VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

VoIP klient pro operační systém Android

VoIP Client for Android Operating System

2014

Daniel Boháč

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Daniel Boháč**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: VoIP klient pro operační systém Android
VoIP Client for Android Operating System

Zásady pro vypracování:

1. Navrhněte a vytvořte VoIP klienta pro operační systém Android.
2. Klient bude vytvořen na základě již existujících klientů s přidanou podporou pro DTMF volbu během video hovoru, bude taktéž přidána podpora pro textovou komunikaci.
3. Klient bude podporovat audio,video přenosy a textovou komunikaci.
4. Vytvořená aplikace bude podporovat SIP, H.263, H.264, DTMF, SOAP, zprávy přes SIP protokol a XMPP protokol.
5. Student vytvoří dokumentaci k vytvořené aplikaci.

Seznam doporučené odborné literatury:

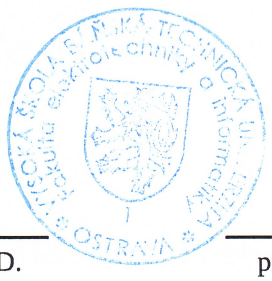
M. Voznak, Voice over IP. VSB-Technical University of Ostrava: College book, 1st. ed., Ostrava, 2008.

Meggelen, J., P., Smith, J., Madsen, L.: Asterisk: The Future of Telephony. O'Reilly Media

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Kapičák**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014



doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Velmi rád bych touto cestou poděkoval vedoucímu své bakalářské práce, panu Ing. Lukášovi Kapičákovi, za odbornou pomoc, konzultace a cenné rady při vedení této práce.

Abstrakt

V současnosti dochází k vysoké penetraci platformy Android na trhu s chytrými telefony. Vzhledem k popularitě této platformy, její jednoduchosti, otevřenosti a rychlosti psaní kódu se Android stává ideální platformou pro vývoj klienta podporujícího technologii VoIP, která tvoří alternativu ke klasické telefonii. Cílem praktické části této bakalářské práce je vytvořit VoIP klienta pro operační systém Android podporujícího audio hovory, video hovory, textovou komunikaci a DTMF. Výhodou tohoto VoIP klienta ve srovnání s konkurencí je především podpora IM komunikace prostřednitvím protokolu XMPP a možnost využití integrovaného internetového prohlížeče v průběhu hovoru. Teoretická část této bakalářské práce se zabývá všeobecnými principy VoIP, platformou Android, nástroji pro vývoj a jednotlivými technologiemi využitými v praktické části této práci.

Klíčová slova: VoIP, SIP, RTP, RTCP, Android, Asterisk, kodek, XMPP, QoS, NAT, bezpečnost, DTMF

Abstract

Nowadays, there is a high penetration of Android in the market of smartphones. Due to the popularity of the platform, its simplicity, openness and fast code writing, Android is an ideal platform for developing client that supports VoIP technology, which forms an alternative to the classical telephony. The practical part of this thesis is about developing a VoIP client for the Android operating system that supports audio calls, video calls, text messaging, and DTMF. The advantage of VoIP client over the competition is about support IM communication using XMPP protocol and the ability to use the integrated web browser during a call. The theoretical part of this thesis deals with the general principles of VoIP, Android platform, development tools and various technologies utilized in the practical part for the thesis.

Keywords: VoIP, SIP, RTP, RTCP, Android, Asterisk, codec, XMPP, QoS, NAT, security, DTMF

Seznam použitých zkratek a symbolů

AAPT	–	Android Asset Packaging Tool
ACK	–	Acknowledgement
ADB	–	Android Debug Bridge
ADT	–	Android Development Tools
AES	–	Advanced Encryption Standard
API	–	Application Programming Interface
ART	–	Android RunTime
BSD	–	Berkeley Software Distribution
CD	–	Compact Disc
DDMS	–	Dalvik Debug Monitor Server
DLCI	–	Data Link Connection Identifier
DNS	–	Domain Name System
DTLS	–	Datagram Transport Layer Security
DTMF	–	Dual-Tone Multi-Frequency
DVM	–	Dalvik Virtual Machine
DX	–	Dalvik Cross-Assembler
GNU	–	Gnu's Not Unix
GPL	–	General Public License
GPS	–	Global Positioning System
GUI	–	Graphical User Interface
HD	–	High-Definition
HTTP	–	HyperText Transfer Protocol
HW	–	hardware
IAX	–	Inter-Asterisk eXchange
ICE	–	Interactive Connectivity Establishment
IDE	–	Integrated Development Environment
IETF	–	Internet Engineering Task Force
IM	–	Instant Messaging
IMEI	–	International Mobile Equipment Identity
IP	–	Internet Protocol
IPC	–	Inter-Process Communication
IRC	–	Internet Relay Chat
ISDN	–	Integrated Services Digital Network

ITU	–	International Telecommunication Union
IVR	–	Interactive Voice Response
JID	–	Jabber ID
MGCP	–	Media Gateway Control Protocol
MOS	–	Mean Opinion Score
MSRP	–	Message Session Relay Protocol
NAT	–	Network Address Translation
NAT-T	–	NAT Traversal
NB	–	narrowband
OHA	–	Open Handset Alliance
OS	–	operační systém
OSI	–	Open Systems Interconnection
PBX	–	Private Branch eXchange
PVC	–	Permanent Virtual Circuit
QoE	–	Quality of Experience
QoS	–	Quality of Service
RFC	–	Request For Comments
RSVP	–	Resource Reservation Protocol
RTCP	–	RTP Control Protocol
RTP	–	Real-time Transport Protocol
SAP	–	Session Announcement Protocol
SD	–	Secure Digital
SDK	–	Software Development Kit
SDP	–	Session Description Protocol
SHA	–	Secure Hash Algorithm
SIMPLE	–	Session initiation protocol for Instant Messaging and Presence Leveraging Extensions
SIP	–	Session Initiation Protocol
SMS	–	Short Message Service
SOAP	–	Simple Object Access Protocol
SQL	–	Structured Query Language
SRTP	–	Secure Real-time Transport Protocol
SS7	–	Signalizační Systém č. 7
SSL	–	Secure Sockets Layer
STUN	–	Session Traversal Utilities for NAT
SVC	–	Switched Virtual Circuit
SW	–	software
TCP	–	Transmission Control Protocol
TLS	–	Transport Layer Security

TURN	–	Traversal Using Relays around NAT
UA	–	User Agent
UAC	–	User Agent Client
UAS	–	User Agent Server
UDP	–	User Datagram Protocol
URI	–	Uniform Resource Identifier
URN	–	Uniform Resource Name
UWB	–	ultra-wideband
VC	–	Virtual Circuit
VoIP	–	Voice over Internet Protocol
VPN	–	Virtual Private Network
WB	–	wideband
XML	–	Extensible Markup Language
XMPP	–	Extensible Messaging and Presence Protocol
ZRTP	–	Zimmermann Real-time Transport Protocol

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

S rozvojem bezdrátových sítí a mobilních služeb se VoIP (Voice over Internet Protocol) komunikace stala nedílným prvkem mobilních telefonů, které jsou běžnou součástí našeho života. Velké popularitě se v současnosti těší tzv. „chytré telefony“, založené na rozsáhlé open source platformě Android.

Pro svou popularitu ze strany vývojářů i běžných uživatelů se Android stal nejoblíbnější platformou dnešní doby a v době psaní této práce dominuje trhu s dotykovými zařízeními. Obrovská popularita této platformy je také důvodem, proč jsem si ji zvolil pro vývoj VoIP klienta v rámci této bakalářské práce. Popularita, jednoduchost a rychlý vývoj aplikací pro Android zapříčinil také rozmach aplikací včetně těch využívajících technologii VoIP, která tvoří alternativu ke klasické telefonii. Z těchto důvodů jsou v následující kapitole popsány obecné principy telefonních sítí a rozdíly mezi VoIP a klasickou telefonií. Signalizaci v mnou vytvořené aplikaci zajišťuje protokol SIP (Session Initiation Protocol), který je dnes značně rozšířen a v podstatě nahradil starší protokol H.323, který je příliš robustní a složitý. O signalizačních protokolech, komunikačních protokolech a technlogoii VoIP pojednává kapitola č. 3.

Značnou výhodou vytvořené aplikace je podpora přenosu zpráv, proto v kapitole č. 4 pojednávám o podporovaném protokolu XMPP (Extensible Messaging and Presence Protocol) umožňujícím IM (Instant Messaging) komunikaci a rozšíření SIMPLE (Session initiation protocol for Instant Messaging and Presence Leveraging Extensions) protokolu SIP. Podmínkou spolehlivé VoIP komunikace je mimo jiné zajištění dostatečné kvality hovoru, který musí probíhat v reálném čase. K tomuto účelu je využíván mechanismus QoS (Quality of Service), který zajišťuje kvalitu služeb prostřednictvím přidělování přenosové kapacity podle priorit datového provozu. O mechanismu QoS a dalších problémech či omezeních, mezi které spadá např. bezpečnost a NAT (Network Address Translation), pojednává kapitola č. 5.

Šestou kapitolu tvoří tónová volba neboli DTMF (Dual-Tone Multi-Frequency). Jedná se o způsob kódování a přenosu informací, který funguje na principu vysílání tónu, složeného ze dvou sinusových signálů o určité frekvenci vhodné pro přenos telekomunikačními telefonními linkami. DTMF slouží např. k řízení hlasových komunikací, komunikaci mezi radiostanicemi, nebo konfiguraci ústředen. Vzhledem k tomu, že cílem mé bakalářské práce je vytvořit VoIP klienta pro již dříve zmíněnou platformu Android, analyzoval jsem tuto platformu v kapitole č. 7.

Za účelem úspory objemu přenášených dat ve VoIP sítích podléhají data před odesláním kompresi, kterou zajišťují kódovací a dekódovací algoritmy zvané kodeky, proto jsem tyto algoritmy popsal v kapitole č. 8 následovanou kapitolou zabývající se vývojem aplikace pro platformu Android. Jako stručný úvod poskytující informace o jednotlivých prvcích vytvořeného VoIP klienta pak slouží kapitola č. 10, kde se nachází popis individuálních prvků, ze kterých se vytvořená aplikace skládá. Podrobnější popis architektury aplikace formou programátorské dokumentace je obsažen v příloze A. Poslední kapitola popisuje testování správné funkčnosti aplikace.

2 Obecné principy přenosu dat v telefonních sítích

V této kapitole je pojednáno o vlastnostech jednotlivých druhů sítí, jejich principech, výhodách a nevýhodách.

2.1 Sítě založené na principu přepojování okruhů

Jedná se o technologii využívanou především v tradičních telefonních sítích a pro přenosy dat přes vytáčené spojení. Technologie přepojování okruhů je velice stará, její původ sahá do počátků telegrafie a telefonie vůbec [1].

Sítě založené na principu přepojování okruhů jsou také označovány jako „komutované". V případě potřeby přenosu dat je zde vyhrazen okruh od odesílatele až k příjemci. Tento okruh je vyhrazen po celou dobu přenosu. Každá ústředna na cestě k příjemci musí pro tyto data (hovor) vyhradit potřebné prostředky. Jestliže toho některá z ústředen není schopna, je odeslán signál obsazenosti linky. Všechna odesílaná data zde putují po stejné cestě, díky čemuž jsou doručena ve správném pořadí. Vzhledem k tomu, že je navázaná cesta k dispozici po celou dobu komunikace, až do ukončení spojení, je zde nevýhoda ve formě existence spojení v době, kdy se žádná data nepřenáší. Za účelem sdílení kapacity systému se provádí multiplexování. Přepojování okruhů lze podle způsobu multiplexování rozdělit na:

- Přepojování časové – Využití časového multiplexu.
- Přepojování prostorové – Přepojování pomocí elektromechanických prvků ústředen.

2.2 Sítě založené na principu přepojování paketů

Tento typ sítí se stal základem moderního Internetu. Data jsou přenášena po menších částech v tzv. paketech, které v sobě kromě samotných přenášených dat nesou i hlavičku obsahující dodatečené informace, mezi které patří např. cílová a zdrojová IP (Internet Protocol) adresa, nebo sekvenční číslo. Při přenosu zde neexistuje sestavený okruh mezi zdrojovou a cílovou stanicí. Jednotlivé pakety jsou sítí směrovány zvlášť, tzn. každý z nich může putovat individuálně odlišnou cestou, která se může dynamicky měnit. Doručení, správnost pořadí a zamezení duplicity paketů obvykle zajišťuje cílová stanice, nikoli síť.

