VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

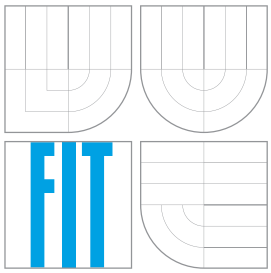
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROSLAV SENDLER

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DATAMINING Z JABBERU

DATAMINING FROM JABBER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV SENDLER
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOZEF MLÍCH
SUPERVISOR

BRNO 2011

Abstrakt

Předmětem této bakalářské práce bylo seznámení se s problematikou komunikace přes Jabber síť, která zde byla rozebrána. Konkrétním cílem bylo vytvoření jednoduchého Jabber klienta, který by byl schopen získávat statistická data. Nashromážděná data sloužila pro pozdější analýzu a grafickou reprezentaci informací z nich získaných.

Abstract

The objective of this thesis was acquaint oneself with problems of communication via Jabber network, which was also analyzed. The specific objective was to create a simple Jabber's client which would be able to obtain statistical data. The collected data was used for analysis and graphic representation of information.

Klíčová slova

Jabber, XMPP, robot, datamining, dolování dat, RapidMiner.

Keywords

Jabber, XMPP, robot, datamining, RapidMiner.

Citace

Jaroslav Sendler: Datamining z jabberu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Datamining z jabberu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Sendler
4. května 2011

Poděkování

Tímto bych chtěl poděkovat mému vedoucímu bakalářské práce Ing. Jozefovi Mlíchovi za ochotu a kladný přístup při konzultacích. Dále za poskytnutí hardware na němž běžel program a sbíral data.

Obsah

Kapitola 1

Úvod

Dnes mezi velmi se rozšiřující technologie na poli síťo Cílem této práce je získat neznámé informace z real–time komunikační sítě Jabber. Předmětem této bakalářské práce bylo seznámení se s problematikou komunikace přes Jabber síť, která zde byla rozebrána. Konkrétním cílem bylo vytvoření jednoduchého Jabber klienta, který by byl schopen získávat statistická data. Nashromážděná data sloužila pro pozdější analýzu a grafickou reprezentaci informací z nich získaných.

Kapitola 2

XMPP

V následující kapitole jsou, pro usnadnění a jednodušší pochopení, rozebrány základní stavební kameny protokolu Extensible Messaging and Presence Protocol (XMPP). Konkrétně jsou zde popsány stávající vlastnosti implementace, architektura protokolu XMPP obecně [22, 23] a další detaily protokolu [1, 24, 14]. Vzhledem k požadavkům na dolování v datech popsaných v následující kapitole je kladen důraz na vybraná rozšíření [21, 12]. Tato rozšíření tvoří základ pro některé rozšířené statusy, jako je například User Tune [18], User Mood [20], User Location [6] a další. Další informace použité pro popis a pochopení XML jazyka byly čerpány z [10, 9].

Vznik samotného protokolu XMPP je datován do roku 2004 (březen), kdy na něj byl přejmenován Jabber. Původní projekt Jabber byl vytvořen roku 1998 autorem Jeremie Millerem, který ho založil za účelem vytvořit svobodnou otevřenou IM službu. Uvedený projekt měl obsahovat tři základní vlastnosti, do kterých se zahrnují jednoduchost a srozumitelnost pro implementaci, jednoduchost v oblasti šíření a otevřenost podobě veřejně dostupného popisu samotného protokolu. Základní vlastnosti a výhody klientů a serverů budou podrobněji popsány níže. Roku 1999, 4.ledna byl vytvořen první server se jménem Jabber. Komunita vývojářů se chopila iniciativy a vytvořila klienty, kteří dokázali se serverem komunikovat, pro různé platformy (Linux, Macintosh, Windows). Roku 2004 byl protokol XMPP přidán mezi RFC[1] dokumenty. Základní norma popisující obecnou strukturu protokolu je RFC 3920 [22] a RFC 3921 [23], který se zaměřuje na samotný instant messaging a zobrazení stavu. Další zdokumentovaná rozšíření jsou vydávána v podobě tzv. XEP (XMPP Extension Protocol) dokumentů, které jsou známé také pod starším názvem JEP (Jabber Enhancement Proposal). Dnešní počet těchto norem se blíží k číslu 300. Každý XEP obsahuje stav vývoje (schválení), ve kterém se zrovna nachází.

Jako bezpečnostní prvky jsou zde podporovány SASL, TLS a GPG. XMPP protokol je postaven na obecném značkovacím jazyce XML, proto vlastnosti popsané dále v této kapitole platí i pro tento protokol.

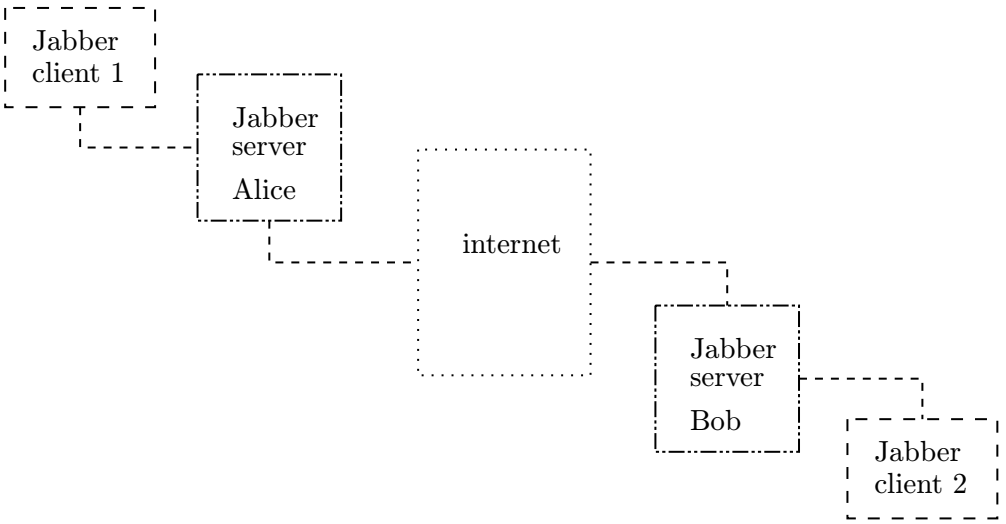
2.1 Architektura

Dobře navržená internetová technologie je tvořena správně fungujícími komponenty, které mezi sebou dokáží vytvořit spojení a následně započít komunikaci. Pro popis Jabber architektury v této práci bylo čerpáno z [1, 24]. Tato struktura se nejvíce podobá struktuře posílání e–mailů. Hlavní předností Jabber sítě je, tak jako u elektronické pošty, její decentra-

[1] RFC request of comments – žádost o komentáře

lizace. V případě Jabberu je decentralizace chápána jako možnost provozovat vlastní server, na rozdíl od jiných komunikačních systémů jako je například facebook, kde existuje pouze jediný poskytovatel služby. V případě serveru je kladen důraz na spolehlivost a rozšiřitelnost a u klienta na uživatele. Každý server pracuje samostatně, což znamená, že chod ani výpadek jiné datové stanice žádným způsobem jeho běh neovlivní. V případě výpadku jiného serveru bude nedostupný pouze seznam kontaktů a služeb, které registrovaným uživatelům poskytoval.

Obrázek 2.1 znázorňující distribuovanou architekturu Jabberu byl převzat z [1] a doplněn o názvy jednotlivých komponent. Komunikace dvou Jabber klientů probíhá za účasti jejich serverů a sítě, která je spojuje. Spojení mezi nimi bývá často šifrováno.



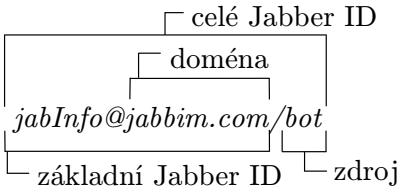
Obrázek 2.1: Distribuovaná architektura Jabber.

Architektura Jabber serverů využívá velké množství mezi–doménových připojení podobně jako internetový systém elektronické pošty. Komunikace klienta z jedné domény s klientem z jiné na rozdíl od e–mailového modelu nevyžaduje spolupráci třetích stran. Klient se spojí s „domácím“ serverem, který přímo naváže spojení se servem požadovaného klienta. Tyto vlastnosti jsou zárukou pro bezpečný přenos zpráv, znemožňující „krádeže“ JID[2], který je popsán níže, a spamování.

Jabber ID

Jabber ID (JID) je jednoznačný virtuální identifikátor uživatele na síti. V případě založení účtu nejsou rozlišována velká a malá písmena, což znamená, že Jabber není case–sensitive. Jednoznačný Jabber identifikátor je složen ze dvou částí: *Jabber bare* nebo–li čisté ID a *resource* [22]. Základní část na první pohled připomíná e–mailovou adresu *user@server*. Druhá část slouží k přesné identifikaci jednotlivých spojení. Je použita ke směrování síťového provozu s uživateli v případě otevření většího množství spojení pod jedním uživatelem. Společně Jabber bare a resource tvoří tzv. *full JID* — *user@server/resource* například *jabInfo@jabbim.cz/bot*. Jednotlivé části uživatelského jména popsané v tomto odstavci jsou ukázány v obrázku 2.2.

[2] uživatelské jméno



Obrázek 2.2: Rozebraná struktura Jabber ID.

Další vymoženost JabberID oproti e–mailové adrese je jeho možnost používat prakticky libovolné národní znaky u doménových jmen a uživatelských účtů [24]. Využíváním kódování UNICODE, se XMPP stává plně mezinárodní a není jako jiné protokoly omezen rozsahem ASCII tabulky. Přestože je tato vymoženost k dispozici, doposud není žádným výrazným způsobem využívána.

Klient

Klient je často jednoduchá aplikace pracující se vzdálenými službami, které jsou provozovány serverem. V této práci je zastoupen robotem s konzolovým rozhraním. XMPP svou architekturou nutí, aby byl co nejjednodušší. Vlastnosti, které by měl mít, jsou shrnuty, podle [14] do tří bodů:

1. komunikace s jedním Jabber serverem pomocí TCP soketu, který garantuje spolehlivé doručení zpráv na rozdíl od UDP. Nad tímto transportním protokolem dále běží kryptografický protokol TLS, který zabezpečuje komunikaci klient–server a server–server.

2. rozparsování a následná interpretace příchozí XML zprávy „stanza“ (kapitola 2.3)

3. porozumění sadě zpráv (*message, iq, presence*) z Jabber jádra [22]

Server

Informace použité pro popis XMPP serveru byly čerpány z [14]. K hlavním charakteristikám serveru oproti klientovi, jehož základní vlastností byla jednoduchost, patří stabilita a bezpečnost. Je pro něj vyhrazen TCP port 5222. Komunikace mezi servery je realizována přes port 5269. Každý server uchovává seznam zaregistrovaných uživatelů, který nevykazuje žádný jiný server. Zaregistrovaní uživatelé v daném seznamu se mohou do sítě připojovat pouze přes něj. To zajišťuje nemožnost „krádeže“ účtu. Protože XMPP komunikace probíhá přes síť, musí mít každá entita adresu, v tomto případě nazvána JabberID. XMPP spoléhá na DNS což znamená, že používá jména na rozdíl od IP protokolu.

