VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

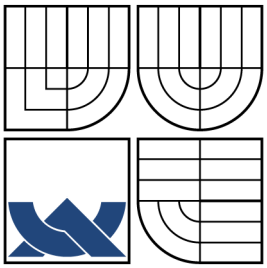
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATION

SIP KLIENT V JAZYCE JAVA

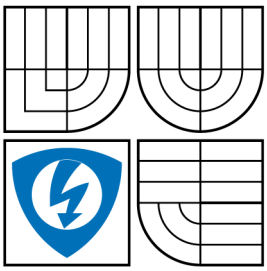
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN ZUKAL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATION

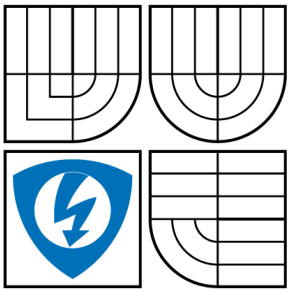
SIP KLIENT V JAZYCE JAVA
SIP CLIENT IN JAVA PROGRAMMING LANGUAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ZUKAL
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR ČÍKA
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Zukal Martin	***ID:***	78430
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

SIP klient v jazyce JAVA

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se signalizačním protokolem SIP. Navrhněte softwarové řešení VoIP klienta v jazyce JAVA tak, aby splňoval požadavky dle standardu RFC 3261. Implementujte základní metody pro obsluhu hovorů. K testování funkčnosti výsledné aplikace použijte VoIP server IceWarp.

DOPORUČENÁ LITERATURA:

[1] SINNREICH, H., JOHNSTON, A. B. Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol, Canada: Wiley, 2006. ISBN 978-0471776574
[2] JOHNSTON, A. B. SIP: Understanding the Session Initiation Protocol, Second Edition. London: Artech House Publishers, 2004. ISBN 978-1580536554

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Petr Číka

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

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práve třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být 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.

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Martin Zukal
Bytem: Lýskova 1043/21, 63500, Brno - Bystrc
Narozen/a (datum a místo): 12.10.1985, Brno

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: SIP klient v jazyce JAVA
Vedoucí/školitel VŠKP: Ing. Petr Číka
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

Tato bakalářksá práce se věnuje návrhu a realizaci aplikace typu klient, která využívá protokol SIP. V první části jsou popsány základní vlastnosti protokolu SIP. Důraz je kladen na porozumění nejzákladnější činnosti tohoto protokolu, nikoli na popis celé problematiky spjaté s protokolem SIP. Dále následuje stručný popis jazyku Java, ve kterém je aplikace vytvářena. Nejrozsáhlejší částí práce je popis návrhu a implementace vyvíjené aplikace. V této části bude čtenář seznámen s postupy, které byly použity během vývoje. Není opomenut ani výběr knihoven a zdůvodnění učiněných rozhodnutí. Práce je zakončena popisem testovacích scénářů ověřujících funkčnost aplikace.

KLÍČOVÁ SLOVA

SIP, Session Initiation Protocol, VoIP telefonie, Internet, Java, programování, SDP, Session Description Protocol, RTP, Real-time Transport Protocol, Model View Controller, applet

ABSTRACT

This bachelor's thesis deals with a design of a client-based application with the utilization of the Session Initiation Protocol (SIP). The first part describes SIP with an emphasis on basic concepts. Then a brief description of the Java programming language which will be used for the implementation follows. The most extensive part is devoted to the description of a design and implementation of the application. Some techniques used during the development of the applicaton are being discussed in this part. The thesis is finished up with test scenarios verifying functionality of the application.

KEYWORDS

SIP, Session Initiation Protocol, VoIP telephony, Internet, Java, programming, SDP, Session Description Protocol, RTP, Real-time Transport Protocol, Model View Controller, applet

ZUKAL, M. *SIP klient v jazyce JAVA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 50 s. Vedoucí bakalářské práce Ing. Petr Číka.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „SIP klient v jazyce JAVA“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce 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 bakalářské práce Ing. Petru Číkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Pro přenos hlasu přes Internet (velice módní je označení VoIP, což je zkratka pro Voice over IP) existuje dnes několik možností. Jejich společným jmenovatelem je oddělení signalizace hovoru od přenosu samotných multimediálních dat. Signalizace je řešena jednotlivými architekturami různě, ale pro přenos multimediálních dat je téměř vždy použit Real-time Transport Protocol (RTP) standardizován v RFC 3550 [13]. Pro spolehlivé spojení na bázi VoIP je nutné, aby síť umožňovala zajištění tzv. kvality služeb, zkráceně QoS (Quality of Service).

Nejstarší archikekturou umožňující přenos hlasu přes jakoukoli paketovou síť je doporučení ITU-T H.323. Jedná se o sadu protokolů a kodeků, které dohromady zajišťují vybudování, trvání a zrušení spojení. Nejčastější je H.323 využíván ve videokonferenčních systémech. Toto doporučení je implementováno například v softwaru NetMeeting od společnosti Microsoft, nebo v open source aplikaci Ekiga.

Poměrně novým protokolem je IAX (Internet Asterisk eXchange). Tento protokol zatím nebyl standardizován žádnou standardizační organizací. Na webových stránkách projektu [14] však lze najít přesný popis vlastností tohoto protokolu. Jedná se o open source protokol a jako takový nalézá uplatnění v projektech s otevřeným zdrojovým kódem. Již podle jména je jasné, že bude použit v pobočkové ústředně Asterisk, ale i jiné open source projekty přidávají podporu tohoto protokolu. Za všechny jmenujme například OPAL (Open Phone Abstraction Library), nebo YATE (Yet Another Telephony Engine).

V dnešní době nejpopulárnějším je protokol Session Initiation Protocol – SIP. První verze SIPu byla standardizována v RFC 2543, současná verze je standardizována v RFC 3261 [11]. Existují i novější verze RFC, které přidávají k SIPu další funkcionalitu. Samotný SIP je pouze signalizační protokol, který využívá dalších standardů (např. RTP pro přenos dat, SDP pro popis relace). Velikou výhodou SIPu je jeho jednoduchost a také to, že na rozdíl od architektury H.323 umožňuje spojení mezi klienty za NATem (Network Address Translation).

Cílem této práce je návrh a následná implementace aplikace, která bude umožňovat volání přes Internet s využitím právě posledně zmíněného protokolu SIP. V práci budou popsány dostupné knihovny, které je možné využít při vývoji aplikace a je vybrána ta najvhodnější. Samozřejmou součástí je popis architektury a budou zmíněny i některé testy, kterými aplikace prošla.

1 TEORETICKÉ MINIMUM

1.1 Protokol SIP

Session Initiation Protocol (zkráceně SIP) je protokol používaný pro sestavení, řízení a ukončení multimediální relace mezi dvěma koncovými body. SIP může být také využit pro přenos tzv. IM zpráv (Instant Messages). Vlastnosti protokolu, chování jednotlivých prvků architektury, přesný formát zpráv atd. jsou popsány v dokumentu RFC 3261. Protokol SIP je textově orientovaný protokol, který definuje 2 druhy zpráv:

- žádosti (request),
- odpovědi (response).

Nutnou podmínkou pro vytvoření relace je existence síťového spojení mezi klienty. SIP byl navržen pro nasazení v síti Internet, která je postavena na vrstvovém modelu TCP/IP. Protokol SIP je v síťovém modelu umístěn na aplikační vrstvě. Pro přenos SIP zpráv se využívá buď protokol TCP, který poskytuje spolehlivou, spojovanou službu, nebo protokol UDP, který poskytuje nespolehlivou, nespojovanou službu. Pokud je vyžadován šifrovaný přenos je možné využít ještě protokol TLS.

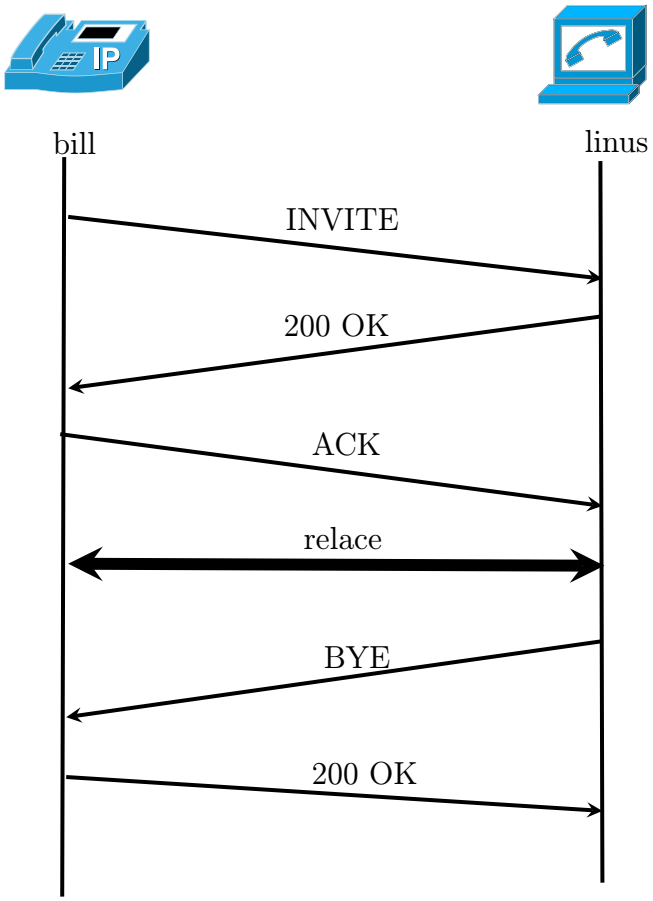
Architektura SIP má jedinou povinnou součást a tou je klient. Ostatní části (Proxy servery, Redirect server apod.) povinné nejsou a relace může být uskutečněna i bez nich. Označení klient je však, pokud mluvíme o problematice SIPu, poněkud nešťastné. Pro tuto práci bych raději zavedl označení User Agent – UA (uživatelský agent), využívané i v samotném RFC 3261. Každý user agent tedy musí obsahovat klienta (User Agent Client – UAC), který má nějaké požadavky (generuje žádosti) a server (User Agent Server – UAS), který je schopen tyto požadavky zpracovávat a vykonávat určité činnosti v závislosti na přijaté žádosti.

Podstatné je, že SIP je pouze signalizační protokol. To znamená, že popisuje pouze samotnou relaci, ale nepopisuje data, která jsou během relace vyměňována. Je tedy jedno, zda SIP použijeme pro řízení telefonního hovoru nebo pro řízení videokonference.

SIP je textový protokol. Zpráva (typicky zabalená v paketu) tedy není nijak kódována a pokud použijeme protokolový analyzátor k zachycení provozu na lince, budou zachycené pakety přímo čitelné bez nutnosti dešifrování.

1.1.1 Ukázka komunikace

Obrázek 1.1 ukazuje průběh nejjednoduššího spojení mezi dvěma agenty.



