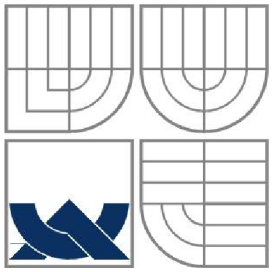
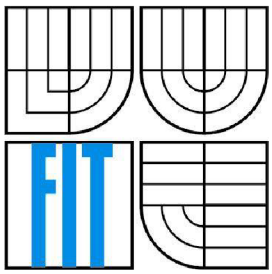


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

JABBER/XMPP ROBOT PRO VYHLEDÁVÁNÍ POMOCÍ GOOGLE

JABBER/XMPP ROBOT FOR SEARCHING BY MEANS OF GOOGLE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Jiří Hrazdil
AUTHOR

VEDOUCÍ PRÁCE — Mgr. Marek Rychlý
SUPERVISOR

BRNO 2007

Jabber/XMPP robot pro vyhledávání pomocí Google

Zadání

1. Seznamte se s protokolem Jabber/XMPP a jeho open-source implementacemi.
2. Seznamte se se službami vyhledávače Google a s rozhraním webové služby Google (Google Web APIs service).
3. Návrhněte vhodný formát dotazu (strukturovaný prostý text), který pokryje funkce a služby poskytované vyhledávačem Google.
4. Navrhněte a implementujte robota pro Jabber, který bude odpovídat na dotazy podle výsledků hledání pomocí služby Google. Robot by měl umožnit přizpůsobit své uživatelské rozhraní podle požadavků uživatele.
5. Diskutujte výsledky práce a navrhněte možná rozšíření.

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Cílem této práce bylo seznámit se s principem a fungováním komunikační sítě Jabber/XMPP a jeho open-source implementacemi, dále pak seznámit se s rozhraním pro vyhledávání webových stránek pomocí rozhraní webové služby vyhledávače Google a implementovat robota, který na dotazy položené prostřednictvím protokolu XMPP odpoví výsledky získanými z webového vyhledávače pomocí protokolu SOAP.

Klíčová slova

Jabber, XMPP, robot, Google, Yahoo, MSN Live, SOAP, vyhledávání na webu.

Abstract

The aim of this thesis was to get acquainted with principles and basics of Jabber/XMPP communication network and its open-source implementations, learn about Google web search application interface and implement a robot, which will respond to search queries sent via XMPP protocol with adequate search results acquired through SOAP.

Keywords

Jabber, XMPP, robot, Google, Yahoo, MSN Live, SOAP, web search.

Citace

Jiří Hrazdil: Jabber/XMPP robot pro vyhledávání pomocí Google, bakalářská práce, Brno, FIT VUT v Brně, 2007

Jabber/XMPP robot pro vyhledávání pomocí Google

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Marka Rychlého.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Hrazdil
2007-05-10

Poděkování

Rád bych poděkoval svému vedoucímu Mgr. Markovi Rychlému za vstřícný přístup a hodnotné připomínky.

Obsah

Seznam příkladů

Seznam obrázků

Úvod

Téma Jabber/XMPP robot pro vyhledávání na Googlu jsem si vybral, protože jsem se chtěl dozvědět více o pozadí protokolu XMPP, a také kvůli dřívějším zkušenostem s vyhledávacím rozhraním od firmy Google. Cílem práce je vytvořit aplikaci, která bude zároveň fungovat jako klient XMPP protokolu i protokolu SOAP. Smyslem práce je poukázat na možnosti provázání protokolu XMPP s moderním využitím sítě Internet – webovými službami.

V první kapitole nejdříve seznámím čtenáře s obsahem pojmů použitých v této práci, s použitými protokoly, se sítí Jabber/XMPP, její historií, jejím fungováním. Dále si shrneme požadavky na funkčnost robota a nastíníme možnosti jejich realizace.

Ve druhé kapitole se seznámíme s programovými prostředky použitými při vývoji robota – s vývojovým prostředím, podpůrnými knihovnami, ale i servery a klienty sítě Jabber.

Ve třetí kapitole se budeme zabývat implementací robota – jeho návrhem, popisem jednotlivých jeho součástí, rozebereme si služby poskytované vyhledávači.

Na závěr zhodnotíme posoudíme dosažené výsledky, shrneme přínosy bakalářské práce a nastíníme možnosti dalšího rozvoje projektu.

1 Teorie

1.1 Použité pojmy

V této kapitole se seznámíme s pojmy a protokoly použitými v bakalářské práci.

1.1.1 Instant messaging (IM)

Instant messaging je fenoménem posledních několika let. Umožňuje dvěma nebo i více lidem komunikovat v reálném čase. Právě jeho rychlost a interaktivita jej odlišuje od e-mailové komunikace. Neméně důležitá je také možnost zjistit, zda je uživatel přítomen na druhém konci drátu nebo ne.

Za jeden z prvních instant messaging systémů lze považovat, pomineme-li možnost zasílání zpráv ostatním přihlášeným uživatelům na UNIXových systémech, IRC (internet relay chat), který vznikl v roce 1988. Jedná se o centralizovaný systém, kdy se koncoví klienti připojují k serveru, který jim umožňuje komunikovat s ostatními klienty připojenými ke stejnému serveru. Servery je možné spojovat do sítí, čímž získáme jednak odolnosti proti výpadku (odpojení jednoho serveru má za následek většinou pouze odpojení uživatelů daného serveru, přičemž ve zbytku sítě je možno v komunikaci dále pokračovat), jednak to umožní komunikaci mezi uživateli připojenými nejen ke stejnému serveru, ale i k ostatním serverům sdruženým v síti. Základ protokolu IRC je sice standardizovaný v RFC 1459 [1], avšak mnoho vývojářů si do serverů přidávalo další vlastnosti, což se projevilo v nekompatibilitě na úrovni serverů – většinou není možné spojit dva servery, které běží na odlišných implementacích IRC démonů. Klientů se toto omezení netýká, naprostá většina klientů je schopna se připojit k libovolnému IRC serveru. IRC protokol je čistě textový, což má jak výhody, tak samozřejmě i nevýhody. Za výhodu můžeme považovat jednoduchou implementaci klienta, nevýhodou je naopak malá odolnost vůči odposlechu, protože data putují v nezašifrované podobě (pokud není použito SSL šifrování, které se však v praxi zatím moc neprosadilo).

Komunikace přes IRC probíhá převážně na tzv. kanálech, kde spolu může hovořit mnoho lidí současně (konverzace typu many-to-many). Protokol však umožňuje, aby komunikace probíhala i nerušeně pouze mezi dvěma uživateli (query, one-to-one). Každý uživatel je v rámci jedné IRC sítě identifikován svou přezdívkou, pod kterou vystupuje. Protokol IRC sám o sobě nepodporuje registraci přezdívek, platí princip kdo první přijde, dostane přezdívku.

V České republice nejpoužívanější IM systém spatřil světlo světa v listopadu 1996, kdy izraelská společnost Mirabilis vydala první verzi klienta ICQ. Pro komunikaci se používá protokol OSCAR (Open System for Communication in RealTime), který je, ač to podle názvu nevypadá, proprietárním protokolem vyvinutým firmou AOL. Uživatelé sítě ICQ se musí před prvním přihlášení

do sítě zaregistrovat, je jim přidělen jedinečný číselný identifikátor (UIN), který slouží jako přihlašovací jméno. Oproti IRC tedy máme relativní jistotu, že když komunikujeme s někým s UIN 123-456-789, bude se jednat o stejnou osobu jako před měsícem. Protokol ICQ podporuje mimo jiné i přenosy souborů a uživatelské obrázky (avatary).

Dalším používaným IM systémem je MSN Messenger (později Windows Live Messenger) od firmy Microsoft, který je taktéž založen na proprietárním protokolu – MSNP (Microsoft Notification Protocol) [2]. První verze byla vydána v polovině roku 1999, od té doby probíhá neustálý vývoj a přidávání nových vlastností. Uživatelským jménem, používaným při přihlášení ke službě, je e-mailová adresa.

1.1.2 XML

XML (eXtensible Markup Language – rozšiřitelný značkovací jazyk) [3] je značkovací jazyk, který byl vytvořen pro zjednodušení výměny dat. Je odlehčenou verzí standardu SGML (Standard Generalized Markup Language). Tvůrcem specifikace je W3C (World Wide Web Consortium), která vydala doporučení XML ve verzi 1.0 v únoru 1998. Specifikace obsahuje jak syntaktické požadavky jazyka, tak i požadavky na parsery. Každý XML dokument je složen ze značek – párových (mají otevírací i ukončující značku) a/nebo nepárových (pouze jedna značka, která slouží zároveň jako otevírací i uzavírací). Každý dokument taktéž musí obsahovat právě jeden kořenový prvek, který tvoří „obálku“ pro obsah dokumentu.

Z hlediska zpracování rozlišujeme 2 typy dokumentů – správně sestavené a validní. Správně sestavený dokument vyhovuje všem syntaktickým pravidlům stanoveným ve specifikaci (tato pravidla jsou uvedena dále). Nesprávně sestavené dokumenty musí být parserem odmítnuty.

Validní dokument je správně sestavený dokument, který splňuje další sémantická pravidla, která mohou být stanovena například v definici typu dokumentu (DTD) nebo XML schématu (XML Schema). Dokument tedy může být správně sestavený, ale když bude obsahovat neznámou značku (z pohledu DTD nebo XML schématu), není validní.

Požadavky na správnou sestavenost dokumentu jsou následující:

- neprázdné elementy musí mít otevírací i uzavírací značku (např. `<p></p>`)
- prázdné elementy mohou být zapsány zkráceným zápisem (`<br />` je syntakticky ekvivalentní `<br></br>`)
- hodnoty atributů musí být uzavřeny v uvozovkách nebo apostrofech
- značky do sebe mohou být vnořeny, avšak nesmí se křížit; každý nekořenový prvek musí být uzavřen do jiného prvku
- dokument musí obsahovat pouze znaky určené znakové sady (ta může být určena externě – například v HTTP hlavičce Content-type, nebo v XML deklaraci na začátku dokumentu)

- záleží na velikosti písmen v názvech značek (nelze zapsat `<P>…</p>`)

Příklad – Správně sestavený XML dokument

```
<?xml version="1.0" encoding="utf-8"?>
<html>
<body>
  <p>Text v odstavci může být <b>tučný</b></p>
</body>
</html>
```

Příklad - Nesprávně sestavený dokument (překřížené značky)

```
<?xml version="1.0" encoding="utf-8"?>
<html>
<body>
  <p>Text v odstavci může být <b>tučný</p></b>
</body>
</html>
```

U první ukázky nemůžeme rozhodnout o validitě, neboť kód sám o sobě nenese žádnou definici sémantiky. Doplníme-li definici typu dokumentu (DTD), získáme z první ukázky XML dokument, který ač je správně sestavený, není validní proti specifikaci XHTML 1.0 (chybějící značka `<head>`).