Server Jabber je systém spravující tok dat mezi jednotlivými komponentami, které společně tvoří Jabber služby. Například *Jabber Session Manager* (JSM) poskytne funkce pro IM komunikaci a práci se seznamem kontaktů. Komunikace mezi jednotlivými servery, jak je uvedeno na obrázku 2.1, je zprostředkována za pomocí komponenty *S2S* (server to server). Při připojení klienta k serveru je komunikace řízená pomocí *C2S* (client to server). Jak již bylo řečeno, Jabber síť využívá doménová jména místo špatně zapamatovatelných IP adres. Pro tento způsob identifikace je určena služba *dnsrv*, která se stará o překlad názvů. V podstatě je to komponenta, která zajišťuje směrování paketů na jiný server.

V tabulce 2.1 jsou shrnuty informace o serverech Jabberu. První sloupec tvoří jméno, následuje programovací jazyk, v němž je napsán. Většina aplikací pro servery je vydávána pod licencí GPL[3]. U všech aplikací byla zkoumána nejaktuálnější verze. Její číslo lze naleznout ve třetím sloupci. Všechny servery lze provozovat na operačním systému Linux a Windows. Na platformě Mac OS mohou být použity všechny zde jmenované vyjma jabberd2. Pět z šesti zde představených programů pro server Jabber jsou stále vyvíjeny, tedy kromě jabberd14. Hlavním účelem tabulky je prezentovat důležité vlastnosti serverů v oblasti podpory rozšířených statusů. Jedná se o standardy *pubsub*[4] (XEP–0060) [12] a o jeho verzi zaměřenější více na uživatele *pep*[5] (XEP–0163) [21]. Obě tato rozšíření tvoří nezbytnou základnu pro *rozšířené statusy* a proto je jejich podpora jak u serverů, tak klientů vyžadována. Podrobněji toto téma bude rozebráno v některé následující podkapitole.

Server	Jazyk	Verze	XEP–0060	XEP–0163
ejabberd	Erlang/ Top	2.1.6	ANO	ANO
Openfire	java	3.6.4	ANO	ANO
jabbred2	c	2.2.11	NE	NE
jabbred14	c, c++	1.6.1.1	ANO	NE
Prosody	lua	0.7.0	NE	ANO
Tigase	java	5.0.0	ANO	ANO

Tabulka 2.1: Přehled Jabber serverů.

Z výše uvedené tabulky je zřejmé, že aplikace pro servery, které jsou stále ve vývoji, podporují tzv. *rozšířené statusy*. Tedy kromě programu jabbred2.

2.2 XML

Jazyk XML (eXtensible Markup Language) [9], metajazyk pro deklaraci strukturovaných dat, je jádrem protokolu XMPP. Samotný jazyk vznikl rozšířením metajazyka SGML, jež slouží pro deklaraci různých typů dokumentů. Základní vlastností je jednoduchá definice vlastních značek (tagů). Dokument XML se skládá z elementů, které můžeme navzájem zanořovat. Vyznačujeme je pomocí značek — počáteční a ukončovací. Pomocí tohoto jazyka je tvořena *stanza* popsaná v následující kapitole.

Ukázka možné struktury dokumentu psaného jazykem XML je zobrazena na příkladu 2.1. Standardně je předpokládáno, že je psán v kódování UTF–8 [10], ale je–li jako v tomto případě použito jiné, musí být konkrétní kódování uvedeno na jeho počátku. V opačném případě nemusí být obsah správně zobrazen. Na začátku dokumentu se také uvádí verze XML, ve které je dokument psán (1. řádek příkladu). Následuje kořenový element, který je uzavřen na samotném konci dokumentu. 4. řádek prezentuje možnost použití prázdného elementu, který obsahuje jeden atribut s názvem zkratky fakulty. Velký význam zde mají úhlové závorky. Jsou jimi z obou stran obaleny všechny elementy.

[3] General Public License — všeobecná veřejná licence GNU
[4] Publish–Subscribe
[5] Personal Eventing Protocol

```xml
<?xml version="1.0" encoding="iso-8859-2"?>
<fakulta>
  <název>Fakulta informačních technologií</název>
  <zkratka fakulty="FIT" />
  <typy studia>
    <bakalářské titul="Bc."></bakalářské>
    <magisterské></magisterské>
    <doktorské></doktorské>
  </typy studia>
</fakulta>
```

Příklad 2.1: Ukázka základního XML dokumentu.

2.3 Stanza

Základní jednotkou pro komunikaci založenou na XML je stanza. Z jednoduššího pohledu je možné se na ni dívat jako na jeden dlouhý XML soubor. Při zahájení komunikace se tento soubor „otevře“. Jeho samotné uzavření probíhá až při odhlášení od sítě, nebo–li přepnutí klienta do stavu offline. Stanzu je tedy možné vnímat jako stream, který obsahuje všechna data probíhající komunikace. Mezi elementy používané pro komunikaci klienta se serverem patří tyto tři: *message*, *presence* a *iq*. Každý zde uvedený člen má svůj jednoznačný význam. V následujících odstavcích jsou jednotlivé části stanzy blíže definovány a na reprezentativních příkladech jsou ukázány jejich základní struktury a možnosti využití v praxi.

První prvek, který bude charakterizován je označen anglickým výrazem *message* (zpráva). Jak již název napovídá, slouží k posílání zpráv všeho druhu. Je to základní metoda pro rychlý přenos informací z místa na místo. Zprávy jsou typu „push“, což znamená, že jsou odeslány a není očekávána žádná aktivita od příjemce, která by přijetí potvrdila. Jedno z dosavadních využití se nachází v klasické komunikace po internetu, tzv. instant messaging (IM). K dalším možným použitím patří skupinový chat a oznamovací nebo upozorňující zprávy. Každá z těchto zpráv je tvořena z minimální povinné struktury. Tak jako u klasické poštovní korespondence nesmí chybět adresa odesílatele a adresa příjemce, kterému je zpráva adresována. Podle možnosti použití jsou zprávy děleny do kategorií. Jmenovitě toto rozdělení implementuje atribut *type*, který může nabývat jednu ze čtyř hodnot. Jsou rozlišovány zprávy pro komunikaci mezi dvěma entitami, skupinový chat, upozornění, chybová zpráva a v neposlední řadě zpráva bez kontextu vyžadující odpověď příjemce. Nakonec nesmí být opomenut blok zprávy, pro uživatele IM nejdůležitější, nesoucí vlastní obsah.

Základní použití struktury elementu *message* je prezentováno na příkladu 2.2. Na prvním řádku je uveden atribut, značící odesílatele. Druhý řádek obsahuje JID klienta, který zprávy přijímá. Následuje informace o typu zprávy a poté je uveden element *body* nesoucí samotný obsah.

```xml
<message from="user@jabbim.com"
         to="jabinfo@jabbim.com/bot"
         type="chat"
    <body> Kolik je hodin? </body>
</message>
```

Příklad 2.2: Použití elementu *message*.

Další částí stanzy je poskytována struktura pro *request–response* (žádost–odpověď)

vazbu, podobnou metodám GET, POST a PUT z protokolu HTTP [24]. Zkráceně je označována pomocí dvou počátečních písmen *Info/Query* nebo–li IQ. Na rozdíl od elementu *message* tvoří *iq* spolehlivější přenos, optimalizovaný pro výměnu dat (binární data). K dalším rozdílům patří povinnost příjemce odpovědět na každou přijatou zprávu, nebo–li potvrdit její doručení. Skutečnost, že je na právě požadovanou zprávu odpovězeno, zajišťuje parametr *id*. Iq dotaz nebo odpověď musí obsahovat stejnou hodnotu tohoto atributu jako zpráva vytvořená žádajícím subjektem. Další povinný atribut rozděluje iq na čtyři typy. Jednotlivé žádosti na proces nebo akci jsou posílány samostatně [23]. V příloze E je uvedena rozsáhlejší struktura tohoto elementu. Použití nachází v případech, které nastavují, žádají nebo informace posílají. Tato struktura je využívána pro novou registraci, posílání seznamu kontaktů a další.

Příklad 2.3 znázorňuje základní použití elementu *iq*. Uživatel *user* posílá dotaz na získání seznamu kontaktu (řádek 5.).

```
1  <iq from="user@jabbim.com/doma"
2      to="user@jabbim.com"
3      id="uhhfw23648"
4      type="get"
5    <query xmlns="jabber:iq:roster"/>
6  </iq>
```

Příklad 2.3: Použití elementu *iq*.

Poslední a pro tuto práci nejdůležitější prvek stanzy je *presence*. V případě, že nemá určeného příjemce, tak funguje způsobem jako broadcast. Což znamená, že jsou informace směrovány všem klientům, kteří jsou zaregistrování k jejímu odběru. Presence v českém překladu informace o stavu (přítomnost) rozesílá dostupnost ostatních entit v síti. Jedná se tedy o nastavení uživatelské dostupnosti tak jako na jiných real–time komunikačních a sociálních systémů.

Existuje několik základních stavů statusů, které reprezentují aktuální dosažitelnost uživatele. Tento jev je vyjádřen pomocí elementu *show*, který disponuje čtyřmi možnostmi. První oznamuje, že je uživatel k dispozici a schopen aktivní komunikace. Druhá často se vyskytující možnost naznačuje, že je subjekt krátkou dobu pryč od svého IM klienta. Tento a další dva stavy, popsané dále, jsou často změněny bez lidského zásahu (pomocí pc nebo jiného zařízení) prostřednictvím funkce známé jako „auto–away“. Poslední dva stavy charakterizují delší časové období nečinnosti. Tato oznámení o změně stavu uživatele jsou často zasílána pouze kontaktům, které se nacházejí v režimu online. Tato optimalizace přispívá ke snížení síťového provozu, jelikož presence v reálném čase při komunikaci využívá velké množství šířky pásma.

Základní použití *presence* je zobrazeno v příkladu 2.4. Kontakt *jabinfo@jabbim.com/bot* (1. řádek) posílá informace o svém stavu (řádek č. 2) a svůj status (č. 3).

```
1  <presence from="jabinfo@jabbim.com/bot"
2    <show> online </show>
3    <status> Jsme zde. </status>
4  </presence>
```

Příklad 2.4: Použití elementu *presence*.

Obsáhlejší struktura elementu *presence* je zobrazena v příloze E, kde je rovněž k nalezení

přehled všech možných stavů.