Obr. 1.1: Sestavení relace

Zpráva INVITE může mít následující tvar:

```
INVITE sip:linus@freesoftware.org SIP/2.0
Via: SIP/2.0/UDP redmond.microsoft.com:5060;branch=z9hG4bKf0591c7ff61
Max-Forwards: 70
To: Linus <sip:linus@freesoftware.org>
From: Bill <sip:bill@microsoft.com>;tag=ccb158e3
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 INVITE
Contact: Bill <sip:bill@redmond.microsoft.com:5060;transport=udp>
Content-Type: application/sdp
Content-Length: 161
```

```
v=0
o=bill 0 0 IN IP4 redmond.microsoft.com
s=subject
c=IN IP4 100.101.102.103
t=0 0
m=audio 18404 RTP/AVP 3 0 8 101
a=rtpmap:0 PCMU/8000
```

V následujícím textu budou blíže popsány jednotlivé řádky zprávy.

Řádky zprávy jsou označovány jako pole hlavičky a nesou název podle klíčového slova na začátku řádku. Každé pole hlavičky je od dalšího pole odděleno znakem ukončení řádku a přechodu na začátek nového řádku (CRLF – Carriage Return Line Feed).

První řádek (označován jako start line) obsahuje název metody, SIP URI (Uniform Resource Identifier) žádaného zdroje a označuje verzi protokolu SIP. Všechny tyto informace jsou odděleny mezerou. Metoda je v tomto případě INVITE. Tato metoda je používána pro sestavení relace.

Uniform Resource Identifier má tvar např. `sip:linus@freesoftware.org`. Za řetězcem `sip:` je umístěn řetězec podobný emailové adrese (má i stejnou funkci). Je to řetězec znaků, který jednoznačně identifikuje účastníka relace na globální úrovni. SIP tedy umožňuje jednoznačnou identifkaci účastníka ne podle adresy rozhraní (spjaté s jedinou jednoktou – např. PC), ale právě podle URI, což umožňuje, aby se účastník mohl připojit odkudkoli a vždy bude znám pod stejným jménem. Bližší informace o formátu URI lze nalézt v [10].

První pole (`Via`) se používá pro směrování zpráv. Obsahuje jméno a verzi protokolu SIP, dále použitý transportní protokol (ve výše uvedeném případě UDP) a nakonec obsahuje adresu uzlu, přes který zpráva prošla. Uzlem se zde rozumí jak servery tak klienti. Když klient vygeneruje žádost, automaticky do ní vloží pole `Via`. Za adresou uzlu je ještě parametr `branch`, který se používá pro identifikaci transakce. Každá odpověď na jednu žádost bude mít stejný parametr `branch`. Polí Via může být v hlavičce více, protože každý server, přes který zpráva prochází, přidá do hlavičky své pole `Via`.

Pole `Max-Forwards` udává kolikrát může být zpráva přeposlána (přes kolik uzlů sítě může projít). Každý server jeho hodnotu dekrementuje o jedna. Těmito vlastnostmi se podobá parametru Time To Live známého z protokolu IP.

Dalšími poli obsaženými v hlavičce jsou pole `To` a `From`, které obsahují jméno a SIP URI účastníků, kteří se mají účastnit relace. Jméno je zpravidla zobrazeno účastníkovi při příchozím volání, protože je člověku bližší než URI. Za pole `To` je ještě připojen parametr `tag`, který je vygenerován jako náhodná sekvence znaků a čísel.

Jeho úkolem je jednozančná identifikace dialogu (na straně každého user agenta).

Pole `Call-ID` je jedinečný identifikátor hovoru. Je vytvořen tak, že klient vygeneruje unikátní sekvenci znaků, připojí znak `@` a za něj připojí své doménové jméno. Tím je zajištěna jeho globální jedinečnost. Toto pole společně s parametry `tag` u obou polí `To` a `From` slouží k jendoznačné identifikaci dialogu.

Dalším polem je pole `CSeq`, které obshuje sekvenční číslo a název metody. Číslo je s každou další žádostí o jedničku inkrementováno.

Pole `Contact` je dalším polem, které musí být obsaženo v hlavičce SIP paketu. Toto pole obsahuje billovo SIP URI. Uvedené URI může být použito pro přímé spojení s agentem. Lze tak obejít všechny případné servery v cestě a komunikovat přímo s druhým agentem.

Nakonec následuje pole `Content-Type` a `Content-Length`, které popisuje typ a velikost zprávy obsažené v tělě SIP paketu. Touto zprávou je v tomto případě hlavička SDP paketu. Session Description Protocol se používá pro popis multimediální relace jako takové a také pro popis formátu dat vyměňovaných během relace. Informace v SDP paketu se používají pro vyjednání kodeků. User agent v této podobě oznamuje svému protějšku jaká data je schopen přijímat a vysílat. Podrobnější informace o vyjednávaní formátů je možné nalézt v [12].

Poté, co linus obdrží zprávu INVITE (a rozhodne se, že chce s billem zahájit hovor), odpoví mu ve formě zprávy 200 OK:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP redmond.microsoft.com:5060;branch=z9hG4bKf0591c7ff61
;received=100.101.102.103
From: Bill <sip:bill@microsoft.com>;tag=ccb158e3
To: Linus <sip:linus@freesoftware.org>;tag=1e442909
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 INVITE
Contact: Linus <sip:linus@users.freesoftware.org>
Content-Type: application/sdp
Content-Length: 155

v=0
o=linus 15964 15964 IN users.freesoftware.org
s=subject
c=IN IP4 111.112.112.113
t=0 0
m=audio 10948 RTP/AVP 3 0 8 101
a=rtpmap:0 PCMU/8000
```

Tato zpráva vznikne v podstatě kopií zprávy INVITE se změnou jen v minimálním počtu polí, či parametrů. Změnil se první řádek, který nyní signalizuje, že jde o konečnou kladnou odpověď.

Důležité je, že nedochází k záměně polí `To` a `From`, jak by se někdo mohl mylně domnívat. To proto, že pole `From` odkazuje na tvůrce žádosti a pole `To` odkazuje na příjemce žádosti. 200 OK není považováno za žádost (je to odpověď), proto k záměně nedochází. Jedinou změnou v těchto polích je přidání parametru `tag` k poli `To`. Tentokrát byl parametr `tag` vygenerován agentem na pravé straně.

Ke změně došlo i v těle zprávy, tedy v hlavičce SDP paketu. Ta nyní obsahuje data, která popisují možnosti agenta vpravo (linus).

Aby bylo sestavení relace úspěšně dokončeno je třeba odeslat kladné potvrzení (ACK):

```
ACK sip:linus@users.freesoftware.org SIP/2.0
Via: SIP/2.0/UDP redmond.microsoft.com:5060;branch=z9hG4bKc9662452be0
To: Linus <sip:linus@freesoftware.org>;tag=1e442909
From: Bill <sip:bill@microsoft.com>;tag=ccb158e3
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 ACK
Content-Length: 0
```

Jak je vidět, ke změně došlo opět na prvním řádku, dále v poli `CSeq`, kde sice zůstává stále číslo 1, ale metoda byla změněna na ACK. Metoda ACK slouží k potvrzení zprávy INVITE. Po přijetí zprávy ACK se relace považuje za zahájenou a agenti si mezi sebou vyměňují multimediální data. Navázání spojení v SIPu probíhá ve třech krocích (stejně jako u protokolu TCP) a proto se často označuje jako „three-way handshake“. Poslední změnou je změna parametru `branch`, který byl nově vygenerován, protože ACK je považováno za novou transakci.

Pokud se jeden z účastníků relace rozhodne tuto relaci ukončit, musí druhému agentovi poslat zprávu BYE. Zpráva může vypadat takto:

```
BYE sip:bill@redmond.microsoft.com SIP/2.0
Via: SIP/2.0/UDP users.freesoftware.org:5060;branch=z9hG4bKf05c143b13
Max-Forwards: 70
To: Bill <sip:bill@microsoft.com>;tag=as03c052d8
From: Linus <sip:linus@freesoftware.org>;tag=1e442909
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 BYE
Contact: <sip:linus@users.freesoftware.org:5060>
Content-Length: 0
```

Nyní byla žádost vygenerována agentem vpravo (tj. linus), proto pole `Via` obsahuje jiné doménové jméno než zprávy výše. I přesto, že pole `Via` je rozdílné, oba agenti jsou schopni určit, že zpráva BYE je součástí jednoho dialogu podle pole `Call-ID` a parametru `tag` u polí `To` a `From`. Fakt, že zprávu odesílá linus, způsobil i záměnu v polích `To` a `From` oproti zprávám, které byly uvedeny výše.

K tomu aby byla relace korektně ukončena, musí být zpráva BYE ještě potvrzena druhým agentem:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP users.freesoftware.org:5060;branch=z9hG4bKf05c143b13
;recieved=111.112.112.113
Max-Forwards: 70
To: Bill <sip:bill@microsoft.com>;tag=as03c052d8
From: Linus <sip:linus@freesoftware.org>;tag=1e442909
Call-ID: 155ffd70f62abce5ec286877d29e1f0e@redmond.microsoft.com
CSeq: 1 BYE
Contact: <sip:linus@users.freesoftware.org:5060>
Content-Length: 0
```

Po obdržení této zprávy je relace ukončena.

Výše uvedená ukázka komunikace je opravdu tou nejjednodušší variantou, která se může vyskytnout. Bohužel v praxi komunikace tak jednoduchá není. Do hry totiž vstupují i další síťové prvky.

Těmito prvky mohou být proxy servery, sloužící ke směrování zpráv mezi jednotlivými sítěmi. Dále registrar servery, které udržují databázi, ve které se nachází údaje o tom, kde se momentálně nachází jednotliví agenti. V neposlední řadě se mohou objevit i SIP brány, které mohou spojovat SIP agenty do klasické telefonní sítě.

Možností je celá řada, ale jejich podrobný popis je příliš rozsáhlý. Další informace je možné nalézt v [11] nebo [7].

1.1.2 Další metody

V předchozí části jsme si ukázali základní metody, které se používají při sestavení relace. Byly jimi

- INVITE,
- ACK,
- BYE.

V této části budou popsány další metody, které je možné použít při komunikaci. Tyto další metody se používají například pro změnu parametrů hovoru, případně pro zjištění dalších možností druhého agenta, případně serveru.

Tab. 1.1: Další metody

Metoda	**Povinná pole v hlavičce**	**Stručný popis**
REGISTER	`Call-ID CSeq From To Via Contact Max-Forwards`	Používá se pro oznámení o přítomnosti agenta. Každý User Agent by měl při přihlášení k síti vyslat zprávu REGISTER, aby proxy server věděl kam má posílat žádosti určené tomuto agentovi.
CANCEL	`Call-ID CSeq From To Via Max-Forwards`	Slouží k ukončení relace (zrušení probíhajících transakcí) předtím, než bylo zasláno potvrzení ACK. Poté, co je přijato ACK, musí se relace ukončit pomocí metody BYE.
OPTIONS	`Call-ID CSeq From To Via Max-Forwards`	Metoda OPTIONS se používá pro zjištění možností jiného agenta, případně serveru. Odpověď vždy obsahuje pole `Allow`.
REFER	`Call-ID CSeq From To Via Max-Forwards Refer-To Contact`	Používá se pro odeslání odkazu na jiné URI (případně i URL). URI (resp.URL), které má být kontaktováno, se nachází v poli `Refer-To`.
SUBSCRIBE	`To Call-ID CSeq Max-Forwards Via Contact Event Allow-Events`	Metodu SUBSCRIBE použije user agent, který chce být upozorňován na určitou událost (např. přihlášení jiného agenta). K upozornění se používá metoda NOTIFY.