Příklad – Správně sestavený nevalidní dokument (XHTML 1.0)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<body>
  <p>Text v odstavci může být <b>tučný</b></p>
</body>
</html>
```

Velkou předností XML je možnost umístit do jednoho XML souboru více sémanticky odlišných typů dat (např. do XHTML souboru můžeme vložit část v MathML – značkovacím jazyce používaném k zápisu matematických vzorců). Přiřazení správné sémantiky k jednotlivým částem kódu probíhá pomocí jmenných prostorů.

1.1.3 SOAP

SOAP (Simple Object Access Protocol - protokol pro jednoduchý přístup k objektům) [4] je protokol, jež slouží k přenosu zpráv mezi dvěma aplikacemi. Pro přenos SOAP požadavků se může používat jak HTTP, tak i SMTP, nicméně v naprosté většině případů se pro přenos používá HTTP. Za nevýhodu můžeme považovat použití XML zpráv, neboť oproti binárním zprávám (u systémů jako CORBA nebo DCOM) dochází k větší režii při zpracování, a také, není-li použita komprese, k přenosu většího množství dat.

Historie protokolu SOAP sahá do roku 1999, kdy byla jako výsledek spolupráce firem DevelopMentor, Microsoft a UserLand vydána specifikace SOAP 1.0. Cílem jejich úsilí bylo vytvořit protokol, který by umožňoval vzdálené volání procedur (RPC) pomocí požadavků v jazyce XML. Od roku 1998 sice již existovala specifikace protokolu XML-RPC, ten se ale ukázal jako příliš jednoduchý a neflexibilní. Aktuální verzí SOAP protokolu je 1.2, vydaná jako doporučení W3C v červnu 2003.

Typická SOAP komunikace probíhá tak, že klient odešle serveru požadavek, který server zpracuje, provede a klientovi zašle zpět výsledek (odpověď). Jak jsem již uvedl, komunikace mezi klientem a serverem využívá protokolu HTTP. Proč byl zvolen zrovna tento protokol? Hlavní jeho výhodou je jeho všeobecná dostupnost, pro zprovoznění webové služby přístupné přes SOAP není třeba žádných zásahů do síťové infrastruktury (jako je např. povolení na firewallu). Pro zasílání požadavků se používá metoda buď metoda GET nebo POST protokolu HTTP. Metoda GET spočívá v zaslání požadavku, který obsahuje URI služby, většinou doplněné o jedinečný identifikátor. (Ve specifikaci SOAP je uveden příklad s rezervací letenek, kdy pomocí GET požadavku získáme SOAP odpověď obsahující podrobné informace o provedené rezervaci.)

Teprve požadavky typu POST nám umožní plné využití protokolu SOAP (na rozdíl od metody GET nám umožní i zasílání SOAP požadavku). Hlavičku POST požadavku tvoří informace o URI volané služby, informace o použitém kódování a délce těla SOAP požadavku. Při přenosu pomocí protokolu HTTP musí požadavek obsahovat hlavičku `SOAPAction`, která jednak jednoznačně identifikuje, že jde o SOAP požadavek (na což mohou reagovat např. firewally), jednak můžeme pomocí tohoto parametru specifikovat URI volané webové služby. Tělo požadavku POST tvoří XML se SOAP požadavkem. Formát odpovědi je stejný, tedy hlavička obsahující stavový kód HTTP (zda požadavek proběhl bez problémů nebo došlo k chybě), informaci o MIME typu odpovědi (pro SOAP se používá `application/soap+xml` nebo `text/html`), délce odpovědi a v těle odpověď obsahuje SOAP zprávu s odpovědí.

Požadavek ve formátu SOAP je XML dokument, který je tvořen kořenovou značkou `Envelope`, do níž jsou uzavřeny dva elementy – `Header` a `Body`. Značka `Header` je nepovinná, může obsahovat např. identifikaci uživatele, přístupové informace (jméno, heslo) apod. Ve značce Body je potom samotný obsah SOAP požadavku nebo SOAP odpovědi, tzn. třeba jméno volané funkce a její parametry. Informace, které má požadavek obsahovat, je možné vyjádřit třeba pomocí WSDL popisu (viz dále).

Příklad - Odchozí SOAP požadavek pro vyhledání fráze na Googlu (kompletní HTTP paket)

```
1: Host: api.google.com
2: User-agent: SOAPpy 0.12.0 (pywebsvcs.sf.net)
3: Content-type: text/xml; charset="UTF-8"
4: Content-length: 1085
5: SOAPAction: "urn:GoogleSearchAction"
```

```
6: <SOAP-ENV:Envelope
7:  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
8:  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
9:  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
10:  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
11:  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
12:  <SOAP-ENV:Body>
13:    <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch" SOAP-ENC:root="1">
14:      <v1 href="#i1"/>
15:      <v2 href="#i2"/>
16:      <v3 xsi:type="xsd:int">0</v3>
17:      <v4 xsi:type="xsd:int">3</v4>
18:      <v5 xsi:type="xsd:boolean">False</v5>
19:      <v6 href="#i3"/>
20:      <v7 xsi:type="xsd:boolean">False</v7>
21:      <v8 href="#i4"/>
22:      <v9 href="#i5"/>
23:      <v10 href="#i5"/>
24:   </ns1:doGoogleSearch>
25:   <v1 xsi:type="xsd:string" id="i1" SOAP-ENC:root="0">
Ef9FKWlQFHI6pbYKWeBwTXM2PdstQB+H</v1>
26:   <v2 xsi:type="xsd:string" id="i2" SOAP-ENC:root="0">normy</v2>
27:   <v6 xsi:type="xsd:string" id="i3" SOAP-ENC:root="0">countryCZ</v6>
28:   <v8 xsi:type="xsd:string" id="i4" SOAP-ENC:root="0">lang_cs</v8>
29:   <v9 xsi:type="xsd:string" id="i5" SOAP-ENC:root="0">utf-8</v9>
30: </SOAP-ENV:Body>
31: </SOAP-ENV:Envelope>
```

Na výše uvedeném paketu vidíme na prvních pěti řádcích hlavičku HTTP POST požadavku, řádky 6 až 31 obsahují samotný SOAP požadavek. Jak je vidět, neobsahuje nepovinnou hlavičku. V těle vidíme volání funkce doGoogleSearch, a to s několika parametry. U řetězcových parametrů je uveden odkaz na hodnoty, které se vyskytují až pod voláním funkce, konkrétně jde o parametry API klíče, hledaného výrazu, omezení země a jazyka a znakovou sadu, ve které je hledaný výraz zaslán.

Na tento požadavek vyhledávací server Googlu odpoví následovně (z výpisu byly pro zpřehlednění vynechány položky, které robot nevyužívá):

Příklad – SOAP odpověď

```
1: HTTP/1.0 200 OK
2: Content-Type: text/xml; charset=utf-8
3: Content-Length: 2258
4: Cache-control: private
5: Date: Tue, 08 May 2007 20:02:26 GMT
6: Server: GFE/1.3
7: Connection: Keep-Alive
```

```
8: <?xml version='1.0' encoding='UTF-8'?>
9: <SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
10: <SOAP-ENV:Body>
11: <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
12:   <return xsi:type="ns1:GoogleSearchResult">
13:    <endIndex xsi:type="xsd:int">1</endIndex>
14:    <resultElements
xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns3:Array"
ns3:arrayType="ns1:ResultElement[1]">
15:      <item xsi:type="ns1:ResultElement">
16:        <URL xsi:type="xsd:string">http://www.normy.biz</URL>
17:        <snippet xsi:type="xsd:string">Technické normy - Ing. Jiří
Hrazdil. Prodáváme technické normy...</snippet>
18:        <title xsi:type="xsd:string">NORMY.biz</title>
19:      </item>
20:    </resultElements>
21:  <searchQuery xsi:type="xsd:string">normy</searchQuery>
22:  <searchTime xsi:type="xsd:double">0.019675</searchTime>
23:  <startIndex xsi:type="xsd:int">1</startIndex>
24:  </return>
25: </ns1:doGoogleSearchResponse>
26: </SOAP-ENV:Body>
27: </SOAP-ENV:Envelope>
```

HTTP hlavičky víceméně odpovídají původnímu požadavku. V těle odpovědi vidíme, že nám vyhledávání vrátilo jeden výsledek, který je reprezentován položkou `<item>` (řádky 15 až 19). Jednotlivé výsledky jsou vnořeny do „pole“ `resultElements`. Vrácená SOAP zpráva taktéž obsahuje informace o tom, jaký řetězec jsme vlastně hledali, a jak dlouho hledání trvalo.

1.1.4 WSDL

Protokol WSDL [5] je taktéž založen na XML. Jeho úkolem je poskytovat popis rozhraní pro webové služby. Obsahuje jak popis funkcí, které webová služba poskytuje, tak i specifikaci formátu dat – parametrů. WSDL je nezávislé na formátech zpráv a komunikačních protokolech, avšak nejčastěji se používá pro popis zpráv protokolu SOAP. Současnou verzí je 2.0, která se zřejmě stane specifikací W3C, nyní je ve stádiu pracovního návrhu.

Příklad – Běžná struktura WSDL souboru