Jak již bylo zmíněno v části o Jabber ID, Jabber podporuje práci s více současně připojenými klienty k jednomu Jabber účtu. Vysvětlení funkčnosti bude prezentováno na příkladu uživatele přihlášeného na stolním počítači a z klienta v mobilním telefonu. U obou těchto připojení je použit stejný Jabber bare, ale odlišného resource, například *domov* a *mobile*. Právě tento rozdíl v tzv. „full“ adrese účtu zajišťuje jednu ze dvou možných podmínek pro správnou adresaci zpráv. Druhá možnost, která bude uplatněna při použití adresy účty pouze ve formě Jabber bare, je nastavení priority u jednotlivých programů. Priorita je číslo v rozsahu hodnot od -128 do 127, kde klient s větší prioritou má přednost před klientem s nižší. Nastane–li případ připojení více klientů se stejnou prioritou, každý server se při rozesílání zpráv zachová podle vlastní implementace. Některé rozešlou zprávy všem klientům, jiné naopak jen poslednímu přihlášenému.

2.4 Rozšíření

Dále se tato práce zabývá rozšířeními protokolu XMPP o další vlastnosti k jejichž popisu slouží XEP. Pro tuto práci jsou nepostradatelné „statusy“, pro které tvoří základ standardy XEP–0060 [12] a XEP–0163 [21] zkráceně PEP[6]. Obě tato rozšíření umožňují strukturovaně pracovat, používat a přenášet další XEP protokoly. Jako příklady relevantní k práci jsou zde uvedeny protokoly *User Location* (kde se uživatel právě nachází) [6], User Tune (co uživatel poslouchá za hudbu) [18], User Mood (aktuální nálada uživatele) [20] a User Activity (co uživatel právě dělá) [11]. Jsou to tedy protokoly založené na PEP, které vyžadují podporu nejen v klientech, ale i na straně serveru (zobrazuje tabulka 2.1). S touto informací úzce souvisí další protokol XEP–0115 [7], který umožňuje zjistit podporované schopnosti klienta, případně, které informace je ochoten přijímat. Tato vlastnost bude popsána níže v části zabývající se podporovanými vlastnostmi.

Všechna tato rozšíření by mohla být přidána přímo do statusu viz příklad 2.4, avšak ten je primárně určen k informování o přítomnosti na IM síti. Hlavní rozdíl mezi PEP a obyčejným posílání stavu pomocí presence je v pravomoci klienta přijmout nebo odmítnou informaci, na rozdíl od presence, jež je přijata vždy.

Základ přenosu informací začíná na straně klienta, který chce všechny ve svém roster listu (seznam kontaktů) informovat o statusu. Zašle zprávu obalenou v elementu *iq* serveru. Ukázka této zprávy je prezentována na příkladu 2.5, který znázorňuje zaslání informace o druhu hudby, kterou v danou chvíli uživatel poslouchá. Využívá k tomu rozšíření User Tune, definovaném na řádku číslo 5. Základ zprávy oznamující začátek vysílání informací o rozšířených statusech je vždy stejný. Liší se pouze řádkem 3. a obsahem elementu item v příkladu 2.5.

V případě úspěšného přijetí *iq* zprávy serverem, každý, kdo se zaregistroval k odebírání rozšířených statusů, obdrží oznámení ve formě *message*. Oznámení bude také doručeno všem resources. Celá zpráva i všechny další náležitosti jsou uvedeny v příloze E.4.

Podporované vlastnosti

Jednotlivá rozšíření protokolu XMPP jsou nepovinná, a proto nemusí být ve všech klientských aplikacích podporována. Pro zjištění podporovaných rozšíření se používá XEP–0115 Entity Capabilities [7]. Toto rozšíření výrazně snižuje počet a velikost komunikací a přenosů

[6] Personal Eventing via Pubsub

```
<iq from='user@jabbim.com' type='set' id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='http://jabber.org/protocol/tune'>
      <item>
        <tune xmlns='http://jabber.org/protocol/tune'>
          <artist>Daniel Landa</artist>
          <length>255</length>
    ...
```

Příklad 2.5: Začátku vysílání rozšířeného statusu.

zpráv mezi uživateli. Dotazem zobrazeným na příkladu 2.6 je zjištěna schopnost jednotlivých klientů, kterou následně server využije pro správné směrování rozšířených statusů. Všechny zde zmiňované rozšíření a protokoly z této kapitoly je možné u každého klienta (seznam klientů obsahuje tabulka v příloze E.4) vyčíst z atributu *ver* (druhá část u atributu node), který je vypočítán ze všech podporovaných protokolů klienta, viz [7].

```
<iq from="user@jabbim.com" id="disco1"
    to="jabinfo@jabbim.com/bot" type="get">
  <query xmlns="http://jabber.org/protocol/disco#info"
      node="http://code.google.com/p/exodus#QgayPKawpkPSDYmwT/WM94uAlu0=" />
</iq>
```

Příklad 2.6: Dotaz na podporované protokoly.

Další rozšíření

V následujících několika odstavcích budou přiblíženy specifikace jednotlivých rozšíření XEP, které slouží jako zdrojová data pro dolování a jsou relevantní k tématu práce.

Prvním rozšířením, nad rámec základních vlastností Jabberu, které zde bude podrobněji rozebráno, je elektronická verze klasické vizitky nebo–li *VCard*. Jeho specifikací se zabývají dva standardy. Jelikož novější verze XEP dokumentu [13] se v době psaní této práce nacházela ve stavu „experimental“, což znamená, že ještě není schválena jako standard, je pouze ve stavu návrhu. Proto bylo použito verze starší [16]. Jednoduše řečeno je VCard struktura, která nese informace o uživateli jako je jméno, příjmení, e–mail, adresa bydliště i zaměstnání a další údaje. Data jsou dále zveřejňována na sítí, z čehož vyplývá, že jsou dostupná ostatním uživatelům. Vyplnění těchto osobních údajů je dobrovolné a tak se u některých uživatelů nachází pouze přezdívka a JID, které jsou často předdefinovány automaticky. Nedílnou součástí všech sociálních a komunikačních systému jsou malé fotografie, loga nebo ikony, kterými se uživatelé prezentují. V síti Jabber tomu není jinak, a proto je samotný obrázek zahrnut přímo do VCard v položce *photo*. Podrobnější informace o jeho nastavení a přijímání je možné nalézt v *vCard–Based Avatars* [19], který jej definuje.

Díky základní podmínce XMPP protokolu (otevřenost) existuje mnoho různých aplikací, pomocí kterých lze v síti Jabber komunikovat. S programy, používanými uživateli, úzce souvisí další zde implementované rozšíření. Jedná se o realizaci *Software Version* dokumentu [17], který se právě zabývá získáváním informací o samotných aplikacích. Je–li toto rozšíření podporováno je díky němu možné zjistit jméno a verzi používané aplikace. Informace o operačním systému často nejsou kvůli bezpečnosti ani vyplněny. Podrobnější informace o softwarové výbavě klienta je možné zjistit pomocí XEP [7], o kterém již bylo dříve psáno v

odstavci zabývajícím se podporovanými vlastnostmi klientských aplikací.

S rozšířením tzv. „chytrých“ mobilních zařízení mezi širší veřejnost vzniklo několik nových disciplín spojených s určováním zeměpisné polohy, jako je například geocaching. Geografická poloha je přenášena ve formě souřadnic popisující přímo zeměpisnou šířku a délku. Současně lze informaci o poloze přenášet i slovně ve formě adresy. Příkladem slovního popisu je ulice, číslo popisné, město a další. Mnoho aplikací, které mají k dispozici GPS přijímač, vysílají a aktualizují zeměpisné informace automaticky, například po určité době nebo změně polohy o určitou vzdálenost. Toto a další níže popsané rozšíření jsou postaveny na již zmiňovaném PEP. Některé části protokolů jsou zjednodušeny a připraveny tím pro „mobilní instant messaging“.

Pro sdělení informací o stavu klienta není v základní verzi Jabberu mnoho. Pomocí presence je možné „pouze“ prozradit, zda je uživatel připraven komunikovat nebo je momentálně nedostupný a to v několika verzích lišících se délkou nepřítomnosti. Pokročilejší nastavení statusu nabízí *User Mood* [20] a to ve formě sdělení současné nálady, jako je například radost. Další možné upřesnění činnosti uživatele jsou definovány v *User Activity* [11], kde každá činnost je složena z povinné obecné kategorie a nepovinné, která informaci upřesňuje. Příkladem může být *eating* a *having_a_snack* tj. uživatel jí, uživatel svačí.

K poslednímu rozšíření implementovanému v této práci patří *User Tune* [18], které umožňuje uživateli šířit informace o aktuálně poslouchané hudbě. Některé dnešních hudební přehrávače dokáží automaticky spolupracovat s IM klientem a předávat informace o hudbě bez nutného lidského zásahu. Ve zprávě jsou tedy přenášeny informace o skladbě, interpretovi, albu a další informace, které mohou být získávány z MP3 ID3v1 nebo novější ID3v2 tag.

Podpora rozšíření v aplikacích je ukázána v tabulce v příloze G. Z této tabulky vyplývá, že rozšířenost aplikací podporující výše popsaná rozšíření je poměrně malá. Například v předcházející zmiňované části o poslouchané hudbě, při stavu, kdy program toto rozšíření nepodporuje, je posíláno pomocí normální presence. Jméno skladatele, alba a další podrobnosti jsou shrnuty do statusu, tudíž jsou doručeny všem uživatelům ze seznamu kontaktů.

Kapitola 3

Data mining

Třetí kapitola se zabývá procesem dobývání znalostí z databází. Popisuje jej jako disciplínu, která vznikla za účelem vytěžení informací z dat, která jsou v nepřeberném množství ukládána v databázích. Díky velikosti dnešních disků, objem ukládaných dat neustále roste. S tím také úzce souvisí zvětšující se poměr nepotřebných a zašumělých dat vůči užitečným informacím. V této kapitole jsou mimo jiné popsány metody používané k dolování z dat, které jsou relevantní k této práci.

Na začátku kapitoly je rozebrán pojem získávání znalostí databází, jehož jednu podstatnou část tvoří samotný data ming. Dále je vysvětlena základní terminologie, pro kterou bylo čerpáno z [8]. Cílem první podkapitoly je přiblížení způsobu, jakým byla data uložená v databázi, připravena k samotnému data miningu. Celá druhá podkapitola je věnována vybraným metodám pro dolování dat a vlastnostem, které je od sebe navzájem odlišují. Jsou zde rozebrány *asociační pravidla*, pro jejichž popis bylo čerpáno z [2]. Pro ostatní metody, které jsou popsány dále, byla jako zdroj informací použita kniha [5]. Poté následuje třetí podkapitola, která se podrobněji zabývá jednou z metod pro dolování dat a to *shlukováním*. Obsahem této části jsou již konkrétní algoritmy pro shlukování dat [27, 3] a také metoda *k–Means* využívaná v praktické části této práce. Kapitolu uzavírá stručný přehled vybraných programů pro data minig a podrobnější seznámení s nástrojem *RapidMiner*, který je v této práci využíván pro samotné dolování.

Terminologie

Pojem data mining nebo–li česky dolování dat se začal ve vědeckých kruzích objevovat počátkem 90. let 20. století. První zmínka pochází z konferencí věnovaných umělé inteligenci (IJCAI'89[1]–mezinárodní konference konaná v Detroitu, AAAI'91[2] a AAAI'93–americké konference v Californii a Washingtonu, D.C) [2].

