtyř vrstvou IMS architekturu a zmiňuje podporované služby. Druhá kapitola se zabývá IMS protokoly. Převážná část kapitoly se zaměřuje na signalizační protokol SIP. Dalšími zahrnutými protokoly jsou UDP, RTP a TLS. Třetí kapitola se věnuje praktické části této práce. Na základě získaných poznatků o IMS architektuře, byla vytvořena java aplikace podporující přenos hlasu přes IP sítě. Hlavní části aplikace tvoří klientská část, databáze MySQL a Servlet aplikace určená pro komunikaci databáze s klientskou částí. Celá aplikace byla vyvíjena v prostředí SDS Sony Ericsson 4.1. FD1. Při vývoji aplikace byl také testován klient OpenIC lite a následně byl srovnán s vytvořeným klientem. Pro monitorování komunikace mezi uživateli slouží webové rozhraní vytvořené pomocí programovacích jazyků PHP a ActionScript2 (Flash). Webová aplikace administrátorovi umožňuje spravovat data jednotlivých uživatelů a uživatelům hlasové služby poskytuje přehled o užívání služby. Pro názorné zobrazení provolaného času slouží uživateli vytvořený graf, který se nachází ve výpisu jeho hovorů.

KLÍČOVÁ SLOVA

IMS, SIP, ICP, PHP, ActionScript, Java, SDS

ABSTRACT

This paper presents a description of the IP Multimedia Subsystem (IMS) architecture and IMS services. IMS is a standardized next-generation networking architectural framework providing advanced services on mobile and fixed networks.

The first chapter describes four-layer IMS architecture and also mentions supported services. The second chapter deals with IMS protocols and primarily focuses on the SIP signaling protocol. UDP, RTP and TLS protocols are also included in this chapter. The third chapter is dedicated to practical part of this paper. A VoIP Java application has been created based on the findings gained throughout the thesis. The main part of the created application consists of client-side application, database MySQL and Servlet application for communication between database and client-side application. The whole system was created using SDS Sony Ericsson 4.1. FD1. An OpenIC lite client has been tested during development on our system and results were compared with created client-side application. A web application based on PHP, MySQL and ActionScript then handles administration and monitoring of customers using voice services.

KEYWORDS

IMS, SIP, ICP, PHP, ActionScript, Java, SDS

NOVOTNÝ, F. *Tvorba real-time aplikace pro platformu IMS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 61 s. Vedoucí diplomové práce Ing. Ľuboš Nagy.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Tvorba real-time aplikace pro platformu IMS jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Ľubošovi Nagyovi z Ústavu telekomunikací za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne ……..... ……..

podpis autora

Obsah

Úvod

Náplní této práce je vytvoření real-time aplikace, která bude podporovat hlasový přenos v IMS sítích. Teoretická část je tedy věnována architektuře IMS a nejdůležitějším protokolům, které zajišťují provoz celé sítě. Praktická část obsahuje samotnou tvorbu aplikace.

IMS neboli IP-Multimedia Subsystem definuje architekturu pro IP sítě. IMS byl původně navržen pro mobilní sítě třetí generace, avšak bylo rozšířeno na bezdrátové WiFi sítě a pokračuje v rozšiřování s cílem nezávislosti na připojení. IMS nabízí telekomunikačním operátorům možnost vybudovat otevřenou IP infrastrukturu orientovanou na služby, která umožní snadné nasazení nových multimediálních služeb zahrnující telekomunikační a zároveň datové služby.

První kapitola je věnována popisu čtyř vrstev sítě IMS. Dále jsou zde popsány prvky architektury IMS a jednotlivá rozhraní IMS sítě.

Druhá kapitola popisuje signalizační protokol SIP, který je zodpovědný za sestavení, dohled a ukončení multimediálního spojení. Tento protokol je používán společně s jinými protokoly, jako jsou RTP pro přenos dat, RADIUS nebo DIAMETER pro autentizaci uživatelů apod. V kapitole jsou uvedeny např. protokoly RTP, UDP a TLS.

Ve třetí kapitole je popsána vlastní tvorba real-time aplikace. Celá aplikace byla vytvořena pomocí tří programovacích jazyků: PHP, ActionScript a Java. IMS aplikace byla naprogramována v jazyce JAVA a webová aplikace pomocí jazyku PHP a ActionScript. Pro tvorbu IMS aplikace bylo zvoleno prostředí SDS (Service Development Studio) Ericsson verze 4.1. FD1 a pro testování IMS sítě v prostředí SDS byl použit klient OpenIC lite. Součástí tohoto projektu je i webová aplikace sloužící pro zaznamenávání údajů o hovorech jednotlivých uživatelů. Je zde popsána uživatelská část, kde si mohou uživatelé zobrazit výpis svých hovorů ve formě tabulky a flashového grafu, a také administrátorská část pro spravování uživatelů a hovorů administrátorem.

V závěru třetí kapitoly je srovnání vytvořeného klienta s testovaným klientem OpenIC lite.

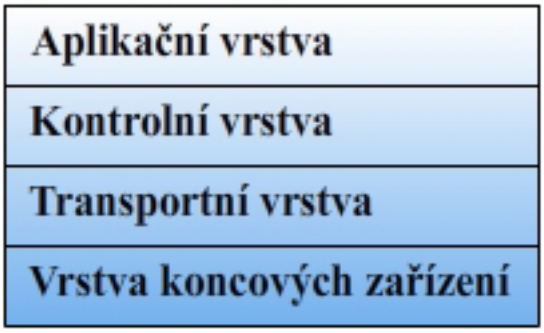
Některé částečné výsledky byly publikované ve sborníku EEICT 2011.

1 Architektura IMS

Architektura IMS [11] podporuje širokou škálu služeb založených na protokolech SIP. IMS poskytuje multimediální služby, které mohou být přístupné pomocí různých zařízení přes IP síť nebo telefonní systém. Mezi takové služby např. patří:

- push to talk
- video telefonie
- instant messaging (posílání zpráv)
- presenční služby
- přenos hlasu

Základním požadavkem pro přístup k službám IMS je, aby klient měl IP připojení. IMS se snaží být nezávislé na připojení a to tak, že IMS služby mohou být poskytovány přes jakoukoli IP síť. Síťovou architekturu lze rozdělit do čtyř vrstev viz obr. 1.1.



Obr. 1.1: Vrstvy IMS sítě [11]

1.1 Vrstvy sítě IMS

1.1.1 Vrstva koncových zařízení

Vrstva koncových zařízení je nejnižší vrstvou IMS architektury [11]. IMS architektura poskytuje širokou škálu možností pro použití koncových zařízení. V IMS lze použít zařízení jako například počítače, mobilní telefony, PDA nebo digitální telefony. Analogové telefony, které se nemohou připojit přímo do IP sítě, se připojují pomocí PSTN brány.

1.1.2 Transportní vrstva

Transportní vrstva je zodpovědná za sestavení a ukončení SIP relace. Poskytuje konverzi přenášených dat mezi analogovým a digitálním formátem [11]. IMS zařízení přistupují k transportní vrstvě pomocí různý přenosových médií. Mezi taková média patří WiFi, WiMAX, DSL, GPRS, WCDMA apod. Vrstva také umožňuje uskutečnit hovor do a z PSTN sítí.

1.1.3 Kontrolní vrstva

Kontrolní vrstva se nachází mezi transportní a aplikační vrstvou. Směruje signalizaci, říká transportní vrstvě, jaký přenos má být povolen a generuje informace o poplatcích za užívání sítě [11]. Jádrem kontrolní vrstvy je CSCF (Call Session Control Function). CSCF zajišťuje SIP registraci koncových bodů a zpracovává SIP signály od příslušného aplikačního serveru v aplikační vrstvě. Protokoly této vrstvy jsou SIP a Diameter.

Další součástí kontrolní vrstvy je databáze HSS (Home Subscribe Server). Tato databáze uchovává unikátní profil každého koncového uživatele. V profilu jsou informace typu uživatelské IP adresy, telefonní záznamy, seznam kontaktů apod. Díky těmto informacím v HSS může poskytovatel služeb centralizovat uživatelská data vůči všem službám v IMS.

1.1.4 Aplikační vrstva

Nejvyšší vrstvou IMS architektury je aplikační vrstva. Nižší vrstvy IMS zajišťují integrovaný a standardizovaný síťový systém umožňující poskytovatelům nabízet různé multimediální služby řídící vrstvy [11]. Veškeré služby běží na aplikačních severech. Aplikační servery jsou zodpovědné za poskytování a vykonávání služeb a vytváří rozhraní pro kontrolní vrstvu pomocí SIP protokolu. Jeden aplikační server může poskytovat více služeb najednou. Aplikační vrstva definuje standardní rozhraní pro společné funkce:

- konfigurace paměti, správa identity, uživatelský status; zajišťované HSS
- zpoplatněné služby; zajišťované CGF (Charging Gateway Function)
- kontrola hlasových služeb, video služeb a zpráv; zajišťované kontrolní vrstvou

1.2 Prvky IMS architektury

1.2.1 CSCF (Call Session Control Function)

CSCF [12] slouží jako centralizovaný prvek směrování, správce bezpečnostních pravidel a jako bod, který prosazuje bezpečnostní pravidla pro zajištění doručení různých real-time aplikací založených na IP přenosu. CSCF je aplikačně založený a používá dynamické informace ke správě síťových zdrojů.
Rozlišují se tři typy CSCF [12]:

- P-CSCF
- S-CSCF
- I-CSCF

Proxy-CSCF

Proxy-CSCF je prvním bodem, který uživatelé kontaktují. Tento bod je zodpovědný za bezpečnost zprávy mezi sítí a uživatelem a přiděluje zdroje pro datový tok.

Interrogating-CSCF

Interrogating-CSCF je prvním bodem, který kontaktují sítě sobě si rovné. I-CSFC se dotazuje HSS na S-CSCF pro uživatele. Chrání S-CSCF a HSS před nepovolenými přístupy. Slouží tedy jako firewall, který zabraňuje nechtěným vniknutím z jiné sítě.

Serving-CSCF

S-CSCF je zodpovědné za zpracování záznamů o umístění v síti každého uživatele, uživatelskou autentizaci a za proces volání. Jedná se vlastně o SIP server, který zajišťuje řízení relace uživatelů. Komunikuje s databází HSS a s AAA (Access, Authorization and Accouting) servery.

S-CSCF a aplikační server

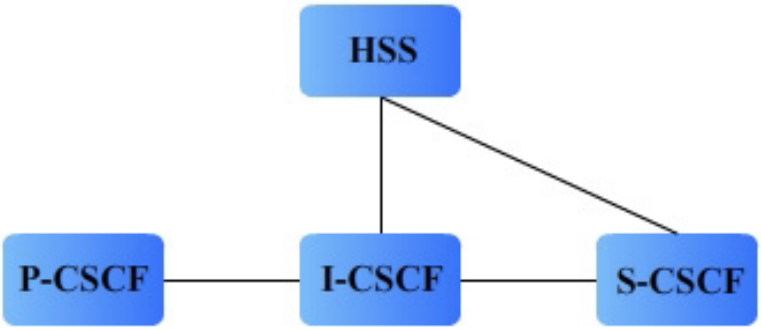
Úkolem S-CSCF je ovládání registračních procesů, udržování statusu spojení a je zodpovědné za směrování.

S-CSCF [12] používá pro komunikaci s aplikačním serverem iFC (Initial Filter Criteria). S-CSCF přeposílá zprávy na každý aplikační server a to v pořadí podle

filtračních kritérií iFC. Po kontaktování posledního aplikačního serveru, je zpráva poslána na místo určení.

IMS obsahuje tzv. SPTs body (Service Point Triggers). Jsou to body v SIP signalizaci, na kterých může být iFC nastaven. Tyto kritéria jsou rozmístěny mezi S-CSCF, HSS a IMS aplikačním serverem. iFC kritéria jsou uloženy v HSS jako část profilu služby. Jsou aktivní až do vypršení registrace nebo do doby než se změní profil služby a měly by obsahovat tyto informace [12]:

- Adresu aplikačního serveru
- Prioritu aplikování filtračního kritéria
- Výchozí nastavení pokud aplikační server nelze kontaktovat
- Volitelné informace, které se přidávají do těla zprávy, než je zaslána na aplikační server



Obr. 1.2: Komunikace HSS s CSCF [12]

Během registrační fáze se S-CSCF používá pro kontrolu uživatelských služeb. Uživatelův profil služby obsahující iFC je stažen z HSS do S-CSCF. Pokud S-CSCF obdrží SIP požadavek shodující se s iFC, je vyvolána služba, která přepošle tento požadavek na aplikační server, který je definován v iFC. iFC se používá pouze pro iniciační SIP požadavek.

1.2.2 MGCF (Media Gateway Control Function)

Tato funkce komunikuje s CSCF [13] a kontroluje spojení pro mediální kanály v IMS-MGW. Ovládá ty části hovoru, které spadají do médií kanálů v MGW. Zajišťuje převádění protokolů mezi ISUP a SIP.

1.2.3 MGW

Propojuje PSTN a IMS sítě. MGW [13] se stará o konverzi medií, ale také o užitečné zatížení zpracování (kodek, echo nebo tvoří most mezi konferencemi).

1.2.4 SLF (Subscription Locator Function)

Pracuje jako samostatný server nebo je součástí jiné funkce. SLF [13] obsahuje adresní prostory, které jsou přiřazeny každému z HSS, a proto je schopen určit polohu HSS, který obsahuje data konkrétního uživatele. Data vyhledává na podnět dotazu z I-CSCF nebo aplikačního serveru. Komunikace SLF a HSS probíhá pomocí protokolu DIAMETER.