```
<definitions>
  <types>
    definice typů
  </types>
  <message>
     definice zpráv používaných při komunikaci s webovou službou
  </message>
  <portType>
     definice portů (služeb poskytovaných webovou službou)
  </portType>
  <binding>
     definice komunikačních protokolů používaných webovou službou
  </binding>
</definitions>
```

Definice typů, používaných v dané webové službě, je kvůli zachování nezávislosti na platformě popsána v jazyce XML Schema.

Zprávy používané při komunikaci s webovou službou si můžeme představit jako obdobu parametrů, které předáváme volané proceduře, v běžných programovacích jazycích. Struktura zprávy se opírá o datové typy, definované v předchozím bodu.

Definice portů je nejdůležitější součástí WSDL dokumentu. Popisuje funkce, které webová služba poskytuje, a také zprávy (parametry), které přijímá nebo vrací. Protějškem v běžných programovacích jazycích jsou knihovny funkcí nebo třídy. WSDL definuje čtyři situace, které mohou při komunikaci s funkcí webové služby nastat. První je jednosměrná komunikace, kdy služba obdrží zprávu, na kterou však neodešle odpověď (v praxi třeba zavolání funkce, která nastaví nějakou hodnotu). Následuje nejběžnější typ požadavků, kdy funkce obdrží nějakou zprávu a na tuto odpoví (např. požadavek na vyhledání nějaké hodnoty a vrácení výsledku). Poslední dva typy komunikace se zabývají situacemi, kdy služba sama o sobě zašle požadavek a buď očekává odpověď nebo ne.

Poslední část definic, element binding, slouží k navázání definovaných portů na konkrétní přenosový protokol.

1.1.5 XMPP

1.1.5.1 Historie protokolu

S myšlenkou na vytvoření otevřeného komunikačního protokolu přišel v roce 1998 Jeremie Miller. V průběhu následujícího roku došlo k rychlému vývoji, a to jak jabberového serveru (jabberd), tak i několika klientů. Taktéž byly specifikovány základy protokolu pro výměnu zpráv a zjišťování stavu klienta. První verze jabberd byla vydána v květnu roku 2000. O měsíc později zaslal Miller s kolegy pracovní skupině IMPP návrh jabberových protokolů. Platnost tohoto prvního návrhu však vypršela, aniž by došlo k nějakému pokroku na standardizačním poli. V srpnu 2001 byla založena JSF (Jabber Software Foundation). Jejím cílem bylo zastřešit jak komerční, tak i nekomerční projekty využívající technologie Jabber, vyvíjet a přijímat rozšíření protokolu.

Na přelomu let 2001 a 2002 došlo k dalšímu pokusu o standardizaci protokolu. IETF byl pod záštitou JSF zaslán další návrh, na jehož základě se začalo uvažovat o vytvoření pracovní skupiny XMPP. Ta byla vytvořena v listopadu 2002. V roce 2003 dokončila XMPP WG práce na formalizaci Jabber protokolu, který od té doby odpovídal požadavkům IETF na IM protokoly (definované v RFC 2779). Provedené změny se zaměřovaly hlavně na zvýšení bezpečnosti (SASL, TLS) a podporu internacionalizace. Činnost pracovní skupiny byla formálně ukončena ke konci roku 2004, kdy došlo k vydání internetových standardů RFC 3920 [6], 3921 [7], 3922 a 3923. Další vývoj protokolu již probíhá pomocí rozšíření, které má na starosti XSF (XMPP Software Foundation – nástupce JSF).

1.1.5.2 Vlastnosti protokolu

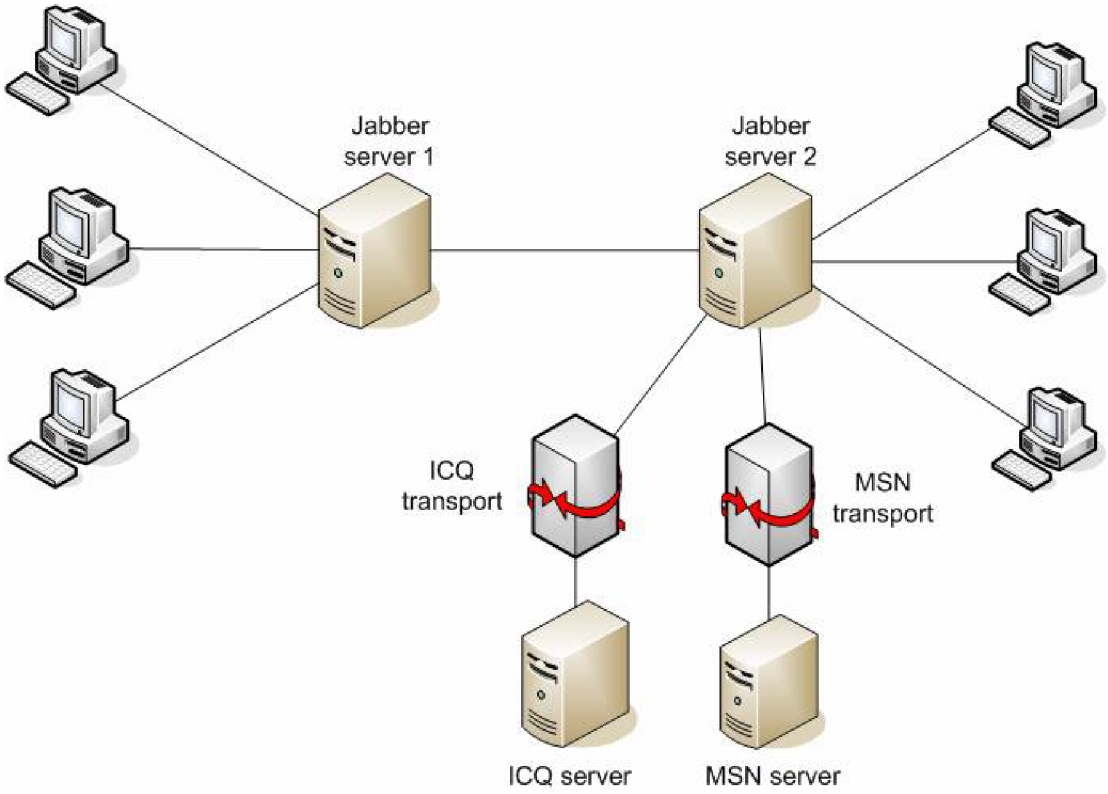
XMPP je otevřený protokol. Znamená to, že je jeho kompletní specifikace volně dostupná komukoliv, a to bez jakýchkoliv poplatků. Otevřenost umožnila vznik mnoha implementací klientů i serverů. Výhodou pro koncové uživatele je možnost volby, a to jak na úrovni výběru použitého programu, tak i na úrovni jabber serveru. V současné době na poli IM převažují uzavřené protokoly – MSN, ICQ, Skype. Uživatelé těchto sítí pak musí využívat jediného klienta, dodávaného provozovatelem IM sítě, který nemusí obsahovat všechny vlastnosti, požadované uživateli, popřípadě nemusí podporovat operační systém, používaný uživatelem.

Další vlastností protokolu XMPP je jeho decentralizace. Provoz v síti není řízen jedním centrálním serverem. Každý uživatel má vytvořen účet na „svém“ serveru a s ostatními klienty komunikuje skrze tento server. Výpadek jednoho serveru neohrozí funkčnost celé sítě, dojde pouze k omezení uživatelů, kteří jsou na tomto serveru registrováni, a nedostupnosti služeb, které tento server poskytuje.

Na obrázku 1 si můžeme prohlédnout schéma XMPP sítě. Klienti připojení k serveru 1 mohou komunikovat jak mezi sebou, tak i s klienty připojenými k serveru 2. Server 2 podporuje 2 transporty (ICQ a MSN), a klienti k němu připojení tedy mohou komunikovat i s klienty těchto dvou sítí.

Nyní se dostáváme k bezpečnosti protokolu XMPP. Ve specifikaci je nastíněno několik možností zabezpečení. Začněme oblíbeným srovnáním s e-mailovými zprávami. Chceme-li poslat e-mail s podvrženou adresou odesílatele, není to žádný problém. V síti Jabber ověřuje server při přijetí klienta jeho autentizační údaje a veškerá další komunikace probíhá v tomto ověřeném sezení. Zaslat zprávu s cizí identitou tedy není možné. Během komunikace jsou používány dva šifrovací protokoly. Prvním je SASL (Simple Authentication and Security Layer), který se používá při navazování spojení mezi klientem a serverem a v průběhu autentifikace klienta. Po vytvoření spojení je veškerá následující konverzace šifrována pomocí TLS (Transport Layer Security). Použití šifrování je volitelné, avšak lze jej vynutit konfigurací serveru.

Obrázek 1 – Schéma XMPP sítě



Každý uživatel sítě Jabber je jednoznačně identifikován svým Jabber ID (JID). Tento identifikátor se podobá e-mailové adrese. V první části je uživatelské jméno, následuje znak zavináče a doménové jméno serveru, na němž je účet zaregistrován. Poslední, nepovinnou, částí je lomítkem oddělené označení zdroje (klienta), přes který uživatel přistupuje k serveru. To umožňuje jednomu uživateli být připojen ke stejnému serveru zároveň pomocí více klientů. JID robota, tvořeného v rámci bakalářské práce, je: `xmpp@jabbim.cz`

Další užitečnou vlastností XMPP protokolu je možnost propojení se sítěmi, které komunikují pomocí jiného protokolu. Tuto vlastnost realizují tzv. transporty (brány), které běží na serveru a připojeným klientům zprostředkovávají připojení jak do ostatních komunikačních sítí (třeba již zmíněného MSN nebo ICQ), ale i k jiným službám – např. SMTP transport umožňuje uživateli z Jabber klienta odesílat e-maily. Pro dosažení požadované funkčnosti není třeba na straně klienta žádných zvláštních rozšíření ani úprav, protože transporty fungují zcela transparentně.

1.1.5.3 Rozšíření

Rozšíření – XEP (XMPP Extension Protocol) – jsou základním prvkem pro doplňování nových vlastností do protokolu XMPP. Nad tvorbou, formalizací a standardizací nových rozšíření bdí XSF. Postup schvalování nového rozšíření dokumentuje XEP-0001 [8].

Pozornost si zaslouží následující rozšíření:
XEP-0004 Data Forms [9]
Data Forms umožňují klientovi zobrazit formulář, který může uživatel vyplnit a odeslat zpět. Lze jej využít třeba pro konfiguraci nějaké služby.

XEP-0030 Service Discovery [10]
Rozšíření Service Discovery popisuje možnosti zjišťování informací o entitě v síti Jabber, konkrétně funkcí, které poskytuje, a protokolů, které podporuje.

XEP-0045 Multi-User Chat [11]
Toto rozšíření umožňuje společnou komunikaci více uživatelům zároveň, stejně jako je tomu v komunikační síti IRC.

XEP-0071 XHTML IM [12]
Klientům podporujícím toto rozšíření je možno zasílat zprávy ve formátu XHTML. Tímto způsobem lze text různě strukturovat, zvýrazňovat a obarvovat. XHTML IM je podmnožinou XHTML, která obsahuje pouze základní značky a základní podporu stylování.

XEP-0096 File transfer [13]
Rozšíření umožňující spolehlivý přenos souborů mezi klienty. Řeší některé problémy, které má dříve používaný protokol Out-of-Band Data. Dodává k souborům metadata (jméno souboru, velikost, popis, datum poslední změny souboru a také hash souboru), umožňuje přenos souborů i klientům, kteří jsou skryti za firewallem. Nepovinnou vlastností je také možnost přenosu jen části souboru.

1.1.5.4 XML streamy

Jak jsme si řekli dříve, komunikace pomocí protokolu XMPP probíhá výměnou XML zpráv. Představme si nyní modelovou situaci, kdy se klient připojí k serveru, zašle mu požadavek a server odpoví zpět. Celá komunikace probíhá v následujících krocích:

1) klient otevře TCP spojení na server
2) klient zašle zprávu serveru ve formátu XML
3) server zpracuje zprávu a vrátí odpověď v XML

4) dojde k ukončení TCP spojení

Takto v dávných dobách fungoval i protokol HTTP. Později do něj bylo implementováno rozšíření, které umožňovalo přes jedno otevřené spojení zaslat několik požadavků a odpovědí, což vedlo ke snížení režie vytváření spojení.

Protokol XMPP toto chování podporuje v podobě proudů XML (XML streams). Strana, která iniciuje spojení, zašle otevírací značku `<stream>`. Dále následuje samotná komunikace (XML značky sloužící k nastavení parametrů spojení, autentizace klienta apod. a XML stanzy). Jakmile jedna ze stran tento XML proud uzavře (pošle ukončovací značku `</stream>`) nebo dojde k chybě v podobě zaslání nesprávně sestaveného XML, je spojení ukončeno. XML proud slouží pro přenos v rámci jednoho směru. Pro oboustrannou komunikaci musí být tedy otevřeny proudy 2 – v každém směru jeden.

V tuto chvíli se naskýtá jedna logická otázka. Jakým způsobem klient zpracovává XML proud, když tento ještě není uzavřen? Víme, že každý XML dokument musí obsahovat jeden kořenový element, který uzavírá obsah XML souboru, jinak se nejedná o správně sestavený XML dokument, a musí být tedy parserem odmítnut. Nelze tedy použít standardní XML parser. Vzhledem k tomu, že obsahem streamu mohou být pouze níže definované stanzy, je nutno využít upravený parser, který bude XML stream dělit na jednotlivé stanzy, a ty potom dále zpracovávat.

1.1.5.5 XML zprávy (stanzy)

Protokol XMPP definuje tři typy XML zpráv, které se používají pro komunikaci mezi entitami v XMPP síti. Všechny stanzy jsou z pohledu struktury XML proudu přímými potomky značky `<stream>`. Samy o sobě mohou sloužit jako obálka pro další XML obsah.

Definovanými typy zpráv jsou:

message – textová zpráva

presence – informace o dostupnosti

iq – dotaz/odpověď

Textové zprávy mohou být čtyř typů: jednoduchá zpráva, konverzace, skupinová konverzace a nadpis. Jednoduchá zpráva a konverzace jsou určeny pro komunikaci mezi dvěma klienty. Skupinová konverzace je obdobou kanálů z chatů typu IRC, kdy spolu může hovořit více osob zároveň. Nadpis je jednoduchá zpráva, zpravidla automaticky rozesílaná, na niž se neočekává odpověď (například burzovní informace, RSS feed apod.) V závislosti na typu zprávy by měl klient umět přizpůsobit své rozhraní.

Informace o dostupnosti je realizována XML stanzou presence. Ta obsahuje volitelný parametr „type“. Tento parametr slouží k předání informace o tom, že entita není dostupná, žádosti o „autorizaci“ (jak ji známe z jiných IM systémů – tzn. entita požádá jinou entitu o povolení ke

sdělování její přítomnosti), zrušení „autorizace“, zjištění stavu entity ze strany serveru a nebo chyba, která se vztahuje k předchozí presence stanze. Není-li tento parametr přítomen, považujeme entitu za připojenou (online). Presence stanza v sobě může obsahovat některou ze tří značek `<show/>`, `<status/>` a `<priority/>`. Značka `<show>` je nepovinná, její obsah může nabývat jedné z hodnot: away, chat, dnd, xa, které označují stavy klienta. Jestliže tato značka není v presence stanze přítomna, považuje ze stav klienta za on-line. Další značkou je `<status>`, která může obsahovat textové upřesnění stavu (např. „jsem na obědě“), sděleného pomocí značky `<show>`. Poslední značkou je `<priority>`, která obsahuje prioritu zdroje (klienta). Na základě hodnoty obsažené v této značce potom server provádí doručování zpráv.

Třetím typem stanzy je iq (info/query – informace/dotaz). Tato stanza umožňuje entitám provádět dotazy a dostávat na ně odpovědi. Obsah požadavku a odpovědi je jednoznačně určen jmenným prostorem, specifikovaným v atributu xmlns. Návaznosti odpovědi na požadavek je dosaženo pomocí atributu id, který musí být u odpovědi stejný jako u požadavku. Rozlišujeme 4 typy iq stanz – get (žádost o hodnotu), set (nastavení hodnoty), result (odpověď na požadavek get/set) a error (došlo k chybě při zpracování předchozího get/set požadavku).

Příklad – Textová zpráva – požadavek na vyhledání fráze „FIT VUT“