NOTIFY	`To From Call-ID CSeq Max-Forwards Via Contact Event Subscription-State Allow-Events`	Slouží k upozornění na určitou událost.
MESSAGE	`To From Call-ID CSeq Max-Forwards Via`	Používá se pro přenos IM (Instant Message) zpráv pomocí protokolu SIP. Zpráva pak většinou obsahuje i nepovinné pole `Subject`.
INFO	`Call-ID CSeq From To Via Max-Forwards`	Metodu INFO používá user agent pro zasílání signalizačních zpráv jinému user agentovi, se kterým má sestavenou relaci. Avšak nedochází ke změně parametrů hovoru.
UPDATE	`To From Call-ID Cseq Max-Forwards Via Contact`	Tato metoda se používá pro změnu parametrů hovoru (resp. relace obecně) předtím, než je doručena zpráva ACK. Po doručení zprávy ACK již musí být použita nová zpráva INVITE.

V tabulce 1.1 je pouze základní přehled dalších metod. Podrobnější popis metod lze nalézt v [7].

1.1.3 Odpovědi

S jednou odpovědí jsme se již setkali. Byla to odpověď 200 OK, která značila, že žádost byla kladně zpracována. 200 OK samozřejmě není jediná možná odpověď, proto budou popsány další možnosti, kterých je celá řada.-

Podobně jako protokol HTTP i SIP dělí odpovědi do několika skupin, které jsou označeny třímístným číslem. Každá skupina začíná jiným číslem, podle kterého je možné určit, kde nastala chyba. Zbývající dvě čísla pak chybu specifikují blíže. Za toto trojčíslí je pak připojen ještě text, ve kterém je stručný popis stavu, který nastal. Tento text není součástí žádného standardu, takže každý user agent (i server) může posílat odpovědi s libovolným textem a tyto odpovědi budou platnými SIP zprávami.

1xx – Informační

Zprávy z této skupiny jsou také někdy označovány jako dočasné (provisional). Pokud zpráva začíná číslem jedna, je to pro agenta signál, že jeho žádost byla doručena ke druhému agentovi a ten pracuje na jejím zpracování. Zprávy 1xx nejsou konečné, to znamená, že nevypovídají nic o úspěšnosti zpracování vyslaného požadavku. Do této skupiny patří zprávy, které uvádí tabulka 1.2.

Tab. 1.2: Odpovědi 1xx

Kód	**Stav anglicky**	**Popis**
100	Trying	Odesláno agentem okamžitě po přijetí zprávy INVITE.
180	Ringing	Indikuje, že u volaného začalo vyzvánění.
181	Call Is Being Forwarded	Upozorňuje na přesměrování hovoru.
182	Call Queued	Oznamuje, že požadavek volajícího byl zařazen do fronty a čeká na zpracování.
183	Session Progress	Informuje volajícího o stavu volání (např. obsazeno).

2xx – Úspěch

Odpovědi začínající číslem dvě informují příjemce o tom, že jeho zpráva byla úspěšně zpracována. Do této skupiny patří pouze dvě odpovědi, které jsou uvedeny v tabulce 1.3.

Tab. 1.3: Odpovědi 2xx

Kód	**Stav anglicky**	**Popis**
200	OK	Oznamuje úspěšné zpracování žádosti druhou stranou.
202	Accepted	Indikuje, že user agent dostal a pochopil žádost, ale server neumožňuje její provedení.

3xx – Přesměrování

Tyto odpovědi nejčastěji posílá prvek sítě nazývaný jako redirect server. Informuje tak user agenta o adrese, na které se nachází požadované URI (případně požadovaná služba). Zprávu o přesměrování může posílat i user agent a to v případě, že má nastavené přesměrování hovorů.

Tab. 1.4: Odpovědi 3xx

Kód	**Stav anglicky**	**Popis**
300	Multiple Choices	Zpráva obsahuje několik polí `Contact` a user agent by měl zkoušet kontaktovat jednotlivá URI v takovém sledu, v jakém jsou umístěny v této odpovědi.
301	Moved Permanently	Indikuje, že požadované URI je trvale dosažitelné na jiné adrese. Tato adresa může být uložena a použita pro příští žádosti.
302	Moved Temporarily	Upozorňuje, že požadované URI je dočasně dostupné na jiné adrese. Tato adresa naopak nemá být ukládána.
305	Use Proxy	Pro zjištění, kde se nachází požadované URI, je třeba kontaktovat proxy server, jehož adresa je v poli `Contact` této odpovědi.

4xx – Chyba na straně klienta

Odpovědi z této kategorie indikují, že žádost nemůže být splněna. Důvod nemožnosti zpracování žádosti je pak popsán v odeslané odpovědi. To, že se jedná o chybu na straně klienta, znamená ve většině případů, že žádost měla špatný formát (např. chybělo některé povinné pole v hlavičce apod.). V této kategorii je definováno více než třicet odpovědí. Bylo by nesmyslné zde popisovat všechny z nich, proto jsou v tabulce 1.5 uvedeny jen ty, se kterými se lze běžně setkat.

Tab. 1.5: Odpovědi 4xx

Kód	**Stav anglicky**	**Popis**
400	Bad Request	Značí, že žádost nebyla serverem pochopena. Nejčastějším důvodem je, že chybí některé z povinných polí.
401	Unauthorized	Indikuje, že žádost vyžaduje autentizaci (prokázání totožnosti).
403	Forbidden	Žádost byla doručena, měla správný formát (tj. byla pochopena), ale z nějakého důvodu (jiného než nutnost autentizace) je zamítnuta.
404	Not Found	Tato odpověď značí, že server nemá žádané URI ve své databázi.
405	Method Not Allowed	Žádost byla doručena, byla pochopena, ale server (nebo user agent) na ní nechce odpovědět.
407	Proxy Authentication Required	Před zpracováním žádosti je vyžadována autentizace.
408	Request Timeout	Vypršel čas pro zpracování žádosti. Žádost je pak většinou zaslána znovu a pole `Expires` obsahuje vyšší hodnotu než v předchozí žádosti.
480	Temporarily Unavailable	Požadované URI je dočasně nedostupné. Text většinou udává důvod nedostupnosti.
486	Busy Here	Tato odpověď je ekvivalentem tónu obsazeno v klasické telefonii.
491	Request Pending	Používá se k zabránění změny parametrů hovoru oběma agenty zaráz (tj. když oba agenti vyslali re-INVITE zprávu).

5xx – Chyba na straně serveru

Do této kategorie patří odpovědi, které značí, že žádost nemůže být zpracována z důvodu chyby na serveru. Odpověď většinou obsahuje pole `Retry-After`, ve kterém je zapsán časový interval, který udává, kdy je možné se opět pokusit o zaslání žádosti.

Tab. 1.6: Odpovědi 5xx

Kód	**Stav anglicky**	**Popis**
500	Server Internal Error	Došlo k chybě na serveru a server nemůže zpracovat žádost. Text za číselným kódem většinou udává důvod selhání serveru.
501	Not Implemented	Indikuje, že server není schopen zpracovat žádost. Používá se když zpráva obsahuje neznámou metodu.
502	Bad Gateway	Tuto odpověď zasílá proxy server, který slouží jako brána do jiné sítě. Oznamuje, že v druhé síti došlo k chybě a žádost nemůže být zpracována.
503	Service Unavailable	Služba je dočasně nedostupná.
504	Gateway Timeout	Zpráva nemohla být zpracována kvůli vypršení časového limitu v jiné síti (tj. opět ji posílá proxy server ve funkci brány).
505	Version Not Supported	Bude použito v budoucnu, kdy bude existovat více verzí protokolu SIP (nyní existuje pouze jedna a to 2.0).
513	Message Too Large	Zpráva nemohla být zpracována, protože byla příliš velká.

6xx – Globální chyba

Odpovědi z této skupiny může zaslat server v případě že ví, že žádost nemůže být zpracována nikde v síti. Pokud user agent dostane takovouto odpověď znamená to, že by neměl opakovat vysílání žádosti než uplyne doba obsažená v poli `Retry-After`. Odpovědi z této skupiny jsou popsány v tabulce 1.7.

Tab. 1.7: Odpovědi 6xx

Kód	**Stav anglicky**	**Popis**
600	Busy Everywhere	Obdoba odpovědi 486. Při obdržení odpovědi 600 však user agent nemá znovu žádost opakovat.
603	Decline	Žádost byla zamítnuta. Odpověď neobsahuje žádné informace o důvodu zamítnutí.
604	Does Not Exist Anywhere	Obdoba odpovědi 404 s tím rozdílem, že server ví, že požadované URI neexistuje nikde v síti.
606	Not Acceptable	Využívá se k vyjednávání parametrů hovoru. Obdržení této zprávy značí, že relace nemůže být sestavena, protože neexistuje formát pro přenos dat, který by vyhovoval oběma stranám.

1.2 Technologie Java

Technologie Java byla vyvinuta společností Sun Microsystems. První verze vývojové sady JDK (Java Development Kit) byla uvolněna v roce 1995. Od té doby se na této technologii intenzivně pracuje a nyní je dostupná verze 6.

Technologie Java se skládá ze dvou částí:

- programovacího jazyka,
- platformy.

Programovací jazyk Java patří mezi vyšší programovací jazyky. Je podobný jazyku C++, od něhož přebírá kladné vlastnosti (např. objektově orientovaný přístup, přenositelnost) a naopak odstraňuje jeho nedostatky (např. práci s ukazateli). Říká se, že Java je jednodušší na naučení než C++. Na rozdíl od C++ se programátor nemusí starat o uvolňování paměti, když již některý objekt nepotřebuje. To za něj obstarává tzv. garbage collector (český ekvivalent uváděný v [1] a [2] zní sběrač neplatných objektů). Jazyk Java dále například kontroluje meze polí. Při přístupu za hranice pole je sice program ukončen s běhovou výjimkou (tzv. runtime exception), ale nedojde k žádné škodě, ke které by mohlo lehce dojít např. v jazyce C (buffer overflow). Daní za tyto výhody je v některých případech nižší výkon aplikací vyvíjených v Javě (ale vývoj v této oblasti jde stále dopředu a nyní už se rozdíl pozná jen u opravdu rozsáhlých a náročných aplikacích).

Při překladu zdrojového kódu se napřed vytvoří „bajtový kód“, který je pak spouštěn na tzv. Java Virtual Machine (JVM). Pro Java Virtual Machine se v češtině používá název virtuální stroj jazyka Java. Tento stroj pak převede bajtový kód na instrukce právě použitého hardwaru.

Z výše uvedeného plyne, že programy napsané v Javě jsou nezávislé na platformě i architektuře klientského počítače, proto se s výhodou používají pro programování webových aplikací a appletů.