1.2.5 BGCF (Breakout Getaway Control Function)

S-CSCF [14] vyžaduje službu od BGCF pokud SIP hovor má být směrován do sítě s přepínáním okruhů (CS). BGCF funkce je zodpovědná za to, kde se bude přistupovat do CS sítí. Výsledek výběru může být do stejné sítě, kde se nachází BGCF nebo do jiné sítě. Pokud se jedná o stejnou síť BGCF, vybere pro další správu relace MGCF. Pokud se jedná o jinou síť BGCF, předá relaci BGCF ve vybrané síti. BGCF je také schopné sbírat informace o uživatelském účtu a shromažďovat statistické informace.

1.2.6 MRF (Media Resource Function)

Funkce MRF se dělí na dvě části – MRFC a MRFP. Zastává funkci míchání médií, analýzu médií a překódování médií. MRF je možné provozovat pouze v domácí síti.

MRFC (Multimedia Resource Function Controller) [14] podporuje konferenční hovory, plní oznamovací služby uživateli, popř. provádí překódování signálu. MRFC zpracovává SIP signalizaci přijatou skrze S-CSCF a používá Media Gateway Control Protocol (MEGACO), který zajišťuje kontrolu nad MRFP.

MRFP (Multimedia Resource Funciton Processor) zpracovává požadavky od S-CSCF a od aplikačního serveru. MRFP poskytuje širokou škálu funkcí pro multimediální zdroje. Mixuje příchozí tok dat, zpracovává mediální tok (např. kódování audia, analýza média) a produkuje záznamy pro zpoplatnění hovoru. Je to procesor řízený MRFC.

1.2.7 HSS (Home Subscribe Server)

HSS je hlavní úložiště [14] pro všechny informace uživatelů a data IMS. HSS obsahuje data o uživatelské identitě, registrační informace, přístupové parametry

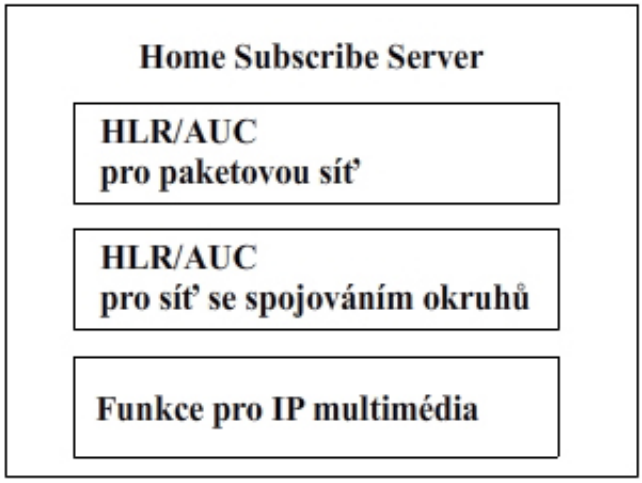
a informace o spuštění služby. Ke komunikaci s ostatními prvky sítě používá HSS protokol DIAMETER.

Rozlišují se dva typy uživatelské identity [14]:

- Soukromá identita
- Veřejná identita

Soukromá identita je uživatelská identita, která je přidělena operátorem sítě a je používána pro registraci a autorizaci. Veřejnou identitu používá uživatel ke komunikaci s ostatními uživateli.

IMS přístupové parametry jsou používány k sestavení relace. Jsou to informace o autentizaci, roamingové autorizaci a přiřazená S-CSCF jména. Informace o spuštění služby umožňují spuštění SIP služby. HSS také ukládá uživatelovy požadavky na S-CSCF a tím umožní I-CSCF vybrat nejlepší S-CSCF pro uživatele.



Obr. 1.3: Architektura HSS [14]

HLR (Home Location Register) poskytuje podporu prvkům v paketové síti (PS). Umožňuje účastníkovi přístup do PS. Podobně poskytuje podporu pro CS sítě a tím umožňuje přístup do GSM/UMTS sítí.

V AUC (Authentication Center) je uložen bezpečnostní klíč každého mobilního účastníka a je využíván k dynamickému generování bezpečnostních dat pro každého takového účastníka. Bezpečnostní data jsou také používána k zajištění integrity a k šifrování radiové komunikace mezi uživatelským zařízením a sítí.

1.3 Rozhraní IMS

1.3.1 Cx rozhraní

Pokud se uživatel přihlásí, je nutné data uložené v HSS spravovat pomocí I-CSCF a S-CSCF. Pro tyto účely existuje rozhraní Cx. Komunikace probíhá pomocí protokolu DIAMETER. Procedury [14] jsou děleny do třech kategorií: zpracování uživatelských dat, uživatelská autentizace a umístění uživatele v síti.

1.3.2 Dx rozhraní

Při výskytu [14] více HSS entit, I-CSCF ani S-CSCF neví, která HSS entita má být kontaktována. Nejprve je nutné kontaktovat SLF, který obsahuje mechanismus pro vyhledávání adresy HSS v síti v případě, že síť obsahuje více HSS. Zavádí se tedy rozhraní Dx, které se používá pro spojení s Cx rozhraním. Ke komunikaci slouží protokol DIAMETER.

1.3.3 Sh rozhraní

Aplikační server potřebuje uživatelská data, aby věděl, na který S-CSCF má vyslat SIP požadavek [14]. Tento typ dat je uložen v HSS. Rozhraní zajišťující tuto komunikaci přes protokol DIAMETER se nazývá Sh. HSS spravuje seznam aplikačních serverů, které k těmto datům mohou přistupovat.

1.3.4 Dh rozhraní

Podobně jako [14] u Dx rozhraní při výskytu více HSS entit síti, I-CSCF ani S-CSCF nejsou schopné určit, se kterou HSS entitu se mají spojit. Nejprve je nutné kontaktovat SLF, který obsahuje mechanismus pro vyhledávání adresy HSS v síti v případě, že síť obsahuje více HSS. Dh rozhraní se používá vždy pro spojení s Sh rozhraním. Porotokol používaný v tomto rozhraní je DIAMETER.

Pro získání HSS adresy aplikační server pošle SLF Sh požadavek určený pro HSS. Po obdržení adresy od SLF aplikační server pošle Sh požadavek na HSS.

1.3.5 Mm rozhraní

Mm rozhraní [14] slouží pro komunikaci mezi IMS a jinými multimediálními IP sítěmi. Umožňuje I-CSCF přijmout požadavek na relaci z jiného SIP serveru nebo terminálu.

1.3.6 Mg rozhraní

Toto rozhraní [14] umožňuje MGCF přeposílání příchozí signalizace relace ze sítí se spojováním okruhů do I-CSCF. MGCF převádí příchozí ISUP signalizaci na SIP. Používaným protokolem pro referenční bod Mg je SIP.

1.3.7 Mi rozhraní

Jestliže S-CSCF [14] zjistí, že relace má být směrována do CS domén, použije k tomu referenční bod Mi. Ten přepošle relaci do BGCF. Používaný protokol pro komunikaci mezi S-CSCF a BGCF je SIP.

1.3.8 Mj rozhraní

Když BGCF [14] přijme signalizaci relace z referenčního bodu Mi, vybere vhodnou CS doménu. Pokud se jedná o jinou síť, pak je relace přeposlána do BGCF v jiné síti přes referenční bod Mk. Používaný protokol pro komunikaci je SIP.

1.3.9 Ut rozhraní

Jedná se o referenční bod mezi uživatelským přístrojem a aplikačním serverem. Přes tento bod si uživatelé bezpečně spravují a konfigurují síťové parametry související s aplikačním serverem [14]. Komunikačním protokolem pro referenční bod Ut je HTTP.

1.3.10 Mr rozhraní

Mr rozhraní používá SIP protokol pro komunikaci S-CSCF a MRFC [14]. Rozhraní není plně standardizované.

1.3.11 Mp rozhraní

MRFC využívá Mp referenční bod ke kontrole multimediálního přenosu [14]. Toto rozhraní je plně vyhovující pro standard H.248.

1.3.12 Go rozhraní

Go zajišťuje komunikaci mezi IMS a GPRS sítí [14]. Původně byl zaveden pro zajištění QoS a kontrolu zdrojové a cílové adresy v určených.

2 IMS protokoly

2.1 Protokol SIP

SIP (Session Initiation Protokol) je signalizační protokol aplikační vrstvy typu klient-server. Zajišťuje sestavení, dohled a ukončení multimediálního spojení (například VoIP). SIP podporuje jak dvoucestný telefonní hovor, tak i multimediální konference. Pro úplnou multimediální architekturu je používán společně s jinými protokoly. Většinou tyto architektury obsahují protokoly jako RTP pro přenos real-time dat, DIAMETER pro autentizace uživatelů či LDAP protokol pro ukládání a přístup k datům na adresářovém serveru.

SIP podporuje pět hlavních prvků pro sestavení a ukončení multimedilní komunikace [2].

2.1.1 Pět prvků pro sestavení a ukončení relace

User location

Zjištění koncového systému, který bude použit pro komunikaci. Najít umístění uživatele, vyžaduje schopnost přeložit uživatelské jméno na IP adresu právě použitého počítače. Důvod proč je to tak důležité je, že uživatel může použít jiný počítač nebo může mít jinou IP adresu, která identifikuje počítač v síti (při použití DHCP). Program registruje uživatele na serveru pomocí SIP tak, že serveru poskytne uživatelské jméno a IP adresu. Díky tomu ostatní uživatelé znají jeho umístění v síti.

User availability

Jedná se o schopnost volané strany určit, zda se mohou zapojit do komunikace [1]. Uživatel se například může nastavit jako nedostupný, zaneprázdněný nebo dostupný.

User capabilities

Jde o vymezení schopností uživatelských programů na obou stranách. Následuje vyjednávání, kterou z nich lze použít během relace.

Session setup

Nastává při spojení účastníků. Volaný účastník je upozorněn například vyzváněním a má možnost komunikaci přijmout nebo odmítnout. Jestliže ji přijme, dojde k nastavení vyjednaných parametrů relace na obou stranách a umožnění komunikace.

Session management

Je to funkce umožňující upravovat relaci za běhu. Během relace jsou data přenášena mezi účastníky a typ komunikace se může změnit. Například se účastník může rozhodnout změnit hlasovou konverzaci na videokonferenci. Během komunikace mohou účastníci také pozvat nebo vynechat jiného účastníka, přidržet hovor, přesměrovat nebo ukončit komunikaci.

2.1.2 SIP URI (Uniform Resource Indicators)

SIP používá zavedené metody pro identifikaci a propojení koncových bodů. Lze to vidět na adresním schématu, který používá pro identifikaci různých SIP účtů. Používá adresy podobné e-mailovým adresám [1]. Hierarchická URI zobrazuje doménu, kde se uživatelův účet nachází a jméno hostitele nebo telefoní číslo, které slouží jako uživatelský účet. Například SIP: uzivatelske_jmeno@domena.cz.

Takováto adresa musí mít unikátní uživatelské jméno v adresním prostoru. Jelikož jsou uživatelská jména uložena na centralizovaných serverech, je server při zakládání nových účtů schopen určit, zda je dané uživatelské jméno zabrané či ne.

Základní schéma URI [1]: sip:user:password@host:port;uri-parameters?headers

V základním schématu URI je znázorněné uživatelské heslo (password). Pokud syntax URI umožňuje přítomnost tohoto pole, nedoporučuje se jej používat, protože posílání autentizačních údajů v čistém textu je bezpečnostní risk. Pole pro heslo je jen dodatečné rozšíření a „user:password“ lze tedy považovat za jednotný řetězec.

Do pole host se zapisuje poskytovatel SIP. Tato část obsahuje doménové jméno nebo numerickou adresu IPv4 nebo IPv6. Pole port je číslo portu, na který je vyslán požadavek.

URI parametry ovlivňují požadavek. Jsou umístěny za „host:port“ a jsou odděleny čárkami. Parametry se zapisují takto: jméno-parametru = hodnota-parametru. Jméno parametru musí být unikátní.

2.1.3 SIP architektura

Aby SIP dobře fungoval, je zapotřebí různých zařízení a protokolů. Používají se nejen zařízení, které umožňují komunikaci mezi účastníky, ale také potřebné servery pro propojení účastníků. Navíc je použito mnoho protokolů pro přenos hlasu a dat mezi zařízeními. To celé pak tvoří architekturu SIP.

Existují dva hlavní komponenty, které SIP používá:

- Uživatelské agenty; UA (user agent). Tvoří je koncové zařízení.
- SIP servery. Reprezentují je počítače v síti, které zpracovávají požadavky klientů a posílají zpět odpovědi.

User Agents

Uživatelský agent (UA) je počítač [2], který je použitý pro hovor, ale i počítač, který je volán. Vytváří se tak dvoubodové komunikační spojení. Existují dvě komponenty UA - klient a server. Pokud UA vytvoří požadavek (zahájí relaci), jedná se o klienta UAC (user agent client). Ti UA, kteří odpovídají na požadavek, jsou UAS (user agent server). Během relace se role UA mění dle potřeby.

Aby mohla být SIP relace zahájena, musí existovat UA. Jeden UA pozve druhého UA do relace, která je řízena a podle potřeby ukončena pomocí SIP. Uživatelský klient (UAC) využívá SIP k posílání požadavků pro uživatelský server (UAS), který požadavek zpracuje a odpoví. UAC a UAS si vymění zprávy a pak si role prohodí.

Uživatelského agenta si lze představit jako softwarovou aplikaci nainstalovanou na počítači, ale jako i PDA nebo USB telefon, který se připojuje k počítači nebo jako bránu propojující síť s PSTN.

SIP server

Uživatelský agent se registruje na SIP serveru pomocí jeho uživatelského jména a IP adresy [2]. Tím určí jeho aktuální polohu v síti a potvrdí, že je online. Protože UA neví IP adresy ostatních UA, vyžádá si je od SIP serveru. SIP server poté identifikuje, zda je osoba online a pokud ano, pomocí uživatelského jména vyhledá IP adresu, aby

zjistil její umístění v síti. Jestliže uživatel není součástí domény (užívá tedy jiný SIP server), je zaslán požadavek na jiný server.
Server tak pracuje v několika rolích:

- Registrar server
- Proxy server
- Redirect server

Registrar server

Registrační servery jsou používány k registraci umístění uživatelského agenta, který se přihlásil do sítě. Server si uloží do systému IP adresu uživatele, kterou spojí s uživatelským jménem. Tak se vytvoří adresář těch, co jsou přihlášeni do sítě a jejich umístění v síti. Pokud si někdo přeje vytvořit relaci s jedním z těchto uživatelů, je odvolán na informace registrar serveru a tím zjistí IP adresy těch, co jsou do relace zapojeni.