Tradiční metoda získání informací z dat je realizována jejich manuální analýzou a interpretací. V praxi ji například nalezneme v odvětví zdravotnictví, vědy, marketingu (efektivita reklamních kampaní, segmentace zákazníků) a dalších. Pro tyto a mnoho dalších disciplín je manuální zpracování příliš pomalé, drahé a vysoce subjektivní. Další důvod k přechodu na jiné metody je objemnost dat, která dramaticky vzrostla a tudíž se manuální analýza stává zcela nepraktická. Databáze rychle rostou ve dvou následujících kategoriích:

1. počet záznamů nebo–li objektů v databázi

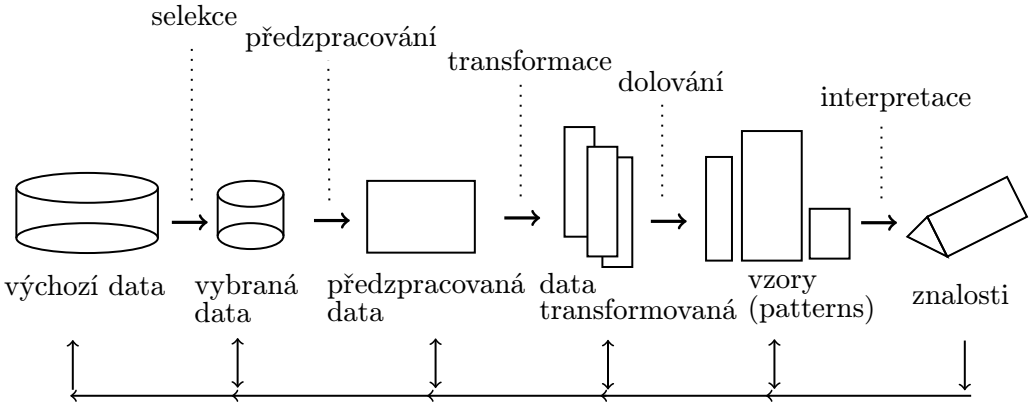
[1] International Joint Conference on Artificial Intelligence
[2] Association for the Advancement of Artificial Intelligence

2. počet polí nebo–li atributů objektů v databázi

Proces data mining je pouze jedna část z odvětví nazývané dobývání znalostí z databází nebo–li KDD[3] definované níže v definici 3.0.1. Vznik disciplíny KDD je důsledkem nepřeberného množství automaticky sbíraných dat, která je potřeba dále využívat. Podstatným znakem celého procesu je správnost reprezentace výsledků formou, která má k uživateli nejblíže. Jako příklad bude uvedena implikace ve tvaru rozhodovacích pravidel, asociační pravidla, rozhodovací stromy, shluky podobných dat a další. Základem KDD je praktická použitelnost metod. Očekává se zjištění nových skutečností namísto prezentování již známých informací.

Definice 3.0.1 KDD je chápáno jako interaktivní a iterativní proces tvořený kroky selekce, předzpracováním, transformace, vlastního „dolování“ (data–mining viz 3.0.2) a interpretace [2].

Grafické znázornění definice 3.0.1 je popsáno schématem na obrázku 3.1, který prezentuje časový harmonogram v KDD. Schéma znázorňuje následnost jednotlivých procesů, které tvoří KDD. KDD je iterativní proces, z čehož vyplývá, že skutečnosti nalezené v předešlých částí zjednoduší a zpřesní vstupy pro následující fáze. Jakmile jsou znalosti získány, jsou prezentovány uživateli. Pro přesnost může být část procesu KDD ještě upravena. Tím budou získány „přesnější a vhodnější“ výsledky.



Obrázek 3.1: Proces dobývání znalostí z databází podle knihy autora Fayyad [4].

Vzhledem k obrázku 3.1, který prezentuje jednotlivé kroky získávání znalostí z databází budou dále tyto procesy popsány. Prvním část v KDD je tvořena výchozími daty, které slouží jako zdroj pro ostatní fáze. Samotný popis získávání těchto dat je popsán v předcházející kapitole. Procesu selekce dat, je kladen za cíl, vybrat co možná „nejužitečnější“ množinu dat a tím i zmenšit její celkový objem. V této části získávání znalostí se při vybírání dat bere ohled na to, jak se jednotlivá data vztahují ke konkrétnímu uživateli. Následující proces nazvaný transformace se zabývá převedením dat do vhodného formátu pro samotné dolování informací. Tato část je popsána v následují podkapitole, kde hlavní úlohu při transformaci je čas. Vybrané metody pro dolování dat jako je například shlukování a další, jsou taktéž v této práci popsány níže.

[3]Knowledge Discovery in Database

Získávání znalostí z databází je proces složen z několika kroků vedoucích od surových dat k formě nových poznatků. Iterativní proces je složen, tak jak je prezentováno v [5], z následujících kroků:

- **čištění dat** – fáze, ve které jsou nepodstatné údaje odstraněny z kolekce.
- **integrace dat** – kombinování heterogenních dat z několika zdrojů do společného jediného zdroje.
- **výběr dat** – rozhodování o relevantních datech.
- **transformace dat** – také známý jako konsolidace dat. Fáze, ve které jsou vybraná data transformována do formy vhodné pro dolování.
- **data mining** – zásadní krok, ve kterém jsou aplikovány vzory na data.
- **hodnocení modelů** – vzory dat zastupují získané znalosti.
- **prezentace znalostí** – konečná fáze, zjištěné poznatky jsou reprezentovány uživateli. Tento základní krok využívá vizualizační techniky, které pomáhají uživatelům porozumět a správně interpretovat získané výsledky.

Jak je uvedeno v [5], běžně jsou některé z těchto kroků kombinovány dohromady. Kroky čištění dat a integrace dat mohou být provedeny společně, tak jako to prezentuje schéma na obrázku 3.1.

V této podsekci jsou ve stručnosti vysvětleny základní nejdůležitější pojmy dále v práci využívané.

Definic výrazu data mining se v odborné literatuře nachází několik. Zde uvedená je kombinací dvou „definic“ z [15].

Definice 3.0.2 Data Mining je proces objevování znalostí, který používá různé analytické nástroje sloužící k odhalení dříve neznámých vztahů a informací z velmi rozsáhlých databází. Výsledkem je predikční model, který je podkladem pro rozhodování [15].

Mezi další četně se vyskytující pojmy v tomto odvětví patří například data, znalosti a informace. Tyto termíny jsou často mezi sebou zaměňovány, proto jsou níže jejich významy striktně definovány tak jako v [8].

Jedna z několika existujících definic pojmu data je uvedena v definici 3.0.3, která je popisuje z pohledu informačního. Data často nemají sémantiku (význam) a bývají zpracována čistě formálně.

Definice 3.0.3 Data jsou z hlediska počítačového pouze hodnoty různých datových typů.

Informace lze chápat jako data, která byla obohacena o sémantiku (význam), jsou tedy již zpracovaná a interpretována uživatelem. Znalosti, jsou řazeny do stejné kategorie jako informace, ale jejich interpretace bývá ještě složitější. Často bývají tvořeny shluky informací, proto jsou reprezentovány jako odvozené informace. Podle studijní opory [8] jsou znalosti informace, které jsou zařazeny do souvislostí.

3.1 Transformace dat

Transformace dat tvoří třetí část z celkového procesu dobývání znalostí z databází. Než se data dostala do tohoto stavu, bylo na nich provedeno několik kroků, ve kterých byla upravována. V první fázi byla sbírána Jabber komunikace, která je popsána v první kapitole. Druhá fáze byla zaměřena na zúžení výsledné množiny a proto byla vybrána jen relevantní data. I přes tyto kroky relační databáze obsahuje velké množství dat, která se nenachází ve stavu, aby mohla být použita jako zdroj pro data minig. Jak bude popsáno v následující podkapitole většina metod pro dolování dat pracuje pouze s daty, která obsahují kvantitativní proměnné. Za tímto účelem je potřeba všechny atributy tabulky z databáze, které mají být nadále používány, převést na měřitelné hodnoty.

V této práci se bude pracovat s atributy nesoucí informace o jak aktuálních tak minulých stavech uživatelů, kteří si přidali účet *jabInfo@jabbim.com* do svého seznamu kontaktů. A tak byla jejich každá změna statusu uložena do databáze. Z důvodu nečíselného hodnoty stavů, jako je například available, away a další, je třeba provést jejich transformaci na kvantitativní hodnoty. Proces byl proveden pomocí bijektivního zobrazení. Kde zobrazení je, podle [8], funkce s definičním oborem S a oborem hodnot T, která je nazývána binární relace $f \subseteq S \times T$. V této relaci se nevyskytují dvě různé dvojice (s,t_1) a (s,t_2), kde $s \in S$ a t_1, $t_2 \in T$. Prvky t_1 a t_2 jsou různé. Z toho vyplývá, že každému prvku s z množiny S je přiřazen jednoznačně právě jeden prvek $t \in T$. Tuto definici je možné zapsat ve tvaru:

$$f : S \to T,$$

kde S a T jsou množiny (D_f, H_f).

Konkrétní případ transformace z této práce tedy bude obsahovat množinu S, kde

$$S = \{Available, Chat, Away, DND, XA, Unavaliable\}$$

a množina T, kde

$$T = \{120, 110, 90, 70, 50, 0\},$$

do které budou jednotlivé prvky z množiny S bijektivně zobrazeny. Kdy bijekce je zobrazení, které každému prvku z cílové množiny, konkrétně z množiny T, přiřazuje právě jeden prvek z množiny počáteční, tedy S.

Při výběru velikosti hodnoty, pro výslednou množinu T, bylo čerpáno z programu Gajim, který je multiplatformní klient s velkou podporou standardů a rozšiřujících protokolů.Hodnoty v množině T byly zvoleny tak, aby měly sestupné uspořádání, a aby bylo možné je dobře mezi sebou porovnávat. Jsou–li vybrány dvě hodnoty například available a unavailable, vzdálenost mezi nimi musí být větší než vzdálenost například u hodnot chat a away. Na druhou stranu prvky ze vstupní množiny S jsou striktně definovány podle Jabber standardu, který je popsán v RFC [22].

Temporální data

Druhá podstatná transformace, pro kterou bylo čerpáno z [25], se tak jako první nezabývá transformováním dat textových na data, jejichž obsah by byl tvořen kvantitativními proměnnými. V této části jsou řídká temporální data transformována na hustá. Pro následné vyhodnocení a data mining je potřeba řádkům z tabulky presence přidat konečné časové razítko, které by vymezilo interval doby platnosti těchto dat.

Jak již bylo uvedeno, hlavním rozdílem mezi temporálními databázemi a ostatními je schopnost uchovávat časové údaje. Své uplatnění nachází v odvětvích, kde je potřeba zpracovávat stará a zároveň nová data, například v oblastech medicíny, finančnictví, monitorování a dalších. Jednotlivé záznamy, které jsou závislé na čase, jsou v databázích ukládány jako samotné body, tedy diskrétně. Přestože v reálném světě je většina těchto údajů z pohledu času spojitých.