O správnou funkci programů na rozdílných platformách se stará již zmíněný virtuální stroj jazyka Java, který je samozřejmě pro každou platformu jiný, ale je dostupný pro všechny dnes používané platformy (Windows, Linux/UNIX, Mac OS atd.).

Součástí vývojového prostředí jazyka Java je také standardní knihovna jazyka Java, které zajišťuje klíčové funkce a velmi unsadňuje programátorovi práci, protože obsahuje řadu užitečných tříd a metod pro řešení běžně se vyskytujících programátorských úkolů (např. setřídit prvky v poli apod.).

2 POPIS ŘEŠENÍ

Při práci na návrhu jsem se částečně nechal inspirovat zdrojem [8]. Během práce na implementaci částí týkajících se protokolu SIP jsem se opíral zejména o zdroj [9]. V obecných implementačních postupech jsem využíval zejména zdrojů [1] a [2].

2.1 Cíle práce

Hlavním cílem práce je samozřejmě vytvoření funkční aplikace umožňující volání přes Internet.

Skutečnost, že výsledná aplikace má být psána v jazyce Java, obecně umožňuje zaprvé, aby byla aplikace přenositelná mezi různými platformami (Windows, Linux, Mac OS) a zadruhé, aby ji bylo možné spouštět z internetového prohlížeče.

Když jsem hledal na Internetu, zda již podobný projekt byl někdy realizován, zjistil jsem, že mnoho takových projektů neexistuje. Mezi projekty psanými v jazyce Java existuje jeden zajímavý open source projekt, který nese název SIP Communicator. Jedná se o poměrně rozsáhlou aplikaci, která kromě SIPu podporuje ještě celou řadu dalších protokolů.

Realizace SIP klienta, který by byl spustitelný z webového prohlížeče, je projekt velmi ojedinělý. Pokud vím, takováto aplikace existuje pouze jedna a bude o ní zmínka v kapitole 2.2.8.

Za cíl práce bych také označil prokázání vhodnosti programovacího jazyka Java pro takovýto druh aplikace.

Samozřejmým cílem je získání nových zkušeností jak v oblasti problematiky protokolu SIP, tak v oblasti programování v jednom z moderních a stále se vyvíjejících jazyků.

Následujících několik bodů shrnuje a částečně doplňuje cíle popsané výše.

- Implementace základních metod protokolu SIP pro řízení spojení
- Přenositelnost mezi různými platformami
- Možnost spouštění buď jako normální aplikace nebo spouštění z prohlížeče ve formě appletu
- Schopnost spolupráce se SIP serverem firmy IceWarp
- Schopnost spolurpáce s různými klienty
- Jednoduchost a stabilita

2.2 Výběr knihoven

Realizace celého SIP klienta by byla časově značně náročná, proto jsem se rozhodl využít některé z mnoha knihoven, které by mi usnadnily práci. SIP klient se v podstatě skládá ze tří hlavních částí, jejichž funkčnost zajišťují následující části software:

- SIP stack,
- Media stack,
- Knihovna pro grafické uživatelské rozhraní (GUI).

Použité slovo stack nemá vhodný český ekvivalent, dalo by se přeložit přibližně jako zásobník.

SIP stack obstarává veškerou funkcionalitu spojenou se samotným protokolem SIP. Samozřejmostí je parser zpráv (který přicházející textovou zprávu vhodně rozdělí do proměnných), jejich příjem a odesílání. Ve většině případů SIP stack poskytuje funkce pro diagnostiku hovorů.

Media stack řeší problematiku digitalizace a kódování hlasu, dále má na starosti přenos zvukového streamu pomocí protokolu RTP. V neposlední řadě je úkolem media stacku dohodování kodeků, které budou použity během hovoru.

Grafické uživatelské rozhraní je jediná věc, se kterou normální uživatel aplikace přijde do styku. Mělo by proto být jednoduché a intuitivní na ovládání. Dalším požadavkem na tuto část je, aby byla graficky na výši („aby pěkně vypadala“) a aby se s jednotlivými prvky z programátorského hlediska jednoduše pracovalo.

Vytvoření SIP stacku je úkol přibližně na rok pro tým programátorů. Nepřipadá tedy v úvahu, abych se o jeho realizaci pokoušel. To stejné platí i pro media stack s tím, že tato část je navíc závislá na platformě, takže by bylo nutné vytvořit různé media stacky pro různé platformy.

Udělal jsem tedy průzkum, zda existuje nějaká open source knihovna (resp. knihovny), která by SIP stack a media stack již implementovala. Stručný přehled zkoumaných knihoven následuje v dalším textu.

2.2.1 Sofia SIP

Jedná se o knihovnu založenou na SIP stacku od firmy Nokia. Podporuje RFC 3261 a s ním související doporučení. Umožňuje i posílání IM zpráv pomocí protokolu SIMPLE. Existuje možnost šifrování přenosu pomocí TLS nebo SSL. Tato knihovna je licencována pod licencí GNU Lesser General Public License (LGPL [5]). Sofia SIP je napsána v jazyce C. Problémem by mohlo být, že je primárně určená pro Linux/BSD operační systémy. Na portaci pro windows se sice pracuje, ale zatím

není k dispozici stabilní verze. Dalším podstatným nedostatkem je, že Sofia SIP neobsahuje media stack. Bylo by sice možné použít nějaký již existující media stack (např. gstreamer), ale výsledná aplikace by pak byla velmi složitá a náročná na realizaci.

Protože neexistuje jednoduché řešení, jak volat příkazy jazyka C z Javy a také proto, že knihovna neobsahuje media stack, rozhodl jsem se tuto knihovnu ve své práci nepoužít.

2.2.2 oSIP, eXosip, Antisip

Všechny tyto tři knihovny jsou vyvíjeny jedním autorem panem Aymericem Moizardem a jsou napsány v jazyce C. Podporují protokol SIP podle RFC 3261. oSIP a eXosip jsou licencovány pod LGPL licencí, Antisip je komerční knihovna. Primárně jsou knihovny určeny pro Linux, ale je možné je využít i ve Windows. Podle vyjádření pana Moizarda, není použití těchto knihoven nikterak jednoduché a vyžaduje velmi pokročilé znalosti jak problematiky SIPu, tak programování v jazyce C. Stejně jako Sofia SIP ani jeden z těchto tří projektů neobsahuje media stack.

Ani jedna z těchto knihoven se mi nezdá vhodná pro použití v mé práci, jednak z důvodu složitosti použití a také kvůli faktu, že jsou napsány v jazyce C.

2.2.3 PJSIP

Tento projekt podporuje protokol SIP dle RFC 3261. Zdrojové kódy jsou napsány v jazyce C a obsahují dostatek názorných komentářů. PJSIP zajišťuje funkcionalitu SIP stacku, zatímco přidružený projekt PJMEDIA zajišťuje funkcionalitu media stacku, což by velmi usnadnilo práci při vývoji výsledné aplikace. Možným problémem by zde mohly být licence, protože hlavně media stack využívá alternativní licencování. Některé části softwaru jsou majetkem třetích stran a ne vždy jsou zveřejněny pod volnou licencí.

Z důvodu nemožnosti volání funkcí v jazyce C z Javy jsem se rozhodl tuto knihovnu nepoužít i přesto, že obsahuje media stack.

2.2.4 YATE (Yet Another Telephony Engine)

Jedná se o poměrně rozsáhlý projekt, který je napsán v C++. Tato knihovna umožňuje realizaci jak klienta tak dalších částí (jako proxy server, registrar atd.). Kromě SIPu (podporuje verzi podle RFC 3261) je zde implementován i standard H.323 a protokol IAX, takže se nabízí využití tohoto projektu jako překladače mezi protokoly. Dalším možným využitím je realizace brány do PSTN sítě.

Jazyk C++ je Javě velmi podobný a proto by se dalo uvažovat o konverzi této knihovny do programovacího jazyka Java. Avšak rozsáhlost tohoto projektu tomu klade do cesty velké překážky.

Protože tato knihovna obsahuje pro potřeby této práce naprosto zbytečené funkce (viz H.323, IAX apod.) a také proto, že je psána v jazyce C++, rozhodl jsem se nevyužít možností tohoto projektu v mé práci.

2.2.5 sipXtapi

SipXtapi je projekt z dílny společnosti SIPfoundry. Je licencován pod LGPL licencí a je psán v jazyce C. Podporuje protokol SIP podle RFC 3261 a s ním souvisejících doporučení. Samozřejmostí je zpracování a transport audia, v blízké budoucnosti by měla být implementována i podpora přenosu videa. SipXtapi je plně přenositelné mezi dnes běžně používanými platformami (vývoj je sice směřován hlavně na platformu windows, ale existují i portace pro ostatní platformy). Součástí projektu je rozsáhlá wiki, kde jsou zpracovány jak návody na použití, tak odpovědi na nejběžnější problémy.

I když se jedná o povedený projekt, rozhodl jsem se jej nepoužít, protože je psán v jazyce C.

2.2.6 Java SIP toolkit

Zde se jedná o komerční produkt izraelské firmy Radvision, která je světovým leaderem v oblasti hlasové komunikace přes Internet. Mezi zákazníky této firmy patří např. společnosti Cisco Systems, SIEMENS, LG apod. Podařilo se mi kontaktovat pana Jonniho Niemanna z firmy Radvision, který mi poskytl informace uvedené níže.

Samozřejmostí je podpora SIPu podle nejaktuálnějšího doporučení (tedy RFC 3261). Podrobnější informace o tom, co všechno tento produkt nabízí je možné nalézt v příloze, kde je připojena reklamní brožura, kterou jsem od pana Niemanna obdržel. Jejich SIP toolkit je napsán v programovacím jazyce Java a obsahuje i jednoduchý media stack. Tyto dva fakty jsou veliké plus, ovšem zásadní mínus je vysoká cena produktu. Dalším omezením jsou i licenční podmínky, které dovolují použití knihovny pouze pro demonstrační účely a to po dobu jednoho roku.

Zejména kvůli ceně a licenčním podmínkám nemá cenu o této knihovně uvažovat jako o možnosti pro mou práci.

2.2.7 MjSip

Jedná se o open source implementaci kompletního SIP stacku napsaného v programovacím jazyce Java. MjSip je licencován pod GPL (GNU General Public License) licencí. Tento projekt je vyvíjen na univerzitě v Římě a v Parmě a je poměrně nový. Zdrojové kódy obsahují dostatek komentářů v angličtině, ale z letmého pohledu jsem usoudil, že některé části jsou ještě stále ve vývoji (zakomentované části pravděpodobně nefunkčního kódu apod.). Výhodou tohoto projektu je, že k dispozici je i jednoduchý klient, na kterém je ukázáno jak využívat knihovních funkcí MjSipu. Nevýhodou MjSipu je fakt, že neobsahuje media stack.

Hlavně z důvodu „čerstvosti“ tohoto projektu jsem se rozhodl nevyužít jej ve své práci.

2.2.8 JAIN SIP

JAIN je zkratka pro Java APIs for Integrated Networks. Jde o popis rozhraní, která byla vytvořena společností Sun Microsytems ve spolupráci se zástupci firem jako např. Cisco, British Telecom apod.