```
<message type="chat" to="xmpp@jabbim.cz/bot" id="aac9a">
<body>FIT VUT</body>
<active xmlns="http://jabber.org/protocol/chatstates"/>
<nick xmlns="http://jabber.org/protocol/nick">nightfish</nick>
</message>
```

Příklad – Informace o stavu klienta

```
<presence from="hrazda@njs.netlab.cz/Miranda" xml:lang="en"
to="hrazda@njs.netlab.cz/Psi">
<priority>0</priority>
<show>dnd</show>
<status>Pracuji na bakalářské práci</status>
</presence>
```

Příklad – Iq stanza pro stažení seznamu kontaktů

```
<iq type="get" id="aac3a" >
<query xmlns="jabber:iq:roster"/>
</iq>
```

1.1.5.6 Ukázka XMPP komunikace

Příklad – XMPP komunikace mezi klientem (k) a serverem (s)

```
k: <?xml version='1.0'?>
   <stream:stream
       to='example.com'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       version='1.0'>
```

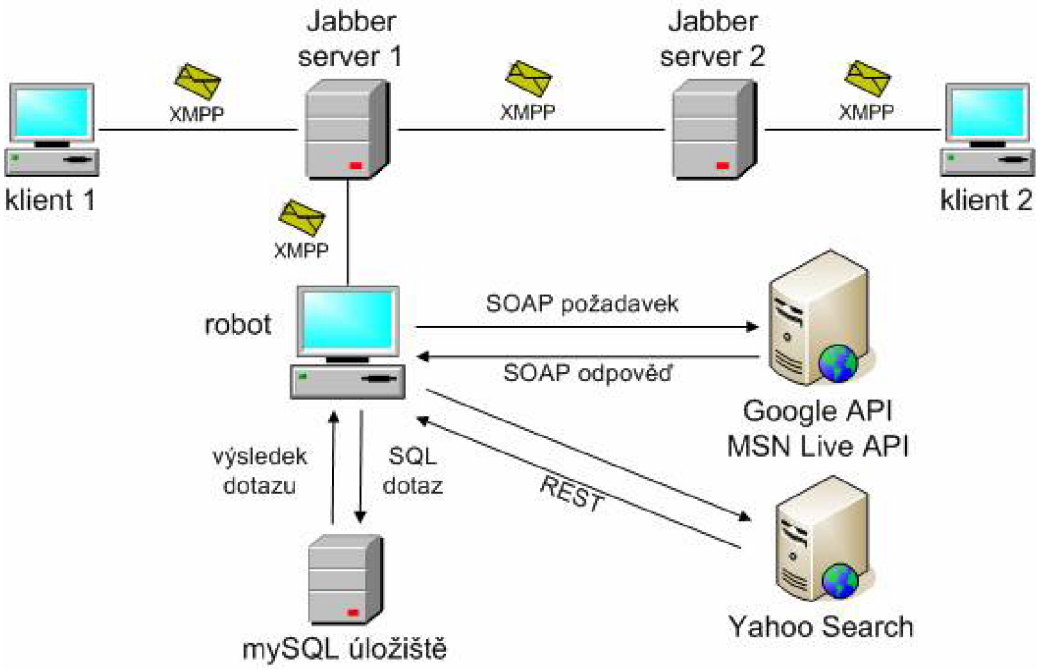
```
s: <?xml version='1.0'?>
   <stream:stream
       from='example.com'
       id='someid'
       xmlns='jabber:client'
       xmlns:stream='http://etherx.jabber.org/streams'
       version='1.0'>
...  nastavení šifrování, autentifikace klienta ...
k:   <message from='juliet@example.com'
             to='romeo@example.net'
             xml:lang='en'>
k:     <body>Art thou not Romeo, and a Montague?</body>
k:   </message>
s:   <message from='romeo@example.net'
             to='juliet@example.com'
             xml:lang='en'>
s:     <body>Neither, fair saint, if either thee dislike.</body>
s:   </message>
k: </stream:stream>
s: </stream:stream>
```

Na ukázce komunikace vidíme modelovou komunikaci mezi klientem (k) a serverem (s). Na prvních několika řádcích dochází k otevření dvou XML proudů (každý v jiném směru), následuje nastavení šifrování a autentifikace klienta, a pak už samotná komunikace, tzn. odeslání jedné zprávy z klienta jinému klientovi a potom přijetí odpovědi. Na závěr komunikace uzavírají klient i server otevřené XML proudy a ukončují spojení na transportní vrstvě.

1.2 Funkce vyhledávacího robota

Podle zadání bylo mým úkolem vytvořit Jabber robota, který bude ostatním klientům komunikační sítě Jabber poskytovat výsledky vyhledávání na webových stránkách. Při návrhu funkce celého systému jsem se dostal ke schématu, vyobrazenému na následujícím obrázku. Ústřední roli hraje robot, jehož mám implementovat. Tento na jedné straně komunikuje s Jabber serverem 1, který zprostředkovává služby robota ostatním klientům sítě Jabber, na druhé straně komunikuje se servery poskytujícími vyhledávání, a to pomocí protokolů SOAP nebo REST a v neposlední řadě spolupracuje i s mySQL databází, do níž si ukládá konfigurační údaje.

Obrázek 2 – Schéma komunikace robota



2 Prostředky použité při implementaci

2.1 Python

Python je vysokoúrovňový interpretovaný programovací jazyk, který byl vytvořen Guido van Rossumem v roce 1991. Rossumovým cílem bylo vytvořit jazyk, který upřednostňuje čitelnost a srozumitelnost před rychlostí. Ponechává určitou míru volnosti – programátoři mohou zvolit jak objektově orientovaný styl, tak i strukturované programování. V novějších verzích taktéž přibyly některé vlastnosti známé z funkcionálního programování – podpora lambda (anonymních) funkcí, funkce `filter()`, `map()` a `reduce()`.

Jazyk je dynamicky typovaný a obsahuje integrovanou správu paměti, založenou na počítání odkazů. Funkční bloky nejsou tvořeny uzavřením do složených závorek (tak jak to známe třeba z Javy nebo C), nýbrž odsazením, což zvyšuje čitelnost zdrojového kódu.

Významným plusem Pythonu je rychlost psaní aplikací, a to jak u jednoduchých prográmků, kde se projevuje jednoduchost syntaxe a stručnost zápisu, tak i u rozsáhlých aplikací, kde můžeme naplno využít vlastností jako je podpora jmenných prostorů (která nám umožní lépe strukturovat aplikaci), použití výjimek pro ošetřování chybových stavů aplikace, standardně dodávané prostředky pro psaní testů a pod.

Za zmínku také stojí možnosti propojení Pythonu s jinými programovacími jazyky. Je například možné výpočetně náročné části kódu naprogramovat v jazyce C nebo C++ a do aplikace v Pythonu je připojit jako modul. Nebo můžeme prostřednictvím kompilátoru JPython [14] psát v Pythonu programy, které mohou využívat vlastností tříd jazyka Java, a které potom přeložíme přímo do javového bajtkódu.

2.2 mySQL

mySQL je multiplatformní vícevláknový víceuživatelský databázový systém, založený na dialektu dotazovacího jazyka SQL. Byl vytvořen švédskou firmou MySQL AB. Licencování je dvojího typu – databázový server je k dispozici jak pod bezplatnou GPL licencí (MySQL Community Server), tak i komerční licencí (MySQL Enterprise).

Co se podpory nejrůznějších vlastností jazyka SQL týká, nebylo na tom MySQL v minulosti moc dobře. Autoři se rozhodli jít cestou optimalizace rychlosti, a to i za cenu menší podpory nejrůznějších funkcí. Avšak od současné řady 5 již mySQL podporuje většinu vlastností, které byste od databázového stroje očekávali – poddotazy, uložené procedury a funkce, triggery, pohledy a podpora různých znakových sad. Samozřejmostí je podpora transakcí.

Pro ukládání dat nabízí mySQL několik typů uložišť, které se odlišují svou rychlostí a svými možnostmi. Nejpoužívanějšími jsou úložiště typu MyISAM, které je založeno na starším ISAM (Indexed Sequential Access Method – indexově-sekvenční vyhledávání). Dalším, v současné době stále populárnějším, je úložiště typu InnoDB, které podporuje transakce a referenční integritu. Dalším velmi zajímavým typem úložiště je Memory. V tomto případě je tabulka vytvářena v paměti RAM, na disk se zapíše pouze informace o její struktuře (popis sloupců). Výhodou je obrovská rychlost práce s těmito tabulkami, nevýhodou je jejich nepersistence. Po vypnutí serveru obsah tabulky zmizí, při znovuspuštění je tabulka vytvořena podle uloženého schématu, avšak neobsahuje žádná data.

MySQL je dostupný pro mnoho operačních systémů – Microsoft Windows, různé verze Linuxu (RHEL, SuSE LES), MacOS X, dále pak pro nejrůznější UNIXové systémy – Solaris, HP-UX, IBM AIX apod. Na závěr by bylo vhodné zmínit, že k MySQL jsou taktéž dostupné zdrojové kódy, takže kdokoliv toužící po studiu databázových systémů má jedinečnou možnost.

2.3 xmpppy

xmpppy je framework vytvořený v jazyce Python. Jeho smyslem je co nejvíce zjednodušit aplikace komunikující pomocí protokolu XMPP. Obsahuje jak třídy pro implementaci Jabber serveru, klientů, tak i transportů. Na základě poskytovaném tímto frameworkem vzniklo několik implementací transportů – jmenovitě IRC transport, Yahoo transport, SMTP transport. xmpppy je také základem populárního Jabber klienta Gajim.

Existují i další pythonové frameworky pro komunikaci pomocí protokolu Jabber, jako třeba Twisted Words nebo jabber.py (na němž je xmpppy založen), avšak tyto projekty jsou již delší dobu neudržované a neposkytují dostatečnou podporu pro vývojáře (v podobě dokumentace, příkladů apod.)

2.4 soappy

soappy je frameworkem napsaným v Pythonu, který programátorům jednoduchým způsobem zpřístupňuje komunikaci pomocí protokolu SOAP. Ke zpracování odpovědi je využit parser postavený na základě SAXu. Framework také obsahuje podporu WSDL. Pro vytvoření požadavku mu tedy stačí ukázat WSDL soubor a zaslat parametry předávané v požadavku. O zbytek se knihovna sama postará a vrací objekt, přes jehož metody a vlastnosti můžeme přistupovat k jednotlivým položkám odpovědi.

2.5 mysqldb

mysqldb je wrapper napsaný v Pythonu, zprostředkovávající aplikaci připojení k mySQL databázi přes rozhraní, které je kompatibilní s Python DB API interface verze 2 [15]. Toto rozhraní nám umožňuje vytvářet aplikace nezávislé na použité databázi, neboť všechny databázové třídy implementují stejnou množinu vlastností a funkcí.

2.6 Jabber servery a klienty

2.6.1 jabberd14

Jabberd14 [16] je originální implementací Jabber protokolu na serveru. Verze 1.0 tohoto serveru byla zveřejněna v květnu 2000. Podporuje jak původní verzi Jabber protokolu, tak i novější verzi, standardizovanou v RFC 3920 a RFC 3921. Server je navržen modulárně, takže je možné jeho jednotlivé části spouštět v samostatných procesech, popřípadě na samostatných strojích. Další vlastností je taktéž běh v clusteru, kdy máme díky redundanci zajištěnu větší odolnost proti chybám, a také větší výkon.