Proxy server

Proxy servery jsou počítače, které jsou použity k přeposílání požadavků na jiné počítače. Pokud SIP server obdrží od klienta požadavek, může jej přeposlat na jiný SIP server v síti. Jestli pracuje jako proxy server, pak může SIP server poskytovat funkce jako kontrolu přístupu do sítě, zabezpečení, autentizaci a autorizaci.

Redirect server

Redirect servery SIP používá k tomu, aby přesměroval klienty k UA, kterého se snaží kontaktovat. Pokud UA vyšle požadavek, redirect server mu zašle IP adresu kontaktovaného UA. Tímto se liší od proxy serveru, který přeposílá požadavek dál, zatím co redirect server se snaží, aby klient UA kontaktoval sám.

Redirect server také umožňuje rozdělit hovor do různých míst. Pokud je hovor určen určitému uživateli, může být rozdělen do několika různých míst tak, že bude zvonit zároveň ve všech z nich. Hovor obdrží to z míst, které odpoví jako první a v ostatních místech zvonit přestane.

Location service

Tato služba uchovává databázi uživatelů registrovaných pomocí SIP serveru a jejich umístění v síti. REGISTER požadavek je proveden, pokud se uživatelský agent

registruje na registrar serveru. Pokud server požadavek akceptuje, získá tak SIP adresu a IP adresu uživatelského agenta a přidá ji do location servis, tedy lokalizační služby, pro jeho doménu. Tato databáze poskytuje aktuální seznam každého kdo je online a jeho pozici, čehož využívají proxy a redirect servery pro získání informaci o uživatelských agentech. Pomocí toho servery propojují uživatelské agenty nebo přesměrovávají požadavky do určitých míst.

2.1.4 Klient-server a peer-to-peer architektura

V síťové komunikaci existují dva různé typy architektury[2]:

- Klient-server
- Peer-to-peer (P2P)

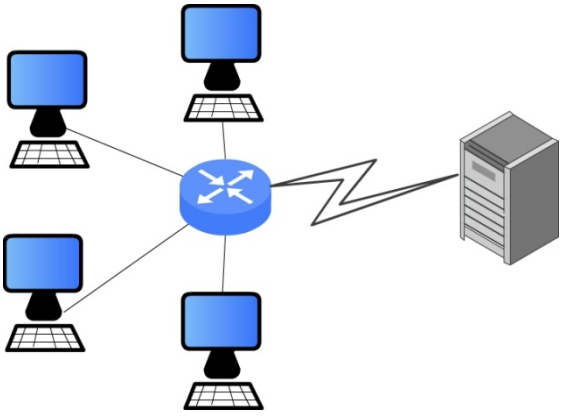
Klient-server

V tomto typu architektury je vztah mezi počítači rozdělen do dvou rolí.

- Klient, který požaduje určité služby nebo zdroje.
- Server, který vyřizuje požadavky pomocí požadované služby nebo zdroje.

Klasický model klient-server je při užívání internetu. Počítač, na kterém běží webový prohlížeč, bude v tomto případě klient, který požaduje webovou stránku od webového serveru. Webový server obdrží požadavek a odpoví tím, že pošle webovou stránku klientovi. U VoIP stejný případ nastává, když klient odešle požadavek na registraci u registrar serveru nebo vznese požadavek na proxy či redirect server, který umožňuje spojení s jiným UA. Role klienta je požadovat služby a zdroje a role serveru je naslouchat a očekávat požadavky, které může zpracovat nebo odeslat na jiný server.

Další důležitá funkce serveru je, že narozdíl od klientů, kteří mohou být vypnuti či odpojeni od sítě, servery jsou většinou aktivní a naslouchají klientským požadavkům. IP adresa serveru se v zásadě nemění, proto jej klienti mohou vždy v síti najít. To je důležité pro vyhledávání jiných počítačů v síti.

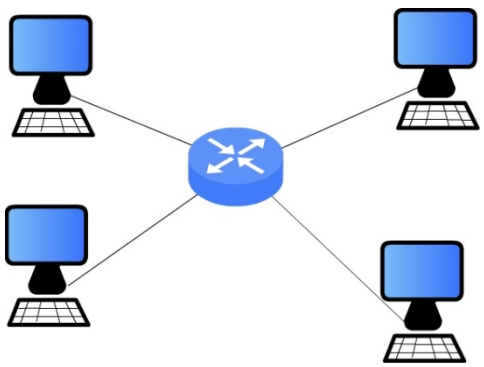


Obr. 2.1: Klient-server architektura

Peer-to-peer

Tato architektura se liší od klient-serveru tím, že použité počítače mají podobné schopnosti a mohou zahájit relaci jeden s druhým a poskytovat odpovědi jeden druhému. Každý počítač poskytuje služby a zdroje. Proto když jeden z nich je nedostupný, je kontaktován jiný pro výměnu zpráv nebo přístupových zdrojů. V tomto případě uživatelští agenti mohou pracovat současně jako klient i server a jsou tak rovnocenní.

Pokud je UA schopen navázat spojení s druhým UA, je nastavena peer-to-peer architektura a stroje jsou schopny vzájemně vznášet požadavky a odpovědi. Jeden stroj pracující jako UA klient vznese požadavek a druhý pracující jako UA server mu na něj odpoví. Každý ze strojů si mohou role prohodit a jednat tak rovnocenně v síti. Například u aplikací podporujících sdílení souborů UA klient si vyžádá soubor od UA serveru a stáhne si jej. Během toho si mohou vyměňovat zprávy nebo komunikovat pomocí VoIP. Pro ukončení komunikace, vznese jeden z nich požadavek pro ukončení relace. Jak je vidět každý z počítačů má roli jak klienta, tak serveru.



Obr. 2.2: Peer-to-peer architektura

2.1.5 SIP a signální zprávy

SIP je textově založený protokol. Je používán pro přenos informací mezi klienty a servery, jako sled požadavků a odpovědí. Existuje několik signálních příkazů pro požadavek:

- **REGISTER** – Používá se, pokud UA se objeví poprvé online. Registruje svoji SIP adresu a IP adresu na registrar serveru.
- **INVITE** - Vyzve dalšího UA ke komunikaci a poté sestaví mezi nimi SIP relaci.
- **ACK** - Používá se pro přijmutí relace a k potvrzení spolehlivého doruční zpráv.
- **OPTIONS** - Získá informace o schopnostech dalšího UA.
- **SUBSCRIBE** - Používá se k získání aktuálního statusu jiného UA. Tyto statusy mohou být online, busy, offline, atd.
- **NOTIFY** - Posílá aktuální informace o UA, tj. jestli je online, busy, offline, atd.
- **CANCEL** - Ruší nevyřešené požadavky, bez toho aniž by ukončil relaci.
- **BYE** - Tento příkaz ukončí relaci. Může jej použít jak UA, který relaci zahájil, tak i UA, který byl volán.

Pokud je vznesen dotaz na SIP server nebo na jiného UA, je možná jedna z několika odpovědí. Odpovědi se dělí do šesti různých skupin s troj číselným kódem, který začíná příslušným číslem skupiny. Tyto skupiny jsou popsány níže.

- **Informační odpovědi 1xx -** Požadavek byl přijat a bude zpracován.
- **Úspěšná žádost 2xx** - Žádost byla úspěšně přijata potvrzena.
- **Přesměrování 3xx -** Žádost nemůže být dokončena a je potřeba dalších kroků (např. Přesměrování UA na jinou IP).
- **Chyba na straně klienta 4xx -** Žádost obsahuje chyby, proto server nemůže zpracovat požadavek.
- **Chyba na straně serveru 5xx -** Žádost byla přijata, ale server ji nemůže zpracovat. Chyba tohoto typu se týká serveru a neznačí, že žádost nemůže zpracovat jiný server.

- **Globální chyba 6xx -** Žádost byla přijata a server ji není schopen zpracovat. Chyba tohoto typu značí, že by se mohla vyskytnout na jakémkoli serveru. Proto není požadavek přeposlán na jiný server.

Tab. 2.1: Příklad odpovědí, jejich popis a číselný kód [2]

Číselný kód	Kategorie	Popis
100	Informační odpovědi	Zkoušení
180	Informační odpovědi	Vyzvánění
200	Úspěšná žádost	OK
300	Přesměrování	Více možností
305	Přesměrování	Použití proxy
380	Přesměrování	Náhradní služba
400	Chyba na straně klienta	Chybná žádost
401	Chyba na straně klienta	Neautorizovaný
404	Chyba na straně klienta	Nenalezen
500	Chyba na straně serveru	Vnitřní chyba serveru
503	Chyba na straně serveru	Nedostupná služba
505	Chyba na straně serveru	Verze SIP není podporována
600	Globální chyba	Zaneprázdněn
606	Globální chyba	Neakceptováno

2.2 UDP protokol

UDP je nespojový protokol transportní vrstvy, který zprostředkovává jednoduchý nespolehlivý přenos dat. Nikdy není používán pro posílání důležitých dat, jako jsou webové stránky nebo databázové informace. UDP je určen pro streamované audio a video, protože je rychlejší než TCP protokol. Tato rychlost je dána tím, že UDP nemá kontrolu chyb při přenosu dat. To má za následek také nižší kvalitu audia a videa.

UDP je schopen posílat pakety mezi několik aplikací běžících na stejném počítači. Pro tento účel byly zavedeny porty [3]. Existuje 65535 TCP a UDP portů, se kterými lze přenášet data. Tyto porty jsou přiřazeny IANA autoritou (Internet Assigned Numbering Authority). To znamená, že každý port 0-1023 bude mít v každém systému stejný význam. Například port 80 slouží pro HTTP spojení po celém světě. Porty 1-1023 jsou

tzv. dobře známé porty, porty 1024-49151 jsou registrované porty a porty 49152-65535 se používají pro komunikaci klineta se serverem.

32bit	
16bit	16bit
zdrojový port	cílový port
délka	kontrolní součet
data	

Obr. 2.3: Hlavička UDP protokolu [4]

Význam polí v hlavičce UDP protokolu:

- **Zdrojový port -** Je volitelné pole, protože odesílatel nemusí požadovat odpověď a značí port výchozího zdroje dat. Lze jej považovat za port, na který má být zaslána odpověď. Pokud není použit, je nastaven na hodnotu nula [4].
- **Cílový port -** Má význam jen s cílovou internetovou adresou.
- **Délka -** Je délka tohoto datagramu v oketech zahrnující hlavičku i data. Nejmenší délka je tedy osm bajtů.
- **Kontrolní součet -** Je to 16-bitový kontrolní součet, zahrnující hlavičku i data. Lze jej vynechat, ale většinou se používá.
- **Data -** Pole pro přenášená data.

2.3 RTP protokol

RTP neboli real-time protokol poskytuje tzv. end-to-end datovou službu pro real-time data. Jedná se o audio nebo video data přes unicastové nebo multicastové spojení. RTP jako takový neposkytuje žádný mechanismus pro zajištění doručení dat v určitém čase nebo QoS, spoléhá na to, že to zajistí nižší vrstva. Nezajišťuje také doručení dat a doručení dat v pořadí.

Aplikace pro audiokonference používané každým z účastníků, posílají data po malých částech, které trvají, řekněme 20 ms [6]. Před každou částí audio dat se nachází RTP hlavička. RTP hlavička a data jsou pak převedeny do UDP paketu. RTP

hlavička určuje jakým typem je audio kódováno (PCM, ADPCM nebo LPC) v každém paketu. Odesílatelé tak mohou změnit kódování během konference a to například proto, aby se přizpůsobili novému účastníkovi.

Internet jako paketová síť příležitostně ztrácí, přeskupuje pořadí paketů a zpožďuje je o různý čas. Aby se s tím RTP vyrovnalo, má obsaženo v hlavičce časové razítko (timestamp) a sekvenční číslo (sequence number), které umožňuje příjemci rekonstrukci časové produkce zdroje. To způsobí, že se bude audio plynule přehrávat každých 20ms. U videa je důležité sekvenční číslo, aby bylo možno určit správnou polohu paketu pro správné dekódování. Sekvenční číslo může také indikovat, kolik paketů bylo ztraceno.

2	3	4	8	9	16bit	32bit
V	P	X	CSRC count	M	Payload type	Sequence number
Timestamp						
SSRC						
CSRC						
Data						

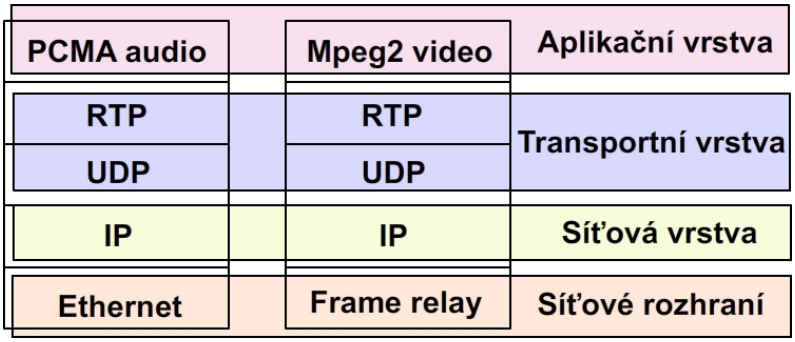
Obr. 2.4: Hlavička RTP protokolu [6]

Význam polí v hlavičce RTP protokolu [6]:

- **V-version:** Identifikuje verzi RTP.
- **P-padding:** Pokud je toto pole nastaveno, paket obsahuje jeden nebo více dodatečných oketů na konci, které nejsou součástí dat.
- **X-extension bit:** Pokud je bit nastaven, za pevnou hlavičkou následuje právě jedno rozšíření s definovaným formátem.
- **CRSC count:** Obsahuje počet CSRC identifikátorů, které následují za pevnou hlavičkou.
- **Payload type (PT):** Jde o typ dat přenášených v paketu. PT má délku 7 bitů a může tak nabývat hodnot 0-127. Je definováno několik statických hodnot. Např. 0 vyjadřuje G.711 μ-Law, 8 značí G.711 A-Law a 18 reprezentuje G.729. Interval mezi 96-127 je rezervován pro dynamický typ.