Z názvu je patrné, že jde pouze o API (tedy rozhraní), avšak implementace těchto rozhraní je na programátorovi. Naštěstí existuje referenční implementace tohoto API, která byla vyvinuta na National Institute of Standards and Technology (NIST) v USA. Hlavním leaderem projektu je pan Mudumbai Ranganathan z NIST, který se podílel i na tvorbě standardu JAIN SIP. Jedná se o open source knihovnu kompatibilní s protokolem SIP podle RFC 3261. Zdrojové kódy jsou psány v jazyce Java a obsahují velké množství užitečných komentářů ve stylu javadoc. Součástí projektu je i množství příkladů jak s knihovnou pracovat. Příklady popisují problematiku od jednoduchého vytvoření a zrušení spojení, přes práci se zprávou CANCEL až po poměrně náročnou ukázku využití zprávy REFER. Další ukázkou způsobu práce s referenční implementací rozhraní JAIN SIP je aplikace JAIN SIP applet phone. Jak název napovídá jde o applet (tedy aplikaci spustitelnou z webového prohlížeče), který umožňuje uskutečnění hovoru pomocí protokolu SIP přímo z webové stránky.

Vývoj tohoto projektu započal již v roce 2003 a od té doby se na něm stále pracuje a je stále udržován v takovém stavu aby implementoval nejaktuálnější verzi RFC.

Tato knihovna našla uplatnění v několika open source projektech, za všechny jmenujme již zmíněný SIP Communicator.

Nedostatkem tohoto jinak velmi zdařilého projektu je, že neobsahuje media stack.

2.2.9 Výběr nejvhodnějšího kandidáta

Z výše uvedeného přehledu je jasné, že nelze použít knihovny psané v jiném jazyce než v Javě. V úvahu by tedy připadaly poslední tři zmiňované knihovny.

Knihovna od firmy Radvision sice jako jediná z knihoven psaných v Javě obsahuje media stack, ale má omezené licenční podmínky a je velmi drahá.

Knihovna MjSip je zase projekt teprve na začátku vývoje a není jisté, že jeho vývoj bude nadále pokračovat. Proto jsem se rozhodl pro svou práci využít knihovnu JAIN SIP.

Nedostatek v podobě chybějícího media stacku je v případě jazyka Java poměrně jednoduché odstranit. Existuje totiž zdarma dostupné API (i s implementací) pro práci s médii v jazyce Java, které je vyvíjeno taktéž společností Sun Microsystems. Jedná se o Java Media Framework (JMF), který má všechny funkce, které jsou vyžadovány od media stacku. Mezi tyto funkce patří například schopnost přehrání streamovaného zvuku, který je přenášen protokolem RTP, dále kódování zvuku přicházejícího z mikrofonu a jeho posílání opět pomocí protokolu RTP atd. Abych toto tvrzení podepřel nějakým faktem, zmíním ještě jednou projekt SIP Communicator, ve kterém je JMF využito jako media stack. Více informací o JMF je možné nalézt na [15].

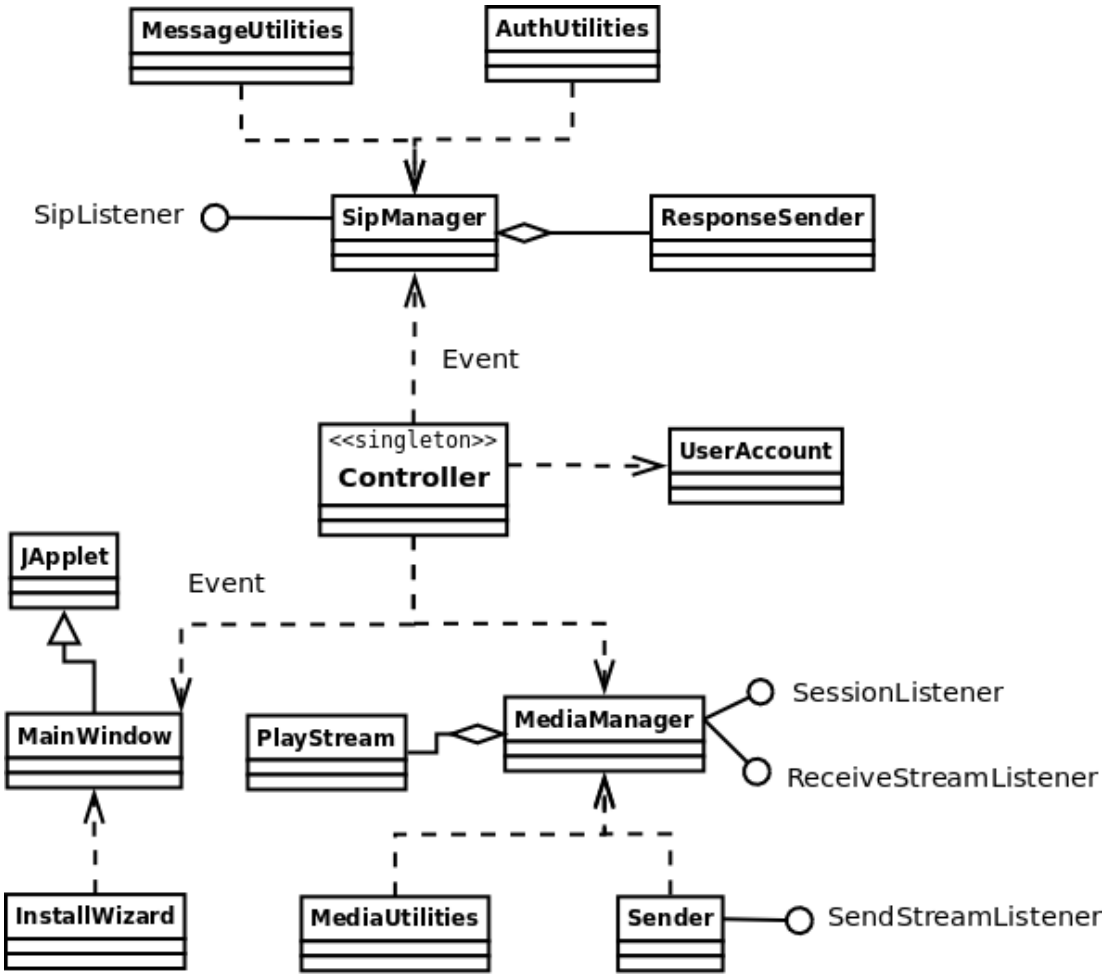
2.2.10 Grafické uživatelské rozhraní

Mohlo by se zdát, že ve výše uvedeném přehledu byla poněkud opomenuta volba knihovny pro grafické uživatelské rozhraní. Není tomu tak. Součástí vývojového prostředí Java je totiž i rozhraní Swing. Rozhraní Swing obsahuje velké množství komponent, které jsou nutné pro tvorbu kvalitního a interaktivního uživatelského rozhraní. Jednou z výhod je, že je možné si vybrat i z různých motivů vzhledu (standardní s názvem Metal a dále systémové vzhledy systému Windows, Mac OS i Linux).

Práce s rozhraním Swing je poměrně jednoduchá a to i přesto, že některé komponenty jsou velmi složité. Každá z komponent byla pečlivě navržena (jak po grafické, tak programové stránce) a obsahuje mnoho metod, které programátorovi velmi usnadňují práci. Pro tvorbu GUI bude tedy využito rozhraní Swing.

2.3 Architektura

Architektura aplikace je jednou z klíčových otázek, kterou je třeba vyřešit před samotným návrhem aplikace. Měla by vycházet z důkladné analýzy řešeného problému a měla by zohledňovat všechny jeho aspekty. Po provedení analýzy bylo rozhodnuto, že aplikace bude vytvořena podle návrhového vzoru, který se nazývá MVC (Model View Controller) [16].



Obr. 2.1: Class diagram aplikace

Zásadní vlastností tohoto návrhového vzoru je to, že odděluje aplikační logiku od prezentační vrstvy aplikace. Dalším kladem je, že v každé části jsou řešeny problémy, které spolu logicky souvisejí. Rozhraní mezi jednotlivými částmi pak umožňuje odhalit chyby, které by jinak byly odhalitelné jen velmi těžce.

Za zápor je možné považovat fakt, že některé problémy je nutné řešit ve více (někdy ve všech) částech aplikace, čímž se aplikace stává složitější a zvyšuje se riziko vzniku chyb.

Podle terminologie jazyka UML (Unified Modelling Language) lze objektový návrh aplikace popsat Class Diagramem (český ekvivalent diagram tříd se příliš často nepoužívá), který popisuje logické souvislosti jednotlivých tříd mezi sebou.

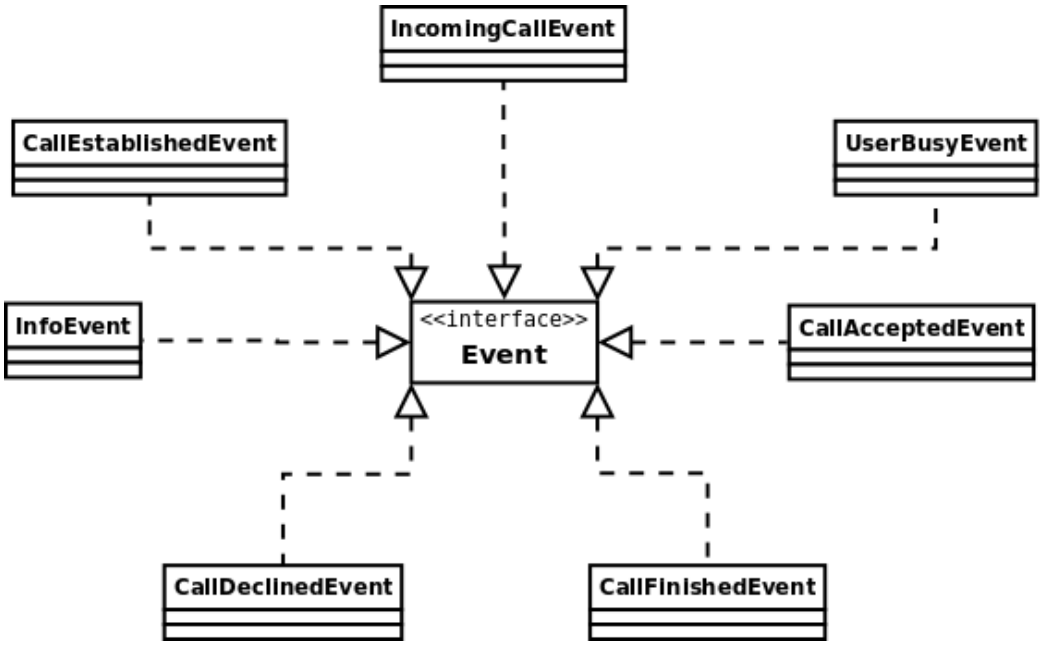
Jak bylo zmíněno výše, návrh vychází z návrhového vzoru MVC. Ten byl mírně upraven tak, aby aplikace nebyla příliš složitá, ale aby zároveň poskytovala výhody, které tento návrhový vzor nabízí.

Class diagram aplikace je zachycen na obrázku 2.1. Dále následuje stručný popis jednotlivých tříd.