U těchto sítí není v okamžiku odeslání paketu známa cesta, kterou bude paket putovat. Paket postupně prochází směrovači, které vybírají nejvhodnější cestu k dalšímu bodu sítě podle směrovacích tabulek a momentálního stavu linky. Mezi parametry této volby patří např. propustnost, zpoždění nebo spolehlivost linky. Komunikační cesta při tomto procesu není žádným způsobem rezervována. Na straně příjemce je pak přenášená informace získána sestavením přijatých paketů. Sítě založené na přepojování paketů mohou být i spojové, a to prostřednictvím mechanismu virtuálních okruhů (VC). Slovo virtuální vychází z důvodu neexistence skutečného fyzického spojení mezi koncovými body. Jednotlivé virtuální okruhy jsou identifikovány identifikátorem DLCI (Data Link Connection Identifier). Díky technologii VC je vždy nejprve vytvořen virtuální okruh

pomocí signalizace a až následně mohou být skrz VC odesílány samotné pakety. Přenášená data nenesou informace identifikující cílové zařízení, ale pouze identifikaci VC, který propojuje koncové body. Existují dva typy virtálních okruhů:

- Permanentní virtuální okruhy (PVC) – Okruhy, které jsou pevně dané a existují nezávisle na existenci přenášených dat.
- Komutované virtuální okruhy (SVC) – Vytvářejí se dynamicky na základě okamžitých potřeb komunikace a po ukončení přenosu automaticky zanikají.

2.3 Srovnání paketových a komutovaných sítí

Výhodou paketových sítí je vysoká efektivita využití šířky pásma a levnější technologie, která tento přenos umožňuje. Navíc tato technologie nevyžaduje, aby komunikující zařízení měla stejnou přenosovou rychlost nebo využívala stejný způsob přenosu (synchronní, arytmický). Mezi další výhody lze zařadit krátké úseky dat, které pakety nesou a poskytují prostřednitvím nižší chybovosti spolehlivější a kvalitnější přenos. Nevýhodou je kolísání kvality hovoru při nezajištění dostatečné kvality služeb a vyšší režie z důvodu potřeby přenosu nadbytečných dat v hlavičkách paketů [2].

Mezi výhody komutovaných sítí patří stálá kvalita hovoru. Naopak nevýhodou této technologie je obvykle velice nízká efektivita využití pásma.

3 VoIP

Tato technologie umožňuje přenos digitalizovaného hlasu ve formě paketů prostřednictvím počítačové sítě nebo jiného datového spojení založeného na IP protokolu, který je základním protokolem třetí vrstvy OSI (Open Systems Interconnection) modelu a definuje způsob směrování paketů mezí sítěmi. VoIP tak tvoří alternativu ke klasické telefonii, založené na použití sítí s přepojováním okruhů přes veřejnou telefonní síť. K dosažení minimálního objemu dat při přenosu se využívají kódovací a dekódovací algoritmy, nazývané kodeky. Kodeky jsou standardizovány a často také patentovány. Za účelem dosažení dobré kvality hovoru se pak využívají služby QoS a samozřejmostí je i potřeba zajištění dostatečné datové propustnosti připojení, jeho nízká latence, jitter a chybovost.

Pro přenos hovorových dat se ve VoIP na transportní vrstvě OSI modelu využívá především protokol UDP (User Datagram Protocol), jelikož je vhodnější pro přenos dat v reálném čase, který je pro VoIP charakteristický. Možný je také přenos prostřednictvím protokolu TCP (Transmission Control Protocol). Na páté vrstvě OSI modelu se využívá protokol RTP (Real-time Transport Protocol), který slouží k přenosu multimediálních dat a obvykle nese 20–30 ms fragmenty hovoru. Rodin VoIP protokolů existuje celá řada. Mezi ty nejznámější patří SIP a starší H.323, mezi ty méně známé můžeme zařadit IAX (Inter-Asterisk eXchange), který byl vyvinut společností Asterisk nebo například protokol Skinny společnosti Cisco Systems. VoIP protokoly lze rozdělit do dvou skupin [4]:

- Signalizační – Tyto protokoly slouží k inicializaci spojení, řízení toku a ukončení spojení. Mezi nejznámější z nich patří výše zmíněný SIP a H.323.
- Komunikační – Protokoly sloužící k vlastnímu přenosu multimediálních dat v reálném čase (RTP, RTCP).

3.1 Signalizační protokoly

3.1.1 Protokol SIP

Jedná se o signalizační protokol, který se využívá v IP telefonii k zahájení, modifikaci a ukončení hovorů, je specifikován v RFC (Request For Comments) 3261 [6]. SIP je textově orientovaný a podobný protokolu HTTP (HyperText Transfer Protocol), který je určen pro přenos hypertextových dokumentů. Byl vyvinut skupinou IETF (Internet Engineering Task Force), jejíž cílem je vývoj a podpora internetových standardů. Je velmi otevřený a flexibilní, proto také jeho popularita roste a prakticky nahradil starší standard H.323. Pro popis vlastností relace se ve spojení s protokolem SIP obvykle využívá SDP (Session Description Protocol) a pro přenos samotného audio či video záznamu protokol RTP [3].

V hlavičkách protokolu SIP jsou umístěny položky To, From a Subject, podobně jako u e-mailové komunikace prostřednictvím SMTP protokolu. SIP obvykle využívá služeb transportního protokolu UDP na portu 5060, ale je možné využít i služeb TCP či TLS (Transport Layer Security) [3].

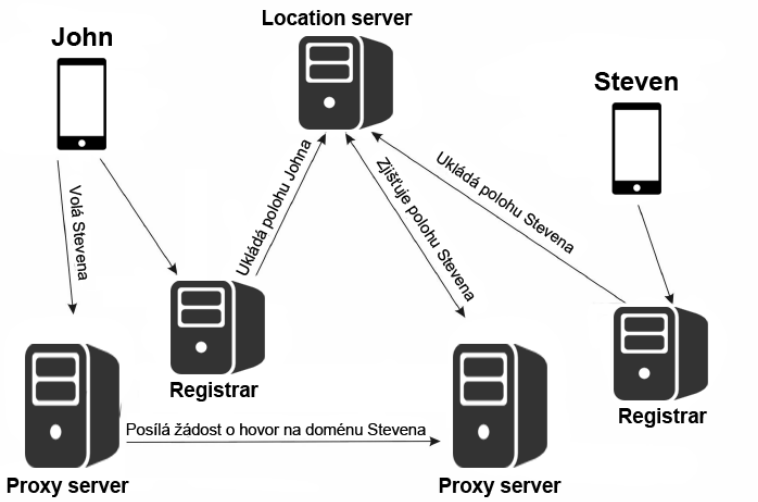
SIP byl vyvinut tak, aby byl co nejjednodušší. Jeho koncept je založen na relacích, jež mohou být zveřejňovány protokolem SAP (Session Announcement Protocol). Inteligence, která slouží ke směrování hovorů je rozmístěna v jednotilivých zařízeních v síti, která jsou nazývána UA (User Agent). Zařízení typu UA existují dva typy:

- Klienti (UAC) – Inicializují spojení zprávou INVITE.
- Servery (UAS) – Zpracovávají a reagují na zprávu INVITE.

Mezi typické UA zařízení můžeme zařadit např. IP telefony. V případě, že tyto telefony hovory přijímají, představují UAS, pokud naopak hovor iniciují, jedná se o UAC.

Architektura sítě, využívající protokol SIP, je znázorněna na obrázku č. 3.1. V sítích založených na tomto protokolu se vyskytují následující servery:

- Registrar – Registrace klientů.
- Redirect – Přesměrování UA na jiný server.
- Proxy – Směrování signalizace pro UA klienty.
- Location – Překlad adres pro redirect a proxy servery.



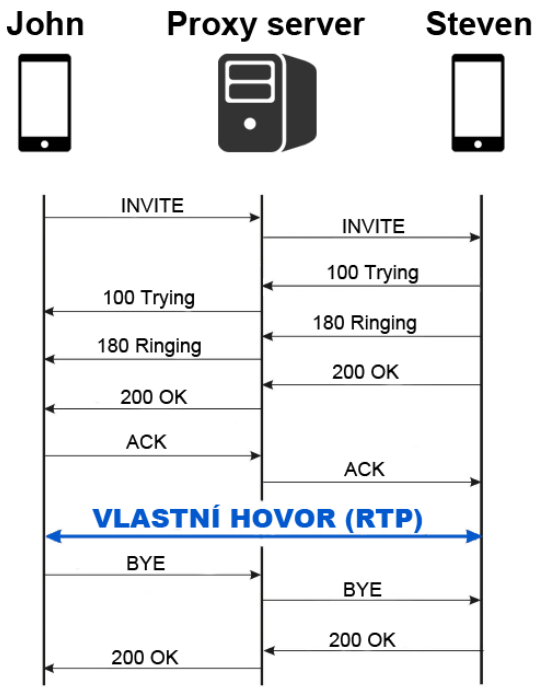
Obrázek 3.1: Příklad sítě s využitím protokolu SIP

Pro adresaci SIP využívá URI (Uniform Resource Identifier) začínající prefixem „sip:“. Obecný tvar tohoto identifikátoru má tvar `sip:user@host`.

Inicializace hovoru prostřednictvím protokolu SIP začíná zprávou INVITE UA klienta. Pokud se chce cílový UAS připojit k relaci, odešle odpověď. Následně UAC, který spojení inicioval, odešle serveru potvrzení (ACK). Nyní se mohou začít přenášet datové proudy mezi hlasovými bránami pomocí RTP [5].

V případě, že bude v síti existovat server redirect, bude UAC odesílat zprávu INVITE právě redirect serveru, který se na cestu k cíli dotáže location serveru. Následně redirect server odesílá UAC informace o dalším postupu (přesměrování). V okamžiku, kdy UAC zjistí adresu cílového UAS, může dojít ke vzniku přímého spojení [5].

Obdobně bude komunikace fungovat při použití proxy serveru, který umožňuje centralizované řízení komunikace a její správu. UAC odešle INVITE zprávu proxy serveru, ten se na cestu k cíli dotáže location serveru. Proxy server slouží jako prostředník pro výměnu informací mezi UAC a UAS. UAS komunikuje s proxy serverem a vyměňuje si s ním parametry hovoru, které jsou následně odeslány proxy serverem klientovi. Ten následně odešle ACK (Acknowledgement) UAS prostřednictvím proxy serveru. Po této procedůře může být zahájen přenos datového proudu RTP. Průběh komunikace protokolu SIP je vyobrazen na obrázku č. 3.2 [5].



Obrázek 3.2: Průběh komunikace protokolu SIP

Aby mohlo dojít ke správnému vytvoření a řízení relace, musí SIP zajistit:

- lokalizaci účastníka hovoru,
- navázání spojení,
- řízení spojení,
- zjištění stavu účastníka,
- zjištění možností účastníka (přenosová rychlost, kodeky…).

Standard RFC 3261 specifikuje celkem šest metod pro komunikaci [6]:

- INVITE – Žádost o změnu parametrů nebo inicializaci spojení.
- CANCEL – Zrušení nesestaveného spojení.
- REGISTER – Žádost k registraci, případně odregistrování
- OPTIONS – Metoda sloužící ke zjištění vlastností určité SIP entity.
- ACK – Potvrzení přijetí odpovědi na žádost INVITE.
- BYE – Slouží k ukončení sestaveného spojení.

Metody definované v jiných RFC:

- NOTIFY – Slouží k informování o události - RFC 3265 [7].
- INFO – Zajišťuje přenos informací v průběhu relace - RFC 2976 [8].
- SUBSCRIBE – Přihlášení k upozorňování na výskyt událostí - RFC 3265 [7].
- PRACK – Potvrzování dočasných odpovědí - RFC 3262 [9].
- UPDATE – Aktualizace stavu určité relace – RFC 3331 [10].
- MESSAGE – Umožňuje přenos zpráv - RFC 3428 [11].
- REFER – Připouští řízení spojení třetí stranou - RFC 3515 [12].
- PUBLISH – Aktualizace prezenčního stavu - RFC 3903 [13].

Typy odpovědí:

- 1xx – informační odpovědi (100 Trying, 180 Ringing),
- 2xx – indikace úspěchu (200 OK),
- 3xx - přesměrování (301 Moved Permanently),
- 4xx – chyba na straně klienta (401 Unauthorized, 486 Busy Here),
- 5xx – chyba na straně serveru (501 Not Implemented),
- 6xx – fatální chyba (600 Busy Everywhere).

3.2 Komunikační protokoly

3.2.1 Protokol RTP

Protokol RTP zajišťuje přenos multimediálních dat v reálném čase. Byl vyvinut organizací IETF a poprvé publikován v roce 1996 jako standard RFC 1889 [24]. Využívá se především v systémech tzv. proudového přenosu. Pro přenos dat RTP využívá především UDP protokol transportní vrstvy. Ve svém záhlaví přenáší informace o typu přenášených dat, zdroj synchonizace, sekvenční číslo, časovou značku a způsob kódování. Pomocí sekvenčního čísla a časových razítek jsou na přijímací straně pakety uspořádány do správného pořadí.