K dalšímu popisu temporálních databází nyní budou charakterizovány tři důležité pojmy, pomocí nichž jsou databáze dále děleny. Prvním pojmem je *granualita*, která udává nejmenší časovou jednotku, kterou databáze rozlišují. Hodnoty, které může nabývat, jsou hodina, den, rok a další. Velikost granuality ovlivňuje velikost objemu dat, který je přímo úměrný s přesností záznamů. Dalším pojmem je *čas platnosti*, která reprezentuje období, kdy je daný fakt v modelovém světě pravdivý. Posledním termínem je *čas transakce*, která definuje přítomnost faktu v databázi a možnost jej získat. Čas transakce a čas platnosti jsou na sobě nezávislé a definují dvě rozdílné časové osy, kdy každá může disponovat s jinou granualitou. Souhrnný název pro výše uvedené tři pojmy, který se používá je systém časových razítek. Při použití těchto systému lze následně tvořit dotazy zaměřené na různá časová období, jako je minulost, přítomnost a budoucnost. Tyto dotazy jsou velmi jednoduché a to díky rozšířenému jazyku TSQL, který vychází z klasické podoby dotazovacího jazyka SQL.

Temporální databáze jsou rozděleny, podle systému časových razítek, na tyto základní: *snímková*, *transakční*, *platného času*, *obojího času* (bitemporální). Entity z databáze, která je použita v této práci, je možné zařadit do kategorie *snímkových tabulek* (snapshot). K jejím hlavním rysům patří zaznamenávání stavu dat v jistém okamžiku. Čas transakce ani čas platnosti zde nejsou uplatněny.

Konkrétní příklad možné transformace je ukázán na části tabulky *presence* 3.1, která se ve stejném formátu nachází i v databázi, a modifikované tabulce *presecence_modify* 3.2. Tabulka 3.2 se od původní liší přidáním sloupce *dateEnd* (podbarven šedě), který s atributem *dateStart* vymezuje interval, kdy daná hodnota byla nebo je platná. Tato entita je pouze příkladem jak by mohla daná transformace vypadat. V této práci se žádná nová tabulka nevytvářela a ani se nemodifikovala již vytvořená. Celá transformace je popsána v další části této podkapitoly.

id	date	toj	presence
87365	2011–4–4 13:53:59	JabInfo@jabbim.cz	Available
87369	2011–4–4 15:43:07	JabInfo@jabbim.cz	Away
...	...	...	...

Tabulka 3.1: Ukázka tabulky *presence*.

id	dateStart	dateEnd	toj	presence
87365	2011–4–4 13:53:59	2011–4–4 15:43:07	JabInfo@jabbim.cz	Available
87369	2011–4–4 15:43:07	INF	JabInfo@jabbim.cz	Away
...	...	...	...	...

Tabulka 3.2: Ukázka modifikované tabulky *presence_modify*.

Postup jednotlivých kroků při transformaci časových údajů z tabulky *presence*, je zobrazen v následujícím výčtu. Tento zjednodušený popis algoritmu vyžaduje dva vstupní

parametry. První je JabberID uživatele, jehož položky v databázi mají být transformovány. Druhá nutná položka je datum, které bude sloužit jako upřesňující vstupní interval.

Data z tabulky presence jsou transformována na vektor ϑ o 288 dimenzích, kde z pohledu časového je jedna dimenze období vymezené 5 minutami. Den je rozdělen na úseky po 5 minutách ($\delta = 300s$), kterých je 288. Tedy

$$\vartheta = (\vartheta_1, \vartheta_0, \ldots, \vartheta_N),$$

kde $N = 288$.

1. Vstupním parametrem je datum, které vymezuje data pro transformaci. Toto datum je převedeno na dvě data (počátek a konec dne), která tvoří hraniční body v intervalu ι.
2. Výběr dat z databáze, která splňují časové období definované intervalem ι a uživatel ID. Uložení těchto dat do množiny Γ.
3. Pokud zadanému dotazu neodpovídá žádný řádek z tabulky, jsou data označena jako prázdná. Výstupní transformovaný vektor ϑ je naplněn hodnotami reprezentujícími stav *Unavailable*. Algoritmus je **ukončen**.
4. Zjištění prvního statusu toho dne.
   - 4.1 Převod data o den dřívějšího na interval ι_1, například *<2011–08–22 00:00:00, 2011–08–22 23:59:59>*.
   - 4.2 Výběr dat vyhovujícímu intervalu ι_1 a uživateli ID z databáze.
   - 4.3 Výběr posledního záznamu z množiny dat získaných z předešlého dotazu.
   - 4.4 Nalezení poslední presence a uložení její hodnoty do λ.
5. Výběr následujícího záznamu z množiny Γ.
6. Výpočet zda je časový interval mezi vybraným a následujícím záznamem větší jak δ.
   - 6.a Časový interval je větší než interval δ.
     - 6.1 Do výsledného vektoru ϑ je ukládána hodnota presence daného záznamu.
     - 6.2 Opakuj předešlý bod **6.1** kolikrát je interval δ menší než rozdíl mezi časy vybraného a následujícího záznamu.
   - 6.b Časový interval je menší než interval δ.
     - 6.1 Jsou vybírány další záznamy z množiny Γ dokud rozdíl mezi časy v sekundách není větší než interval δ.
     - 6.2 Jednotlivé presence záznamů jsou ukládány do pomocného pole, transformované do kvantitativních hodnot, jak je uvedeno v úvodu této podkapitoly.
     - 6.3 Každý prvek pole je vynásoben počtem sekund, zastoupených v daném intervalu.
     - 6.4 Výběr největšího prvku z pomocného pole a uložení jej do výsledného vektoru ϑ.
7. Celý proces je opakován od bodu **5**, dokud množina Γ není prázdná.

3.2 Metody dolovaní dat

Základ metod dolování dat je založen na statistice, posledních poznatcích z umělé inteligence či strojového učení. Hlavní cíl těchto netriviálních metod je společný — snaha zjištěné výsledky prezentovat srozumitelnou formou. Pro většinu používaných metod je společná vlastnost předpoklad, že objekty popsané pomocí podobných charakteristik patří do stejné skupiny (učení na základě podobnosti similarity–based learning). Objekty obsahující atributy, lze převést na body v n–rozměrném prostoru, kde n reprezentuje počet atributů. Vychází se z představy podobnosti bodů tvořící určité shluky v prostoru.

Další rozdíly mezi metodami, které byly prezentovány v [2], spočívají ve:

- schopnosti reprezentace shluků (např. otázka lineární separability)
- srozumitelnosti nalezených znalostí pro uživatele (symbolické vs. subsymbolické metody)
- efektivnosti znovupoužití nalezených znalostí
- vhodnosti typů dat
- a další ...

Problémy, které data mining řeší, se rozdělují do několika skupin. Do výčtu vybraných z nich, které budou následně rozebrány, patří *asociační pravidla*, *klasifikace*, *modely*, *predikce* a *shlukování*.

Při popisu asociačních pravidel, která jsou založena na syntaxi *IF–THEN*, bylo čerpáno z [2]. Jejich rozšíření se datuje do 90. let 20. století, kdy byly panem Agrawalem představeny v souvislosti s analýzou „nákupního košíku“. Použitelnost bude vysvětlena právě na příkladu analýzy nákupního košíku. Podstata příkladu je tvořena zákazníkem a jeho systémem nakupování. Jsou zjišťovány produkty, které jsou nakupovány současně. Hledají se nebo–li jsou vytvářeny společné vazby (asociační pravidla) mezi výrobky a určuje se jejich spolehlivost. Na základě těchto závislostí je upravováno umístění jednotlivých výrobků. Obecně jsou tedy asociační pravidla považována za konstrukci, která z hodnot jedné transakce odvozuje možnost výskytu závislostí v jiných transakcích. Jsou tedy hledány všechny vnitřní závislosti existující mezi daty.

K dalším metodám pro data minig patří klasifikace, která bude opět vysvětlena na příkladu, převzatého z [8]. Podle obsahu databáze nebo dotazníku bude každý klient banky zařazen do různých krizových skupin. Na základě těchto skupin pracuje „credit skóring“, jež klientovi poskytne nebo odepře například úvěr v bance. Další příklady využití jsou například ve zdravotnictví. Na základě zdravotního stavu pacienta a jeho příznaků, je pacient zařazen do tříd, které reprezentují jednotlivé nemoci. Klasifikací jsou, podle [5], jednotlivé zkoumané elementy rozděleny (podle hodnot atributů) do vhodných kategorií, které jsou předem vytvořeny z navzájem podobných objektů (tvorba profilů třídy). Při této metodě je upřednostňována přesnost před jednoduchostí a rychlostí. Zdroje klasifikovaných objektů jsou většinou tvořeny jednotlivými řádky v databázi. Vzory dat vytváří instance, Jejichž vlastnosti reprezentují atributy vyjádřené číselnou hodnotou.

Na modelech je založen třetí typ metod pro dolování dat. Základem modelů jsou trénovací data. Dále uvedené příklady vybraných klasifikačních modelů, byly čerpány z [5]. Především jsou to rozhodovací stromy, neuronové sítě, statistické metody, klasifikační pravidla a další.

Metodam která je postavena na myšlence, kde chronologicky seřazená data a vývoj jejich hodnot v minulosti tvoří základ pro určení hodnot budoucích, se nazývá predikce. Je řazena mezi velmi známé procesy, které na základě získaných znalostí předpovídají následující vývoj. Předpokládá se, že na základě informací získaných z dat v minulosti, bude možné postavit modely, které se budou chovat stejně nebo alespoň podobně i v budoucnu. Využití naleznou v předpovědi počasí (z naměřených meteorologických hodnot se určují budoucí předpokládané teploty), při vývoji cen na burze a dalších. Podklady pro popis predikce byly čerpány z [2, 5].

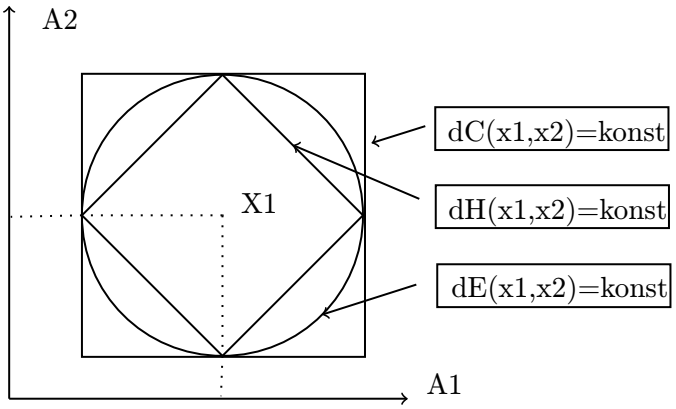
Poslední zde uvedená metoda je zaměřená na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků. Tato část, pro kterou bylo čerpáno z [27, 3], bude podrobně rozebrána v následující podkapitole.