Server je implementován v jazycích C a C++, což přináší několik výhod:

- velké množství programátorů, kteří znají tento jazyk a mohou do serveru doplňovat další funkce
- kompilace do strojového kódu přináší větší rychlost oproti interpretovaným jazykům (nebo jazykům kompilovaným do bajtkódu)

K jabberd14 také existuje velmi dobrá dokumentace programového rozhraní, striktně dodržuje standardy, je velmi flexibilní, co se týká konfigurace a možností nasazení.

2.6.2 jabberd2

Jabberd2 [17] není, ač by se to mohlo zdát, novou verzí jabberd14. Jde o dva oddělené projekty, každý s vlastním vývojářským týmem.

Jabberd2 je modulární systém, jehož základními komponentami jsou:

- router – tvoří páteřní propojení mezi ostatními komponentami serveru, je zodpovědný za předávání XML zpráv mezi komponentami
- s2s (server-to-server) – obhospodařuje spojení mezi Jabber servery, předává mezi nimi pakety a provádí autentifikaci ke vzdáleným serverům
- resolver – slouží k překladu doménových jmen pro komponentu s2s

- sm (session manager) – spravuje sezení – předává zprávy, informace o dostupnosti klientů, spravuje seznamy uživatelů a autorizace
- c2s (client-to-server) – komponenta obhospodařující spojení serveru s klienty – umožňuje připojení klientů k serveru, autentifikaci klientů, registruje uživatele, spolupracuje se session managerem

2.6.3 ejabberd

Ejabberd [18] je v současné době jabber serverem s nejaktivnějším vývojem. Je naprogramován v Erlangu, běží na mnoha platformách (Windows, Linux, MacOS X, FreeBSD, NetBSD). Stejně jako předchozí dvě implementace umožňuje provozování v clusteru s real-timovou replikací všech důležitých informací, z čehož plyne odolnost proti výpadku jednoho nebo více uzlů.

Dále je také nutno zmínit jednoduchost správy serveru. Díky použití Erlangu není třeba instalovat žádné externí databázové systémy ani webový server. Dostupné jsou jak zdrojové kódy, tak i binární instalátory pro většinu podporovaných systémů. ejabberd umožňuje také provozovat více jabberových domén v rámci jedné instance serveru, podporuje IPv6 jak pro propojení klienta se serverem, tak i serverů navzájem. Administraci je možno provádět přes webový prohlížeč (HTTPS), příkazovou řádku, ale i přes jabber klienta, který podporuje Service Discovery.

Co se podpory standardů týká, je ejabberd na výborné úrovni. Podporuje plně jak základní RFC 3920 a 3921, tak i mnoho rozšíření. Autentizace uživatelů může probíhat proti LDAP serveru, popřípadě je možno ukládat informace do relačních databází (buď nativně podporovaných mySQL a PostgreSQL, nebo přes ODBC). Samozřejmostí je také podpora transportů (bran) do ostatních IM sítí (AIM, ICQ, MSN).

2.6.4 PSI

Psi [19] je multiplatformní klient s otevřeným kódem. Podporuje jak Windows, tak i Linux a MacOS X. Nesnaží se za každou cenu podporovat co nejvíce funkcí, vývojáři se spíše zaměřují na stabilitu a podporu rozšíření, které byly Jabberovou komunitou standardizovány. Samozřejmostí je podpora kódování Unicode, přenosu souborů, service discovery, podpora šifrování, skupinový chat, podpora více uživatelských účtů, registrace transportů, avatary. V nejnovějších vývojových verzích je již také implementována podpora zobrazování XHTML zpráv.

2.6.5 Gajim

Gajim [20] je klient naprogramovaný v Pythonu s využitím grafické knihovny GTK+. Co se množství implementovaných funkcí protokolu XMPP týká, je Gajim na špici. Podporuje záložky v okně chatu, které velmi zpřehledňují komunikaci s více uživateli, podporu skupinového chatu, přenosy souborů, šifrování přenosu, registraci transportů, service discovery, více účtů, zobrazování XHTML zpráv.

2.6.6 Miranda

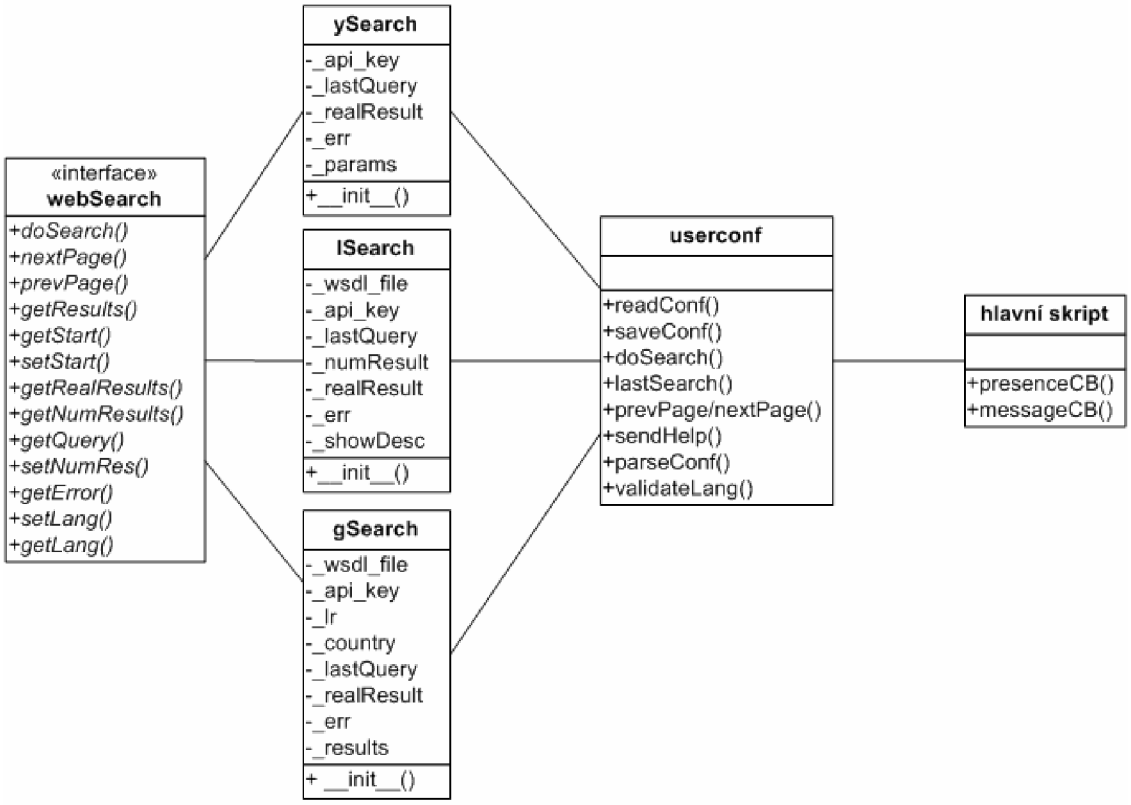
Miranda[21] je IM klient pro operační systém Microsoft Windows. Pomocí zásuvných modulů podporuje mnoho komunikačních sítí – AIM, ICQ, IRC, MSN, Yahoo, ale i Jabber. Počet podporovaných rozšíření (XEP) je poměrně velký – nechybí data forms, multi-user chat, service discovery, přenos souborů, avatary.

3 Detaily implementace

3.1 Struktura aplikace

Aplikace je složena z jednoho řídicího skriptu, tříd zajišťujících komunikaci s rozhraním vyhledávače, jedné třídy zajišťující správu uživatelského nastavení a jedné pomocné třídy.

Obrázek 3 – Struktura aplikace



3.1.1.1 Hlavní řídicí skript – jabber.py

Stará se o připojení k serveru, přihlášení robota, zaregistrování obsluhy příchozích zpráv. Běh programu probíhá v nekonečné smyčce. Definovány jsou následující funkce:

`sendMsg()` – odesílá zprávu klientovi. Má tři parametry – prvním je označení uživatele, kterému se zpráva odešle, druhým je samotný text zprávy a třetím (nepovinným) je ekvivalent zprávy v XHTML

`presenceCB()` – zpracovává registrace klientů na robotovi, posílá uvítací zprávu nově registrovaným klientům

`messageCB()` – zpracovává příchozí zprávy, na základě zaslaných příkazů provede příslušnou akci

`xmpp_iq_version()` – reaguje na žádost o zaslání názvu aplikace a její verze

3.1.1.2 Vyhledávání – třídy gSearch, lSearch, ySearch

Tato trojice tříd provádí komunikaci s jednotlivými vyhledávači a vrací výsledky. Všechny třídy implementují „rozhraní“, které obsahuje metody `doSearch()`, `nextPage()`, `prevPage()`, `getResults()`, `getStart()`, `getNumResults()`, `getQuery()`, `setNumRes()`.

`doSearch()` provádí samotné hledání – sestavení SOAP požadavku, zpracování výsledků hledání a nastavení soukromé proměnné obsahující výsledky.

`nextPage()` a `prevPage()` provedou posun na další, resp. předchozí stránku výsledků vyhledávání

`getResults()` vrací pole obsahující výsledky posledního provedeného vyhledávání

metody `getStart()` a `getNumResults()` se používají pro vizualizaci pořadí výsledků vyhledávání

Třídy `gSearch` a `lSearch` využívají knihovny `soappy` pro vytvoření SOAP klienta na základě dodaného WSDL souboru. Třída `ySearch` využívá třídu dodanou přímo firmou Yahoo. Nejdříve je pomocí „factory třídy“ vytvořen objekt zodpovědný za vyhledávání, poté jsou mu předány parametry, na jejichž základě sestaví HTTP požadavek, provede hledání a vrátí kolekci (seznam) výsledků.

3.1.1.3 Uživatelské nastavení – třída userconf

Třída `userconf` se používá pro správu uživatelského nastavení. Komunikuje s databází, ve které je konfigurace uložena, ukládá do ní případné změny. Dále komunikuje s jednotlivými vyhledávacími třídami, kterým posílá požadavky na vyhledávání. Zformátované výsledky hledání vrací řídicímu skriptu, který je rozesílá uživatelům.

3.1.1.4 Pomocné nástroje – třída utils

Tato třída byla založena kvůli pomocným funkcím. Nakonec vyvstala potřeba použití jenom jedné funkce, a to `strip_html()`, jejímž úkolem je odstranit z řetězce HTML značky. Dále jsou v této třídě definovány konstanty odpovídající chybovým stavům.

3.2 Ukládání dat

Vyhledávací robot potřebuje pro svou činnost spolehlivé a trvalé úložiště dat. Na toto úložiště nejsou kladeny žádné speciální požadavky, protože ukládaná data nejsou ani složitě strukturovaná, ani jich není mnoho. Pro tuto činnost jsem zvolil databázi MySQL, částečně kvůli jejímu rozšíření, jednoduché administraci, ale i dostupnosti implementace pro přístup k ní z prostředí Pythonu.

Aplikace si do databáze ukládá informace o uživatelích a jejich nastavení, a také historii vyhledávání (sbírání podkladů pro možnou budoucí optimalizaci ovládání a vyladění výchozích hodnot parametrů). Máme tedy tabulku `config`, která obsahuje 5 sloupců:

`jid` – JID identifikátor uživatele (slouží zároveň jako primární klíč)

`lang` – jazyk, ve kterém se vyhledává

`se` – vyhledávač, který se pro hledání používá

`numres` – počet vracených výsledků

`desc` – zda-li se má v textové verzi výsledků zobrazovat popis stránky

Tabulka určená pro záznam historie hledání se jmenuje `log` a obsahuje 8 sloupců – číselný primární klíč, jid klienta, který inicioval hledání, datum a čas hledání, hledaný výraz, použitý jazyk, vyhledávač, počet výsledků na stránku a zobrazená stránka výsledků.