- **Sequence number:** Sekvenční číslo začíná náhodnou hodnotou a zvyšuje se s každým poslaným RTP paketem. To napomáhá k určení paketů, které jsou mimo pořadí.
- **Timestamp:** Timestamp neboli časové razítko je podobné sekvenčnímu číslu. Začíná také náhodnou hodnotou. Hodinová frekvence záleží na typu PT.
- **SSRC -** Synchronization Source Identifier je 32 bitový identifikátor původce audio/video streamu.
- **CSRC -** Identifikují podporující zdroje dat obsažených v paketu.

Ikdyž RTP data přenáší UDP protokol, který nezajišťuje spolehlivost doručení dat, RTP poskytuje určitou schopnost spolehlivosti přenosu dat mezi uživatelskými agenty. Pro tento účel protokol používá Real-time control protokol (RTCP). RTCP definuje tři typy zpráv: Receiver report (RR), Sender report (SR) a Source description (SDES). Zprávy obsahují statistiky typu kolik paketů bylo posláno, počet ztracených paketů a statistika jitter. RTCP tedy monitoruje QoS a zpracovává informace o uživatelích během relace.



Obr. 2.5: Zařazení RTP do TCP/IP modelu [6]

2.4 TLS protokol

TLS (Transport Layer Security Protocol) je protokol, který zajišťuje bezpečnou komunikaci mezi aplikacemi a jejich uživateli na internetu. TLS lze použít s jinými protokoly jako například s UDP. Používá se pro bezpečné spojení mezi klientem a serverem nebo UA klientem a UA serverem. TLS je nástupcem SSL (Secure Sockets Layer). Protokol se skládá ze dvou vrstev [7]. První vrstva je TLS Record protokol a druhá TLS Handshake protokol.

2.4.1 TLS Record protokol

Připojení je soukromé [7]. Pro šifrování dat je použita symetrická kryptografie (DES, RC4 atd.). Klíče pro šifrování jsou unikátně generovány pro každé spojení a jsou založené na tajném vyjednávaní pomocí jiného protokolu, jako je TLS Handshake protokol. TLS jak Record protokol lze použít i bez šifrování.

Připojení je spolehlivé. Ve zprávě je obsažena kontrola integrity používající MAC (Message Authentication Code). Pro výpočet MAC jsou použity hashovací funkce SHA, MD5 atd.

TLS Record protokol se používá pro zapouzdřování protokolů vyšších vrstev. Po zapouzdření protokol TLS Handshake umožní serveru a klientovi prokázat svoji totožnost a sjednat šifrovací algoritmus a kryptografické klíče ještě dříve, než aplikační protokol pošle nebo obdrží jeho první data.

2.4.2 TLS Handshake protokol

Uživatelská identita [7] může být prokázána asymetrickou kryptografií nebo veřejným klíčem (RSA, DSS). Tato autentizace je volitelná, ale často bývá vyžadována alespoň od jednoho z uživatelů.

- Vyjednávání o společných tajných datech je bezpečné.
- Vyjednávání je spolehlivé. Žádný útočník nemůže změnit komunikaci, aniž by si toho všimli účastníci komunikace.

Pro SIP použití TLS protokolu není důležité. Pro obvyklou VoIP komunikaci mezi přáteli je použití TLS zbytečné. Tento protokol se používá tehdy, pokud je nutné zajistit bezpečnost citlivých dat. To se týká především firemních VoIP hovorů, kde dochází k výměně důvěrných informací.

2.5 SDP protokol

Při použití VoIP, streamovaného videa nebo jiných relací je zapotřebí poskytnout účastníkům metadata, jako jsou například parametry media nebo transportní adresa. SDP zavádí standardní zápis pro přenos těchto informací. SDP je čistě jen formát pro popis relace. Používá se tedy s vhodným transportním protokolem [8].

Popis SDP relace obsahuje tyto informace:

- Název relace
- Typ média; video, audio...
- Transportní protokoly; RTP/UDP/IP, H.320 ...
- Formát média; H.264 video, MPEG video ...

Multikastová IP relace zahrnuje informace:

- Multikastovou skupinovou adresu média
- Přenosový port pro médium

Unikastová IP relace zahrnuje tyto informace:

- Vzdálenou adresu média
- Vzdálený port média

SDP může obsahovat také časové informace. A to jakýkoli seznam, který definuje časový začátek a konec relace nebo lze relaci opakovat v určitém časovém úseku.

2.6 Domain name system (DNS)

Jedná se o hierarchický systém domén pro pojmenování počítačů a síťových služeb [9]. DNS se používá v sítích TCP/IP k nalezení počítačů a služeb pomocí textových adres. Jakmile uživatel zadá DNS jméno do aplikace, DNS služba jej zpracuje do jiné informace spjaté s tímto jménem a to do IP adresy.

Většina uživatelů dává přednost textové formě adresy, protože je dobře zapamatovatelná. Avšak počítače v síti komunikují pomocí numerických adres. Klient tedy odešle požadavek ve formě textové adresy na DNS server a ten mu vrátí příslušnou adresu, která má numerickou formu.

Obr. 2.6: Komunikace DNS klienta a DNS serveru [9]

Architektura DNS se sestává z DNS databáze, DNS serveru a DNS klienta. V DNS databázi jsou uloženy aktuální jména hostů a k nim příslušná IP adresa. U DNS serveru se jedná o software, který na požádání poskytuje informace z databáze. DNS klient, jednoduše řečeno, je program na straně uživatele, který vznáší dotaz na DNS server, když se snaží připojit k jinému počítači. DNS pro přenos používá port 53 pro TCP i UDP.

3 Tvorba aplikace

3.1 Popis aplikace

Práce je zaměřena na vývoj aplikace klient-server, která umožní přenos hlasu. Pro tvorbu IMS aplikace bylo zvoleno prostředí SDS (Service Development Studio) Ericsson verze 4.1. FD1. Toto prostředí je založeno na open source vývojové platformě Eclipse. SDS Ericsson obsahuje emulátory serverů a terminálů, simulátor IMS sítě a ICP platformu. Pro testování IMS sítě v prostředí SDS byl použitý klient OpenIC lite. Tento klient byl použit i při testování vyvíjeného klienta.

Další část práce se věnuje tvorbě webové aplikace sloužící pro monitorování jednotlivých hovorů uživatelů. Tato aplikace je vytvořena pomocí PHP, MySQL databáze a Flash (ActionScript2). Programovací jazyk PHP byl zvolen, protože je to open source software, je zdarma a umožňuje snadné programování webových aplikací. Flash byl použit, protože podporuje vektorovou grafiku a umí animovat objekty. Vytváří tak pro uživatele atraktivnější webové prostředí.

Jako úložiště pro aplikaci byla zvolena databáze MySQL. Je to databáze, která je na rozdíl od PostgreSQL zdarma. Co se týče funkcí, je PostgreSQL pestřejší. Avšak pro účely této aplikace je MySQL databáze dostačující.

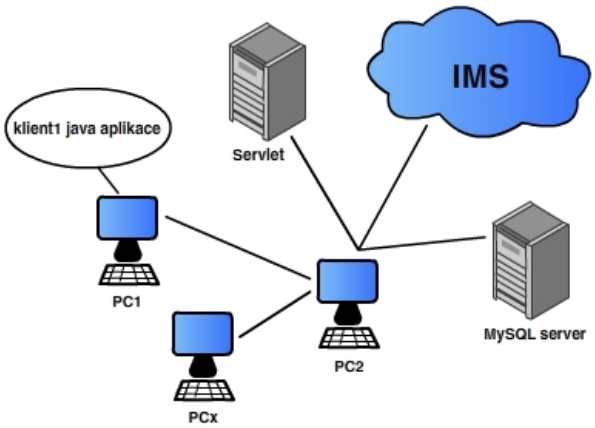
Operační systém, na kterém byl nainstalován SDS Ericsson, byl zvolen Windows XP. Aby prostředí správně fungovalo, Windows XP nesmí obsahovat vyšší aktualizační balíček než Service Pack 2. Taktéž prostředí není funkční na Windows Vista a Windows 7.

Grafické rozhraní aplikace bylo vytvořeno pomocí doinstalovaného balíčku Jigloo (pro nekomerční účely), určeného pro Eclipse. Pomocí tohoto GUI pluginu lze pohodlně vytvářet komponenty, kontejnery a menu aplikace.

3.2 Struktura projektu pro hlasovou službu

Aplikaci tvoří čtyři hlavní části:

- Webová část aplikace
- Aplikace klienta
- Aplikace Servlet
- Databáze (MySQL)



Obr. 3.1: Struktura aplikace podporující hlasovou službu

Obrázek 3.1 znázorňuje realizaci projektu. PC1 tvoří java aplikace pro hlasovou službu. PC2 tvoří aplikace Servlet, MySQL databázi a prvky IMS – konkrétně CSCF, HSS, DNS. Na PC1 je také testována webová aplikace umožňující monitorování hovorů. PCx představuje volaného klienta. Veškeré části aplikace byly testovány na jednom PC.

3.3 Webová aplikace

Vytvořená webová aplikace slouží pro monitorování jednotlivých hovorů uživatelů. Tato aplikace je vytvořena pomocí PHP, MySQL databáze a Flash (ActionScript 2). Aplikace je rozdělena do dvou částí. První část je určena pro administrátora, kde je možné spravovat uživatelovy data, zobrazit výpis volání uživatele a vytvářet nové operátory s cenou hovoru, které lze potom přiřadit jednotlivým uživatelům. Druhá část aplikace je určena pro uživatele. Uživatel si zde může zobrazit provolané minuty za určitý měsíc a to pomocí výpisu do tabulky nebo jako flashový graf.

Aby byl možný vývoj aplikace na lokálním PC, bylo na něj potřeba nainstalovat Apache, PHP a MySQL. Nemusí se tak každá část aplikace testovat na hostitelském serveru a tím se vývoj aplikace výrazně zrychlí. Jako vývojové prostředí pro Windows byl zvolen WampServer, který obsahuje Apache, PHP, MySQL a umožňuje jednoduché nastavení jednotlivých částí. WampServer byl zvolen pro jeho jednoduchou instalaci a nastavení.

3.3.1 Apache

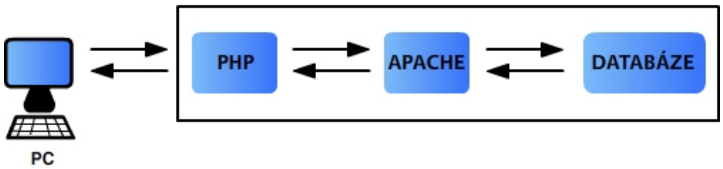
Apache je volně přístupný a zastává funkci softwarového webového serveru. Zpracovává požadavky uživatelů a stará se o jejich zobrazení. První verze Apache byla vydána v roce 1995. Jádro Apache je spravováno skupinou dobrovolných programátorů. Protože je zdrojový kód volně přístupný, kdokoli si může server přizpůsobit svým požadavkům. Vývoj Apache se velice podobá vývoji operačního systému Linux. Původní verze byla napsána pro UNIX. Nyní však existují verze pro Windows a jiné operační systémy.

3.3.2 PHP

PHP (Hypertext Preprocessor) je skriptovací jazyk na straně serveru. Pomocí PHP se programují dynamické internetové stránky. Nejčastěji se tento skriptovací jazyk začleňuje do jazyka HTML. Skript se provádí na straně serveru a uživatel obdrží jen výsledek operace. PHP obsahuje knihovny pro podporu práce s textem, soubory, grafikou a umožňuje přístup k databázi (např. MySQL nebo PostgreSQL). PHP jazyk je založený na jazyce C, Perl, Pascal a Java.

3.3.3 MySQL

MySQL je databázový systém, který společně s PHP umožní uživateli uložení a zobrazení dat pomocí webového prohlížeče. Zpracovává dotazy ve strukturovaném jazyce (SQL – Structured Query Language). Data nejsou ukládány do jedné tabulky, ale do několika různorodých tabulek. Tím je zajištěna maximální efektivita a rychlost.



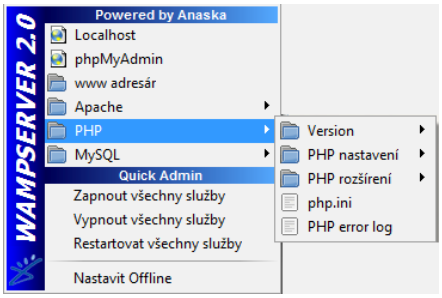
Obr. 3.2: Komunikace klienta s databází

3.3.4 Nastavení WampServeru

Nejpreve [18] je nutné stáhnout instalátor *WampServer2.0h.exe*. To je možné z internetových stránek http://www.wampserver.com/en/. Po spuštění souboru se rozběhne klasická windowsovská instalace, která nepotřebuje bližší pozornost. Po instalaci je nutné nastavit u Apache modul rewrite_module. U PHP nastavení se vypne

safe mode a zapne *file uploads* a *allow url fopen*. V PHP rozšíření se aktivujte *php_curl*, *php_gettext*, *php_gd2*, *php_mysql*, *php_mysqli*, *php_sqlite* a *php_zip* (pro tuto aplikaci není nutné aktivovat vše z tohoto výčtu).