3.3 Shlukování

V této podkapitole je shlukování rozděleno na několik metod shlukové analýzy podle [5]. U každé z nich jsou popsány její základní vlastnosti a uvedeny algoritmy relevantní k práci. Poslední metoda *metoda rozkladu* je rozebrána podrobněji z důvodu jejího praktického využití v této práci.

Shlukování je zaměřeno na dělení objektů do předem neznámých skupin. Proces dělení probíhá na základě specifikace objektů a jejich odlišnosti od ostatních shluků.

Většina níže popsaných metod a algoritmů je založena na výpočtu vzdáleností mezi objekty. Tato vzdálenost lze vyjádřit různými mírami, podle knihy [2] například pomocí *Hammingovy vzdálenosti* (dH), *Euklidovské vzdálenosti* (dE) a *Čebyševovy vzdálenosti* (dC). Rozdíl mezi těmito typy určující vzdálenosti, graficky vyjadřuje obrázek 3.2. Kde $X1$ je střed, od něhož jsou jednotlivými obrazci znázorněny dané vzdálenosti. Konkrétně pomyslné body umístěné po obvodu kruhu jsou všechny stejně vzdáleny od středu $X1$. Tato vzdálenost je označena jako Euklidovská. Další 2D těleso čtverec, který je vodorovný s osami $A1$ a $A2$ prezentuje Čebyševovu vzdálenost. Po obvodu posledního obrazce, čtverce otočeného o 45° podle osy A1, jsou všechny pomyslné body stejně vzdáleny od bodu $X1$ Hammingovou vzdáleností.



Obrázek 3.2: Srovnání výpočtu vzdáleností od bodu x_1 [2].

Jako první jsou zde rozebrány *metody založené na modelu*, které se pokouší přiřadit

data k určitému matematickému modelu na základě společných optimalizovaných vlastností. Většina procesů je založena na předpokladu, že jsou data generována pomocí standardních statistik. Mezi zástupné metody této shlukovací analýzy se řadí Expectation–Maximization (EM) a Self Organizing Oscillator Network, dále jen SOON. Algoritmus SOON je založen na neuronové síti. Je to metoda vycházející z algoritmu SOM[4] [27]. Metoda EM je rozšířením algoritmu *k–means*, který bude podrobně rozebrán v následující části.

Hlavní princip *metody hierarchického shlukování* je založen na tvorbě stromové hierarchie shluků, která je známá pod názvem *dendrogram*. Hierarchické metody, podle [5], mohou být rozděleny do dvou skupin a to na základě principu, kterým jsou dendrogramy vytvářeny. První možnost je *aglomerativní přístup*, který shlukuje menší shluky, kdy výsledkem je jen jeden. Druhý přístup, *divizní*, je založen na opačném předpokladu. Na počátku je tedy jeden velký shluk, který je postupně rozdělován dokud není počet shluků roven počtu objektů [27]. Mezi zástupce této metody například patří algoritmus AGNES[5].

K dalším metodám patří *metody založené na mřížce*, které kvantují datový prostor do konečného počtu pravoúhlých buněk. Tyto buňky jsou uspořádány do víceúrovňové mřížkové struktury. Zmíněná struktura tvoří základ pro shlukové operace. Hlavní výhoda tohoto přístupu je rychlost zpracování, které většinou nebere ohled na počet datových objektů. Čas zpracování závisí pouze na počtu buněk v každé dimenzi kvantovaného prostoru. Mezi zástupce metod založených na mřížce patří metoda STING — STatistical INformation Grid, který pracuje se statickými informacemi uloženými v buňkách mřížky. Algoritmus je rozdělen do dvou částí. První si klade za cíl rekurzivně rozdělit datový prostor na pravoúhlé buňky. Druhá fáze testuje spojitost mezi sousedy relevantních buněk [27].Mezi další metody založené na mřížce patří WaveCluster[6], využívající vlnkové transformace k rozdělení prostoru dat. Tato transformace zdůrazňuje shluky v prostoru a objekty jim vzdálené potlačuje [5].

Metody založené na hustotě vychází z m–rozměrného prostoru, ve kterém jsou zobrazeny objekty ve formě bodů. Místa v prostoru s větší koncentrací objektů ve srovnáním s ostatními oblastmi jsou nazývány shluky. Výchozí předpoklad je existence okolí jednotlivých bodů (sousedství). Jedna z charakteristik metod založených na hustotě je schopnost vypořádat se s vzdálenými hodnotami, označovanými jako šum [27]. Jako příklad je uvedena metoda DBSCAN[7], která je založena na hustotě objektů v prostoru. U jednotlivých objektů je zkoumáno jejich okolí. Algoritmus je ovlivňován dvěma parametry, velikosti shluku ε a minimálním počtem objektů v daném shluku $MinPts$, které spolu úzce souvisí (viz [5]). Bod splňující obě podmínky je označen za jádro. Za pomocí jader je rozšiřována množina objektů spojených na základě hustoty. Obsahuje–li jádro x_1 ve svém okolí další jádro x_2 znamená to, že jádro x_1 je přímo dosažitelné z jádra x_2. Tímto způsobem jsou vytvářeny výsledné *shluky*. V opačném případě, body, které nesplňují dvě zmíněné podmínky, jsou označeny jako *šum*.

Všechny zde doposud zmiňované metody poskytují dobré výsledky pouze s malým počtem dimenzí, tak jak je to popsáno v [8]. S narůstajícím počtem atributů roste počet nerelevantních dimenzí určených pro shlukování. S tímto také přibývá zvětšená produkce zašumění a znesnadnění nalezení relevantních shluků. Data jsou roztroušena do mnoha dimenzí a tím odpadá možnost použití vzdálenostních funkcí.Zmíněné problémy shlukování velkých dat řeší dvě techniky *metoda transformace rysů* a *metoda výběru atributů*. Pro

[4] Self–Organizing Map
[5] AGglomerative Nesting
[6] Clustering Using Wavelet Transformation
[7] Density–Based Spatial Clustering of Applications with Noise

efektivní shlukování je možné použít například algoritmus CLIQUE[8].

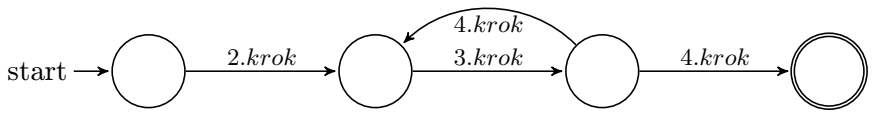
Metody rozkladu

Metody rozkladu rozdělují datové prvky do několika podmnožin, nazývané shluky. Počet shluků musí být znám před zahájením samotného procesu. Přiřazení do konkrétních tříd je, podle [27], jednoznačné nebo probíhá na základě míry příslušnosti objektů do shluků. Pro velký počet objektů, se kterými se pracuje, jsou využívány různé iterační optimalizace.

Hlavním zástupcem u uvedených metod je algoritmus k–means, který je popsán níže. Tvoří základ pro většinu metod shlukování nejen pro metody rozkladu. K dalším metodám se řadí k–medoidů, k–modů, k–histogramů, fuzzy shluková analýza a další.

k–means

Shlukování pomocí algoritmu k–means je používáno pro data obsahující kvantitativní proměnné a pro data, která nejsou příliš zašumělá. Základní proces je tvořen iterativním rozdělováním objektů do tříd na základě vzdáleností od jejich středů. Střed nebo–li centroid shluku je vektor, jehož vzdálenost od součtu vzdáleností objektů v této třídě je minimální. Celý tento proces je prezentován na obrázku 3.3 pomocí jednoduchého schématu konečného automatu. Jednotlivé kroky konečného automatu odpovídají krokům k–means zobrazeného pomocí algoritmu 3.1. Pro výpočet vzdáleností mezi objekty samotnými nebo mezi objekty



Obrázek 3.3: Algoritmus k–means zobrazený pomocí konečného automatu.

a středem je použita euklidovská vzdálenost[9], která je vyobrazena na obrázku 3.2.

K hlavním výhodám algoritmu k–means patří jeho relativní efektivnost. Složitost algoritmu je $O(TKN)$, kde N je počet objektů, K je počet shluků a T je počet iterací. Obvykle platí, že počet objektů je mnohem větší než počet iterací i shluků. Na druhou stranu má i řadu nevýhod, kvůli kterým je často různými způsoby modifikován (k–medoids, k–medians). K hlavním „nedostatkům“ patří předem nutná znalost počtu shluků (tříd) K, do kterých budou objekty zařazeny. Druhý často se vyskytující problém je samotné ukončení algoritmu, které nastane u nalezení lokálního optima namísto optima globálního. Tato nepřesnost vzniká nevhodně zvoleným rozmístěním počátečních středů. Původní nemodifikovaná verze algoritmu nedefinuje jak se má postupovat, jsou–li nalezeny prázdné shluky.

K–menas je algoritmus, kterým jsou přiřazovány objekty (vektory) x_n, kde $n = 1, .., N$, do S_k, kde $k = 1, ...K$, shluků. V prvním kroku jsou určeny počáteční středy tříd, do kterých se budou objekty shlukovat. Určení počátečních centroidů c_k probíhá například náhodným výběrem K objektů nebo K prvních objektů souboru. Druhým krokem jsou zkoumány jednotlivé vzdálenosti objektů x_n od počátečních středů c_j pomocí euklidovské vzdálenosti. Na základě nejmenší zjištěné vzdálenosti mezi objektem a centroidem je objekt zařazen do shluku, kterému náleží právě tento střed. Ve třetím kroku, tak jako u kroku prvního, jsou hledány nové středy shluků. Nyní již však nejsou zvoleny náhodně, ale spočítány. Jsou

[8] CLustering In QUEst

[9] mean = střed, centroid je vektor průměrů

vypočítány na základě průměrných jednotlivých hodnot objektů a uložen jako m–rozměrný vektor. Čtvrtým krokem se algoritmus dostává do konečné fáze, kdy mohou nastat dva možné případy. Nově nalezené středy nejsou příliš vzdáleny od předchozích centroidů a proto je algoritmus ukončen. Druhá častěji se vyskytující možnost iterativně provádí algoritmus od druhého kroku dokud neplatí první možnost nebo dokud se objekty nepřestanou přemísťovat úplně. Při popisu tohoto algoritmu bylo čerpáno z [27, 2]. Níže zobrazený algoritmus 3.1 prezentuje krok po kroku metodu k–means.

1. **náhodně zvol rozklad do** K **shluků**
2. **urči centroidy pro všechny shluky v aktuálním rozkladu**
3. **pro každý příklad** x
   - 3.1 **urči vzdálenosti** $d(x, c_k)$, $k = 1, ..., K$, **kde** c_k **je centorid k–tého shluku**
   - 3.2 **nechť** $d(\mathbf{x}, c_l) =$ **min_k** $d(x, c_k)$
   - 3.3 **není–li** x **součástí shluku** l **(k jehož centoridu** c_l **má nejblíže), přesuň** x **do shluku** l
4. **došlo–li k nějakému přesunu, potom jdi na** 2, **jinak konec**

Algoritmus 3.1: Metoda k–means byla převzata z [2].