Při registraci nového uživatele je do tabulky přidán řádek s výchozími hodnotami nastavení. Jestliže robota kontaktuje již dříve zaregistrovaný uživatel, je jeho nastavení načteno do databáze a udržováno v objektu třídy `userconf`. Při jakékoliv změně nastavení je tato změna ihned uložena do databáze.

3.3 Formát dotazů

Pro dotazování a konfiguraci robota jsem zvolil jednoduchý textový formát, který je založen na zkušenostech s ovládáním robotů v prostředí sítí IRC. Každý pokyn pro robota začíná znakem vykřičníku (s jedinou výjimkou, která je uvedena dále), následuje jedno nebo více klíčových slov nebo jejich zkratek a případně i nějaká hodnota, kterou může být řetězcový literál nebo číslo.

Základními příkazy jsou (v závorce jsou uvedeny použitelné zkratky):

`!query (!q)` – vyhledá pomocí zvoleného vyhledávače výraz následující za příkazem

`!next (!n)` – přesune se na další stránku výsledků vyhledávání a zobrazí výsledky

`!prev (!p)` – přesune se na předchozí stránku výsledků vyhledávání a zobrazí výsledky

`!last (!l)` – zopakuje poslední hledání

`!help (!h)` – zobrazí nápovědu

`!set (!s)` – slouží k výpisu nebo změně nastavení

Vzhledem k tomu, že hlavní činností robota je hledání a naprostá většina požadavků tedy bude obsahovat příkaz `!query`, jehož opakované vypisování by dosti zdržovalo, je vyhledávání považováno za výchozí akci. Když robotovi zašleme libovolný text, který nezačíná vykřičníkem, dojde k automatickému vyhledání zadaného výrazu.

3.4 Kompatibilita

Robot je napsán v jazyce Python, pro nějž existují interprety pro všechny běžné operační systémy, robot tedy není omezen na jednu konkrétní platformu. Navíc Python nativně podporuje kódování UTF-8, které je povinně používáno jak v protokolu XMPP, tak i v SOAP požadavcích.

3.5 Rozhraní pro jednotlivé vyhledávače

Vzhledem k faktu, že společnost Google pozastavila v prosinci 2006 vydávání nových klíčů k SOAP rozhraní pro vyhledávání na webu, a došlo také ke znatelnému zhoršení spolehlivosti služby (zhruba 30% požadavků se vracelo s chybovým hlášením), dohodli jsme se s vedoucím bakalářské práce na implementaci dalších vyhledávačů, konkrétně MSN Live Search a Yahoo Search, z nichž je možno při hledání vybírat.

3.5.1 Google SOAP API

Google poskytuje přístup ke svému vyhledávání a jeho výsledkům pomocí SOAP požadavků [22]. Součástí každého požadavku je jedinečný identifikátor, který si zájemci o používání této služby mohou nechat vygenerovat. (Respektive mohli, neboť Google již vydávání nových klíčů pro SOAP API pozastavil [23] a všechny nové uživatele láká na AJAX Search API, které poskytuje implementaci vyhledávání na bázi JavaScriptu, což je bohužel pro naše potřeby nepoužitelné. V SOAP rozhraní již nedále nebudou prováděny žádné opravy chyb a je otázkou, jak dlouho ještě služba poběží.)

Pro zjednodušení implementace Google dodává WSDL soubor (viz přílohu 1), pomocí kterého lze zautomatizovat sestavení SOAP požadavku.

3.5.2 MSN Live Search

Stejně jako Google i Microsoft poskytuje ke svému vyhledávacímu API WSDL soubor se specifikací poskytovaných funkcí a formátu dotazů. Pro přístup k vyhledávání potřebuje vývojář jedinečný identifikátor, který si může nechat vygenerovat na webu MSN [24].

Vyhledávání od Microsoftu poskytuje nejen výsledky hledání na webových stránkách, ale i z mnoha dalších oblastí – vyhledávání telefonních čísel, informace o počasí, financích apod.

3.5.3 Yahoo! Web Search

Na rozdíl od předcházejících dvou konkurentů nefunguje vyhledávání na Yahoo [25] na principu SOAP požadavků, nýbrž na architektuře REST. Principem je vytvoření HTTP GET požadavku:

- jehož základem URL je identifikace serveru, služby, verze a metody, tzn. například `http://search.yahooapis.com/WebSearchService/V1/webSearch`
- do části obsahující parametry pospojujeme samotné parametry hledání `appid=YahooDemo&query=test+query&results=3`

Podle požadavků specifikace protokolu HTTP musí být hodnoty všech parametrů upraveny tak, aby obsahovaly pouze povolené znaky (např. znaky národních abeced musí být převedeny do hexadecimální reprezentace uvozené znakem procenta).

Stejně jako vyhledávání od Microsoftu, i Yahoo Web Search podporuje vyhledávání ve více typech zdrojů – hledání multimediálního obsahu (zvuky, obrázky, video), hledání ve zprávách. Samotné hledání webových stránek je rozděleno do 4 oblastí – samotné vyhledávání, kterým se zabýváme v rámci této práce; dále pak vyhledávání webových stránek s kontextem, kdy kromě hledaného výrazu zadáme i kontext hledání – několik slov či vět, které hledanou oblast dále specifikují; poslední dva typy jsou zaměřeny na doporučování. První nám k zadanému klíčovému slovu vrátí další relevantní výrazy (related suggestion), které můžeme použít při dalším vyhledávání. Druhý nástroj (spelling suggestion) nám vrací opravené zadané slovo nebo výraz s opravenými překlepy.

3.6 Služby poskytované vyhledávači

V následující tabulce jsou uvedeny možnosti, které nám poskytují vyhledávací rozhraní. Vyhledávače se liší jen v drobnostech, jako je počet dotazů z jedné IP za den, dále potom ve formátech vracených výsledků – u jediného Yahoo je možno díky nepoužití technologie SOAP zvolit i jinou než XML reprezentaci. Za největší nedostatek osobně považuji chybějící podporu češtiny u MSN Live. Přestože webová podoba vyhledávače nemá s hledáním českých stránek problémy, do specifikace rozhraní bohužel náš jazyk zahrnut nebyl.

	Google	**MSN Live**	**Yahoo**
Vyhledávaný řetězec	slovo, přesná fráze, spojování pomocí spojek AND/OR, vyloučení slova z vyhledávání, operátor site	slovo, přesná fráze, spojování pomocí spojek AND/OR, vyloučení slova z vyhledávání, operátor site	slovo, přesná fráze, spojování pomocí spojek AND/OR, operátor site
Určení počtu a pozice vrácených výsledků	ano	ano	ano
Filtr podobných stránek	ano	ne	ano
Adult filtr	ano	ano	ano
Omezení výsledků	podle jazyka (včetně češtiny) podle země (včetně ČR) podle tématu	podle místních zvyklostí (locale) – čeština/ČR chybí	podle jazyka (včetně češtiny) podle regionu (odpovídá existujícím národním doménám yahoo) podle země (včetně ČR)
Omezení	maximální počet slov: 10 počet výrazů site: 1 na dotaz maximální počet vrácených výsledků: 10 maximální pozice prvního vráceného výsledku: 1000	maximální počet vrácených výsledků: 50 maximální pozice prvního vráceného výsledku: 1000	počet výrazů site: maximálně 30 maximální počet vrácených výsledků: 100 maximální pozice prvního vráceného výsledku: 1000
Formát výstupu	XML	XML	XML, JSON, serializovaný PHP objekt
Omezení počtu dotazů	1000 dotazů/den	10000 dotazů/den	5000 dotazů/den

3.7 Ukázky výsledků vyhledávání

Následující tabulka shrnuje ukázkové výsledky vyhledávání „FIT VUT“.

	česky	anglicky
Google	1) Faculty of Information Technology *http://www.fit.vutbr.cz/* 2) Fakulta informacních technologií *http://www.fit.vutbr.cz/.cs* 3) Doctoral Degree Programme *http://www.fit.vutbr.cz/study/phd/*	1) Faculty of Information Technology *http://www.fit.vutbr.cz/* 2) Fakulta informacních technologií *http://www.fit.vutbr.cz/.cs* 3) Speech@FIT http://www.fit.vutbr.cz/*research/groups/speech/*
MSN Live	výsledky z MSN Live jsou pouze v angličtině 1) Faculty of Information Technology *http://www.fit.vutbr.cz/* 2) MSDN AA: FIT VUT v Brně *http://msdn61.e-academy.com/vut_fit* 3) Video server FIT VUT *http://video1.fit.vutbr.cz/*	
Yahoo	1) History of this Page (Dotazník o studijních oborech na FIT VUT (výsledky)) *http://perchta.fit.vutbr.cz:8000/vyuka-gis/6.history* 2) EYSSELT Milos (CZ): studijni poradce na FIT *http://www.fit.vutbr.cz/~eysselt/index.html* 3) Fakulta informacních technologií *http://www.fit.vutbr.cz/.cs*	1) Faculty of Information Technology *http://www.fit.vutbr.cz/* 2) Information and Database Systems Research Group *http://www.fit.vutbr.cz/research/groups/is/team.php* *3)* History of this Page (Dotazník o studijních oborech na FIT VUT (výsledky)) *http://perchta.fit.vutbr.cz:8000/vyuka-gis/6.history*

3.8 Perspektivy dalšího vývoje