Veškeré soubory využívající WampServer se pak ukládají do složky *c:\wamp\www*. Testování skriptu se provádí na http://localhost/. Za touto adresou následuje cesta ke skriptu (např. **http://localhost/dp/php/login.php**).



Obr. 3.3: Nastavení WampServeru

3.3.5 Vytvoření databáze

Prvním krokem pro tvorbu webové aplikace bylo vytvoření návrhu databáze. Na obrázku 3.4 je zobrazen UML diagram, který znázorňuje strukturu celé databáze.

Po spouštění WampServeru se pomocí webového prohlížeče zobrazí úvodní stránka WampServeru http://localhost/. Ná této stránce v nástrojích se zvolí položka *phpmyadmin* a následně záložka *databáze*. Zde byla vytvořena nová databáze s názvem *ims*, která obsahuje tři tabulky:

- calls
- operators
- users

Do tabulky *users* se ukládají údaje o uživateli. Nejdůležitějším údajem je *id***.** Je to unikatní identifikátor uživatele, který ho provází celou databází. Dalším unikátním údajem této tabulky je uživatelovo *URI* a *login*. Ostatní údaje již unikátní být nemusí. K jednotlivým údajům tabulky lze přistupovat podle toho, zda je uživatel přihlášen jako administrátor nebo jako obyčejný uživatel.

Další prvky tabulky users:

- **URI** – jedná se o adresu podobné e-mailové adrese. Tato adresa musí být unikátní a má hierarchickou strukturu. Používá se pro identifikaci různých SIP účtů.
- **login** – je sada znaků, která se používá společně s heslem pro přihlášení do systému. Login do databáze vkládá administrátor. Pro snadnou orientaci by měl začínat pismenem *x*, dále následuje pět znaků z přijmení a končí dvouciferným číslem. Např. xnovot00.
- **password** – pod touto položkou se nachází heslo, které společně s loginem slouží pro přihlášení do systému.
- **name** – jméno uživatele.
- **surname** – příjmení uživatele. Jméno a přijmení se zobrazuje u výpisu hovorů.
- **e-mail** – slouží jako kontaktní e-mailová adresa uživatele.
- **date_last_login** – do této položky se ukládá datum posledního přihlášení uživatele.

Tabulka *operators* obsahuje vytvořené operátory. Operátory vytváří administrátor a přirazuje je jednotlivým uživatelům. Aministrátor volí také název operátora a cenu hovoru za minutu.

Položky tabulky operators:

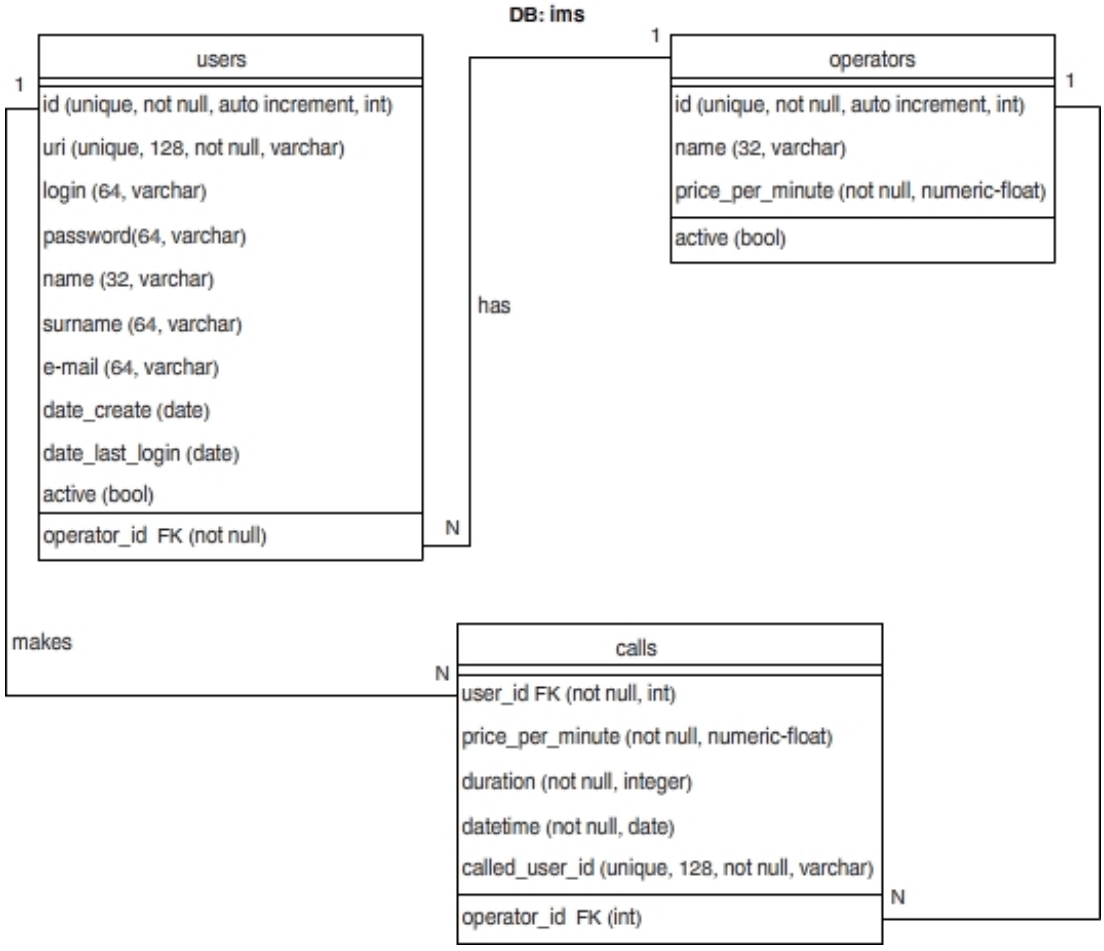
- **id** – unikatní identifikační číslo operátora, které má význam jen v rámci databáze.
- **name** – jméno operátora.
- **price_per_minute** – cena hovoru daného operátora za minutu.

V tabulce *calls* jsou zaznamenány jednotlivé hovory uživatelů. U jednotlivých hovorů se zaznamenává i cena za minutu, protože se cena u operátora může během období změnit.

Položky tabulky calls:

- **user_id** – unikátní identifikátor uživatele, který uskutečnil hovor.

- **price_per_minute** – cena za minutu hovoru.
- **duration** – celkový provolaný čas za uskutečněný hovor. Čas je uložen v sekundách a v aplikaci je potom převeden do formátu MM:SS.
- **datetime** – datum kdy byl hovor uskutečněn.
- **called_user_id** – do této položky se ukládá, komu uživatel volal. Ve výpisu se pak zobrazí URI volaného.



Obr 3.4: UML diagram

3.3.6 Tvorba a popis PHP aplikace

Při tvorbě PHP aplikace pro monitorování hovorů bylo nutné odlišit uživatelskou část od administrátorské části. Rozdělení probíhá již na úvodní stránce při přihlašování do systému a to pomocí php skriptu obsaženém v souboru *login.php*. Na obrázku 3.5 je vidět výpis části php kódu.

Pro administrátora byl zvolen login: **admin** a heslo: **1234**. Tento login a heslo je ve skriptu zadán pevně. Z bezpečnostního hlediska je to zcela nevyhovující, slouží to jen pro účel testování aplikace. Pokud by aplikace měla být dále využívána, heslo a login administrátora by se uchovával v databázi a byla by tak možná jejich změna dle potřeby. Skript by vypadal podobně jako pro přihlášení uživatele.

Protože hlavní úlohou této aplikace je přistupovat do databáze, byla vytvořena třída s názvem *DB*, která je definována v souboru *class_db.php.* Tato třída je pak volána prakticky ze všech php souborů, ze kterých je aplikace složena. Pokud se volá skript, který není obsažen ve stejném souboru, je třeba jej začlenit do kódu pomocí funkce **require '*název_soboru*'**. V případě souboru *login_php* v části pro přihlašování uživatele je to tedy **require 'class_db.php'**, viz zdrojový kód na obrázku 3.4, řádek 16. Na řádku 19 a 20 je volána samotná třída *DB*. Následuje závorka s několika klíčovými slovy:

- ***SELECT*** – určuje co má být z databáze vybráno. Pro přihlášení uživatele jsou to položky login, password (heslo) a acitve. Active má hodnotu 1 nebo 0 a nastavuje ji administrátor. Pokud je uživatel aktivní (má hodnotu 1), uživatel má přístup do systému.
- ***FROM*** – za toto klíčové slovo se píše název databázové tabulky a za název alias této databáze, který se používá v *SELECT* a *WHERE*. Z řádku 19 na obr. 3.4 tedy vyplývá, že se přistupuje do databázové tabulky s názvem *users*, která má alias *u*. Za SELECT tedy následuje *u.login*, tzn., že *u* je alias tabulky *users* a *login* je položka tabulky *users*.
- ***WHERE*** – za tímto příkazem se píše podmínka, která má být splněna. V případě této to aplikace to znamená, že uživatel musí být aktivní a musí souhlasit vyplněný login a heslo.

Pokud byl databázový dotaz úspěšný, uloží se do proměnné *$_result* číslo *1*. Splní se tím podmínka a skript odkáže uživatele na stránku *user.php*. Pokud podmínka není splněna, vypíše se hláška *Neplatné heslo nebo login*.

```php
<?php
session_start();
//    var_dump($_SESSION['admin']);
$msg="";
if(isset($_SESSION['admin']) && $_SESSION['admin']===true){
  header("Location: admin.php");
  exit;
  }
if(isset($_POST['submit'])){
   if($_POST['admin']=="admin" && $_POST['pass']=="1234"){
    $_SESSION['admin']=true;
    header('Location: admin.php');
    exit;
   }
   else {
      require 'class_db.php';
      $admin=$_POST['admin'];
      $passwd=$_POST['pass'];
      $_result=DB::Query("SELECT u.login, u.password, u.active FROM `users` u WHERE u.active=1  AND u.login='{$_POST['admin']}
                          ' AND u.password='{$_POST['pass']}'  ");
        if(mysql_num_rows($_result)==1) {
          $_SESSION['user']=true;
          header('Location: user.php');
          exit;
         }else{
         $msg="Neplatné heslo nebo login";

     }
   }
}
?>
```

Obr. 3.5: Výpis PHP skriptu pro přihlášení do systému

Administrátorská část

Po úspěšném vyplnění přihlašovacích údajů pro administrátora odkáže php skript *login.php* pomocí funkce *header('Location: admin.php');* na stránku *admin.php*. V horní části stránky jsou umístěny funkční tlačítka a nápis **Uživatelé** nebo **Operátoři**, podle toho v jaké sekci se administrátor nachází. Tlačítko *Uživatelé* zobrazí tabulku s registrovanými uživateli, viz obrázek 3.6

.

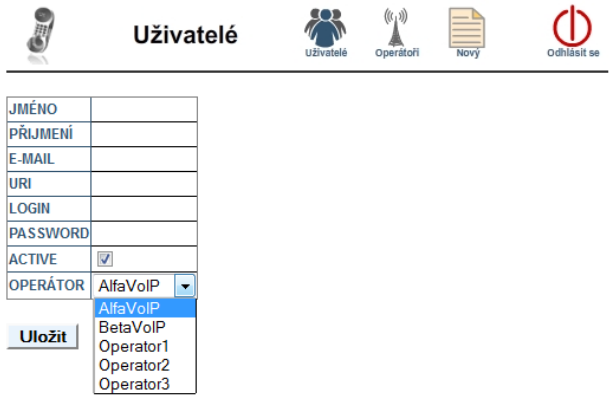
ID	LOGIN	JMÉNO	PŘÍJMENÍ	URI	E-MAIL	AKTIVNÍ	OPERÁTOR	CELKOVÝ ČAS	VOLÁNÍ	upravit	smazat
10	xkoube00	Radek	Koubek	sip:radek@ericsson.com	radek.k@gmail.com	ANO	Operator 1	2 min 5 s			
15	xneusc00	Adam	Neuschl	sip:adam@ericsson.com	filip@cover.cz	ANO	Operator2	6 min 0 s			

Obr. 3.6: Administrace – výpis uživatelů

Výpis hovorů daného uživatele se vyvolá stisknutím tlačítka znázorňující hodiny a to ve sloupci *VOLÁNÍ,* ve výpisu uživatelů (obr. 3.6). Výpis hovorů se zobrazí pro aktuální měsíc a rok. Pomocí šipek *měsíc* a *rok* se pohybuje mezi jednotlivými měsíci a roky. Administrátor má také možnost jednotlivé hovory uživatelů mazat (obr. 3.7).

Obr. 3.7: Administrace – výpis hovorů uživatele

Nový uživatel se vytvoří stisknutím tlačítka *Nový* v sekci *Uživatelé*. V tabulce (viz obr. 3.8) se vyplní uživatelovy údaje, přiřadí se mu operátor a tlačítkem *Uložit* se odešlou data do databáze.



Obr. 3.8: Administrace – vytvoření uživatele

Tlačítko *Operátoři* zobrazí tabulku s existujícími operátory (viz obr. 3.9). Podobně jako u uživatelů, se nový operátor vytvoří pomocí tlačítka *Nový* v sekci *Operátoři*. Do tabulky se vyplní jméno operátora, zda je operátor aktivní a cena za minutu hovoru. Data se uloží do databáze opět pomocí tlačítka *Uložit.*

ID	NÁZEV	AKTIVNÍ	CENA ZA MINUTU	POČET UŽIVATELŮ	upravit	smazat
2	AlfaVoIP	ANO	3	3		
15	BetaVoIP	ANO	5	1		
16	Operator1	ANO	6	0		
19	Operator2	ANO	3	0		
20	Operator3	ANO	3	0		

Obr. 3.9: Administrace – výpis operátorů

Uložená data lze jak v sekci *Uživatelé* tak i v sekci *Operátoři* změnit pomocí tlačítka *upravit*. Tlačítko *upravit* se nachází vždy v zobrazené tabulce s uživateli nebo operátory (např. viz obr. 3.9). Tlačítko *smazat* smaže uživatele nebo operátora z databáze.