3.2.2 Protokol RTCP (RTP Control Protocol)

Jedná se o řídící protokol spolupracující s RTP. Slouží k řízení RTP relace, zajišťuje odpovídající kvalitu služeb a synchronizaci audio a video přenosů. Je definován v RFC 3550 [25]. Využívá periodického vysílání paketů od každého účastníka RTP relace všem ostatním účastníkům. Toto periodické vysílání slouží k účelům diagnostiky a řízení výkonnosti. Mezi činnosti, které RTCP zajišťuje, patří:

- identifikace zdroje RTP,
- poskytování informací o kvalitě vysílaných dat,
- řízení intervalu vysílání RTCP,
- přenos informace o řízení relace.

RTCP standardně využívá port o jedno číslo vyšší než RTP a shromažďuje informace o mediálním spojení, mezi které patří především jitter, latence, počet ztracených a odeslaných paketů. Podle těchto dat může aplikace zvýšit kvalitu služeb např. použitím jiného kodeku. Protokolem RTCP jsou definovány následující typy zpráv:

- Sender Report (informace o odesílateli) – Jsou pravidelně vysílány aktivními účastníky. Obsahují přijímací statistiky a informace o probíhající komunikaci.
- Receiver Report (informace o příjemci) – Informují odesílatele a ostatní příjemce o problémech a kvalitě služeb.
- Application-Specific Message – Umožňuje definici nových zpráv, které nejsou definované ve standardu.
- Source Description Message – Pravidelně odesílané zprávy nesoucí informace o účastníkovi.
- Goodbye Message – Slouží k ukončení streamu.

4 Protokoly umožňující textovou komunikaci

4.1 XMPP (Extensible Messaging and Presence Protocol)

XMPP je protokol, založený na obecném značkovacím jazyce XML (Extensible Markup Language), původně byl vyvíjen Jabber open source komunitou [36]. Nyní má jeho vývoj na starosti organizace XMPP Standarts Foundation. Hlavním cílem XMPP je IM komunikace a zjišťování stavu. Síť, která využívá protokol XMPP není centralizovaná, jako u většiny jiných IM služeb, nýbrž distibuovaná. Každý uživatel je v této síti identifikován jeho identifikačním jménem a názvem serveru, tyto hodnoty jsou spojeny znakem @ a nazývají se Jabber ID, zkráceně JID. Identifikátor JID může, až na několik výjimek, obsahovat libovolné unicode znaky. Na jeden uživatelský účet může být přihlášeno více uživatelů, mezi kterými se rozhoduje podle priority, nebo celé adresy, která může nabývat formátu např. user@example/home nebo user@example/anywhere.

Každý uživatel se připojuje na server, na kterém byl zaregistrován. V případě potřeby komunikovat s uživateli na jiném serveru, si tyto servery mezi sebou vymění potřebné informace. XMPP síť je založena na klient-server architektuře, kdokoli může do této sítě připojit svůj vlastní server a komunikovat s uživateli na ostatních serverech. Protokol XMPP byl navržen jako rozšiřitelný a je ho možné použít také např. pro signalizaci ve VoIP sítích, přenosy souborů a videa, nebo jako víceuživatelský chat podobný IRC (Internet Relay Chat) [36]. Navíc je XMPP otevřeným standardem a kdokoli může implementovat jeho služby pod jakoukoli licencí.

Výhody XMPP protokolu [36]:

- flexibilita,
- bezpečnost,
- decentralizace,
- otevřenost,
- rozšířenost.

4.2 SIMPLE

SIMPLE je sada protokolů určená pro IM komunikaci a prezenci. Tento otevřený standard rozšiřuje SIP protokol o možnosti výměny informací o prezenci uživatelů a přenos zpráv v reálném čase. Jedná se o často používanou alternativu umožňující IM komunikaci, která je využívána především ve VoIP aplikacích využívajících signalizační protokol SIP, na němž je SIMPLE založen.

4.2.1 IM komunikace

Protokol SIP definuje dva režimy IM komunikace [37]:

- session Mode (MSRP protokol),
- page mode (SIP metoda typu MESSAGE).

4.2.2 Prezence

Prezenci specifikovanou protokolem SIMPLE můžeme rozdělit na tři části [37]:

- Jádro protokolu – Poskytuje rozšíření SIP protokolu o metody NOTIFY A SUBSCRIBE.
- Soukromí, politika, zjišťování – Definuje, kdo může být informován o prezenci jednotlivých uživatelů a v jaké míře.
- Prezenční dokumenty – Informace o prezenci je zakódována v XML dokumentech, které jsou přenášeny v těle SIP zpráv.

5 Problémy a omezení komunikace

5.1 QoS

Jedním z problémů technologie VoIP je potřeba zajištění dostatečné kvality služeb. Proto vznikla technologie QoS umožňující v počítačových a telekomunikačních sítích rezervaci přenosové kapacity a řízení datových toků. QoS se využívá výhradně v sítích založených na principu přepojování paketů, kde kompenzuje rozdíly mezi komutovanými a paketovými sítěmi prostřednictvím garance kvality služeb, kterou zhoršují vlivy, mezi které patří:

- jitter,
- ztrátovost paketů,
- zpoždění,
- nedostatečná šířka pásma.

Ve veřejných telefonních sítích dosahuje zpoždění hodnot typicky 50 až 90 milisekund, ve VoIP tato hodnota roste až k hodnotám několika stovek ms. Podle specifikace ITU (International Telecommunication Union) G.114 je akceptovatelné zpoždění do 150 ms, případně 300 ms pro mezinárodní hovory [14, 21]. Ztrátovost paketů nad 5% je rovněž nepřípustná [14, 22].

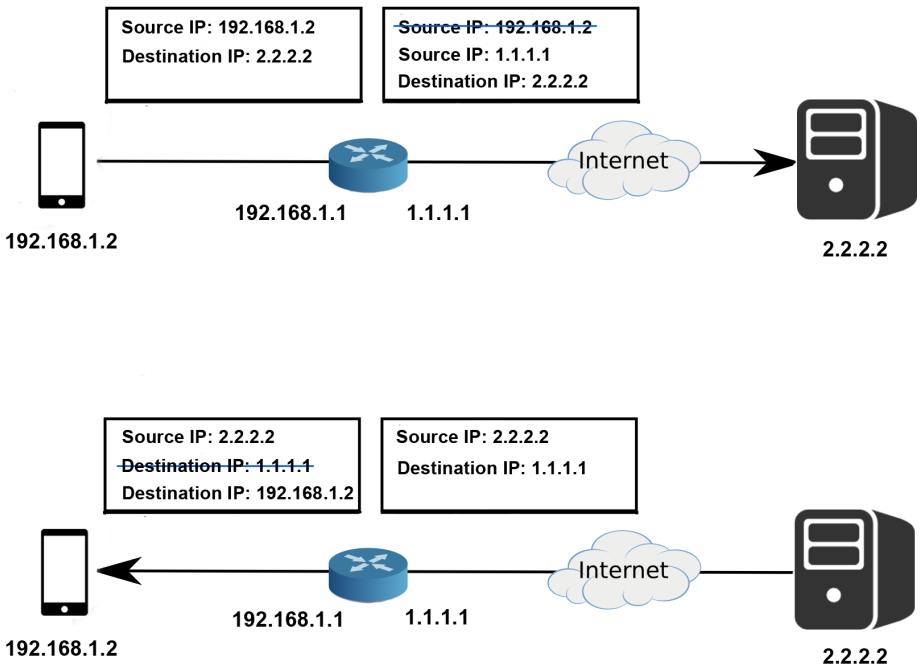
Metody QoS [38]:

- Differentiated services – Do hlaviček paketů se zapisují informace o kategoriích, které představují prioritu dat.
- Integrated services – Implementace parametrizovaného přístupu, kdy aplikace využívají RSVP (Resource Reservation Protocol) protokol k vyžádání a rezervaci zdrojů v síti.
- Best-effort services – Každý paket se snaží dostat k cíli za co nejkratší dobu.

5.2 NAT

Překlad síťových adres neboli NAT je metoda umožňující přepis IP adresy, případně také portu průchozích paketů. Využívá se z důvodu nedostatku veřejných IP adres. Pomocí NAT mohou stanice s privátní adresou komunikovat s vnější sítí prostřednictvím vypůjčené veřejné adresy. Provoz zvenčí působí, jakoby pocházel z NAT serveru, proto lze říci, že je NAT server i jednoduchým proxy serverem. Princip funkce NAT je zobrazen na obrázku č. 5.1.

Kromě jeho základní funkce NAT také napomáhá vytvoření bezpečnější sítě, z důvodu nemožnosti zařízení z vnější sítě iniciovat spojení se zařízením ze sítě nacházejícím se za NATem, pokud neexistuje patřičný záznam v tabulce překladu. Toto řešení sice



Obrázek 5.1: Princip NAT

zvyšuje bezpečnost stanic před útokem z vnější sítě, ale také zamezuje přístupu ke stanicím v případě potřeby. Existují ale způsoby, jakými lze tento problém obejít. Mezi možná řešení patří port forwarding a NAT Traversal (NAT-T).

V mé aplikaci používaný SIP spadá mezi protokoly, kterým činní NAT problémy. SIP společně s ostatními daty odesílá také síťové adresy a čísla portů na aplikační úrovni. V případě, kdy je iniciátor datového přenosu situován za NATem, dojde z důvodu překladu adres k navrácení nesprávné informace, která způsobí neúspěch při samotné signalizaci, nebo při doručení RTP dat.

Pro umožnění komunikace skrz NAT existuje řada řešení, mezi které spadají techniky ICE (Interactive Connectivity Establishment), TURN (Traversal Using Relays around NAT), nebo STUN (Session Traversal Utilities for NAT), které umožňují zjištění typu NAT, veřejné IP adresy a portu. Všechny tyto techniky jsou podporovány mou aplikací. Dalším možným řešením je využití ALG (Application Layer Gateway), která realizuje proxy službu pro SIP a RTP. Posledním řešením je možnost některých SIP klientů definovat externí IP adresu pro použití v hlavičkách protokolu SIP. Signalizace od klienta pak odchází z privátní sítě s hlavičkami s veřejnou adresou SIP proxy serveru.

5.2.1 Rozdílné typy NAT podle STUN

- Full-cone NAT – Všechny požadavky ze stejné vnitřní IP adresy a portu jsou mapovány na stejnou externí IP adresu a port. Externí host může komunikovat s lokálním počítačem za NATem zasláním paketu na mapovanou externí adresu [23].
- Address-restricted-cone NAT – Narozdíl od Full-cone NAT může vnější host zaslat paket lokálnímu, pouze pokud mu dříve zaslal lokální počítač paket [23].
- Port-restricted cone NAT – Funguje obdobně jako Address-restricted-cone NAT, ale omezení zahrnuje čísla portů. Vnější host může zaslat paket na konkrétní port lokálnímu zažízení jen pokud lokální počítač předtím zaslal z tohoto portu paket vnějšímu zařízení [23].
- Symmetric NAT – Veškeré požadavky ze stejné lokální IP adresy a portu na určitou IP adresu a port jsou mapovány na unikátní externí zdrojovou IP adresu a port. V případě odeslání paketu stejným lokálním zařízením se stejnou zdrojovou adresou a portem jinému cíli, je použito odlišné mapování. Pouze vnější zařízení, které obdrží paket, může poslat paket zpět lokálnímu zařízení, přičemž musí odpovídat adresa i port [23].

5.3 Bezpečnost ve VoIP

Vzhledem k roustoucímu počtu uživatelů využívajících služeb VoIP, vznikly také požadavky na zajištění bezpečnosti hovorů. Z těchto důvodů byly vytvořeny protokoly zabezpečující VoIP komunikaci. Mezi nejčastěji používané bezpečnostní protokoly ve VoIP sítích patří SRTP (Secure Real-time Transport Protocol), ZRTP (Zimmermann Real-time Transport Protocol), TLS a DTLS (Datagram Transport Layer Security). Mnou vytvořená aplikace podporuje všechny zmíněné protokoly, vyjma ZRTP.

5.3.1 Protokol SRTP

Jedná se o protokol poskytující šifrování, autentizaci a integritu přenášených RTP zpráv. Byl vyvinut kryptografickými odborníky společnosti Cisco Systems ve spolupráci se společností Ericsson a publikován organizací IETF v roce 2004 pod specifikací RFC 3711 [27]. Jako výchozí je zde využívána symetrická šifra AES (Advanced Encryption Standard) a k ověřování integrity dat je použit algoritmus HMAC-SHA1 produkující 160 bitový tag, který je následně zkrácen na 80 nebo 32 bitů. Tag je připojen ke každému paketu, důsledkem čehož může být ověřena datová integrita. K výpočtu HMAC se využívá informace o užitečném zatížení paketu a dat z hlavičky těchto paketů.

Samotný protokol SRTP nedokáže zajistit výměnu klíčů, ale využívá protokol MIKEY, zapouzdřený v SDP protokolu. Pokud není samotná SIP signalizase šifrována, je možné výměnu klíčů zachytit, proto je doporučováno využití protokolu umožňujícího šifrování.