2.3.1 Controller

Z obrázku 2.1 je vidět, že řízení celé aplikace má na starosti třída `Controller`. Tato třída je navržena podle návrhového vzoru singleton („jedináček“, nebo též množina s jedním prvkem), což znamená, že současně existuje pouze jedna instance této třídy. To je na jednu stranu výhodné, protože `Controller` pak má přehled o tom, v jakém stavu aplikace právě je, na druhou stranu to přináší problémy se sdílením některých prostředků.

Veškerá komunikace mezi jednotlivými částmi aplikace probíhá právě přes tuto třídu. Třída `Controller` komunikuje s ostatními částmi aplikace, buď prostřenictvím událostí, nebo volá příslušné metody daných tříd. UML diagram zobrazující vztahy mezi jednotlivými událostmi je na obrázku 2.2.



Obr. 2.2: Class diagram událostí

Třída `Controller` obsahuje několik metod `processEvent`, které reagují vždy právě na jednu událost z obrázku 2.2. Tento způsob implementace je podle [2] považován z hlediska dobrého objektového návrhu za nejvhodnější. Volání správné metody zajišťuje mechanizmus polymorfismu a tzv. pozdní vazby.

Rozhraní `Event` obsahuje dvě metody: `getInfoMessage` pro získání popisu nastalé události a `getSource` pro získání odkazu na zdroj události. Většina událostí plní svou funkci za použití dvou výše zmíněných metod, ale existují dvě události, které musejí přenášet více informací a proto obsahují více metod.

IncomingCallEvent a CallEstablishedEvent

Tyto dvě události jsou téměř totožné. Obě nesou informace o tom z jaké IP adresy, z jakého portu a v jakém formátu je vysílán RTP stream od druhého účastníka hovoru. Dále nesou informace o URI druhého účastníka.

Liší se pouze kontextem (tedy vlastně okamžikem), ve kterém jsou tyto události generovány. `IncomingCallEvent` je posílána v okamžiku, kdy je zachycen zpráva INVITE a `CallEstablishedEvent` je odeslána po obdržení zprávy 200 OK (potvrzuje úspěšné navázání spojení).

2.3.2 SipManager

Třída `SipManager` je vazebním prvkem mezi knihovnou JAIN SIP a zbytkem aplikace. Implementuje rozhraní `SipListener` z knihovny JAIN SIP a je registrována jako listener, což znamená, že knihovna JAIN SIP může volat metody této třídy při příchodu SIP zpráv. Metody rozhraní `SipListener` jsou následující:

- `void processRequest(RequestEvent e)`,
- `void processResponse(ResponseEvent e)`,
- `void processTimeout(TimeoutEvent e)`,
- `void processIOException(IOExceptionEvent e)`,
- `void processTransactionTerminated(TransactionTerminatedEvent e)`,
- `void processDialogTerminated(DialogTerminatedEvent e)`.

Nejdůležitější jsou první dvě metody `processRequest` a `processResponse`. Z názvu je patrné, že metoda `processRequest` je volána při příchodu žádosti (tedy zpráv jako např. INVITE, BYE apod.) a metoda `processResponse` je volána při příchodu odpovědi (např. 200 OK, 486 Busy Here apod.). Ostatní metody není pro základní obsluhu hovorů potřeba využívat (ale je nutné uvést deklaraci těchto metod).

Všechny zmíněné metody mají pouze jeden parametr – objekt zapouzdřující samotnou zprávu a další potřebné informace, pomocí kterých je možné na událost patřičně reagovat.

Ve zkratce by se dalo říct, že třída `SipManager` obsahuje aplikační logiku, která reaguje na příchozí zprávy v souladu s doporučením RFC 3261. V případě, že je vyžadována akce od uživatele (např. přijmutí hovoru), je poslána odpovídající událost třídě `Controller`.

2.3.3 ResponseSender

Třída `ResponseSender` obsahuje metody pro tvorbu a odesílání odpovědí na přicházející žádosti podle požadavků uživatele. Uveďmě krátký příklad využití této třídy:

Aplikace obdrží zprávu INVITE a ohlásí uživateli, že má příchozí hovor. Uživatel hovor buď přijme nebo odmítne. Pokud byl hovor přijat, třída `ResponseSender` odešle volajícímu kladnou odpověď (200 OK). Pokud uživatel hovor odmítne, třída `ResponseSender` odešle volajícímu zápornou odpověď.

`ResponseSender` je vnořená (někdy též nazývána vnitřní) třída uvnitř třídy `SipManager`. Tím, že je definována jako vnořená, má přístup ke všem členským proměnným nadřazené třídy a není potřeba volat její metody s velkým množstvím parametrů.

2.3.4 MessageUtilities

Vzhledem k tomu, že některé části SIP zpráv se opakují téměř v každé zprávě, která má být odeslána, bylo by vhodné shromáždit všechny metody, které se na tvorbě těchto částí podílí, do jedné třídy. Touto třídou je právě třída `MessageUtilities`.

K tvorbě jednotlivých polí je možné využít tovární třídy knihovny JAIN SIP. Tovární třídy, nebo-li factory classes jsou třídy, které v závislosti na předaných parametrech vracejí instanci určitého objektu (zde pole SIP zpráv). Metody těchto tříd jsou schopny vytvořit jednotlivá pole SIP zpráv podle zadaných parametrů.

Metody třídy `MessageUtilities` shromažďují více volání výše zmíněných továrních tříd a vytváří větší celky (v některých případech přímo celé zprávy), které jsou již přímo použitelné v SIP zprávách.

2.3.5 AuthUtilities

Během komunikace s využitím protokolu SIP je v určitých případech nutné ověřit, že komunikující strany jsou opravdu tím, za koho se vydávají. Proces, který ověří, že účastník má dostatečné oprávnění k provedení určité akce, se nazývá autentizace.

Obecně existují dva typy autentizace:

- autentizace uživatele vůči serveru,
- autentizace uživatele vůči uživateli.

Protokol SIP využívá pro autentizaci uživatelů stejný mechanizmus jako protokol HTTP. Princip činnosti je popsán v dokumentu RFC 2617 ([3]) a nazývá se digest access authentication scheme. Algoritmy pro „výpočet“ odpovědi ze známých údajů jsou popsány ve výše uvedeném dokumentu. Jejich konkrétní implementace je umístěna ve třídě `AuthUtilities`.

Tato třída nachází uplatnění zejména ve chvíli, kdy je třeba provést registraci na serveru pomocí zprávy REGISTER. Jedině po registraci je totiž možné využívat služeb serveru pro vyhledání druhého účastníka v případě, že neznáme přímo jeho kontaktní adresu (IP adresu).

2.3.6 MediaManager

Podobně jako je třída `SipManager` vazebním prvkem mezi knihovnou JAIN SIP a aplikací, tvoří třída `MediaManager` vazební prvek mezi aplikací a knihovnou Java Media Framework. Jak je vidět z obrázku 2.1 třída `MediaManager` implementuje rozhraní `SessionListener` a `ReceiveStreamListener`. Obě tato rozhraní pocházejí právě z JMF.

Mezi hlavní úkoly této třídy patří správná inicializace tříd `PlayStream` a `Sender`, dále vytvoření a ukončení RTP relace a v neposlední řadě také započetí a ukončení vysílání RTP streamu.

Další důležitou funkcí třídy `MediaManager` je i detekce zvukové karty.

2.3.7 PlayStream

Další vnořená třída v této aplikaci. Je inicializována třídou `MediaManager` a má na starosti rozpoznání a přehrávání příchozího RTP streamu.

Způsob, jakým je implementováno zachycení RTP streamu v JMF, umožňuje nedefinovat formát příchozího streamu. Aplikace jej tak rozpozná sama.

2.3.8 Sender

Vysílání RTP streamu není tak triviální jako jeho zachycení a přehrávání, proto třída `Sender` není definována jako vnořená. Úkolem třídy `Sender` je správně nakonfigurovat objekt `Processor` (součást JMF), který je pak schopen zachytávat zvuk ze zvukové karty, převádět jej do požadovaného formátu a následně jej posílat v podobě RTP streamu na určenou IP adresu a port.

2.3.9 MediaUtilities

Třída `MediaUtilities` obsahuje statické metody, které převádí názvy formátů používaných v JMF na payload type používaný v SDP paketu a naopak. Pokud je metoda definována jako statická znamená to, že není zapotřebí vytvářet instanci této třídy aby bylo možné k metodám přistupovat.

Dále obsahuje statické proměnné, ve kterých je definováno, jaké formáty RTP streamu aplikace podporuje. Protože Java Media Framework je zdarma a použití některých kodeků vyžaduje zakoupení licence, je počet podporovaných formátů omezen. Naštěstí je tento nedostatek více patrný při práci s videem, takže to nemá negativní vliv na použitelnost vyvíjené aplikace. Seznam podporovaných formátů lze najít na [15]. Existují i různé pluginy, kterými je možné množinu podporovaných formátů výrazně rozšířit, ale vzhledem k tomu, že podpora audia je poměrně dobrá, nebylo nutné tyto pluginy instalovat.

2.3.10 UserAccount

Aplikace je spouštěna minimálně se dvěma parametry. Těmi jsou uživatelské jméno v podobě emailové adresy a heslo. Parametr uživatelské jméno se skládá ze samotného uživatelského jména, znaku `@` a domény. Po spuštění aplikace je nutné tyto parametry uložit a z uživatelského jména je nutné získat adresu serveru, na kterém bude uživatel registrován. Všechny tyto informace společně s dalšími (jako je např. aktuálně používaný port) jsou uloženy v proměnných třídy `UserAccount`.

Instance třídy `UserAccount` je po inicializaci pomocí jednoho z několika přetížených konstruktorů uložena jako členská proměnná uvnitř třídy `Controller`.

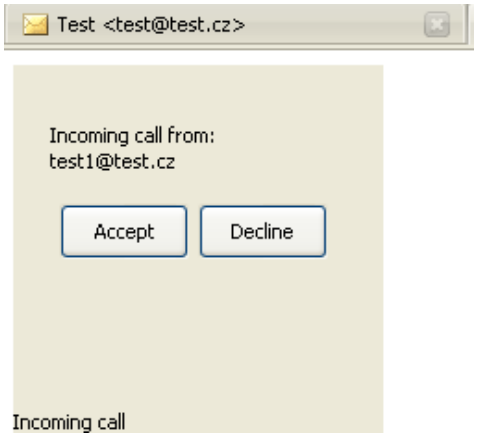
2.3.11 MainWindow

Aplikace může být spuštěna buď jako normální aplikace (z příkazového řádku kvůli předání parametrů), nebo jako applet (tedy spouštěna z webové stránky). Třída `MainWindow` je implementována tak aby byla schopná realizovat obě tyto možnosti. Je potomkem třídy `JApplet` (viz obr. 2.1) a obsahuje tedy metody potřebné pro běh appletu (`init()`, `start()`, `stop()` a `destroy()`), ale stejně tak obsahuje i metodu `public static void main(String[] args)`, která umožňuje spuštění aplikace „normálním“ způsobem.