Možnosti dalšího vývoje by měly klást důraz na použitelnost a jednoduchost uživatelského rozhraní. Textová konfigurace a dotazování je sice velmi jednoduché, avšak uživatelé rádi klikají. Rozšíření XMPP protokolu Data Forms umožňuje vytvářet jednoduché formuláře, zobrazovat je v uživatelských klientech a odesílat získaná data zpět. Pomocí těchto formulářů by šla zjednodušit konfigurace robota. Uživateli by se zobrazil formulář s možností výběru přednastavených hodnot. Taktéž by bylo možné vytvořit formulář pro vyhledávání, kdy by se spolu s vyhledávacím políčkem zobrazily možnosti konfigurace parametrů pro jeden konkrétní dotaz.

Další důležitou vlastností, hlavně z pohledu zahraničních uživatelů, je internacionalizace uživatelského rozhraní. Vytvořit vícejazyčné prostředí není problém. Hlavním problémem je zjistit jazyk, který klient podporuje. Jediným možným řešením, na které jsem přišel, je načtení uživatelovy vizitky (v-card), z níž se dá zjistit země. Tento přístup může narazit ve dvou případech – není-li země vyplněna a nebo nerozpozná-li robot vyplněnou zemi (neexistuje žádné pravidlo, jakým způsobem zemi vyplnit – je třeba uvažovat české i anglické názvy, verze bez diakritiky, různé zkratky apod.)

Dalším z možných rozšíření je práce s historií vyhledávání. Robot zaznamenává všechny provedené dotazy, včetně nastavení a kontextu. Z těchto informací by se po čase používání dalo vyčíst nejlepší výchozí nastavení (počet stránek výsledků, vyhledávač), popřípadě doplnit robota o cache, do níž by se ukládaly výsledky hledání častých dotazů.

Závěr

Pro implementaci robota jsem si zvolil skriptovací jazyk Python. Jedním z důvodů byl požadavek zadání na nezávislost na operačním systému. Dalšími důvody pro použití Pythonu byla rychlost vývoje, která je oproti jazykům jako C++ nebo Java několikanásobně větší, a také nativní podpora kódování Unicode. Díky této vlastnosti odpadlo mnoho problémů s určováním a převodem kódování.

Výstupem bakalářské práce je mimo jiné funkční implementace robota odpovídající zadání, avšak je zřejmé, že pro maximalizaci uživatelské přívětivosti se dá udělat ještě hodně. Velký potenciál skýtá rozvoj uživatelského rozhraní, které je zatím velmi spartánské. Dalším důležitým bodem je podpora více jazyků, protože Jabber komunita hovořící anglicky je mnohem větší než česká. Do budoucna zůstává také na zvážení, zda-li by nebylo výhodnější tuto aplikaci implementovat ne jako robota, nýbrž jako službu.

Vinou vnějších vlivů – konkrétně kvůli zásahu provozovatele vyhledávače Google, jsme s vedoucím byli nuceni hledat alternativy k nespolehlivému Google SOAP rozhraní, které není do budoucna použitelné z důvody ukončeného vydávání licenčních klíčů. Na zvážení byly možnosti využít rozhraní vytvořené třetí stranou ke hledání přes Google Ajax Search API, a nebo implementovat vyhledávače jiných výrobců. Nakonec jsme se rozhodli pro druhou možnost, a to hlavně kvůli tomu, že rozhraní poskytované třetí stranou nebylo oficiální, nebyla u něj zaručena funkčnost a může být Googlem kdykoliv zablokováno. V robotovi je tedy na výběr ze tří vyhledávačů – Google, MSN Live a Yahoo.

Možnosti rozvoje existují i na straně vyhledávačů. Může se jednat například o podporu hledání multimédií (obrázky, video, zvuky, hudba), která však zatím naráží na straně Jabber klientů (rozšíření XHTML IM sice podporuje zobrazení obrázků v rámci zasílané zprávy, avšak hudební nebo video obsah je možné zobrazovat pouze externí aplikací, což na pohodlnosti mnoho nepřidává). Osobně mě, co se vyhledávačů týká, zklamaly dvě věci – chybějící podpora češtiny u MSN Live Search, a také již zmíněný přístup společnosti Google ke Google SOAP API.

Z pohledu osobního přínosu tvorby bakalářské práce jsem spokojen – měl jsem možnost blíže poznat programovací jazyk Python, a také procvičit své znalosti angličtiny při čtení specifikací jednotlivých protokolů a zkušeností uživatelů zvolených implementačních prostředků.

Literatura

[1] Oikarinen, J., Reed, D.. *RFC 1459 - Internet Relay Chat Protocol* [online]. May 1993 [cit. 2007-05-10]. Dostupný z WWW: <http://tools.ietf.org/html/rfc1459>.

[2] Webová stránka. *Microsoft Notification Protocol*. [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://en.wikipedia.org/wiki/Microsoft_Notification_Protocol>.

[3] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F.: *Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation*. 16 August 2006, edited in place 29 September 2006. [cit. 2007-05-10]. Dostupná z WWW: <http://www.w3.org/TR/REC-xml>.

[4] Gudgin, M., Hadley, M., Moreau, J., Nielsen, H. *SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation*. 27 April 2007 [cit. 2007-05-10]. Dostupné z WWW: <http://www.w3.org/TR/soap12-part1/>.

[5] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. *Web Services Description Language (WSDL). W3C note* [online]. 15 March 2001 [cit. 2007-05-10]. Dostupná z WWW: <http://www.w3.org/TR/wsdl>.

[6] Saint-Andre, P.. *Extensible Messaging and Presence Protocol (XMPP): Core* [online]. Jabber Software Foundation, October 2004 , 2004-08-30 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/rfcs/rfc3920.html>.

[7] Saint-Andre, P. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence* [online]. October 2004 [cit. 2007-05-10]. Dostupná z WWW: <http://www.xmpp.org/rfcs/rfc3921.html>.

[8] Saint-Andre, P. XEP-*0001 XMPP Extension Protocols* [online]. Last Updated: 2006-12-07 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0001.html>.

[9] Eatmon, R., Hildebrand, J., Muldowney, T., Saint-Andre, P. XEP-*0004 Data Forms* [online]. Last Updated: 2007-05-02 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0004.html>.

[10] Eatmon, R., Hildebrand, J., Millard, P., Saint-Andre, P. XEP-*0030 Service Discovery* [online]. Last Updated: 2007-02-15 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0030.html>.

[11] Saint-Andre, P. XEP-*0045 Multi-User Chat* [online]. Last Updated: 2007-04-10 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0045.html>.

[12] Saint-Andre, P. XEP-*0071 XHTML IM* [online]. Last Updated: 2007-03-14 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0071.html>.

[13] Muldowney, T., Miller, M., Eatmon, R. XEP-*0096 File Transfer* [online]. Last Updated: 2004-04-13 [cit. 2007-05-10]. Dostupný z WWW: <http://www.xmpp.org/extensions/xep-0096.html>.

[14] Webová stránka. Jython [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://www.jython.org/Project/index.html>.
[15] *PEP 249 - Python Database API Specification v2.0* [online]. 07 Apr 1999 , Last-Modified: 2007-04-28 [cit. 2007-05-10]. Dostupný z WWW: <http://www.python.org/dev/peps/pep-0249/>.
[16] Webová stránka. *jabberd14 XMPP/Jabber server daemon* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://jabberd.org/>.
[17] Webová stránka. *jabberd2 - Trac* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://jabberd2.xiaoka.com/>.
[18] Webová stránka. *ejabberd | distributed fault-tolerant Jabber/XMPP server in Erlang* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://ejabberd.jabber.ru/>.
[19] Webová stránka. *Psi Jabber Client :: Home.* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://www.psi-im.org/>.
[20] Webová stránka. *Gajim, a Jabber client.* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://www.gajim.org/>.
[21] Webová stránka. *Miranda IM – Home of the Miranda IM client.* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://www.miranda-im.org/>.
[22] Webová stránka. *Google API* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://code.google.com/apis/soapsearch/>.
[23] Forrest, B. *Google Deprecates Their SOAP Search API* [online]. 2006-12-18 [cit. 2007-05-10]. Dostupný z WWW: <http://radar.oreilly.com/archives/2006/12/google_depreciates_SOAP_API.html>.
[24] Webová stránka. *MSN Live API* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://msdn2.microsoft.com/en-us/library/bb251794.aspx>.
[25] Webová stránka. *Yahoo Web Search API* [online]. [cit. 2007-05-10]. Dostupná z WWW: <http://developer.yahoo.com/search/web/>.

Seznam příloh

Příloha 1 – Ukázkový WSDL soubor

V této příloze přikládám WSDL soubor, pomocí kterého můžeme zautomatizovat vytváření SOAP požadavků pro vyhledávání pomocí vyhledávače Google.