5.3.2 Protokol TLS

Dalším podporovaným kryptografickým protokolem je TLS. Jedná se o nástupce protokolu SSL (Secure Sockets Layer) umožňujícího bezpečnou komunikaci v síti. TLS je inicializován na páté vrstvě OSI modelu, kde proběhne tzv. handshake prostřednictvím asymetrické šifry, následně pracuje na vrstvě šesté, kde šifruje komunikaci symetrickou šifrou. TLS zajišťuje šifrování a autentizaci uživatele vůči serveru i proces vzájemné autentizace mezi uživateli systému. Při použití druhé varianty musí být zajištěna bezpečná distribuce klíčů za pomoci certifikační autority. Jako ochrana proti útokům Man-in-the-Middle porovnává klient aktuální DNS (Domain Name System) jméno serveru se jménem certifikátu. Také rozděluje vstupní data na poloviny a na každou z nich aplikuje jiný hashovací algoritmus (SHA-1, MD5), výsledek je sloučen funkcí XOR. Protokol TLS může být také použit k tunelování síťových protokolů a vytvoření VPN (Virtual Private Network).

5.3.3 Protokol DTLS

Standard organizace IETF, který byl specifikován v RFC 4347 [26] v dubnu 2006. DTLS je založen na protokolu TLS a upraven pro spolupráci s protokolem UDP. Je využit v aplikacích tolerujících nízkou ztrátovost paketů, ale vyžadujících nízké zpoždění. Mezi tyto aplikace lze zařadit VoIP.

6 Tónová volba

Tónová volba neboli také DTMF či kmitočtová volba je způsob kódování a přenosu informací využívaný moderními ústřednami, který funguje na principu vysílání tónu složeného ze dvou sinusových signálů o určité frekvenci. Tyto frekvence jsou vyobrazeny v tabulce č. 6.1 a jsou voleny tak, aby je bylo možné přenášet telekomunikačními telefonními linkami, které jsou určeny pro frekvence 300–3400 Hz. DTMF slouží k řízení hlasových komunikací, vytáčení, konfiguraci ústředen a komunikaci mezi radiostanicemi, ale na rozdíl od svého předchůdce (pulzní volba) umožňuje také přenos dat. Standard DTMF přenosu je 50 ms trvání tónu a 50 ms bez tónu (10 tónů za 1 s). Tuto rychlost lze zvýšit, ale při kratších intervalech může dojít k chybnému vyhodnocení. Počet tónů je celkem 16, ale u běžných telefonních přístrojů se nejčastěji setkáme s klávesnicí obsahující 12 kláves — klávesy A–D jsou vynechány a slouží zejména k programování ústředen a jiným speciálním funkcím [20].

6.1 Tvorba DTMF signálu

Každý řádek a sloupec na klávesnici představuje určitou frekvenci. Při stisku klávesy se vygeneruje tón složený z tónu sloupce a tónu řádku, z toho plyne, že jeden DTMF tón tvoří vždy dvě frekvence, tyto frekvence nepodléhají interferenci. Průběhy obou výstupních signálů musí být sinusového charakteru a také by měly mít stejnou amplitudu. Možná odchylka frekvencí je maximálně $\pm 1,5\%$ od nominální hodnoty. Tón o vyšší frekvenci z dané dvojice může mít vyšší hodnotu amplitudy maximálně o 4 dB [20].

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Tabulka 6.1: Kódování tónové volby

6.2 Speciální tónové frekvence

Pro speciální účely jako je indikace stavu linky, zařízení nebo pro oznámení výsledků operací jsou definovány speciální cyklické tóny. Tyto tóny jsou standardizovány pro každou zemi, ale ve většině evropských zemích se využívá jednofrekvenční systém využívající kmitočet 425 Hz. Mezi tyto tóny patří např. napojovací tón, oznamovací tón, vyzváněcí tón, obsazovací tón a odkazovací tón.

7 Android

Android není pouze operační systém, ale rozsáhlá softwarová platforma, která byla navržena především pro mobilní, dotyková zařízení, jako jsou např. tzv. chytré telefony, tablety nebo navigace. Zahrnuje v sobě operační systém, uživatelské rozhraní, aplikace a middleware. I přesto, že společnost Google tento systém prezentuje jako open source, existují části, které jsou uzavřenou technologií této společnosti. Původně byla tato platforma vyvíjena společností Android Inc., kterou v roce 2005 odkoupila společnost Google. Od roku 2007 Android spadá pod konsorcium OHA (Open Handset Alliance), jehož cílem je urychlení vývoje a inovací mobilních zařízení.

Popularita platformy Android neustále roste a to především díky její otevřenosti, nezávislosti na použitém hardware, provázanosti s Google službami, přívětivému vzhledu a dostupnosti aplikací. Ve třetím čtvrtletí minulého roku ovládala 82% trhu s chytrými telefony a v online distribuční službě Google Play Store bylo v říjnu 2012 k dispozici 700 000 aplikací ke stažení [15, 16].

Minimální verze OS (operačního systému) Android, podporovaná mnou vyvinutou aplikací, je verze 3.0 (Honeycomb). Tato verze mimo jiné přináší lepší optimalizaci pro tablety a zařízení s velkým displejem. Hlavním důvodem podpory této verze je nové uživatelské rozhraní a jeho prvky, mezi které spadají fragmenty využívané v mé aplikaci. Fragmenty zvyšují interaktivitu a flexibilitu aplikace a také umožňují řízení přechodů mezi obrazovkami včetně tvorby vícepanelových aplikací, mezi které má aplikace spadá.

7.1 Architektura operačního systému Android

Operační systém Android lze podle účelu rozdělit celkem do pěti úrovní, tvořících jeho architekturu. Ta je představena na obrázku č. 7.1 a tvoří ji následující vrstvy:

- Linux Kernel (jádro OS),
- nativní knihovny,
- Android runtime ,
- aplikační framework,
- aplikace.

7.1.1 Linux Kernel

Jádro operačního systému bylo původně postaveno na Linuxovém jádře verze 2.6.24 [17] a tvoří nejnižší vrstvu jeho architektury, která zajišťuje např. správu paměti, práci se sítí, správu procesů a také obsahuje zabudované ovladače zařízení.

7.1.2 Nativní knihovny

Druhou vrstvu tvoří nativní knihovny psané v jazycích C a C++ (pro vyšší výkonnost), ale volány jsou přes rozhraní jazyka Java. Příkladem těchto knihoven jsou:

- SQLite – Odlehčený relační databázový systém.
- WebKit – Nástroj pro renderování webových stránek.
- Libc – Standardní knihovna libc odvozená z BSD (Berkeley Software Distribution) upravená pro embedded zařízení.
- OpenSSL – Knihovna pro práci s protokolem SSL sloužícího k zabezpečení komunikace.
- Knihovny pro práci s 2D a 3D grafikou.

Vývojáři mohou funkce těchto knihoven využívat prostřednictvím vrstvy Android Application Framework.

7.1.3 Android runtime

7.1.3.1 DVM (Dalvik Virtual Machine) Vrstva Android Runtime zahrnuje virtuální stroj Dalvik (DVM). Ten byl vytvořen tak, aby na jednom zařízení mohlo být efektivně spuštěno velké množství instancí virtuálního stroje a zároveň je optimalizován pro potřeby mobilních zařízení. Je určen ke zpracování souborů v Dalvik Executable (dex) formátu, který vznikne konverzí .class souboru pomocí nástroje DX (Dalvik Cross-Assembler). Dalvik je také optimalizován k minimální náročnosti na paměť, nekomprimovaný soubor formátu dex má obvykle o pár procent menší velikost, než komprimovaný soubor formátu jar. Každá aplikace, je v systému Android spuštěna v individuálním procesu se svou vlastní instancí virtuálního stroje [17].

Vzhledem k tomu, že je právě DVM přisuzována náročnost operačního systému Android, která se projevuje především velkou náročností na paměť a pomalý chod aplikací na méně výkonných zařízeních, rozhodla se společnost Google vytvořit jeho náhradu. Tato náhrada byla pojmenována ART (Android RunTime) a slibuje vyšší výkon i prodloužení výdrže na baterii. ART byl poprvé představen s vydáním OS Android verze 4.4 (KitKat) , kde byl intergrován k získání zpětné vazby uživatelů a především developerů, ale zatím je považován za nestabilní.

7.1.3.2 Core Libraries Také nazývány Dalvik Libraries, implementují API (Application Programming Interface) pro všeobecné účely psaní kódu v programovacím jazyce Java. Můžeme je rozdělit do dvou kategorií:

- Knihovny pro interoperabilitu s programovacím jazykem Java – Umožňují psaní kódu v dobře známém prostředí programovacího jazyka Java. Je to podmnožina standardních Java core knihoven, které byly přizpůsobeny pro použití aplikacemi využívajícími DVM. Nachází se zde běžné třídy sloužící k práci se soubory, řetězci, reflexí, šifrováním aj.
- Specifické knihovny DVM – Tato sada knihoven umužňuje interakci s instancí virtuálního stroje.

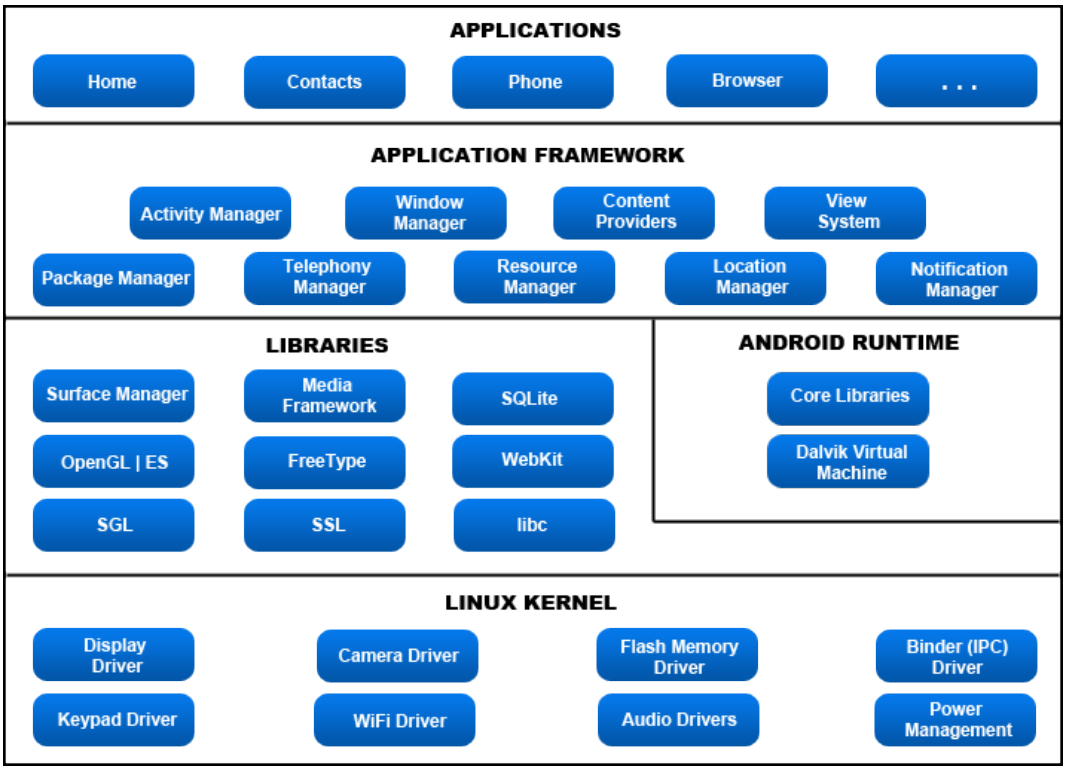
7.1.4 Aplikační Framework

Tato vrstva poskytuje vývojářům služby na vyšší úrovni abstrakce a značně tím ulehčuje práci např. s prvky uživatelského rozhraní, strukturovanými daty, notifikacemi nebo senzory. Mezi třídy tohoto frameworku mimo jiné patří:

- Activity Manager – Slouží k interakci s aktivitami a správě jejich životního cyklu.
- Telephony Manager – Poskytuje informace o telefonních službách a jejich stavech.
- Resource Manager – Podpora přístupu ke zdrojům dané aplikace na vyšší úrovni abstrakce.
- Content Provider – Umožňuje přístup ke strukturované množině dat jiných aplikací.
- Notification Manager – Slouží k zobrazení a editování upozornění ve stavové liště.
- Location Manager – Poskytuje přístup k lokalizačním službám, které zpřístupňují informace o poloze zařízení např. pomocí GPS (Global Positioning System).
- Package Manager – Umožňuje získat data vztahující se k aplikačním balíčkům nainstalovaným na zařízení.

7.1.5 Aplikace

Aplikace tvoří nejvyšší vrstvu architektury operačního systému Android a představují ji aplikace, které využívají jednotliví uživatelé. Patří zde např. webový prohlížeč, správce souborů, aplikace pro práci s fotoaparátem, psaní zpráv a mnoho dalších.