Ke komunikaci aplikace s uživatelem dochází přes jednoduché grafické užitelské rozhraní (GUI). Všechny prvky grafického rozhraní jsou soustředěny uvnitř třídy `MainWindow`.

Grafické uživatelské rozhraní se dynamicky mění podle aktuálního stavu aplikace a umožňuje uživateli rozhodovat o chodu aplikace.

Příkladem může být příchozí hovor. Po spuštění aplikace je obrazovka prázdná. Ve chvíli, kdy je zachycena zpráva INVITE, zobrazí se na obrazovce dvě tlačítka: `Accept` a `Decline` (viz obrázek 2.3). Po stisku jednoho z nich tlačítka zmizí a buď se zobrazí tlačítko pro zavěšení (v případě přijmutí hovoru) nebo je opět zobrazena prázdná obrazovka (v případě zamítnutí hovoru).



Obr. 2.3: Ukázka grafického uživatelského rozhraní

2.3.12 InstallWizard

Pro správnou funkci celé aplikace je nutné, aby na uživatelském počítači byl nainstalován Java Media Framework popsané v předcházejícím textu.

V části 2.3.1 byla popsána třída `Controller`, která mimojiné obsahuje metodu `checkJMF()`, která je schopna detekovat, zda je JMF nainstalován nebo ne. Pokud JMF nainstalován není je vytvořena instance třídy `InstallWizard`, která stáhne instalační soubor a spustí instalaci Java Media Framework. To vše v závislosti na platformě používaného PC (jsou různé instalační balíky pro Linux a pro Windows).

3 TESTY A JEJICH VÝSLEDKY

Mezi techniky analýzy a návrhu patří tzv. extrémní programování (XP), které je blíže popsáno v [1]. Jednou z hlavních myšlenek této „filozofie“ je vytvoření testovacích scénářů (tzv. unit testů), které jsou napsány dříve než samotná třída. Těmito testy je pak ověřena všechna požadovaná funkčnost dané třídy.

Během vývoje aplikace jsem nešel až do takového extrému, ale v hlavě jsem si navrhl, jak má třída fungovat a pak ji implementoval. Abych mohl tyto předpoklady ověřit, obsahuje většina tříd metodu `public static void main(String[] args)`, která umožňuje třídu spouštět. V těle této metody je pak vždy umístěn kód, kterým je možné otestovat funkčnost třídy.

Samozřejmostí u podobně rozsáhlých aplikací jsou tzv. ladicí výpisy. Jsou to zprávy, které se vypisují na standardní výstup a informují vývojáře (případně i uživatele) o tom, co právě aplikace vykonává (např. se vypsiují parametry funkcí, návratové hodnoty apod.). Jsou užitečné zejména v případě pádu aplikace, kdy je možné z nich poznat, v jaké části došlo k chybě. Pro normálního uživatele jsou mnohdy spíše na obtíž a proto je možné je vhodnou konfigurací vypnout.

V této kapitole bych se rád zaměřil až na závěrečné testování, které ověřuje funkčnost aplikace jako celku. Vzhledem k povaze vytvářené aplikace nebude prováděna verifikace za pomocí matematického důkazu správnosti programu proti jeho specifikaci.

Testovací scénáře budou popsány vstupními parametry, dále požadavky na prostředí a výsledkem bude splnění nebo nesplnění testované funkce.

3.1 Testování spojení za pomoci SIP serveru ve Windows XP

Požadavky na prostředí

Dva počítače s operačním systémem Windows XP Service Pack 2, zvukovou kartou, mikrofonem, reproduktory, nainstalované Java Runtime Environment (dále jen JRE) a Java Media Framework (dále jen JMF). Funkční SIP server s vytvořenými dvěma účty pro testování.

Vstupní podmínky

Na jednom z PC spusťte aplikaci s parametry prvního účtu (uživatelské jméno a heslo, např. `test1@test.cz` a `heslo`). Na druhém PC spusťte aplikaci s parametry

druhého účtu např. `test2@test.cz` a `heslo`. Použijte ještě třetí parametr označující volaného účastníka (tedy `test1@test.cz`). Obě aplikace se po stisku tlačítka Accept spojí. Otestujte funkčnost spojení a přenášeného hlasu. Hovor ukončete.

Výsledek

Pro testování této funkcionality bylo využito SIP serveru od firmy IceWarp. Test byl vyhodnocen jako úspěšný.

3.2 Testování spojení za pomoci dvou SIP serverů ve Windows XP

Požadavky na prostředí

Dva počítače s operačním systémem Windows XP Service Pack 2, zvukovou kartou, mikrofonem, reproduktory, nainstalované JRE a JMF. Dva funkční SIP servery, na každém z nich bude vytvořen jeden testovací účet. Servery musí patřit do rozdílných domén (např. www.vutbr.cz a www.muni.cz).

Vstupní podmínky

Na jednom z PC spusťte aplikaci s parametry prvního účtu (uživatelské jméno a heslo, např. `test@test1.cz` a `heslo`). Na druhém PC spusťte aplikaci s parametry druhého účtu např. `test@test2.cz` a `heslo`. Použijte ještě třetí parametr označující volaného účastníka (tedy `test@test1.cz`). Obě aplikace se po stisku tlačítka Accept spojí. Otestujte funkčnost spojení a přenášeného hlasu. Hovor ukončete.

Výsledek

Pro testování této funkcionality bylo využito dvou SIP serverů od firmy IceWarp. Test byl vyhodnocen jako úspěšný.

3.3 Testování spouštění z webového prohlížeče ve Windows XP

Požadavky na prostředí

Dva počítače s operačním systémem Windows XP Service Pack 2, zvukovou kartou, mikrofonem, reproduktory, nainstalovaným JRE a JMF. Libovolný webový prohlížeč.

Vstupní podmínky

Spusťte webový prohlížeč a přejděte na stránku s kódem pro spuštění apletu (přiloženo na cd). Applet je digitálně podepsaný, takže budete požádáni o výslovné spuštění appletu. Stiskněte tlačítko Run. Testovací stránka je vytvořena tak, že již obsahuje potřebné parametry. Po načtení aplikace se ve stavovém řádku objeví text např. `You have been succesfuly registered on server expert4me.com`.

Výsledek

Testování probíhalo s prohlížeči Mozilla Firefox, Internet Explorer (verze 6 i 7). Ve všech prohlížečích se aplikace úspěšně spustila. Test byl tedy vyhodnocen jako úspěšný.

3.4 Testování detekce Java Media Framework ve Windows XP

Požadavky na prostředí

Počítač s operačním systémem Windows XP Service Pack 2, zvukovou kartou, mikrofonem, reproduktory, nainstalované JRE. Libovolný webový prohlížeč. Java Media Framework tentokrát nebude nainstalován. Funkční SIP server s vytvořeným účtem pro testování.

Vstupní podmínky

Tento test probíhá taktéž z webového prohlížeče (z důvodu možnosti stažení instalačních balíků pro JMF), postup je tedy ze začátku stejný jako postup pro testování spouštění z webového prohlížeče. Aplikace detekuje, že není nainstalováno JMF a nabídne instalaci. Potvrďte, že chcete JMF nainstalovat. Bude stažen a spuštěn instalační soubor. Projděte procesem instalace, na jeho konci bude vyžádán restart počítače, ten proveďte. Po restartu aplikaci spusťte znova. Tentokrát již aplikace instalaci nenabídne, protože JMF již je nainstalováno.

Výsledek

Test byl vyhodnocen jako úspěšný.

3.5 Testování spojení za pomoci SIP serveru na platformě Linux

Požadavky na prostředí

Dva počítače s operačním systémem Linux, zvukovou kartou, mikrofonem, reproduktory, nainstalovaným JRE a JMF. Funkční SIP server s vytvořenými dvěma účty pro testování.

Vstupní podmínky

Na jednom z PC spusťte aplikaci s parametry prvního účtu (uživatelské jméno a heslo, např. `test1@test.cz` a `heslo`). Na druhém PC spusťte aplikaci s parametry druhého účtu např. `test2@test.cz` a `heslo`. Použijte ještě třetí parametr označující volaného účastníka (tedy `test1@test.cz`). Obě aplikace se po stisku tlačítka Accept spojí. Otestujte funkčnost spojení a přenášeného hlasu. Hovor ukončete.

Výsledek

Test byl vyhodnocen jako neúspěšný.

V tomto případě byla využita 64 bitová verze distribuce Fedora 8. Obě aplikace se spojí (tj. funguje signalizace), ale přenos hlasu není funkční. Příčinou nefunkčnosti přenosu hlasu může být buď chyba v Java Media Framework (tomu by nasvědčovala chybová hlášení `Unable to handle format: ULAW/rtp, 8000.0 Hz, 8-bit, Mono`) a nebo chyba v ovladači zvukové karty, který je součástí distribuce Fedora (tomu zase nahrává chybové hlášení `*** PULSEAUDIO: Unable to connect: Connection refused`).

Další testování na platformě Linux nebylo prováděno jednak z časových důvodů a také proto, že vývoj aplikace byl primárně zaměřen na platformu Windows, kde byla funkčnost ověřena dostatečně. Na platformě Linux šlo pouze o technologický test.

4 ZÁVĚR

Obsahem práce je návrh, realizace a závěrečné testování softwarového řešení SIP klienta. Výsledná implementace byla provedena v jazyce Java, který se ukázal jako velmi vhodný pro tento druh aplikace. Jazyk Java byl navržen tak, aby urychloval vývoj aplikací, což se během realizace několikrát ukázalo jako pravdivé tvrzení. Rychlému vývoji také velice napomohlo zvolené vývojové prostředí s názvem Eclipse.

Obecně mohu říct, že výběr knihoven se ukázal jako správný. Během realizace jsem musel řešit jen dva závažnější problémy, které byly spojeny s výběrem knihoven.

Při přechodu na novější vývojovou verzi knihovny JAIN SIP jsem byl nucen přepsat část kódu pro vytváření odpovědi na SIP zprávy, protože původně použitá metoda byla označena jako deprecated (zavržená, zastaralá) a byla nahrazena metodou jinou. Nová metoda měla odlišné parametry a bylo nutné podrobně prostudovat její dokumentaci, aby nedocházelo k pádům aplikace.

Dalším úskalím se ukázalo být přehrávání přijatého RTP streamu. Po prostudování API k Java Media Framework jsem vytvořil návrh a následně se pokusil přehrávání podle tohoto návrhu implementovat. Bohužel implementace byla nefunkční, takže jsem byl nucen návrh upravit tak, aby původní třída `Receiver` byla definována jako vnitřní, čímž vznikla třída `PlayStream`.

Při pohledu do kapitoly 2.1 je patrné, že kromě přenositelnosti aplikace mezi různými platformami se mi podařilo dosáhnout všech cílů, které jsem si na začátku stanovil. Možnost spouštění aplikace na platformě Linux bude do aplikace doplněna. Do budoucna bude nutné implementovat robustní obsluhu vyjímek, která zatím v aplikaci chybí. Dále by bylo vhodné, aby aplikace umožňovala šifrovaný přenos dat s využitím protokolu TLS. Rozhodně je jasné, že v tomto stavu aplikace není schopna konkurovat již existujícím řešením, ale je to dobrý základ, na kterém je možné stavět.