```
<?xml version="1.0"?>

<!-- WSDL description of the Google Web APIs.
     The Google Web APIs are in beta release. All interfaces are subject to
     change as we refine and extend our APIs. Please see the terms of use
     for more information. -->

<!-- Revision 2002-08-16 -->

<definitions name="GoogleSearch"
             targetNamespace="urn:GoogleSearch"
             xmlns:typens="urn:GoogleSearch"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Types for search - result elements, directory categories -->

  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
                targetNamespace="urn:GoogleSearch">

      <xsd:complexType name="GoogleSearchResult">
        <xsd:all>
          <xsd:element name="documentFiltering"           type="xsd:boolean"/>
          <xsd:element name="searchComments"              type="xsd:string"/>
          <xsd:element name="estimatedTotalResultsCount"  type="xsd:int"/>
          <xsd:element name="estimateIsExact"             type="xsd:boolean"/>
          <xsd:element name="resultElements"              type="typens:ResultElementArray"/>
          <xsd:element name="searchQuery"                 type="xsd:string"/>
          <xsd:element name="startIndex"                  type="xsd:int"/>
          <xsd:element name="endIndex"                    type="xsd:int"/>
          <xsd:element name="searchTips"                  type="xsd:string"/>
          <xsd:element name="directoryCategories"
type="typens:DirectoryCategoryArray"/>
          <xsd:element name="searchTime"                  type="xsd:double"/>
        </xsd:all>
      </xsd:complexType>

      <xsd:complexType name="ResultElement">
        <xsd:all>
          <xsd:element name="summary" type="xsd:string"/>
          <xsd:element name="URL" type="xsd:string"/>
          <xsd:element name="snippet" type="xsd:string"/>
          <xsd:element name="title" type="xsd:string"/>
          <xsd:element name="cachedSize" type="xsd:string"/>
          <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
          <xsd:element name="hostName" type="xsd:string"/>
          <xsd:element name="directoryCategory" type="typens:DirectoryCategory"/>
          <xsd:element name="directoryTitle" type="xsd:string"/>
        </xsd:all>
      </xsd:complexType>

      <xsd:complexType name="ResultElementArray">
        <xsd:complexContent>
          <xsd:restriction base="soapenc:Array">
             <xsd:attribute ref="soapenc:arrayType" wsdl:arrayType="typens:ResultElement[]"/>
          </xsd:restriction>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:complexType name="DirectoryCategoryArray">
        <xsd:complexContent>
```

```
          <xsd:restriction base="soapenc:Array">
             <xsd:attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:DirectoryCategory[]"/>
          </xsd:restriction>
        </xsd:complexContent>
      </xsd:complexType>

      <xsd:complexType name="DirectoryCategory">
        <xsd:all>
          <xsd:element name="fullViewableName" type="xsd:string"/>
          <xsd:element name="specialEncoding" type="xsd:string"/>
        </xsd:all>
      </xsd:complexType>

    </xsd:schema>
  </types>

  <!-- Messages for Google Web APIs - cached page, search, spelling. -->

  <message name="doGetCachedPage">
    <part name="key"            type="xsd:string"/>
    <part name="url"            type="xsd:string"/>
  </message>

  <message name="doGetCachedPageResponse">
    <part name="return"         type="xsd:base64Binary"/>
  </message>

  <message name="doSpellingSuggestion">
    <part name="key"            type="xsd:string"/>
    <part name="phrase"         type="xsd:string"/>
  </message>

  <message name="doSpellingSuggestionResponse">
    <part name="return"         type="xsd:string"/>
  </message>

  <!-- note, ie and oe are ignored by server; all traffic is UTF-8. -->

  <message name="doGoogleSearch">
    <part name="key"            type="xsd:string"/>
    <part name="q"              type="xsd:string"/>
    <part name="start"          type="xsd:int"/>
    <part name="maxResults"     type="xsd:int"/>
    <part name="filter"         type="xsd:boolean"/>
    <part name="restrict"       type="xsd:string"/>
    <part name="safeSearch"     type="xsd:boolean"/>
    <part name="lr"             type="xsd:string"/>
    <part name="ie"             type="xsd:string"/>
    <part name="oe"             type="xsd:string"/>
  </message>

  <message name="doGoogleSearchResponse">
    <part name="return"         type="typens:GoogleSearchResult"/>
  </message>

  <!-- Port for Google Web APIs, "GoogleSearch" -->

  <portType name="GoogleSearchPort">

    <operation name="doGetCachedPage">
      <input message="typens:doGetCachedPage"/>
      <output message="typens:doGetCachedPageResponse"/>
    </operation>

    <operation name="doSpellingSuggestion">
      <input message="typens:doSpellingSuggestion"/>
      <output message="typens:doSpellingSuggestionResponse"/>
    </operation>

    <operation name="doGoogleSearch">
      <input message="typens:doGoogleSearch"/>
      <output message="typens:doGoogleSearchResponse"/>
    </operation>

  </portType>
```

```
  <!-- Binding for Google Web APIs - RPC, SOAP over HTTP -->

  <binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="doGetCachedPage">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>

    <operation name="doSpellingSuggestion">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>

    <operation name="doGoogleSearch">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <!-- Endpoint for Google Web APIs -->
  <service name="GoogleSearchService">
    <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/beta2"/>
    </port>
  </service>

</definitions>
```

Příloha 2 – Uživatelská příručka

Ovládání robota je velmi jednoduché. Nejdříve si do svého seznamu kontaktů musíte přidat xmpp@jabbim.cz. Po přidání vás robot automaticky autorizuje a zobrazí vám uvítací zprávu. Nyní již můžete zasílat buď výrazy určené k vyhledání a nebo konfigurační příkazy.

Příkazy začínají znakem vykřičníku. Nejdůležitějším příkazem je `!query`. Vše, co za něj napíšete, bude vyhledáno na aktuálně zvoleném vyhledávači. Ve výsledcích hledání je možno se posouvat na další nebo předchozí stranu, a to pomocí příkazů `!next` , resp. `!prev`. Chcete-li zopakovat poslední hledání, například po změně nastavení, odešlete příkaz `!last`. Nevíte-li si rady s ovládáním robota, odešlete příkaz `!help`, robot vám zobrazí nápovědu.

Nastavování probíhá pomocí příkazu `!set`. Za ním následuje název parametru, který chcete změnit a jeho nová hodnota. Parametry jsou následující:

`!set se` – změna vyhledávače – povolené hodnoty parametru jsou google, live a yahoo

`!set nr` – změna počtu zobrazených výsledků na jedné stránce – povolenou hodnotou je libovolné kladné celé číslo

`!set lang` – změna jazyka hledání – možné hodnoty tohoto parametru jsou závislé na zvoleném vyhledávači. Při zadání nesprávné hodnoty dojde k vypsání všech povolených hodnot.

`!set desc` – přepíná zobrazení popisku výsledků v textovém zobrazení

Každý z příkazů jde vyvolat také pomocí zkratky, tvořené vykřičníkem a prvním znakem názvu příkazu (tedy `!q`, `!n`, `!p`, `!l`, `!s`, `!h`).

Robot umí zasílat výsledky jak v klasickém textovém režimu, tak i ve strukturovaném HTML kódu. Bohužel strukturované zobrazení nezvládají všechny klienty – podpora je u Gajimu a vývojové verze klienta Psi.

Příloha 3 – Návod k instalaci

Pro instalaci je vyžadována základní administrátorská znalost operačního systému na bázi Linuxu. Doporučeným instalačním prostředím je operační systém Linux. Pomocí balíčkovacího systému nainstalujte Python verze 2.5 a mySQL databázi verze 5.0. Nyní je potřeba stáhnout a nainstalovat následující Pythonové knihovny:

- soappy (http://sourceforge.net/project/showfiles.php?group_id=26590&package_id=18246)
- fpconst (http://cheeseshop.python.org/pypi/fpconst/0.7.2)
- dnspython (http://www.dnspython.org/)
- xmpppy (http://xmpppy.sourceforge.net/)
- mysqldb (http://sourceforge.net/projects/mysql-python)

Instalace je velmi jednoduchá. Všechny zmiňované balíčky jsou vybaveny instalačním skriptem. Nejdříve stáhneme instalační balíček, rozbalíme, a v adresáři se souborem setup.py spustíme

```
python setup.py build
```

a potom pod účtem s administrátorskými právy spustíme

```
python setup.py install
```

Tím máme dokončenu přípravu prostředí, nyní již můžeme z distribučního adresáře zkopírovat soubory

- gsearch.py
- lsearch.py
- ysearch.py
- userconf.py
- utils.py
- jabber.py
- GoogleSearch.wsdl
- LiveSearch.wsdl

do adresáře, z něhož budeme robota spouštět. Souboru `jabber.py` nastavíme práva pro spuštění (chmod +x). V souboru `userconf.py` upravíme hodnoty proměnných `_server`, `_port`, `_user`, `_password` a `_dbname` tak, aby obsahovaly platné hodnoty pro připojení k databázi. Také je potřeba pomocí dodaného souboru `schema.sql` vytvořit databázové tabulky. Dále v souboru `jabber.py` nastavíme JID a heslo robota. Tím je konfigurace dokončena a robot je připraven k použití. Stačí spustit soubor `jabber.py`, robot se automaticky přihlásí k Jabber serveru a začne poskytovat své služby.