Obrázek 7.1: Architektura OS Android

8 Kodeky

Tímto termínem jsou označovány algoritmy, umožňující transformovat datový proud, nebo signál, určený k přenosu. Původně bylo tohle označení odvozeno z počátečních slabik slov „**ko**dér“ a „**dek**odér“. Kodeky existují ve formě SW i HW a jejich cílem je komprese dat. Na použitém kodeku závisí velikost dat, jejich výsledná kvalita i potřebná šířka pásma ke spolehlivému přenosu hovorových dat. Kodeky lze rozdělit na:

- Kodeky ztrátové – Data po dekódování nejsou identická s daty původními, jinými slovy dochází ke ztrátě informace. Ztrátové kodeky těží z nedokonalosti lidských smyslů, proto ztráta dat nemůže být člověkem zaznamenána, nebo způsobí pouze minimální postřehnutelný rozdíl.
- Kodeky neztrátové – Při kompresi tohoto typu kodeků nedochází ke ztrátě informací a data po dekompresi jsou ekvivalentní s těmi původními. Tyto kodeky se využívají tam, kde je potřeba pracovat s daty v původní kvalitě. Nevýhodou je nižší kompresní poměr v porovnání s kodeky ztrátovými.

Kodeky jsou jedním z faktorů, které ovlivňují výslednou kvalitu hovoru. Pro ohodnocení kvality hovoru byla zavedena pětistupňová stupnice MOS (Mean Opinion Score), kterou můžete vidět v tabulce č. 8.1. Hodnotu kvality hovoru lze stanovit jednou ze tří metod:

- subjektivní (hodnocení posluchačů),
- objektivní (srovnání odeslaných vzorků s přijatými),
- výpočetní model (E-model).

MOS	Kvalita	Zhoršení kvality
5	Vynikající	Neznatelné
4	Dobrá	Znatelné, ale ne nepříjmené
3	Akceptovatelná	Mírně nepříjemné
2	Špatná	Nepříjemné
1	Velmi špatná	Velmi nepříjemné

Tabulka 8.1: Stupnice MOS

8.1 Kodeky podporované aplikací

8.1.0.1 Podporované audio kodeky:

- G.711 (PCMA, PCMU),
- GSM,
- iLBC,
- Speex (Speex-NB, Speex-WB, Speex-UWB),
- OPUS,
- G.722,
- G.729.

8.1.0.2 Podporované video kodeky

- VP8,
- MP4V-ES,
- Theora,
- H.264/MPEG-4 AVC (H264-BP, H264-MP),
- H.263,
- H.263+,
- H.263++.

9 Vývoj aplikace

Dnes existují dvě hlavní vývojové prostředí pro platformu Android. Prvním z nich je Eclipse IDE (Integrated Development Environment), pro které je dostupný plugin Android development tools ulehčující vývoj pro tuto platformu. Programátor není nucen využívat vývojové prostředí Eclipse, existují i jiné alternativy. Aplikace pro Android lze psát i v obyčejném textovém editoru a následně zkompilovat pomocí příkazové řádky, vhodnější variantou je využít jiné vývojové prostředí, jako např. Android Studio.

Vývojové prostředí Android Studio je založeno na IDE IntelliJ IDEA a poskytuje integrovaný plugin ADT (Android Development Tools) pro vývoj na platformu Android. Ačkoli má toto vývojové prostředí velký potenciál do budoucna, použil jsem pro vývoj aplikace v této bakalářské práci Eclipse IDE, vzhledem k tomu, že je ověřeno širokou vývojářskou komunitou a osobně s ním mám několikaleté zkušenosti. Naproti tomu Android Studio je v době psaní této bakalářské práce v alfa verzi, a proto obsahuje množství chyb, které mohou vývoj značně ztížit.

9.1 Android SDK (Software Development Kit)

Nástroje a API sloužící k vývoji aplikací pro platformu Android jsou zahrnuty v Android SDK. Mezi tyto nástroje a prvky patří:

- knihovny tříd,
- developerské nástroje,
  - DX – slouží ke konverzi java.class souborů na dex soubory,
  - AAPT (Android Asset Packaging Tool) – převádí aplikaci do .apk souboru,
  - ADB (Android Debug Bridge) – umožňuje komunikaci s emulátorem nebo Android zařízením přes příkazovou řádku,
  - DDMS (Dalvik Debug Monitor Server) – perspektiva umožňující interakci s Android zařízením.
- emulátor a systémové obrazy,
- dokumentace a ukázkové kódy.

9.2 Základní stavební kameny aplikací pro Android

Každá aplikace vytvořená pro platformu Android se skládá z různých komponent. Mezi ty nejpoužívanější z nich patří:

- Activity,
- Intent,
- Service,
- Broadcast Receiver,
- Content Provider,
- Notification.

Tyto komponenty musí být definovýny v souboru AndroidManifest.xml, který je obsažen v kořenovém adresáři projektu a udává jeho strukturu.

9.2.1 Activity

Aktivita představuje základní stavební blok uživatelského rozhraní. V podstatě se jedná o synonymum pro jednu obrazovku aplikace. Každá aktivita dědí ze třídy Activity a obsahuje veškeré grafické rozhraní umožňující interakci aplikace s uživatelem. Grafické rozhraní aktvity je definováno layoutem, který slouží ke specifikaci rozmístění jednotlivých grafických komponent, jako jsou tlačítka, seznamy nebo obrázky.

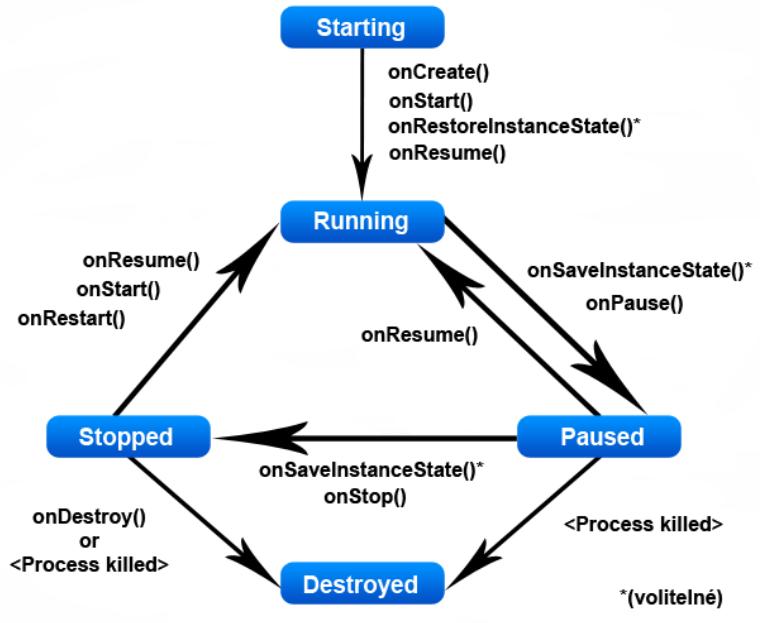
Vzhledem k tomu, že většina aplikací se neskládá pouze z jedné aktivity, byl vytvořen mechanismus umožňující jejich přechody. Hlavní roli v tomto mechanismu hraje třída Intent neboli v překladu záměr. Typické volání jiné aktivity se pak provádí prostřednictvím funkce startActivity(Intent), případně startActivityForResult(Intent, requestCode), pokud je potřeba, aby volaná aktivita vrátila libovolný výsledek po jejím ukončení. Životní cyklus aktivity si můžete prohlédnout na obrázku č. 9.1 [28].

9.2.2 Intent

Jedná se o pasivní datovou strukturu, která udržuje abstraktní popis operace, která se má provést a data k této operaci potřebná. Využívá se k odesílání asynchronních zpáv mezi aktivitami, službami (services) nebo jinými aplikačními komponentami. Ve své podstatě umožňuje pozdní vazbu, tzn. dynamický výběr komponenty (po spuštění), která operaci provede [31].

Záměry lze rozdělit na:

- Implicitní – Specifikuje akci, která se má provést, ale neinformuje systém o tom, která třída tuto operaci zařídí. Výběr správné třídy rozhodne podle dostupných informací systém, případně umožní výběr uživateli prostřednictvím vyskakovacího okna.
- Explicitní – Specifikuje konkrétní komponentu, která danou operaci provede.



Obrázek 9.1: Životní cyklus aktivity

9.2.3 Service

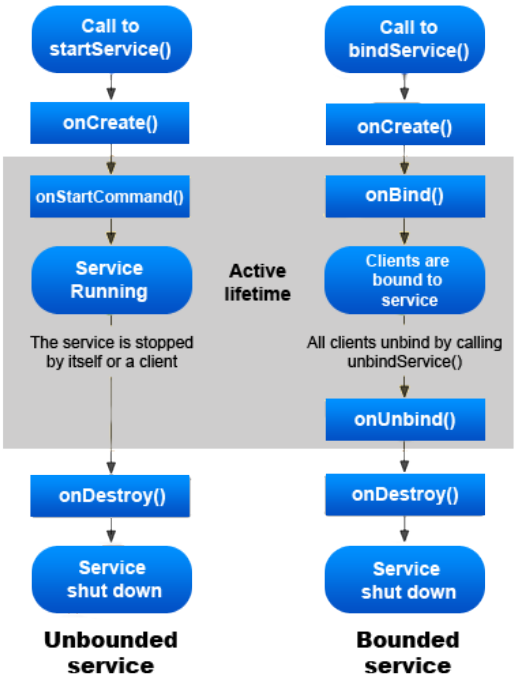
Jedná se o komponentu, která je navržena k vykonávání operací na pozadí. Její životní cyklus představuje obrázek č. 9.2. Mezi běžné operace prováděné touto komponentou patří např. přehrávání hudby, nebo stahování dat. Vzhledem k tomu, že tato komponenta vykonává operace na pozadí, není zde potřeba grafického rozhraní. Service neboli služba, je nezávislá na aktivitách a je implementována jako potomek třídy Service [29].

Službu lze spustit dvěma způsoby:

- Metodou startService – Po spuštění může být služba spuštěna na pozadí po dobu neurčitou i v případě ukončení komponenty, která tuto službu spustila. Obvykle je služba určena k vykonání určité operace a po jejím dokončení by měla být ukončena.
- Metodou bindService – Slouží k vytvoření klient-server architektury, která umožňuje vzájemnou komunikaci, včetně komunikace mezi procesy (IPC). Tento typ služby existuje po dobu existence komponenty, která je s ní svázána.

9.2.4 Broadcast Receiver

Komponenta vykonávající činnost příjemce záměrů (intents) odesílaných jako broadcast zprávy. Slouží k reakci na tyto záměry, umožňuje automatické provádění operací na základě příchozích událostí a tím vytváření událostmi řízené aplikace. V mé aplikaci tuto komponentu využívám např. k vytvoření notifikace při příchodu nové zprávy [32].



Obrázek 9.2: Životní cyklus služby

9.2.5 Content Provider

Vytváří abstrakci pro přístup k množině strukturovaných dat. Mezi množinu těchto dat patří např. obrázky, videa, hudba nebo kontaktní informace. S určitými omezeními jsou tito poskytovatelé obsahu dostupní jakékoli aplikaci a rovněž zajišťují zapouzdření dat a poskytnutí mechanismu definujícího bezpečnost dat [30]. Content provider poskytuje jednoduché rozhraní se standardními metodami, mezi které patří:

- query (nalezení a vrácení dat),
- update (aktualizace dat),
- insert (vložení dat),
- delete (odstranění dat).

9.2.6 Notification

Preferovaná technika získání pozornosti uživatele na vzniklou událost [32]. Je to informace, která není zobrazena ve standardním GUI (Graphical User Interface) aplikace, ale v notifikační oblasti, kde se po vyvolání notifikace objeví ikona reprezentující určitou událost, často doprovázena notifikačním tónem a vibracemi. Notifikace je v aplikaci využita např. k zobrazení informace o aktuálním stavu připojení k SIP serveru, nebo upozornění na přijatou zprávu.

9.3 Výběr vhodného open source klienta

OS Android sice s příchodem verze 2.3 Gingerbread obsahuje zabudované SIP API, ale to neumožňuje video hovory. Proto jsem se rozhodl vycházet z open source projektu. Níže je k dispozici výčet nejznámějších z nich, mezi kterými jsem se rozhodoval:

- Linphone,
- Sipdroid,
- CSipSimple,
- IMSDroid.

Linphone je VoIP klient, který je dostupný pro OS Windows, Linux, OS založených na BSD, Mac OS X, BlackBerry OS, iOS a Android. Využívá SIP stack nazvaný Belle-sip [33]. Nevýhodou tohoto VoIP klienta z hlediska vývojáře je především malá vývojářská komunita a podpora. Druhým kandidátem byl Sipdroid. Jedná se o VoIP klienta určeného výhradně pro OS Android. Jeho značnou nevýhodou je podpora pouze jediného video kodeku, jímž je kodek H.263, proto nesplňuje podmínky specifikované v zadání bakalářské práce. Předposledním a zároveň velice oblíbeným VoIP klientem je CSipSimple. Tento klient nabízí rozsáhlou nabídku podporovaných technologí a služeb, ovšem při jeho testování jsem zjistil závažný nedostatek, jímž jsou video hovory. Tyto hovory jsou zatím pouze experimentální funkcí a při komunikaci s jinými klienty, nebo při použití odlišných rozlišení dochází k chybám v přenosu a zobrazení obrazu.