Další osud aplikace zatím není znám. Firma IceWarp (její SIP server byl použit pro testování) projevila zájem o další spolupráci, ale zatím není jasné, jakým způsobem bude aplikace využita. Uvažuje se o napojení do aplikace Webmail, které by umožňovalo volání přímo z webového prohlížeče bez nutnosti instalace jakéhokoli software.

LITERATURA

[1] ECKEL, B. *Myslíme v jazyku Java : knihovna programátora*. Bogdan Kiszka. 1. vyd. Praha : Grada Publishing, spol. s.r.o., 2001. 432 s. Myslíme v.... ISBN 80-247-9010-6.

[2] ECKEL, B. *Myslíme v jazyku Java : knihovna zkušeného programátora*. Bogdan Kiszka. 1. vyd. Praha : Grada Publishing, spol. s.r.o., 2001. 472 s. Myslíme v.... ISBN 80-247-0027-1.

[3] FRANKS, J., et al. *HTTP Authentication: Basic and Digest Access Authentication* [online]. 1999, [cit. 2008-04-14]. Dostupný z WWW:
<http://tools.ietf.org/html/rfc2617>.

[4] Free Software Foundation, Inc. *GNU General Public License* [online]. c2007, [cit. 2007-11-07]. Dostupný z WWW:
<http://www.gnu.org/licenses/gpl.html>.

[5] Free Software Foundation, Inc. *GNU Lesser General Public License* [online]. c2007, [cit. 2007-11-07]. Dostupný z WWW:
<http://www.gnu.org/copyleft/lesser.html>.

[6] HANDLEY, M., JACOBSON, V., PERKINS, C. *SDP: Session Description Protocol* [online]. 1998, [cit. 2007-11-03]. Dostupný z WWW:
<http://tools.ietf.org/html/rfc4566>.

[7] JOHNSTON, A. B. *SIP : Understanding the Session Initiation Protocol* . 2. vydání, London: Artech House Inc., 2004, 283 s., ISBN 1-58053-655-7

[8] LIBÁK, J. *SIP softphone*. 2007. xvi, 124 s. České vysoké učení technické v Praze. Fakulta elektrotechnická. Vedoucí diplomové práce Ing. Jan Kubr. Dostupný z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/libakj1_2007dipl.pdf>.

[9] O' DOHERTY, P., RANGANATHAN, M. *NIST–SIP: The Reference Implementation for JAIN–SIP 1.2* [online]. [2006], [cit. 2008-04-19]. Dostupný z WWW: <http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/>.

[10] RESNICK, P. *Internet Message Format* [online]. 2001, [cit. 2007-11-03]. Dostupný z WWW: <http://tools.ietf.org/html/rfc2822>.

[11] ROSENBERG, J., et al. *SIP: Session Initiation Protocol* [online]. 2002, [cit. 2007-11-03]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3261>.

[12] ROSENBERG, J., SCHULZRINNE, H. *An Offer/Answer Model with the Session Description Protocol (SDP)* [online]. 2002 [cit. 2008-04-19]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3264>.

[13] SCHULZRINNE, H., CASNER, S., FREDERICK, R., JACOBSON, V., *RTP: A Transport Protocol for Real-Time Applications* [online]. 2003, [cit. 2007-11-03]. Dostupný z WWW: <http://tools.ietf.org/html/rfc3550>.

[14] SPENCER, M., et al. *IAX2: Inter-Asterisk eXchange Version 2* [online]. 2006 [cit. 2008-04-19]. Dostupný z WWW:

<http://www.openiax.org/rfcs/draft-guy-iax-02.txt>.

[15] Sun Microsystems, Inc.. *Java Media Framework API* [online]. c1999-2008 [cit. 2008-04-19]. Dostupný z WWW: <http://java.sun.com/products/java-media/jmf/>.

[16] Wikipedia contributors. *Model-view-controller* [online]. Wikipedia: The Free encyklopedia, [2005], last modified on 19 April 2008 [cit. 2008-04-19]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Model-view-controller>.

[17] ZAKHOUR, S. a kolektiv. *Java 6 výukový kurz* 1. vydání, Brno: Computer Press, 2007, 534 s., ISBN 978-80-251-15775-6

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API Application Programming Interface

BSD Berkeley Software Distribution

CRLF Carriage Return Line Feed

GPL GNU General Public License

HTTP Hypertext Transfer Protocol

IM Instant Message

JAIN Java APIs for Integrated Networks

JDK Java Development Kit

JMF Java Media Framework

JRE Java Runtime Environmemt

JVM Java Virtual Machine

LGPL GNU Lesser General Public License

MVC Model View Controller

NAT Network Address Translation

PSTN Public switched telephone network

QoS Quality of Service

RFC Request For Comments

RTP Real-time Transport Protocol

SDP Session Description Protocol

SIP Session Initiation Protocol

SSL Secure Sockets Layer

TLS Transport Layer Security

UA User Agent

UAC User Agent Client

UAS User Agent Server

UML Unified Modelling Language

URI Uniform Resource Identifier

VoIP Voice over IP

SEZNAM PŘÍLOH

A JAVA SIP TOOLKIT

Java SIP

TOOLKIT

For Developing Java SIP Applications

RADVISION, the leading protocol toolkit vendor, is proud to introduce its new Java SIP toolkit. Based on RADVISION's award-winning SIP toolkit, the new Java SIP toolkit is a powerful and highly versatile set of tools designed to simplify and dramatically reduce development time of Java-based SIP applications, such as *Click to talk, Virtual Call Centers, Web Soft-Phones*, and SIP related *infrastructure based elements*, such as *SIP Proxies* and *SIP Registrars*. Implementing J2SE technology, the high performance Java SIP toolkit is IETF and JCP standards-compliant and provides multiple API layers for full user control and flexibility. The Java SIP toolkit delivers superior performance, performing one scale faster than similar solutions – while never compromising Java's inherent ease-of-use. The Java SIP toolkit is based on RADVISION's industry-leading SIP toolkit, an interoperable, market-proven solution used by hundreds of customers worldwide.

Fully compliant with:

- RFC 3261
- JSR32 v1.2
- RFC 3262

The Java SIP toolkit comes with all the components developers require, including a set of quick start sample applications that demonstrate efficient JSR 32 API usage, a GUI test application and detailed documentation.

SIP – The Emerging Signaling Protocol

The Session Initiation Protocol (SIP) is emerging as the industry choice for real-time communication applications, such as voice and video over IP (V^2oIP), Instant Messaging (IM), and presence. Based on proven Internet protocols, such as SMTP and HTTP, SIP is text-encoded and well-suited for the Internet and other IP environments. SIP provides the mechanisms to implement a broad range of features, including call control, next-generation service creation and interoperability with existing telephony systems. SIP is used as the signaling protocol in the most advanced and emerging architectures, such as IMS, TISPAN and PacketCable.

Java – Enabling Platform-Independent Development

Java is a popular software technology that embodies the concept of *"write once – run all"* – enabling written code to run on any machine or OS with no need to adjust the code to run on a specific platform environment. This is achieved by detaching the hardware processor from the compiled code, using a "virtual machine." The Java programming language is very friendly and easy to develop on, and has become one of the most powerful and useful tools for rapid application development and service deployment.

Java APIs are defined by the JCP (Java Community Process) organization. The JCP's standard for SIP toolkits is the JSR 32 (JAIN SIP API). As a leader in protocol definition and standardization efforts, RADVISION contributed significantly to the development of the latest JSR32 v1.2 standard.

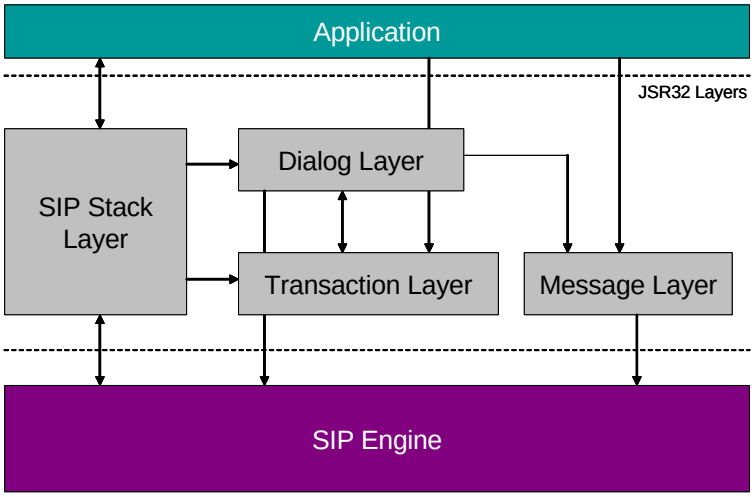
RADVISION
Delivering the Visual Experience

JAVA SIP STACK ARCHITECTURE

The Java SIP Stack architecture follows the standard SIP stack layering model. By implementing the same standard layers, the Java SIP Stack enables superior flexibility as it enables the application layer to approach and manipulate each message.

The Java SIP stack architecture delivers the following advantages:

- Superior performance
- The JAVA SIP Stack inherits all the stability, maturity, scalability, and interoperability of RADVISION's proven legacy SIP Stack.



ARCHITECTURE LAYERS

SIP STACK LAYER – Sets the system configuration, logging, utilities and other resources, such as Providers and ListeningPoints. It is also responsible for the initialization and shutdown of all other layers.

DIALOG LAYER – Creates and manages dialog objects and maps incoming transactions to dialog objects.

TRANSACTION LAYER – Creates and manages transaction objects and maps incoming messages to transaction objects. Each transaction is responsible for maintaining states, and sending and receiving messages and retransmissions using the Transport layer.

SIP ENGINE

The SIP engine delivers superior performance. Whereas basic Java implementations might suffer from reduced performance compared with C-written stacks, RADVISION's Java SIP toolkit overcomes this challenge by using inner C layers, communicated through Java Native Interface (JNI) technology. The C layers are based on RADVISION's industry-leading SIP toolkit, providing outstanding performance and proven stability.

SIP ENGINE COMPONENTS

TRANSMITTER AND TRANSPORT – Responsible for sending and receiving SIP messages. The Transmitter layer performs all address resolution activities based on the message it is about to send. The Transport layer sends and receives messages, handles SIP networking I/O, and manages UDP sockets and TCP connections, as specified in RFC 3261.

MESSAGE AND PARSER – Handles parsing and encoding of SIP messages. Enables browsing and editing of SIP message contents, and comparison of message parts, such as SIP addresses and headers.

USA/Americas
Tel +1 201.689.6300
Fax +1 201.689.6301
infoUSA@radvision.com

APAC
Tel +852.3472.4388
Fax +852.2801.4071
infoAPAC@radvision.com

EMEA
Tel +44 (0) 20 8757 8817
Fax +44 (0) 20 8757 8818
infoUK@radvision.com

Product specifications subject to change without notice. This document is not part of a contract or license as may be expressly agreed.
RADVISION is a registered trademark of RADVISION, Ltd.. All trademarks recognized.

RADVISION
Delivering the Visual Experience

B OBSAH CD