Posledním tlačítkem v horní části stránky je tlačítko *Odhlásit se*, které odhlásí přihlášeného administrátora nebo uživatele ze systému.

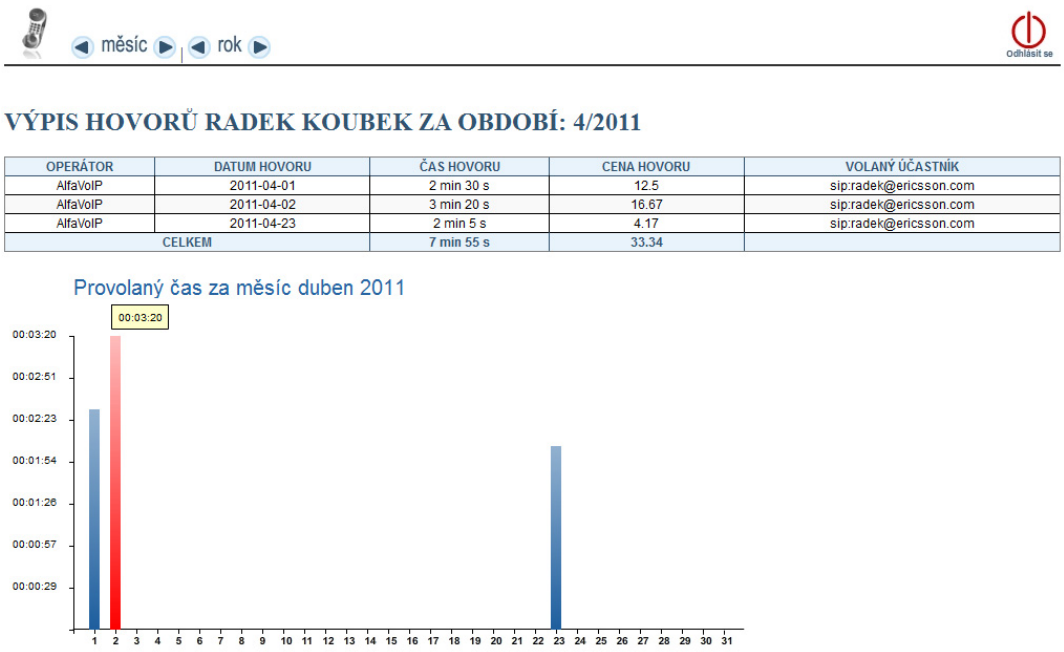
Uživatelská část

Po vyplnění přihlašovacích údajů pro uživatele odkáže php skript *login.php* na stránku *user.php*. Hned po přihlášení se uživateli zobrazí výpis hovorů za aktuální měsíc a pod ním flashový graf, který slouží pro názorné zobrazení provolaných minut. Najetím myši na sloupec grafu, se zobrazí přesný počet provolaných minut za daný den (obr. 3.10). Podrobnější popis flashové části je v kapitole 3.4.

Ve výpisu hovorů uživatel vidí s jakým operátorem hovor uskutečnil, datum hovoru, kolik provolal minut, cenu hovoru za provolaný čas a volaného účastníka.

Podobně jako v administrátorské části uživatel může listovat mezi jednotlivými měsíci a roky pomocí šipek *měsíc* a *rok*, tentokrát umístěných v horní části stránky.

Uživatel nemá možnost vytvářet nové položky a ani je mazat. Tato část aplikace slouží jen pro zobrazení uživatelových záznamů.



Obr. 3.10: Uživatelská část – výpis hovorů uživatele

3.4 Flashová aplikace

3.4.1 ActionScript a Flash

ActionScript je skriptovací jazyk společnosti Macromedia Flash, pomocí kterého je zajištěna komunikace s flashovým programem. Komunikace probíhá dvoucestně. ActionScript říká Flashi jak se má chovat a dotazuje se Flashe co se děje za jeho běhu. To umožňuje vytváření interaktivních flashových animací či aplikací. ActionScript vychází z jazyka JavaScript.

První verze Flashe nabízely jen omezené interaktivní možnosti. Mezi tyto možnosti patřily příkazy jako play, stop, gotoAndPlay, gotoAndStop, které umožňovaly pohyb po takzvané časové ose animace a příkaz getURL, který otevíral URL adresy. Později přibyly funkce např. pro načítání XML souboru nebo pro komunikaci s PHP skriptem. Díky těmto funkcím je možné vytvářet RIA aplikace (Rich Internet Application) přístupné pomocí webového rozhraní. Jedná se o různé hry, grafy, interaktivní animace apod. RIA aplikace se v dnešní době snaží přiblížit svou funkčností desktopovým aplikacím. Umožňují uživatelům editovat prvky stránky či měnit vzhled pomocí funkcí drag and drop.

RIA aplikace mají lepší odezvu při práci než klasické stránky vytvořené pomocí HTML nebo PHP. Taková aplikace je nejprve stažena na klientský počítač a není tak

závislá na další komunikaci se serverem (pokud není třeba ukládat data do databáze na serveru). Proto dojde-li ke změně v aplikaci, není nutné tuto změnu posílat na server, který provede úpravu a vrátí zpět výsledek. Je to jeden z důvodů proč byl vybrán Flash pro grafické znázornění provolaného času. Veškeré změny se provádí u klienta. Zvyšuje se tak rychlost odezvy aplikace a nabízí se také možnost pracovat s RIA aplikací i při výpadku připojení k internetu.

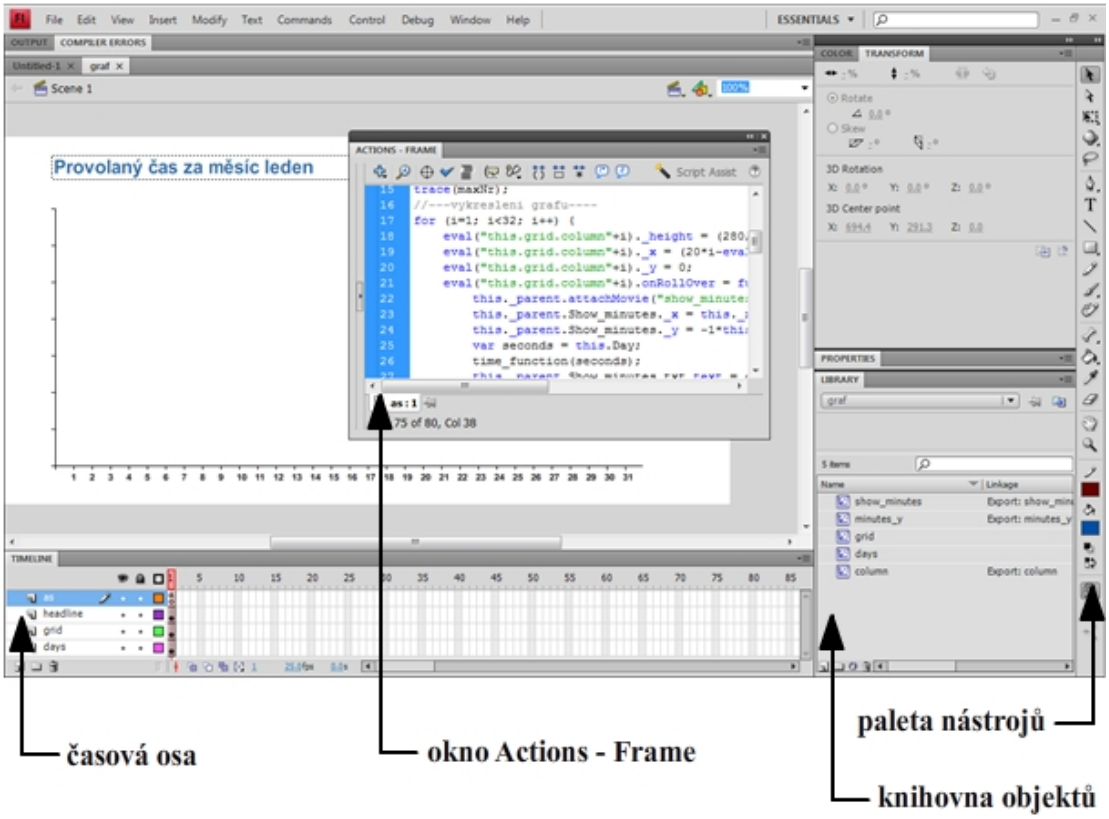
Existují tři verze ActionScriptu:

1. ActionScript 1.0 – je to nejjednodušší forma ActionScriptu
2. ActionScript 2.0 – je dobrý pro výpočetně nenáročné projekty
3. ActionScript 3.0 – má lepší zpracování XML a lépe pracuje s obrázkovými elementy

Pro přehrání flashové RIA aplikace je nutné mít nainstalovaný ve webovém prohlížeči přehrávač požadované verze. Existuje mnoho verzí přehrávačů od nejstaršího Flash Player 2 až po nejnovější Flash Player 10. Poslední čtyři verze s jejich popisem jsou uvedeny níže.

- **Flash Player 7**: Tento přehrávač začíná podporovat ActionScript 2.0 a podporuje CSS stylování pro text.
- **Flash Player 8**: Obsahuje knihovny s API pro práci s bitmapu za běhu aplikace, filtry pro rozostření a vržení stínu.
- **Flash Player 9**: Tento přehrávač již podporuje ActionScript 3.0. Obsahuje podporu pro fullscreen mód. Hlavním cílem tohoto přehrávače byl nárůst celkového výkonu.
- **Flash Player 10**: Kromě x a y osy obsahuje také osu *z* a 3D rozhraní API. Umožňuje rychlejší přehrávání a to především videí ve formátu H.264.

3.4.2 Seznámení s programem Adobe Flash CS4 Professional



Obr. 3.11: Hlavní okno programu Adobe Flash CS4 Professional

Časová osa

Do časové osy se vkládají snímky animace. Snímky pak obsahují jednotlivé části animace a také mohou obsahovat ActionScript, který ovládá objekty animace a samotnou časovou osu. Časová osa se dále dělí do jednotlivých vrstev.

Okno Actions – frame

Do tohoto okna se píše ActionScript, který definuje akce objektů obsažených v animaci nebo ovládá časovou osu pomocí funkcí play(), stop(), gotoAndPlay(), gotoAndStop() apod. Rozlišují se dva typy skriptů: synchronní a asynchronní. Synchronní skripty [15] běží souběžně s přehráváním animace. Náročnější skripty tak mohou zpomalit běh animace. Jedná se o skripty přiřazené jednotlivým snímkům animace a jsou označeny ve snímku symbolem *a*. Asynchronní skripty jsou součástí složitějších programů a nejsou závislé na pozici přehrávání (na poloze na časové ose). Asynchronní skripty náleží nejčastěji tlačítkám nebo MovieClipům.

MovieClip

MovieClip je objekt, který obsahuje vlastní časovou osu. Pomocí metod a vlastností mu lze přiřadit jeho chování. MovieClip může v sobě obsahovat další MovieClipy. Aby mu bylo možné přiřadit metodu nebo vlastnost, je potřeba u něj definovat jméno instance. Důležité také je, na které úrovni (level) se MovieClip nachází. MovieClip, který se nachází na hlavní časové ose scény, je na úrovni _root. Pokud je vytvořen na hlavní časové ose MovieClip s názvem *buttons_mc* a v něm další např. *button_play*, který bude reprezentovat tlačítko přehrát, bude zacílení tohoto tlačítka vypadat takto: *_root.buttons_mc.button_play*. Za tímto zacílením následuje metoda nebo vlastnost, která určuje chování tlačítka. Tomuto zápisu se říká absolutní zápis. Relativní zápis by vypadal takto: *this.buttons_mc.button_play*. Následující skript spustí hlavní časovou osu po kliknutí na MovieClip *button_play* (tlačítko přehrát).

Absolutní zápis:

```
_root.buttons_mc.button_play.onRelease = function() {
_root.play();
};
```

V relativním zápisu by to vypadalo takto:

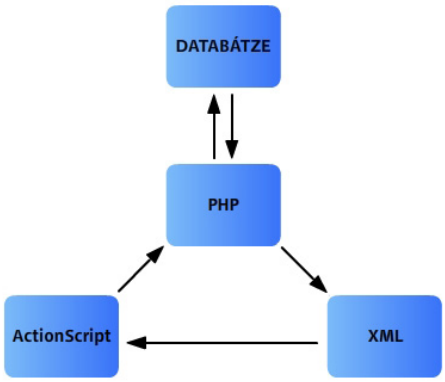
```
this.buttons_mc.button_play.onRelease = function() {
this._parent._parent.play();
};
```

Veškeré MovieClipy, obrázky, zvuky apod., které animace obsahuje jsou zobrazeny v okně LIBRARY.

3.4.3 Tvorba dynamicky plněného grafu

Tento graf slouží pro zobrazení provolaných minut za určitý měsíc. Data jsou načítána dynamicky z databáze na serveru.

Flash jako takový neumí přistupovat přímo k databázi na serveru. Aby bylo možné zpracovat data pomocí Flashe, je nutné je zformátovat do formátu XML. XML soubor lze uložit přímo na server s pevně uloženými daty nebo jej lze generovat dynamicky pomocí PHP. Z důvodu nezahlcování serveru daty, je XML v tomto případě generováno dynamicky.



Obr. 3.12: Generování XML pomocí PHP

XML

XML (eXtensible Markup Language) je jazyk pro kódování dokumentů pro strojní čtení. Nejedná se o programovací jazyk. XML je navrženo spíše pro popis struktury dokumentu než pro popis jeho vzhledu. U XML souboru je nejdůležitější jeho hierarchická struktura, ne jeho vzhled.