Vzhledem ke zmíněným nedostatkům předchozích open source klientů, nízké vývojářské podpory a nedostatku nalezených informací jsem se rozhodl vycházet z projektu IMSDroid. Tento klient podporuje velké množství technologií, audio i video kodeků včetně těch požadovaných v zadání bakalářské práce. Při testování rovněž prokázal své kvality a bezproblémovou schopnost komunikace s jinými VoIP klienty. Velkou výhodou je také rozsáhlá vývojářská komunita a podpora.

9.3.1 IMSDroid

Open source VoIP klient založený na Doubango SIP stacku, vydán pod licencí GNU/GPL (Gnu's Not Unix/General Public License). Podporuje Full HD (High-Definition) video, NAT-T technologie STUN, TURN a ICE. Zabezpečení zajišťuje prostřednictvím protokolů SRTP, DTLS a TLS. Také podporuje více hovorů na lince a MSRP (Message Session Relay Protocol) chat [34].

Mezi další vlastnosti a podporované technologie patří [34]:

- SIP,
- IPv4, IPv6,
- DTMF,
- QoS,
- PRACK,
- Přenos souborů pomocí MSRP.

Podporované audio kodeky:

- iLBC,
- GSM,
- PCMA,
- PCMU,
- Speex (NB, WB, UWB),
- OPUS,
- G.722,
- G.729.

Podporované video kodeky:

- H.263, H.263+, H.263++,
- H.264-BP, H.264-MP,
- MP4V-ES,
- VP8,
- Theora.

9.4 Použitá API

9.4.1 aSmack

Jedná se o open source knihovnu určenou pro IM komunikaci založenou na knihovně Smack. Knihovna je napsána v jazyce Java a poskytuje komplexní řešení od přenosu zpráv po zjišťování prezence uživatelů prostřednictvím protokolu XMPP. V aplikaci jsem využil větev (fork) této open source knihovny nazvanou FlowDalic [39] ve verzi s označením 18-0.8.9.

9.4.2 android-ngn-stack (Doubango framework)

Doubango je experimentální, open source framework určený pro desktopové i embedded zařízení. Poskytuje podporu pro VoIP komunikaci a související technologie. Doubango framework byl navržen tak, aby byl přenosný, vysoce efektivní a bez problémů fungoval i na zařízeních s limitovaným výkonem a pamětí [18].

9.4.3 Android-support-v4

Sada podpůrných knihoven, které poskytují zpětnou kompatibilitu pro Android API, které jsou dostupné pouze pro vyšší verze systému. Tyto knihovny umožňují využití nejnovějších prvků při současné kompatibilitně se staršími verzemi Androidu. Hlavní důvod využití těchto knihoven v mém projektu je zpětná kompatibilita z důvodu využití fragmentů (Fragment, FragmentActivity), které jsou dostupné od verze 3.0 Honeycomb. I přes využití těchto knihoven ale aplikace nefunguje na nižších verzích systému, jelikož dochází k výjimce z důvodu nenalezení třídy dědící z FragmentActivity.

9.5 Ukládání dat

Podle charakteru dat jsem v práci využil několik způsobů jejich ukládání:

- Shared Preferences,
- Externí uložiště (External Storage),
- SQLite databázi.

9.5.1 Shared Preferences

Tato metoda je pravděpodobně nejjednodušší. Slouží k perzistentnímu ukládání primitivních datových typů ve tvaru klíč-hodnota. V aplikaci je tato metoda využita k ukládání hodnot nastavení, jako jsou uživatelské jméno, heslo, adresa serveru apod. Nevýhodou této metody je nemožnost ukládání složitějších datových struktur a nepraktičnost pro zápis většího množství dat.

9.5.2 Externí uložiště

Využití externího uložiště (SD karta nebo interní paměť) přináší výhodu ve formě velké kapacity a možnosti ukládání libovolných datových struktur, ale najdeme zde rovněž i nevýhody – data, která zde uložíme, nejsou privátní, a proto bychom zde neměli ukládat citlivé informace. Navíc může kdykoli dojít k vyjmutí paměti a nedostupnosti média. Tato metoda také není vhodná k práci s velkými objemy strukturovaných dat, zde se uplatní využití databáze. Externí uložiště bylo využito např. k ukládání ikony kontaktu, kterou nebylo možno uložit do Shared Preferences, ale zároveň bylo zbytečné vytvářet novou tabulku v databázi kvůli jedné ikoně.

Za zmínku stojí problémy OS Android verze 4.4, ve které může aplikace třetích stran zapisovat data na SD kartu pouze do složky, kterou si aplikace sama vytvoří, čímž Android znemožňuje práci s ostatními soubory a složkami.

9.5.3 SQLite databáze

Tohle řešení umožňuje ukládání strukturovaných dat do privátní databáze. Je to ideální metoda pro potřeby ukládání velkého množství strukturovaných dat a jejich dotazování. SQLite je odlehčená verze relační databáze, která je obsažena v malé knihovně napsané v jazyce C a umožňuje práci s daty pomocí klasického dialektu SQL (Structured Query Language). V projektu je databáze využita pro ukládání kontaktů, zpráv a historie volání.

10 Architektura aplikace

V následujícím textu jsou popsány jednotlivé balíčky a třídy aplikace. Podrobnější informace o třídách a jejich funkcích lze nalézt v dokumentaci programu v příloze A. Architektura grafického rozhraní je vyobrazena v příloze B. Aplikace je rozdělena celkem do sedmi následujících balíčků podle jejich účelu.

10.1 Balíček activities

V tomto balíčku se nachází veškeré třídy, dědící ze třídy Activity. Potomci této třídy reprezentují grafické rozhraní aplikace a umožňují tak interakci uživatele s touto aplikací.

- Třída AVActivity – Aktivita reprezentující obrazovku audio nebo video hovoru, řídící jednotlivé operace probíhajícího hovoru. Mimo standardních informací zobrazených na obrazovce jako je ikona kontaku, jeho název, doba hovoru, tlačítka pro příjmutí nebo odmítnutí hovoru, zde nalezneme i zválštní funkce, mezi které patří DTMF, zobrazení webového prohlížeče přes polovinu obrazovky, funkce pozdržení hovoru, přepínání reproduktoru, nebo možnost řízení přenosu videa.
- Třída AVQueueActivity – Jedná se o aktivitu reprezentující frontu probíhajících hovorů. Tato aktivita se spustí v případě více hovorů na lince. Uživatel si zde může zvolit, který hovor chce pozdržet a ve kterém chce dále pokračovat.
- Třída CodecsSettingsActivity – Jednoduchý seznam podporovaných kodeků, které lze podle potřeby aktivovat či deaktivovat.
- Třída GeneralSettingsActivity – Aktivita zobrazující obecné nastavení aplikace. Mimo jiné zde nalezneme nastavení umožňující konfiguraci hlasitosti hovorů, zapínání aplikace po dokončení startu operačního systému, potlačení šumu nebo možnost zapnutí režimu celé obrazovky probíhajícího videohovoru.
- Třída IdentitySettingsActivity – V této aktivitě lze nastavit ikonu uživatele a údaje potřebné pro komunikaci prostřednictvím SIP protokolu.
- Třída InterceptCallActivity – Jednoduchá aktivita sloužící k „zachytávání“ hovorů a zobrazení vhodné obrazovky hovoru.
- Třída MainFragmentActivity – Třída dědící z FragmentActivity. Tato třída slouží ke správě jednotlivých fragmentů nacházejících se v balíčku fragments. Mezi její funkce patří např. řízení přechodů mezi fragmenty a jejich inicializace. Dále tato třída poskytuje pomocné metody umožňující ukončení programu, provádění potřebných akcí v případě výpadku sítě a pomocné metody pro fragmenty jako je poskytnutí přístupu k databázi nebo získání data a času ve vhodném formátu.
- Třída MessageTypingActivity – Aktivita sloužící pro odesílání zpráv a zobrazení chatu. Skládá se z jednoduchého seznamu zpráv, tlačítka pro odeslání zprávy a textových polí reprezentujících text zprávy a příjemce zprávy.

- Třída MessagingSettingsActivity – Jednoduchá aktivita zobrazující nastavení XMPP protokolu a umožňující modifikaci. Mezi modifikovatelné položky této aktivity patří:
  - uživatelské jméno,
  - heslo,
  - XMPP server,
  - port serveru.
- Třída NattSettingsActivity – Třída poskytující nastavení technologií umožňujících průchod komunikace skrz NAT.
- Třída NetworkSettingsActivity – Aktivita, ve které lze nastavit vlastnosti připojení k síti. Nalezneme zde např. výběr, zda používat WiFi, nebo mobilní síť, protokol IPv4, nebo IPv6 a výběr protokolu transportní vrstvy.
- Třída QoSSettingsActivity – Jednoduchá třída umožňující nastavení QoS/QoE (Quality of Service/Quality of Experience) a nastavení preferovaného rozlišení videa.
- Třída SecuritySettingsActivity – Jedná se o aktivitu umožňující nastavení zabezpečení komunikace prostřednictvím SRTP protokolu.
- Třída SettingsActivity – Tato aktivita reprezentuje seznam všech možných kategorií nastavení určených k zobrazení další konfigurace.

10.2 Balíček adapters_items

Tento balíček obsahuje sadu tříd, které jsou použity za účelem naplnění seznamů, reprezentovanými třídou ListView, hodnotami. Celkem zde nalezneme dva typy položek:

- Adapater – Potomek třídy ArrayAdapter sloužící jako zdroj dat, poskytující jednotlivé řádky pro seznam tvořený třídou ListView.
- Item – Jednoduchá třída reprezentující položku, která je použita výše zmíněným adaptérem pro naplnění instance třídy ListView.

10.3 Balíček custom_views

Balíček obsahující pouze třídu CustomEditText. Jedná se o standardní třídu EditText s upravenou funkcí onKeyPreIme. Tato funkce je upravena za účelem zrušení tzv. „fokusu“ stiskem tlačítka zpět. Důsledkem toho dojde ke změně stavu prezence XMPP uživatele.

10.4 Balíček database

Součástí tohoto balíčku jsou třídy umožňující interakci s SQL databází a třídy reprezentující objekty, jejichž data jsou do této databáze uložena.

- Třída DatabaseContract – Třída sloužící k formální deklaraci organizace databáze. Obsahuje abstraktní vnitřní třídy, které definují obsah jednotlivých tabulek databáze.
- Třída DatabaseHelper – Pomocná třída pro vytvoření databáze a správu verzí.
- Třída Database – Jedná se o třídu, která vystavuje metody pro práci s SQLite databází. Mezi tyto metody patří množství standardních metod pro vkládání, odstraňování a dotazování se na záznamy databáze.
- Třída CallHistoryRecord – Třída reprezentující záznam historie hovorů, který je uložen do databáze. Tento záznam obsahuje veškeré potřebné informace pro další použití. Mezi tyto informace patří:
  - identifikační číslo kontaktu,
  - název kontaktu,
  - datum a čas hovoru,
  - informace, zda byl hovor odchozí nebo příchozí.
- Třída FavoriteContactsRecord - Třída slouží k reprezentaci oblíbeného kontaktu, který je uložen do databáze. Záznam v databázi obsahuje následující informace o kontaktu:
  - identifikační číslo kontaktu,
  - název kontaktu,
  - SIP číslo,
  - Ikona kontaktu,
  - XMPP jméno.
- Třída MessagesHistoryRecord - Třída představující zprávu uloženou do databáze. Jsou v ní obsaženy následující informace o zprávě:
  - identifikační číslo zprávy,
  - informaci o odesílateli,
  - informaci o příjemci,
  - textový obsah zprávy,
  - datum a čas,
  - ikona kontaktu.

10.5 Balíček db_communicator

Balíček obsahující množinu tříd, rozhraní a výčtových typů všeobecného charakteru. Jednotlivé prvky této množiny jsou podrobněji popsány níže.