Díky jednoduchosti a relativní rychlosti je metoda *k–means* stále výrazně využívána. Uplatnění nachází v široké škále oblastí jako je například biologie nebo počítačová grafika. Vzhledem k enormnímu počtu možného uspořádaní nejsou výsledky vždy přesné, ale často pouze přibližné.

3.4 Programy

V současné době na programovém trhu existuje mnoho systému, které jsou zaměřeny na data mining. Mezi nejrozšířenější a nejdostupnější nástroje patří Weka a RapidMiner. K těmto nástrojům je také možné zařadit program FIT-miner vyvíjený na fakultě informačních technologií v Brně. V této práci byl pro samotný data miningg využit program RapidMiner, který dostal přednost před ostatními. Z pohledu nástroje FIT–miner, který ve své základní části podporuje z databází pouze Oracle, se RapidMiner jevil jako vhodnější. Kompatibilitu pro databáze typu PostgreSQL již měl zabudovanou a tak nebylo potřeba vyvíjet žádné doplňující moduly, jak by to bylo u FIT–mineru. V případě nástroje Weka, RapidMiner působil propracovanějším dojmem a také nabízí lepší grafické zobrazení vyhodnocených výsledků.

Dalším velmi rozšířeným a často používaným nástrojem je jazyk *R*. R vychází z jazyka *S*, který ale není jako jazyk R volně šiřitelný. Statistický a grafický nástroj R je tedy volně dostupným jazykem a prostředím, které je ovládáno pouze z příkazové řádky. Pro jednodušší práci jej lze rozšířit o grafické rozhraní jako je RKWard nebo R Commander. Samotnou aplikaci lze rozšířit o mnoho statistických doplňků, které jsou taktéž zdarma.

Mnoho programů pro dolování dat je založena na přístupu vizuálního programování. Jedná se o proces, při kterém je uživatelem, za pomoci grafických prostředků, navržen algoritmus a další postup práce. Jako příklad lze uvést poloprofesionální aplikaci *Orange*, u které jsou nejdůležitější části psány pomocí C++ a rozšíření lze implementovat v jazyce Python.

Na vybraných technicky zaměřených vysokých školách existují skupiny, které se zabývá výzkumem a vývojem nástrojů pro data mining. Jako příklad lze uvést již dříve zmiňovaný FIT–miner z fakulty informačních technologií v Brně nebo také projekt LISp-Miner z

Vysoké školy ekonomické v Praze. LISp–Miner je otevřený akademický systém určený pro výuku a výzkum metod pro dobývání znalostí z databází.

RapidMiner

RapidMiner je, tak jak je popsán na oficiálních stránkách produktu [26], celosvětově nejpoužívanější open–source systém pro dolování dat. Je možné jej používat jako samotnou aplikaci nebo jej začlenit jako komponentu do vlastních výrobků v podobě knihovny pro jazyk Java. Pro zájemce je nabízen také ve verzích pro firmy, které jsou rozdílné v poplatcích, podpoře pro zákazníka, záruce a dalších balíčcích služeb zajišťující celkovou komplexnost a spolehlivost produktu.

Jak již většina podobných aplikací, je v současné době implementován v jazyce Java, díky které nabízí flexibilní nejen grafické prostředí. K vybraným základním rysům toho nástroje, tak jak jsou prezentovány firmou *Rapid–i*, patří: výkonné, přesto intuitivní grafické uživatelské rozhraní pro návrh procesů, jednoduché řešení pro transformaci dat, kontrola výsledků již při samotném návrhu a další. Nástroj RapidMiner podporuje širokou škálu metod a algoritmů pro data minig. Mnoho algoritmů je implementováno přímo v aplikaci, ale také je použito metod z konkurenčního softwaru Weka. V základní verzi určené pro veřejnost je k nalezení přes 100 procesů k modelování. Jsou zde zastoupeny jak metody klasifikační a asociační, tak i metody shlukovací z nichž lze jmenovat například DBSCAN, k–medoids a hlavně k–means.

K dalším schopnostem RapidMineru je možnost spuštění jeho samotného pomocí grafického rozhraní nebo z příkazové řádky. Jak již bylo uvedeno dříve, je také možné jej použít jako knihovnu v jazyce Java. V této práci jsou použity první dvě možnosti. Pomocí grafického prostředí byl vytvořen experiment, otestována jeho funkčnost a následně pro jednotlivá shlukování použita šablona procesu, která byla volána z příkazové řádky. Tato možnost je k dispozici díky tomu, že jsou projekty v programu RapidMiner ukládány do čitelné a strukturované formy za pomocí značkovacího jazyka xml.

Kapitola 4

Implementace

Obsahem čtvrté kapitoly je popis praktické části této práce. Jsou zde charakterizovány jednotlivé prvky, které byly použity jak pro získání dat, tak pro jejich následné uložení.

Cílem této práce je dolování dat z Jabberu. Jak již bylo dříve napsáno, Jabber je real–time komunikační služba díky níž mohou její uživatelé komunikovat, informovat nebo sdílet svůj status s jinými uživateli. Celá tato vzájemná komunikace skrývá rozsáhlé množství informací o klientech dané sítě. Všechna tato navzájem vyměněná nebo poskytnutá data následně poslouží jako zdroj samotnému dolování. Pro jejich uskladnění je využita databáze, jejichž strukturální návrh prezentuje obrázek 4.1.

Druhá část této kapitoly je zaměřena na Jabber klienta nebo–li robota, který je implementován v jazyce C++ pouze s konzolovým rozhraním. Robot v této práci hraje roli pasivního uživatele, který informace pouze přijímá. Ve vybraných případech dokáže uživatele ze svého seznamu kontaktů vyzvat k zaslání odpovědi s informacemi na odpověď, o kterou žádal. Struktura samotného robota je popsána níže a reprezentována obrázkem 4.2. V části o Jabber robotovi jsou také uvedeny informace o knihovně *gloox*, kterou je zprostředkovány všechny náležitosti Jabber komunikace.

4.1 Databáze

Data, která jsou sbírána robotem, jsou ukládána do objektově–relační databáze PostgreSQL[1]. PostgreSQL nebo–li také Postgres byl použit ve verzi 8.4.7 a je provozován na operačním systému Ubuntu Maverick verze 10.10.

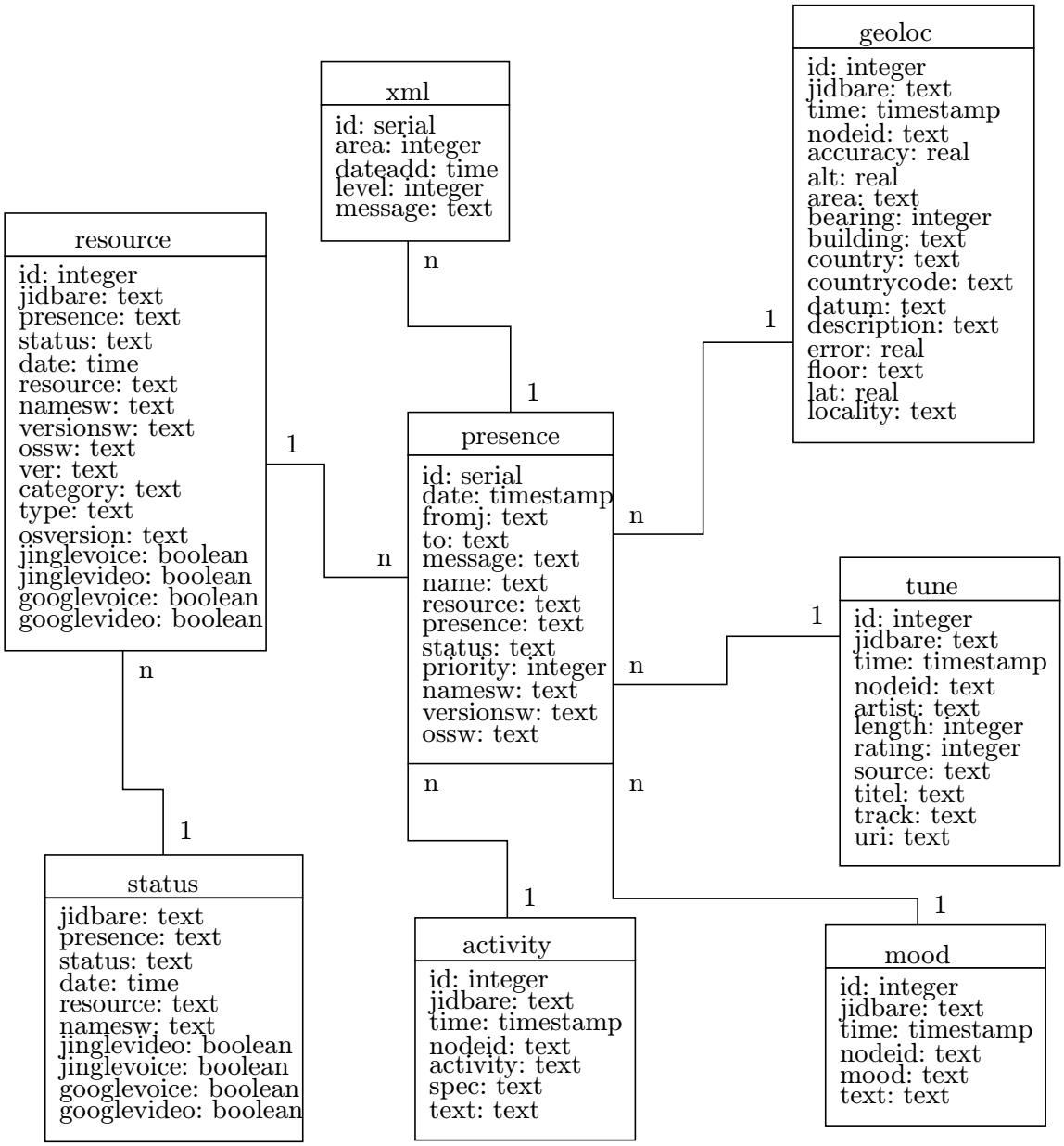
Návrh databáze

Struktura databáze, do které je ukládána veškerá komunikace Jabber robota, využívá relační model. Obrázkem 4.1 jsou prezentovány nejdůležitější části databáze. Celou strukturu návrhu je možné nalézt v příloze F. V jednotlivých částech návrhu databáze je počítáno s druhotným využitím obsahu, které bude popsáno v dalších oddílech této práce.