Podstata XML syntaxe je jednoduchá a je podobná jazyku HTML. XML soubor je složen z obsahu, který je strukturován pomocí značek. Obsahem se míní byty, které představují mezery, interpunkci a další ASCII znaky a čísla. Znaky, které nespadají do ASCII, jsou reprezentovány pomocí entit. Entita je více bytový řetězec, který začíná ampersandem a končí středníkem. Např. entita *>* reprezentuje znak větší než >. V obsahu XML souboru znaky jako <,>,& musí být reprezentovány pomocí entit.

XML tagy (značky) identifikují a vymezují jednotlivé prvky obsahu v XML dokumentu. Tag je uvozen znakem menší než < a ukončen znakem větší než > (např. <name>). Každý tag je párovým tagem. To znamená, že každý začátek obsahu je uvozen tzv. otevíracím tagem (např. <name>) a ukončen zavíracím tagem, který je stejný jako otevírací, ale obsahuje znak / (např. </name>). XML podporuje i prázdný tag. Otevírací tag může obsahovat kromě jména také atribut typu jméno-hodnota, který popisuje objekt spjatý s tímto tagem. Atribut se skládá z názvu a znaku =, za kterým následuje hodnota uzavřená v uvozovkách (např. <name id=”01”>). Komentáře se do XML souboru píší takto: <!-- *zde komnetář* -->.

Návrh struktury XML pro flashový graf

Pro flashový graf zobrazující provolaný čas za jednotlivé dny v určitém měsíci je struktura XML následující:

```
<?xml version="1.0" encoding="utf-8"?>
<content>
<year><![CDATA[2010]]>< /year >
<month><![CDATA[12]]>< /month >
<day id=”1” seconds=”180”></day>
        <day id=”2” seconds =”170”></day>
        <day id=”3” seconds =”120”></day>
        <!-- následují další dny -->
        <day id=”31” seconds =”150”></day>
</content>
```

První řádek obsahuje tag, ve kterém je uvedena verze XML a kódování souboru. Na druhém řádku tag <content> značí začátek obsahu. V obsahu je uveden rok <year>, měsíc <month> a jednotlivé dny měsíce <day>.

V párovém tagu <year> je obsažen další tag <![CDATA[*zde patří text*]]>, který říká flashi, že jeho obsahem je HTML text. Tento text je pak načten do hlavičky samotného grafu. Číselná hodnota měsíce je převedena pomocí ActionScriptu na textovou podobu. V tomto případě hodnota 12 na prosinec. Otevírací tag <day> obsahuje dva atributy. Prvním atributem je *id*, který nese hodnotu dne v měsíci. Druhým atributem je *seconds*, který má hodnotu celkem provolaných minut za daný den. Po vypsání 31 dnů s jejich parametry, je obsah ukončen zavíracím tagem </content>.

Načtení XML do flashe

Na následujícím obrázku 3.13 je částečný výpis zdrojového kódu ActionScript, který načte data z XML. Na začátku skriptu je definovaná proměnná *xml_file*, která definuje název a umístění XML souboru. Dále je potřeba vytvořit proměnnou, do které budou XML data načteny. V tomto případě je to proměnná *xmlData* a je typu *XML*. Aby nebyla ze souboru načítána prázdná místa, přiřadí se hodnotě *xmlData* vlastnost *ignoreWhite* s hodnotou *true*. Funkce *load()* zajistí načtení dat z XML do proměnné *xmlData.*

Deklarace funkce *xmlData.onLoad = function(succ:Boolean)* vrací proměnnou *succ* typu boolean, která určuje, zda se XML podařilo načíst či nikoli. V případě že ano, je proveden skript, který podle struktury XML načte jednotlivá data.

V této aplikaci není načítán XML soubor, ale PHP skript *graph_data.php*, který vytváří XML tzv. OnTheFly (za běhu). Nedochází k ukládání XML souborů na server a zahlcování tak paměťového místa.

Jelikož navržená struktura XML je hierarchicky rozdělena do dvou úrovní, jsou potřeba dva *for* cykly. První cyklus se dostane do úrovně <content> a druhý načítá jednotlivé položky v této úrovni. Cykly se provádí, dokud je zajištěna existence dalšího uzlu v dané úrovni. Uzly v druhé úrovni jsou uloženy do proměnné *item1* typu *XMLNode*. Na základě jména uzlu (*nodeName*) je obsah uzlu nebo atribut uzlu (*atributes.seconds*) uložen do příslušné proměnné.

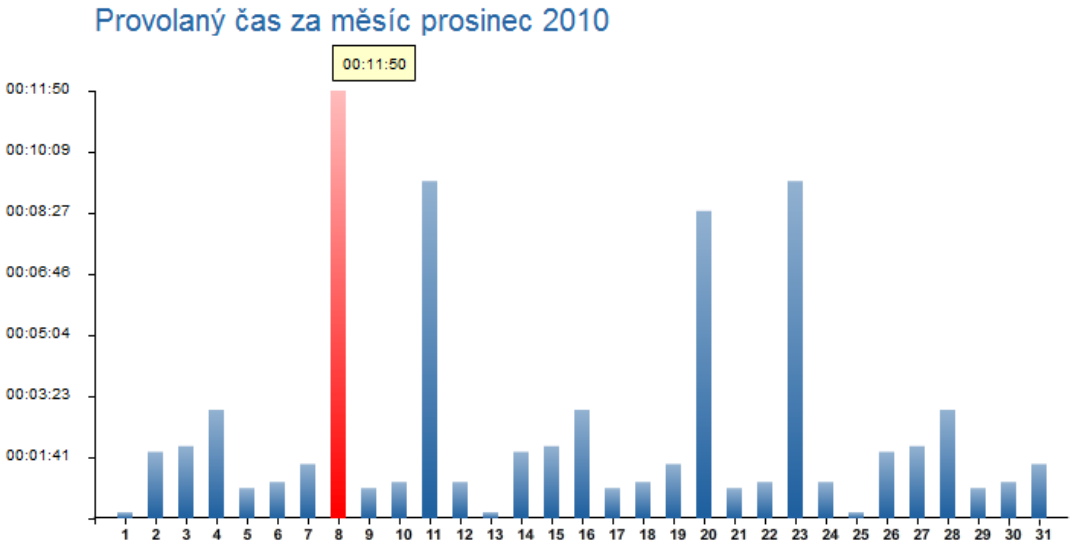
```
xml_file = new String("xml.php");
xmlData = new XML();
xmlData.ignoreWhite = true;
xmlData.load(xml_file);

xmlLoad = function () {
    xmlData.onLoad = function(succ:Boolean) {
        if (succ) {
            for (var item0:XMLNode = xmlData.firstChild; item0 != null; item0=item0.nextSibling) {
                for (var item1:XMLNode = item0.firstChild; item1 != null; item1=item1.nextSibling) {
                    if (item1.nodeName != null) {
                        var item_type = item1.nodeName;
                        switch (item_type) {
                            case "month" :
                                _root.Month = item1.firstChild.nodeValue;
                                break;
                            case "year" :
                                _root.Year = item1.firstChild.nodeValue;
                                break;
                            case "day" :
                                _root.grid.attachMovie("column","column"+i,_root.grid.getNextHighestDepth());
                                eval("_root.grid.column"+i)._x = (20*i-eval("_root.grid.column"+i)._width/2);
                                eval("_root.grid.column"+i)._y = 0;
                                //--
                                eval("_root.grid.column"+i)._height = 10;
                                eval("_root.grid.column"+i).Day = item1.attributes.seconds;
                                //--
                                if (maxNr<int(item1.attributes.seconds)) {
                                    maxNr = item1.attributes.seconds;
                                }
                                i++;
                                break;
                        }
                    }
                }
            }
        } else {
            trace("chyba v načítání dat");
        }
        draw_graph(maxNr);
        osay(maxNr);
        month();
    };
};
xmlLoad();
```

Obr. 3.13: Načtení XML souboru pomocí ActionScriptu

Aplikace grafu zjistí nejvyšší denní hodnotu provolaného času zvoleného měsíce. Z této hodnoty pak vychází výška jednotlivých sloupců. To znamená, že této nejvyšší hodnotě bude náležet nejvyšší sloupec grafu. Při prvotním načtení grafu jsou jednotlivé sloupce animovány. Na počátku mají nulovou výšku a postupně rostou směrem nahoru

od osy x až do výšky, kterou určuje provolaný čas za den, kterému sloupec náleží. Po najetí myši na jeden ze sloupců, se změní jeho barva a zobrazí se přesný provolaný čas. Časová hodnota je pomocí vytvořené funkce převedena ze sekund do formátu HH:MM:SS.



Obr. 3.14: Flashový graf pro výpis hovorů za určitý měsíc

3.5 Aplikace Servlet

Tato aplikace zpracovává požadavky z aplikace klienta a zajišťuje komunikaci s MySQL databází. Spojení s MySQL zajišťuje JDBC ovladač MySQL Connector/J. Tento ovladač poskytuje Java aplikacím přístup do relačních databází. Podrobnější informace o JDBC se nachází v [19].

Pro přístup do databáze jsou použity dvě důležité metody:

- executeUpdate(String sql) – pro vkládání záznamů do databázové tabulky pomocí příkazů: INSERT INTO (vložení záznamu), UPDATE (aktualizace záznamu), DELETE (mazání záznamu)
- executeQuery(String sql) – pro čtení záznamů z databázové tabulky pomocí příkazu SELECT

Návrh databáze pro celou aplikaci je zobrazen na UML diagramu (obr. 3.4) v kapitole 3.3.5. Tato kapitola obsahuje také popis jednotlivých položek vytvořené databáze.

3.6 Aplikace klienta

Pomocí balíčku Jigloo pro tvorbu GUI doinstalovaného do SDS Ericsson byla vytvořena grafická reprezentace aplikace pro klienta. Okno aplikace (viz obr. 3.15) tvoří několik částí:

- Buddy list
- SIP URI
- Nick
- Information
- Tlačítka: Call, Hang Up, Add Buddy, Del Buddy a Exit

Buddy list je seznam uživatelů přidaných do klientského seznamu v databázi. Pomocí tlačítka *Add Buddy* lze do seznamu přidat dalšího uživatele. Naopak tlačítkem *Del Buddy* je uživatel ze seznamu odebrán. Seznam se ukládá do textového souboru *buddylist.txt* v tagovém formátu. Problematika tagů již byla popsána v kapitole 3.4.3. Význam tagů je vysvětlen v tabulce 3.1.

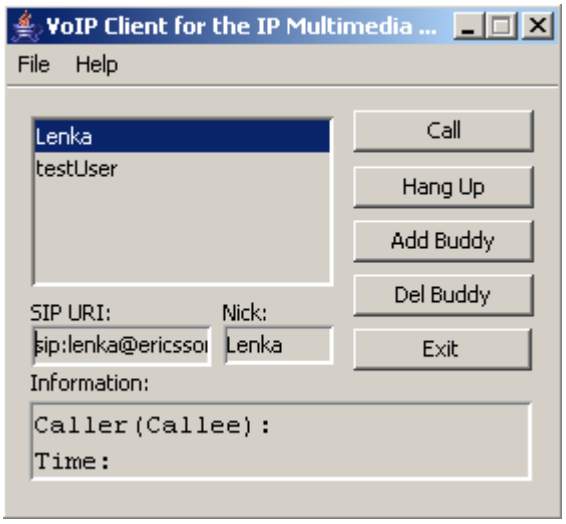
Tab. 3.1: Význam tagů textového souboru *buddylist.txt*

Tag	**Význam**
<buddies>	začátek dokumentu
<user>	definování uživatele
<nick>	přezdívka
<sip>	SIP - identifikátor

V poli SIP URI je zobrazena URI uživatele. Pole Nick obsahuje přezdívku uživatele a pole Information má dvě položky:

- Caller (Callee) – volaný účastník (účastnice)
- Time – provolaný čas za daný hovor

Tlačítko Call zajišťuje volání zvoleného uživatele a tlačítko Hang Up ukončí hovor mezi účastníky. Posledním tlačítkem aplikace je tlačítko Exit, které celou aplikaci ukončí.



Obr. 3.15:Hlavní okno aplikace klienta

Při vytvoření hlasové služby byly využity funkce definované v ICP balíčkách. Informace k balíčkům je možné najít v dokumentaci dostupné po naistalování prostředí SDS; přístupné z nápovědy - ICP API JavaDoc. Jedná se hlavně o rozhraní IVoipCall.

3.7 Zhodnocení projektu

Projekt se skládá z několika částí. První část programovanou v jazyce Java tvoří aplikace klienta a aplikace Servlet. Druhou část projektu tvoří webová aplikace, vytvořená pomocí jazyka PHP (začleněného do HTML a CSS) a jazyka ActionScritpt. Obě části pak sdílí databázi MySQL.

Při vývoji byl testován také klient OpenIC lite a následně byl srovnán s vytvořeným klientem. Výsledky srovnání jsou zobrazeny v tabulce 3.2.