- Třída DBC – Globální objekt definující aplikaci založenou na Doubango frameworku. Jedná se o třídu, která mimo jiné zprostředkovává přístup k instancím tříd PackageManager, AudioManager, SensorManager, KeyGuardManager, ConnectiviryManager a PowerManager. Také umožňuje zisk informací o URN (Uniform Resource Name) zařízení, číslu IMEI (International Mobile Equipment Identity), verzi SDK a mnoho dalších informací.
- Třída DialerUtils – Tato třída obsahuje pomocné metody ulehčující konfiguraci klávesnice, především z hlediska nastavení tónové volby.
- Třída Engine – Jedná se o vstupní bod programu, reprezentující SIP engine, umožňující přístup k veškerým službám, potřebných ke komunikaci prostřednictvím SIP protokolu. Hlavními metodami této třídy jsou metody start a stop sloužící k inicializaci resp. zastavení enginu.
- Třída GlobalBroadcastReceiver – Jednoduchý BroadcastReceiver, který umožňuje automatické spuštění aplikace po zapnutí daného zařízení, pokud je tato volba povolena v nastavení aplikace.
- Rozhraní IBaseScreen – Rozhraní definující metody, které umožňují získávání vlastností tříd, které jej implementují a souvisí se zobrazením určité obrazovky aplikace.
- Rozhraní IScreenService – Další rozhraní, které definuje, jakým způsobem lze s určitými obrazovkami aplikace pracovat.
- Třída Main – Třída, která se spouští při spuštění aplikace. Zajišťuje inicializaci a zastavení SIP enginu. Také má na starosti zobrazování správných obrazovek uživateli při provádění určitých operací. Mezi hlavní funkci lze rovněž zařadit spuštění aktivity MainFragmentActivity, jakmile dojde k inicializace SIP enginu.
- Třída NativeService – Nativní služba běžící na pozadí, která je spuštěna SIP enginem. Úkolem této služby je zpracovávání událostí, mezi které patří:
  - registrace,
  - události související s přenosem zpráv,
  - události protokolu MSRP,
  - SIP události typu INVITE.
- Třída ScreenService – Třída zajišťující práci s jednotlivými obrazovkami aplikace. Poskytuje mechanismy pro zobrazení určité obrazovky, ukončení zobrazení a změnu jejich pořadí v zásobníku aktivit.
- Výčtový typ ScreenType – Výčtový typ reprezentující typ jednotlivých obrazovek.

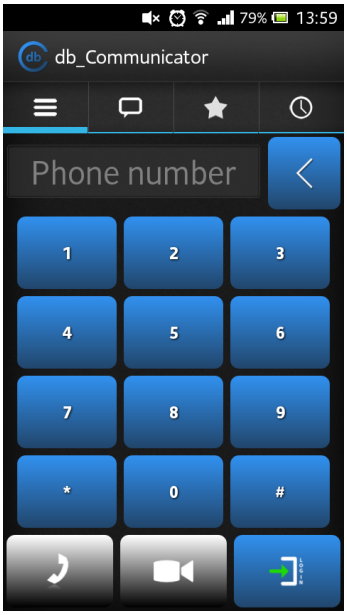
10.6 Balíček fragments

Obsahuje jednotlivé fragmenty spadající pod třídu MainFragmentActivity, která je řídí. Fragmenty, tvořící hlavní část GUI aplikace jsou představeny na obrázcích č. 10.1–10.4. Všechny tyto fragmenty poskytují menu, které umožňuje zobrazení obrazovky nastavení nebo ukončení aplikace. Výpis těchto fragmentů:

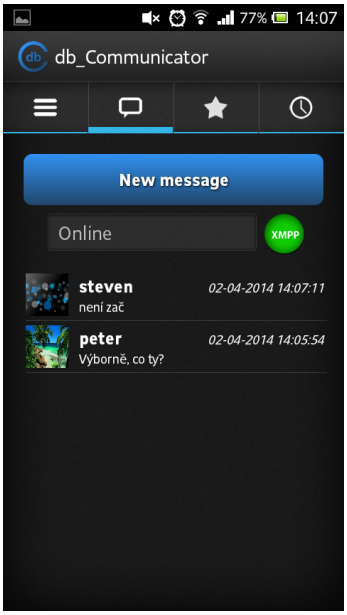
- Třída DialerFragment – Tento fragment zobrazuje klávesnici umožňující vytáčet hovory. Dále se zde nachází tlačítko pro přihlášení či odhlášení od SIP serveru a tlačítka, kterými lze inicializovat audio nebo video hovor. Při psaní čísla volaného se toto číslo zobrazuje v textovém poli v horní části obrazovky, na jehož pravé straně je situováno tlačíko umožňující toto číslo smazat v případě chyby.
- Třída MessagesFragment – Jedná se o fragment, který zobrazuje veškeré zprávy uspořádané do konverzací podle jména kontaku. Z obrazovky reprezentované tímto fragmentem lze také nastavit XMPP status (Online, Away, Offline) a libovolný textový popis stavu. Samozřejmostí je i možnost odeslání nové zprávy prostřednitvím tlačítka umístěného nahoře obrazovky. Toto tlačítko spustí aktivitu s názvem MessageTypingActivity.
- Třída FavoritesFragment – Fragment představující seznam oblíbených kontaktů. Každý kontakt v seznamu je reprezentován ikonou, názvem, SIP číslem a XMPP ikonou reprezentující jeho stav. Dále tento fragment disponuje tlačítkem umožňujícím přidávání nových kontaků, po jehož stisku je zobrazen dialog pro tvorbu nového kontaktu.
- Třída CallHistoryFragment – Poslední z řady fragmentů, jehož cílem je zobrazit seznam obsahující historii hovorů, ve které jsou obsaženy standardní informace, mezi které patří název kontaktu, ikona reprezentující, zda-li se jedná o příchozí nebo odchozí hovor, informace o datu a času hovoru.

10.7 Balíček xmpp

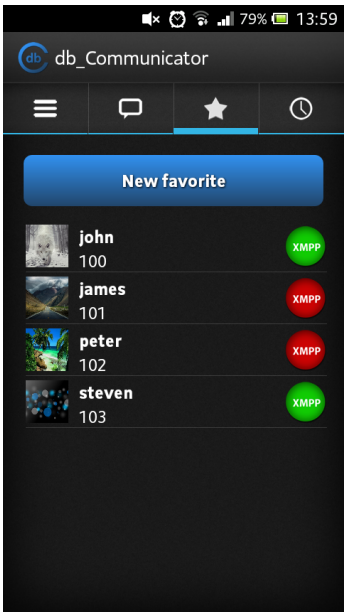
Tento balíček obsahuje jedinou třídu nesoucí název XMPPHelper. Tato třída slouží jako prostředník pro veškerou práci s protokolem XMPP, jako je připojení k serveru, řešení výpadků spojení, přihášení k XMPP serveru, přihlášení k odběru informací o kontaktech, nastevení prezenčního stavu a zjišťování stavu ostatních uživatelů. Dále zajišťuje příjem a odesílání zpráv, notifikace nových zpráv a odhlášení uživatele od XMPP serveru.



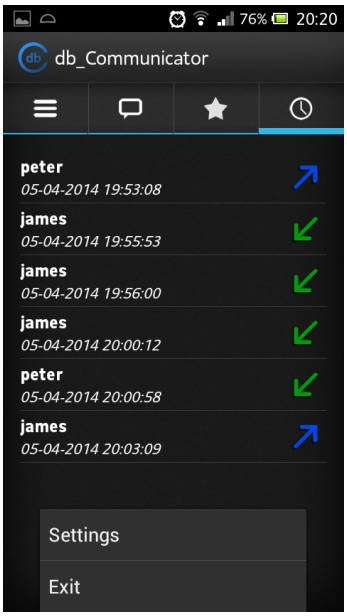
Obrázek 10.1: DialerFragment



Obrázek 10.2: MessagesFragment



Obrázek 10.3: FavoritesFragment



Obrázek 10.4: CallHistoryFragment

11 Testování aplikace

Pro účely testování aplikace bylo zapotřebí zajistit nezbytné vybavení. Jako zařízení představující VoIP klienta jsem využil smartphone Sony Xperia Sola a HTC Sensation XE. Dále byl použit také Android emulátor a SW klienti 3CX Phone, X-Lite, Zoiper, CSipSimple a Spark. VoIP ústředna byla tvořena serverem s open source platformou Asterisk a umožnění textové komunikace prostřednictvím protokolu XMPP zajišťoval server Openfire. Za účelem sledování přenosu dat mezi VoIP klientem a ústřednou byl použit analyzátor síťového provozu Wireshark, který umožnil detekci případných chyb. Aplikace byla testována celkem na třech serverech:

- lokální ústředna Asterisk,
- ústředna Asterisk situovaná ve školní síti,
- Openfire server situovaný ve školní síti.

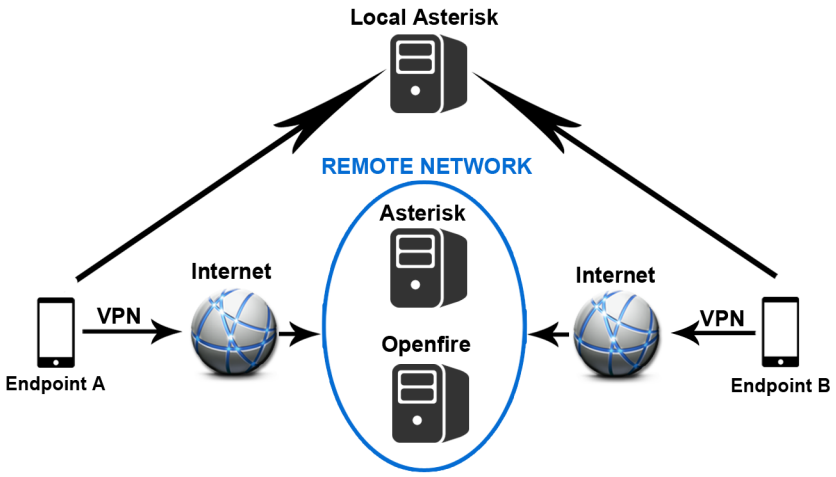
Připojení k serverům ve školní síti bylo z důvodu bezpečnosti zajištěno prostřednictvím VPN. Tohle řešení bylo omezující a nedovolovalo připojení k serverům z více zařízení najednou prostřednictvím jednoho účtu. Z tohoto důvodu byla vytvořena také lokální ústředna asterisk pro účely jednoduššího testování mimo školní síť. Architekturu testovací sítě naleznete na obrázku č. 11.1.

11.1 Sony Xperia Sola, HTC Sensation XE, Android emulátor

Hlavním zařízením, na kterém byla aplikace testována je smartphone Sony Xperia Sola s OS Android ve verzi 4.0.4. Vzhledem k potřebě zajištění správné funkčnosti aplikace na zařízeních s rozdílnými parametry, byl také využit smartphone HTC Sensation XE s OS Android ve verzi 4.0.3 a Android emulátor, který umožnil vytvoření široké škály zařízení s rozdílnými parametry. Funkčnost aplikace byla testována na verzích OS Android 3.0 a vyšších. Také byla ověřena funkčnost na zařízeních s rozdílným rozlišením a velikostí displeje. Kompletní testování aplikace včetně její jednotlivých funkcí a kodeků bylo zajištěno nejen použitím mnou vytvořené aplikace, ale byla také otestována interoperabilita mezi mou aplikací a zmíněnými SW klienty.

11.2 Asterisk

Jedná se o bezplatný open source framework umožňující vytvářet víceprotokolové realtime komunikační systémy. Asterisk je využíván celosvětově malými i velkými společnostmi, call centry i vládními organizacemi. Na světě je již přes milión komunikačních systémů založených na ústředně Asterisk, která slouží jako základ pro kompletní telefonní systémy, nebo pouze jako rozšíření existujících systémů. Dnes je tento projekt udržován společnými silami technologické společnosti Digium a komunitou Asterisk. [19].



Obrázek 11.1: Architektura testovací sítě

Asterisk je open source, spolehlivý, flexibilní a stabilní framework, původně navržen pro OS Linux. Dnes existuje mnoho jeho variací, které podporují mnoho VoIP protokolů, mimo jiné zahrnující:

- SIP,
- IAX,
- H.323,
- MGCP (Media Gateway Control Protocol).

Vyjma VoIP protokolů podporuje také množství tradičních protokolů určených pro sítě založené na přepojování okruhů, jakými jsou např. SS7 (Signalizační Systém č. 7) nebo ISDN (Integrated Services Digital Network).

Asterisk může být spuštěn i na zařízeních s nízkým výkonem, a to především díky jeho vysoké konfigurovatelnosti, modulárnosti a zdrojového kódu psaného v jazyce C [35]. Níže jsou vypsány příklady komplexních řešení, které lze využitím Asterisku vytvořit:

- pobočková ústředna (PBX),
- konferenční server,
- voicemail služby s adresářem,
- interaktivní hlasový průvodce (IVR server),
- packet voice server.

12 Závěr

Cílem bakalářské práce bylo vytvoření VoIP klienta pro operační systém Android. Po úvodním seznámení s platformou Android a se základy VoIP komunikace, jsem na základě nabytých znalostí začal vyvíjet aplikaci podporující audio přenosy, video přenosy a textovou komunikaci. Navzdory zabudovaného SIP API v Androidu verze 2.3 (Gingerbread) a vyšší, jsem byl nucen aplikaci vytvořit na základě již existujících open source klientů, jelikož toto API neposkytuje možnost video hovorů. Z důvodů uvedených v kapitole 9.3, jsem se rozhodl vycházet z projektu IMSDroid využívající Doubango framework. Po vytvoření spolehlivě fungujícího VoIP spojení jsem implementoval mechanismy pro ukládání perzistentních dat, které jsem popsal v kapitole 9.5. Dále jsem aplikaci rozšířil o možnost přenosu zpráv a sledování prezence prostřednictvím protokolu XMPP, který patří mezi nejpoužívanější protokoly pro IM komunikaci. Toho jsem docílil využitím aSmack API.

V bakalářské práci jsem splnil veškeré body zadání, vyjma použití protokolu SOAP (Simple Object Access Protocol) a umožnění přenosu zpráv prostřednictvím protokolu SIP (SIMPLE). SOAP měl původně sloužit pro ovládání multimédiálního systému, ale po dohodě s vedoucím bakalářské práce nebyl implementován a veškeré ovládání se přesunulo do webového rozhraní, které je v aplikaci dostupné při probíhajících hovorech. Po konzultaci s vedoucím bylo rovněž vynecháno použití protokolu SIP (SIMPLE) pro přenos zpráv.

Vzhledem ke splnění všech ostatních bodů bakalářské práce, byl tímto vytvořen univerzální komunikační nástroj, který umožňuje audio i video hovory za použití široké škály kodeků, komunikaci prostřednictvím Wi-Fi i mobilních sítí a také poskytuje řadu dodatečných funkcí. Mezi dodatečné funkce patří např. podpora pro IM komunikaci prostřednitvím XMPP protokolu, DTMF volba, NAT-T, QoS nebo zobrazení webového prohlížeče přes půl obrazovky při probíhajících hovorech. Kompletní výčet funkcí lze nalézt v dokumentaci programu v příloze A nebo na webových stránkách projektu IMSDroid [34], ze kterého byla většina funkcí převzata. Jelikož výsledná aplikace poskytuje také zabezpečení komunikace a rozsáhlé spektrum nastavení, jedná se o konkurenceschopný komunikační nástroj, který jeho uživatelům dokáže snížit výdaje vynaložené na SMS zprávy a nákladné telefonní hovory.

Budoucí vývoj tohoto projektu bude zaměřen na podporu technologie VPN zajišťující bezpečnost komunikace prostřednictvím nedůvěryhodné sítě a dále možnost vzdálené konfigurace ústředny prostřednictvím webových služeb. Také bude přidána podpora pro telefony se starší verzí operačního systému Android.

13 Reference

[1] Svět sítí: Věčné téma: přepojování okruhů či paketů? (1). [online]. [cit. 2014-04-22]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=Vecne-tema-prepojovani-okruhu-ci-paketu-1-1842006

[2] Svět sítí: Věčné téma: přepojování okruhů či paketů? (2). [online]. [cit. 2014-04-22]. Dostupné z: http://www.svetsiti.cz/clanek.asp?cid=Vecne-tema-prepojovani-okruhu-ci-paketu-2-1942006

[3] M. Vozňák. Signalizační protokoly v NGN. Dokument dostupný na URL: http://homel.vsb.cz/~voz29/ss/ss_13pr.pdf

[4] MB data: Úvod do VoIP. [online]. [cit. 2014-04-22]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm

[5] WALLACE, Kevin. VoIP bez předchozích znalostí. Vyd. 1. Překlad Jan Gregor. Brno: Computer Press, 2007, 231 s. ISBN 978-80-251-1458-2.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. RFC 3261 – SIP: Session Initiation Protocol, IETF, 2002. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3261.txt

[7] A. B. Roach. RFC 3265 – Session Initiation Protocol (SIP)-Specific Event Notification, IETF, 2002. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3265.txt

[8] S. Donovan. RFC 2976 – The SIP INFO Method, IETF, 2000. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc2976.txt

[9] J. Rosenberg, H. Schulzrinne. RFC 3262 – Reliability of Provisional Responses in the Session Initiation Protocol (SIP), IETF, 2002. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3262.txt

[10] K. Morneault, R. Dantu, G. Sidebottom, B. Bidulock, J. Heitz. RFC 3331 – Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) - User Adaptation Layer, IETF, 2002. Dokument dostupný na URL: http://tools.ietf.org/rfc/rfc3331.txt

[11] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle. RFC 3428 – Session Initiation Protocol (SIP) Extension for Instant Messaging, IETF, 2002. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3428.txt

[12] R. Sparks. RFC 3515 – The Session Initiation Protocol (SIP) Refer Method, IETF, 2003. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3515.txt

[13] A. Niemi. RFC 3903 – Session Initiation Protocol (SIP) Extension for Event State Publication, IETF, 2004. Dokument dostupný na URL: http://tools.ietf.org/html/rfc3903

[14] L. Sova. VOIP – HLASOVÁ KOMUNIKACE V IP SÍTÍCH, VOIP GSM, 2006. Dokument dostupný na URL: http://www.petrpexa.cz/diplomky/sova.doc

[15] Root: Android stále roste, už má 82% trhu smartphonů. [online]. [cit. 2014-04-22]. Dostupné z: http://www.root.cz/zpravicky/android-stale-roste-uz-ma-82-trhu-smartphonu

[16] Mobilizujeme: Google Play dohnal App Store v počtu aplikací. [online]. [cit. 2014-04-22]. Dostupné z: http://mobilizujeme.cz/clanky/google-play-dohnal-app-store-v-poctu-aplikaci/

[17] M. Krumnikl. Android. Dokument dostupný na URL: http://tamz2.mrl.cz/download/TAMZ2-2012-1.pdf

[18] Doubango: Doubango v2.x. [online]. [cit. 2014-04-22]. Dostupné z: https://code.google.com/p/doubango/

[19] Asterisk: Get Started. [online]. [cit. 2014-04-22]. Dostupné z: http://www.asterisk.org/get-started

[20] HW: DTMF - fámy a skutečnost. [online]. [cit. 2014-04-22]. Dostupné z: http://www.hw.cz/teorie-a-praxe/dokumentace/dtmf-famy-a-skutecnost.html

[21] Voice Quality: Quality of Service for Voice over IP. [online]. [cit. 2014-04-22]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html

[22] VoIP-info: QoS. [online]. [cit. 2014-04-22]. Dostupné z: http://www.voip-info.org/wiki/view/QoS

[23] VoIP-info: STUN. [online]. [cit. 2014-04-22]. Dostupné z: http://www.voip-info.org/wiki/view/STUN

[24] RTP: A Transport Protocol for Real-Time Applications (rfc1889) [online]. [cit. 2014-04-22]. Dostupné z: http://tools.ietf.org/html/rfc1889

[25] RTP: A Transport Protocol for Real-Time Applications (rfc3550) [online]. [cit. 2014-04-22]. Dostupné z: http://tools.ietf.org/html/rfc3550

[26] Datagram Transport Layer Security [online]. [cit. 2014-04-22]. Dostupné z: https://tools.ietf.org/html/rfc4347

[27] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman. RFC 3711 – The Secure Real-time Transport Protocol (SRTP), IETF, 2004. Dokument dostupný na URL: http://www.ietf.org/rfc/rfc3711.txt

[28] Android Developers: Activities. [online]. [cit. 2014-04-22]. Dostupné z: http://developer.android.com/guide/components/activities.html

[29] Android Developers: Services. [online]. [cit. 2014-04-22]. Dostupné z: http://developer.android.com/guide/components/services.html

[30] Android Developers: Content Providers. [online]. [cit. 2014-04-22]. Dostupné z: http://developer.android.com/guide/topics/providers/content-providers.html

[31] Android Developers: Intent. [online]. [cit. 2014-04-22]. Dostupné z: http://developer.android.com/reference/android/content/Intent.html

[32] M. Krumnikl. Application Components. Dokument dostupný na URL: http://tamz2.mrl.cz/download/TAMZ2-2012-2.pdf

[33] Linphone: Software architecture. [online]. [cit. 2014-04-22]. Dostupné z: http://www.linphone.org/eng/documentation/dev/

[34] IMSDroid: Highlights. [online]. [cit. 2014-04-22]. Dostupné z: https://code.google.com/p/imsdroid/

[35] CESNET: Asterisk a embeding. [online]. [cit. 2014-04-22]. Dostupné z: https://sip.cesnet.cz/cs/swahw/asterisk

[36] XMPP Standards Foundation: XMPP Technologies Overview. [online]. [cit. 2014-04-22]. Dostupné z: http://xmpp.org/about-xmpp/technology-overview/

[37] SIMPLE Made Simple: An Overview of the IETF Specifications for Instant Messaging and Presence Using the Session Initiation Protocol (SIP). [online]. [cit. 2014-04-22]. Dostupné z: https://tools.ietf.org/html/rfc6914

[38] CISCO: Quality of Service Overview. [online]. [cit. 2014-04-22]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios/qos/configuration/guide/15_1/qos_15_1_book/qos_overview.html

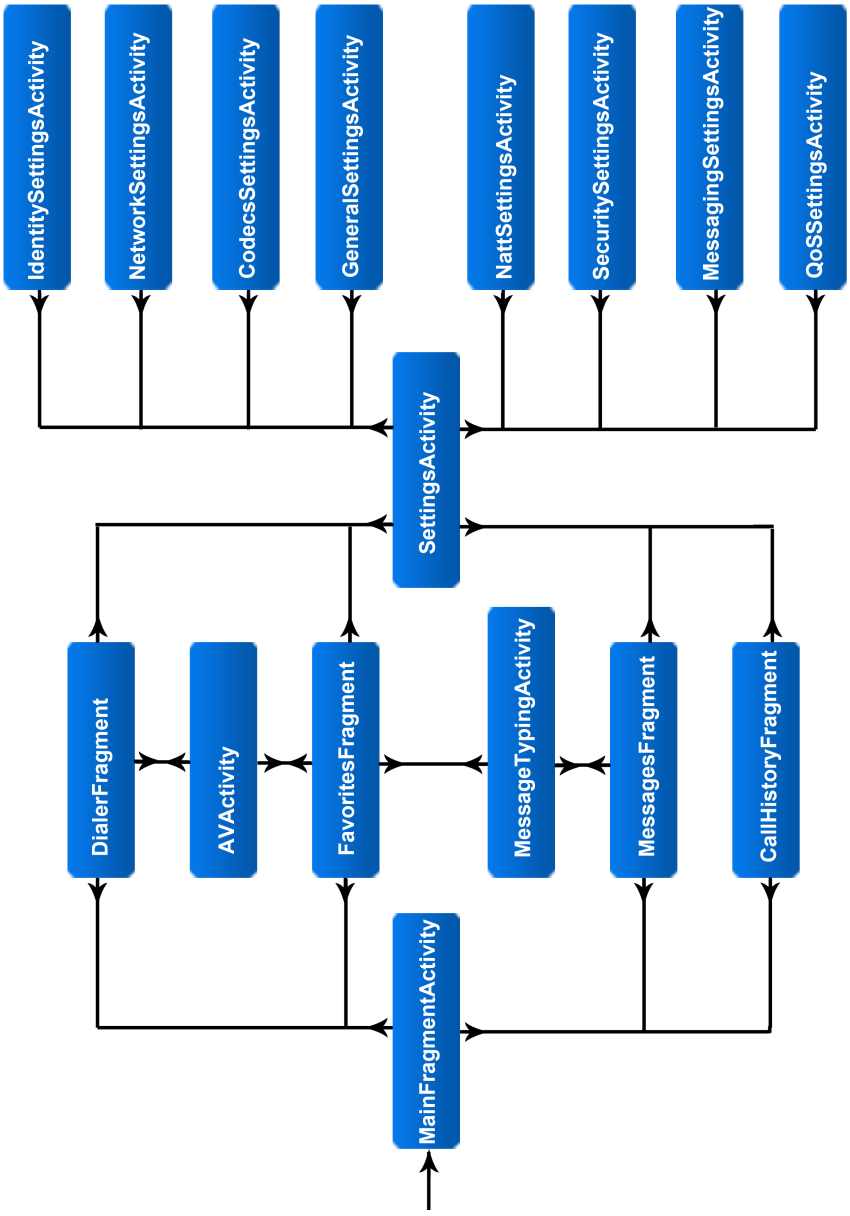
[39] Flowdalic: asmack. [online]. [cit. 2014-04-22]. Dostupné z: https://github.com/flowdalic/asmack

A Adresářová struktura přiloženého CD

Přiložené CD obsahuje následující adresáře:

- Aplikace – Obsahuje vytvořenou aplikaci pro Android a kompletní projekt vytvořené aplikace v zip archívu.
- Dokumentace – Obsahuje dokumentaci programu, vygenerovanou nástrojem Javadoc a dokumentaci Doubango frameworku.
- Obrázky – Obsahuje obrázky použité v této bakalářské práci a snímky GUI vytvořené aplikace.
- Text_bakalářské_práce – Obsahuje text této práce ve formátu pdf.

B Architektura GUI aplikace



Obrázek B.1: Architektura GUI aplikace