Tab. 3.2: Srovnání klientských aplikací

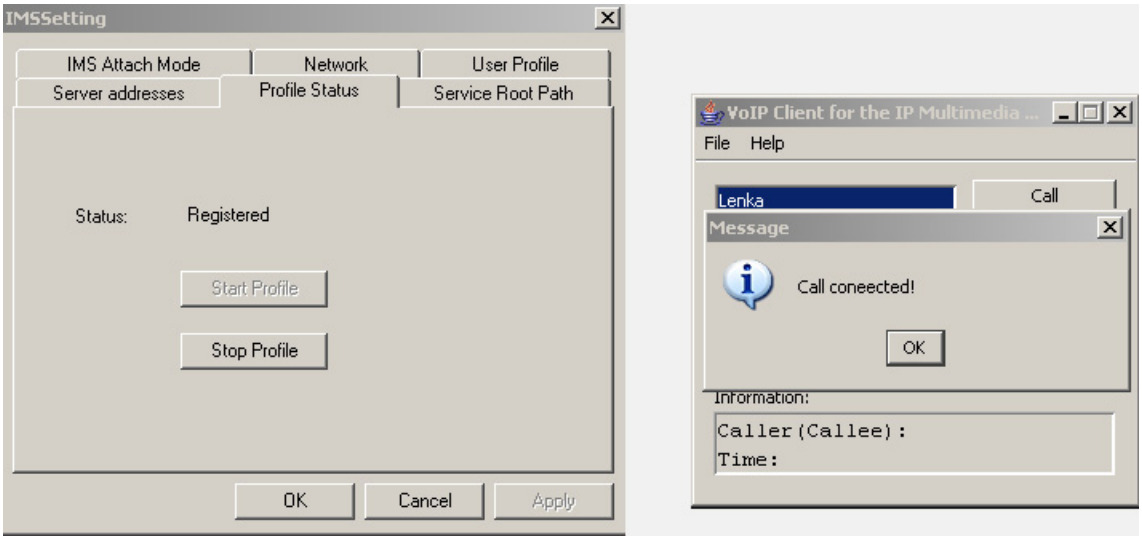
Vlastnosti	**Vytvořený klient**	**OpenIC lite**
Hlasová služba	Ano	Ano
Možnost posílání zpráv	Ne	Ano
Uložení seznamu klientů	Ano (lokální uložení)	Ano (lokální uložení)
Zasílání informací o hovoru	Ano	Ne
Cena aplikace	Free	Free
Nutnost ICP platformy	Ano	Ne

Značnou nevýhodou vytvořené aplikace je nutnost mít v operačním systému nainstalovanou ICP platformu, která je potřebná pro vývoj SDS aplikací v SDS Sony Ericsson.

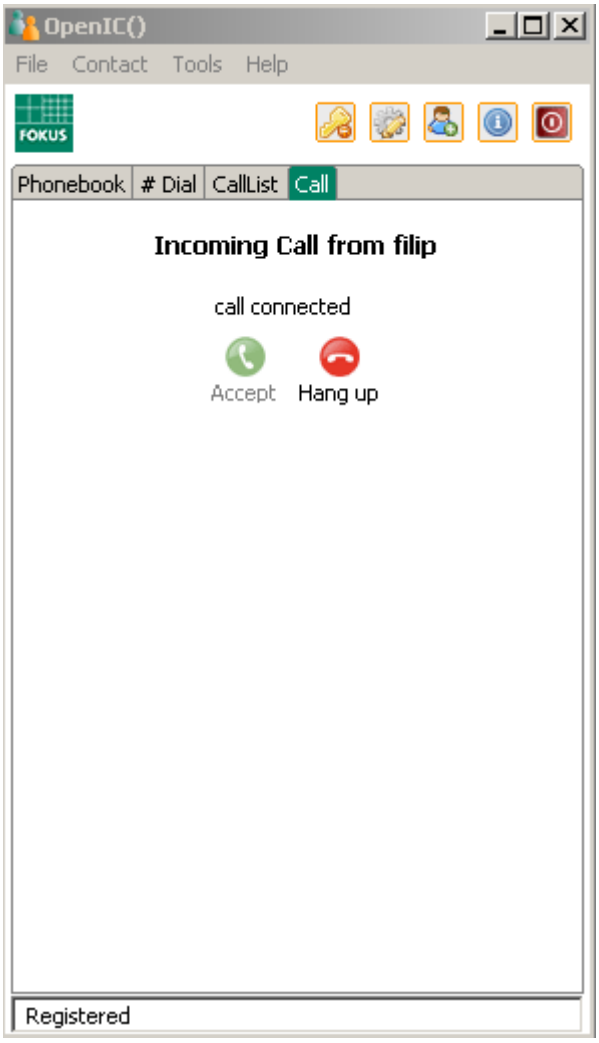
Výhodou aplikace je zasílání informací o hovorech, které je možné zobrazit přes webové rozhraní.

Při testování uživatelského rozhraní navrhovaného klienta a klienta OpenIC lite uživateli (byly vybrány 18 lidí pro testování) byl navrhovaný klient vyhodnocený více jako userfriendly aplikace než OpenIC lite. Tento názor je však individuální u každého jedince, a proto tato vlastnost nebyla zahrnuta v tabulce 3.2.

Následující obrázky znázorňují navázání spojení mezi klienty. Informace ve vytvořeném klientu se zobrazí po odkliknutí okna *Message*.



Obr. 3.16: Navázání spojení mezi klienty; vpravo vytvořená klientská aplikace



Obr. 3.16: Navázání spojení mezi klienty; testovaný klient OpenIC lite

4 Závěr

Tato práce se věnovala problematice IMS (IP-Multimedia Subsystem) a vývoji real-time aplikace. IMS je standardizovaná architektura sítí nové generace, která nabízí pokročilé služby v mobilních i pevných sítích. Jedná se o architekturu orientovanou na služby, jejímž cílem je poskytovat jak existující služby, tak i budoucí internetové služby. IMS nabízí telekomunikačním operátorům možnost vybudovat otevřenou IP infrastrukturu orientovanou na služby, která umožní snadné nasazení nových multimediálních služeb zahrnující telekomunikační a zároveň datové služby.

První kapitola popisuje IMS architekturu a zmiňuje podporované služby. Architekturu IMS lze rozdělit do čtyř vrstev. Nejnižší vrstvou je vrstva koncových zařízení, následuje transportní a kontrolní vrstva a nejvyšší vrstvou je aplikační vrstva. V této kapitole jsou také popsány prvky IMS architektury. Za zmínku stojí prvky CSCF (Call Session Control Function) a HSS (Home Subscribe Server). CSCF slouží jako centralizovaný prvek směrování, správce bezpečnostních pravidel a jako bod, který prosazuje bezpečnostní pravidla pro zajištění doručení různých real-time aplikací založených na IP přenosu. HSS je hlavní úložiště pro všechny informace uživatelů a data IMS. Obsahuje např. data o uživatelské identitě, registrační informace, přístupové parametry a informace o spuštění služby. Poslední problematikou této kapitoly je popis jednotlivých IMS rozhraní.

Druhá kapitola se zabývá IMS protokoly. Převážná část kapitoly se zaměřuje na protokol SIP. Je to signalizační protokol aplikační vrstvy typu klient-server. Zajišťuje sestavení, dohled a ukončení multimediálního spojení (například VoIP). SIP podporuje jak dvoucestný telefonní hovor, tak i multimediální konference. Dalšími zahrnutými protokoly jsou UDP, RTP a TLS.

Třetí kapitola byla věnována praktické části této práce. Na základě získaných poznatků o IMS architektuře, byla vytvořena java aplikace podporující přenos hlasu přes IP sítě. Hlavní části aplikace tvoří klientská část, databáze MySQL a Servlet aplikace určená pro komunikaci databáze s klientskou částí. Jako úložiště pro aplikaci byla zvolena databáze MySQL, protože je na rozdíl od PostgreSQL zdarma. Celá aplikace byla vyvíjena v prostředí SDS Sony Ericsson 4.1. FD1. Při vývoji aplikace byl testován také klient OpenIC lite a následně byl srovnán s vytvořeným klientem. Značnou nevýhodou vytvořené aplikace je nutnost mít v operačním systému nainstalovanou ICP platformu, která je potřebná pro vývoj SDS aplikací v SDS Sony

Ericsson 4.1. FD1. Naopak výhodou vytvořené aplikace je zasílání informací o hovorech, které je možné zobrazit přes webové rozhraní.

Webové rozhraní bylo vytvořeno pomocí programovacích jazyků PHP a ActionScript2 (Flash). Programovací jazyk PHP byl zvolen, protože je to open source software, je zdarma a umožňuje snadné programování webových aplikací. Flash byl použit, protože podporuje vektorovou grafiku a umí animovat objekty. Vytváří tak pro uživatele atraktivnější webové prostředí. Webová aplikace administrátorovi umožňuje spravovat data jednotlivých uživatelů a uživatelům hlasové služby poskytuje přehled o užívání služby. Pro názorné zobrazení provolaného času slouží uživateli vytvořený flashový graf, který se nachází ve výpisu jeho hovorů.

Některé částečné výsledky byly publikované ve sborníku EEICT 2011.

LITERATURA

[1] *SIP Center* [online]. 2011 [cit. 2010-10-20]. SIP Center. Dostupné z WWW: <www.sipcenter.com>.

[2] CHAFFIN, Larry. *Building a VoIP Network with Nortel* [online]. USA : SYNGRESS, June 2006 [cit. 2010-10-22]. Dostupné z WWW: <http://media.techtarget.com/searchVoIP/downloads/Building_a_VoIP_Network_Ch[1]._8.pdf>.

[3] RODRIGUEZ, Erik. *TCP vs. UDP* [online]. [s.l.] : [s.n.], 200? [cit. 2010-10-05]. Dostupné z WWW: <http://www.scribd.com/doc/6644453/TCP-vs-UDP>.

[4] POSTEL, J. *User Datagram Protocol* [online]. [s.l.] : [s.n.], 28 August 1980 [cit. 2010-10-05]. Dostupné z WWW: <http://www.faqs.org/rfcs/rfc768.html>.

[5] TONCAR, Vladimír. About the Real Time Protocol. In *Voice over IP Overview* [online]. [s.l.] : [s.n.], 2010-10-7 [cit. 2011-05-02]. Dostupné z WWW: <http://toncar.cz/Tutorials/VoIP/index.html>.

[6] SCHULZRINNE, H., et al. *RTP: A Transport Protocol for Real-Time Applications* [online]. [s.l.] : [s.n.], July 2003 [cit. 2010-10-22]. Dostupné z WWW: <http://tools.ietf.org/html/rfc3550>.

[7] DIERKS, T.; ALLEN, C. *The TLS Protocol* [online]. [s.l.] : [s.n.], January 1999 [cit. 2010-10-23]. Dostupné z WWW: <http://www.ietf.org/rfc/rfc2246.txt>.

[8] HANDLEY, M., *SDP: Session Description Protocol* [online]. [s.l.] : [s.n.], July 2006 [cit. 2010-10-23]. Dostupné z WWW: <http://tools.ietf.org/html/rfc4566>.

[9] HENTZEN, Whil. *DNS Explained* [online]. [s.l.] : [s.n.], 200? [cit. 2010-10-24]. Dostupné z WWW: <http://www.hentzenwerke.com/wp/dns_explained.pdf>.

[10] *Metaswitch Networks* [online]. 200? [cit. 2010-11-01]. IMS Architecture. Dostupné z WWW: <http://www.metaswitch.com/sbc-session-border-controller/ims-architecture.aspx>.

[11] *IBM* [online]. 2006 [cit. 2010-11-05]. Introduction to IP Multimedia Subsystem. Dostupné z WWW: <http://www.ibm.com/developerworks/webservices/library/ws-soa-ipmultisub1/>.

[12] RUSSELL, Travis. *The IP Multimedia Subsystem (IMS): Session Control and Other Network Operations*. V. Británie: Mc Graw-Hill OSBOURNE, 2008. 242 s. ISBN 0071488537.

[13] A. AHSON, Syed; ILYAS, Mohammad. *IP Multimedia Subsystem Handbook*. USA : Taylor & Francis Group, 2009. 562 s. ISBN 978-1-4200-6459-9.

[14] POIKSELKA, Miikka, MAYER, Gregor, KHARTABIL, Hisham, NIEMI, Aki. *The IMS: IP Multimedia Concepts and Services in the Mobile Domain*. England: WILEY, 2004. 448 s. ISBN 0-470-87113-X.

[15] *FLASHJPW* [online]. 2011 [cit. 2011-02-23]. Příručka pro Flash a ActionScript. Dostupné z WWW: <http://flash.jakpsatweb.cz/>.

[16] *Flash.cz* [online]. 2011 [cit. 2011-02-25]. Práce s XML. Dostupné z WWW: <http://www.flash.cz/ >.

[17] COLE, Timothy W.; HABING, Thomas G. A Brief Overview of XML. In *A Brief Overview of XML* [online]. University of Illinois at Urbana-Champaign : Grainger Engineering Library Information Center, 199? [cit. 2011-04-23]. Dostupné z WWW: <http://dli.grainger.uiuc.edu/Publications/TWCole/AALL_2000/PDF/Overview.pdf>.

[18] *Albireo* [online]. 2010 [cit. 2011-04-23]. WampServer instalace. Dostupné z WWW: <http://www.albireo.eu/wamp-server-instalace>.

[19] REESE, G. *Database programming with JDBC and JAVA*. USA: O'Reilly. 2000. 348 s. ISBN: 1-56592-616-1.

Seznam zkratek

AAA	Access, Authorization and Accouting
AS	Application Server
BGCF	Breakout Gateway Control Function
CGF	Charging Gateway Function
CSCF	Call Session Control Function
DSL	Digital Subscribe Line
GPRS	General Packet Radio Service
HLR	Home Location Register
HSS	Home Subscribe Server
HTTP	Hyper Text Transfer Protocol
I-CSCF	Interrogating Call Session Control Function
ICP	IMS Client Platform
IANA	Internet Assigned Numbers Authority
IMS	Internet Protocol Multimedia Subsystem
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control address
MGCF	Multimedia Gateway Control Function
MRF	Media Resource Function
MRFC	Multimedia Resource Function Control
MRFP	Multimedia Resource Function Processor
P-CSCF	Proxy Call Session Control Function
PHP	Hypertext Preprocessor
PDA	Personal Digital Assistant
PSTN	The Public Switched Telephone Network
QoS	Quality of Service
RIA	Rich Internet Application
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLF	Subscription Locator Function
SSRC	Synchronization Source Identifier

TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol UML
URI	Uniform Resource Identifier
VoIP	Voice over IP
WCDMA	Wideband Code Division Multiple Access
WiFi	Wireless Local Area Networks Technology
XML	eXtensible Markup Language