V době návrhu databáze nebylo zcela zřejmé, která data budou následně analyzována. Z tohoto důvodu se struktura databáze snaží zachytit všechna „důležitá“ data. Za tímto účelem je v návrhu databáze obsažena tabulka *xml*, která je nositelem obsahu jak všech přijatých, tak i odeslaných zpráv. Tabulky *debug*, *level* a *logarea*, které v zúženém návrhu databáze, uvedeném na obrázku 4.1, nejsou zobrazeny, jsou určeny pouze jako doplňkové

[1] vyvinul se z projektu Ingres

informace k typu zprávy. Tabulku xml je možné nahradit jednotlivými dalšími tabulkami, které jsou zaměřeny na konkrétní data. Příkladem entity, která již obsahuje konkrétní data bez xml prvků, je tabulka *message*, která je taktéž vyobrazena v příloze E. Jejím obsahem jsou zprávy vzniklé při Jabber komunikaci mezi uživatelem a robotem, jehož schopnosti budou popsány níže.



Obrázek 4.1: Vybraná část struktury databáze.

Tabulka *presence* z pohledu XMPP standardu prezentuje jeden ze tří základních částí stanzy, element *presence*. Základním cílem této tabulky je shromažďování uživatelských statusů. Jak již bylo zmíněno ve třetí kapitole, v části která se zabývá transformací, atribut presence obsahuje hodnoty typu text. Příklad všech možných hodnot, kterých tento atribut může nabývat je prezentován v příloze E v části zabývající se elementem presence. K dalším

atributům této tabulky patří *message*, který je reprezentován textovým řetězcem a rozšiřuje informace o stavu uživatele. V této části je často obsažen text, který je do statusu přidán automaticky IM klientem. Transformovaný obsah této entity tvoří důležitou část pro další krok v KDD a to v podobě dat, ze kterých budou „dolovány“ informace.

Většina rozšíření standardu XMPP, která jsou popsána ve druhé kapitole, jsou prezentována pomocí pěti tabulek. Konkrétně to jsou tabulky *geoloc*, *tune*, *mood*, *activity* a *vcard*. Pro obsah a názvy atributů všech pěti tabulek s rozšířeními se staly vzory dokumenty jednotlivých XEP, které je definují. Tabulky kromě těchto atributů obsahují také položku *jidbare*, která, jak již název naznačuje, obsahuje pouze čisté JabberID bez resources. Tato skutečnost vyplývá již ze samotného návrhu XEP protokolů. V tabulce *vcard* jsou také položky, jejichž obsah je tvořen fotografiemi. Obrázky jsou zde uloženy ve stavu, tak jak jsou přenášeny po síti, tedy pomocí datového formátu Base64. Base64 je algoritmus, který převádí binární data na řetězec, který je tvořen pouze znaky ASCII tabulky.

S posledním rozšířením nazvaném *Software version* je spjata tabulka resource, jejíž několik sloupců tvoří informace o IM klientovi a operačním systému, na kterém běží. Tyto základní údaje jsou definovány v protokolu [17], kde je také možné čerpat bližší informace. Pokročilejší atributy, jako je *type* a *osversion* nacházejí svůj podklad v XEP dokumentu [7], taktéž jako atribut *ver*. V neposlední řadě se zde nachází zmínka o podpoře audia a videa, ať už v podobě *jingle* nebo *google*. Přesto tyto výše uvedené údaje nejsou hlavním důvodem existence této tabulky. Její primární cíl je uchovávat informace o připojených uživatelích a všech jejich resources.

Entita resources tvoří základ pro tabulku *status*. Její obsah je automaticky generován z obsahu tabulky popsané výše. Počet řádků odpovídá počtu uživatelů, kteří jsou v seznamu kontaktů tohoto robota. Primárním cílem je informování o aktuálním stavu uživatele. Je tedy poskytován stav, ve kterém se uživatel nachází s přihlédnutím na prioritu a resources. Řádek entity obsahuje status a dostupnost uživatele a jeho resources s největší přihlášenou prioritou.

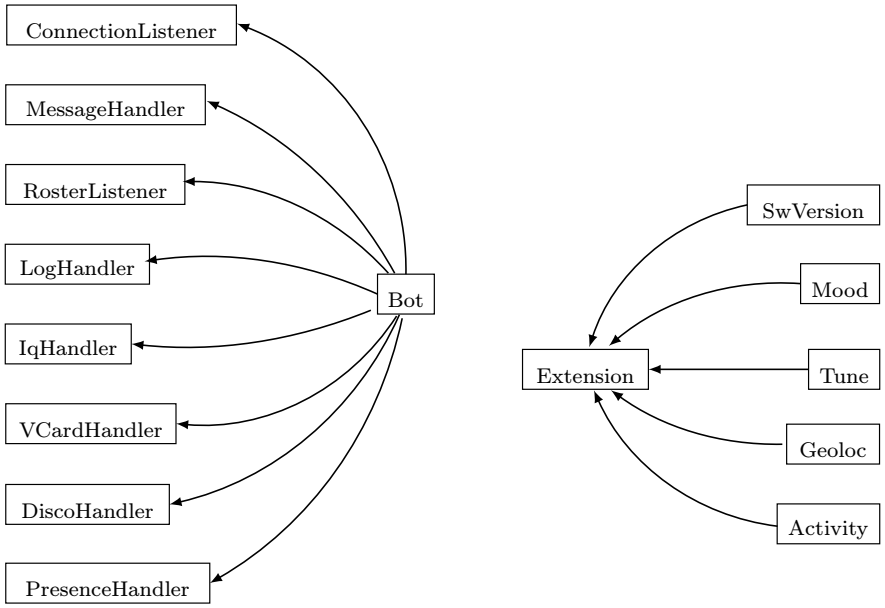
4.2 Robot

V této části bude popsána aplikace, která zajišťuje real–time komunikaci s ostatními entitami Jabber sítě. Konzolový robot, který je vyvíjen pro operační systém Linux, také vhodným způsobem pracuje s databází, kam jsou ukládána jednotlivá získaná data. Jabber jádro síťové komunikace [22], je implementováno pomocí volně dostupné knihovny *gloox*. V porovnáním s jinými knihovnami disponuje lepší podporou a dokumentací. Gloox plně implementuje standart XMPP Core [22] a z větší části i standard XMPP IM [23]. Dodatečně je podporováno kolem třiceti XEP standardů mezi něž například patří vcard–temp [16] a další.

Při implementaci robota byly využity základní prvky objektově–orientovaného programovaní, jako jsou například třídy a dědičnost. Každá třída zde zastupuje jednoznačně oddělitelné elementy robota. Ať už se jedná o blok komunikující s databází, různá rozšíření nebo o jádro robota. Všechny zde jmenované objekty hrají důležitou roli jak v části dekompozice na menší ucelené celky tak i v celkové komplexnosti programu. Jako příklad lze uvést třídy, které reprezentují jednotlivá pro tuto práci vhodná rozšíření, jejichž základ je v podobě XEP dokumentů.

Návrh robota

Struktura návrhu tříd robota, která je v základní části prezentována obrázkem 4.2, je velmi podobná struktuře návrhu databáze, obrázek 4.1. Mnoho atributů z robota a z databáze jsou ve vztahu jedna ku jedné. Samotná komunikace a řízení mezi těmito entitami je zprostředkováno pomocí knihovny *libpq*. Libpq je programátorské rozhraní pro databázi PostgreSQL. Je to sada funkcí v jazyce C, které umožňují klientským programům zadávat dotazy na server a následně na ně obdržet odpovědi. Pomocí této knihovny je řízen přenos dat mezi robotem, který sbírá data, a databází, kam jsou ukládána. Navázání spojení a následné jednotlivé dotazy probíhají v blokujícím režimu. V souboru *connect.h* jsou, v podobě předdefinovaných konstant, uvedeny všechny údaje nutné k navázaní spojení ať už s databází nebo s komunikačním Jabber serverem. Tudíž se stává hlavní částí určenou k editaci připojení a Jabber účtu na němž robot běží. K dalšímu pouze textovému dokumentu patří *const.h*, který je zaměřen k definování jednotlivých příkazů a dotazů pro databázi. Jsou to příkazy pro tvorbu a aktualizaci tabulek, které jsou taktéž definované jako konstanty. Identifikační názvy, pro snadné odlišení, jsou psány velkými písmeny a začínají textem *DB_*.



Obrázek 4.2: Vybraná část struktury robota.

Třída, která využívá zde doposud popsané konstanty a rozhraní libpq, je nazvaná *Database*. Obsahuje mnoho funkcí, které jsou přetěžovány, a které zajišťují samotnou provázanost mezi daty a databází. Pomocí těchto funkcí jsou data získaná z komunikace přenášena a ukládána do jednotlivých tabulek databáze. Pro zahájení samotné práce s databází slouží funkce *start()*, která zajistí připojení k databázi a provede nutné inicializační kroky. Jako je tvorba nových tabulek, při prvním zapnutí aplikace, nebo kontrola existence těchto entit.

Pro rozšíření podle dokumentů XEP popsaných v kapitole druhé, v části zabývající se relevantními rozšířeními, je základ nadtřída Extension. Tato třída obsahuje základní funkce a parametry pro všechna rozšíření. Jako příklad lze uvést uživatelské jméno klienta, který dané rozšíření „šíří“ po síti. Obsahuje také dvě čistě virtuální metody, sloužící k rozparsování části xml zprávy a k vyčistění proměnných. Třídy, které z Extension dědí a jsou

prezentovány na návrhu robota uvedeném na obrázku 4.2, jsou následující: *Geoloc*, *Tune*, *Mood*, *Activity* a *SwVersion*. Samotnou strukturou těchto rozšíření, jak již bylo uvedeno výše, jsou pomocí parametrů kopírovány vlastnosti jednotlivých XEP dokumentů. Tedy ke každému atributu xml zprávy existuje jak proměnná uchovávající její obsah tak i tzv. „get“ a „set“ metoda. Pomocí virtuální metody jsou jednotlivé části rozšíření rozparsovány, uloženy do tříd a následně vloženy do databází. Poslední doposud nezmiňované rozšíření v podobě *vcard*, je využito z knihovny gloox, která jej implementuje.

Základní část, kterou je možné považovat za jádro aplikace, je třída *Bot*. Tato třída je založena na již zmiňované C++ knihovně gloox. Je implementována za pomocí vybraných tříd, ze kterých dědí. Jsou to především „handler“ třídy, kterými je zajišťován přístup k jednotlivým zprávám. Nebo–li jejich prostřednictvím je získán přístup k samotným elementům stanzy, jako je message, iq, a presence, jejichž vlastnosti a využití jsou popsány v druhé kapitole. Konkrétní seznam tříd, ze kterých je děděno, zobrazuje návrh robota na obrázku 4.2. Jako příklad lze uvést třídu, z prostoru jmen gloox, *VcardHandler*, díky níž je možné získat a zpracovávat vcard uživatelů, kteří jsou v seznamu kontaktů robota. Samotné přidávání uživatelů do seznamu kontaktů je prováděno automaticky. Pouze na straně klienta je nutné požádat robota o autorizaci, která však proběhne automaticky. Jednotlivé kontakty nejsou žádným způsobem tříděny do skupin, tudíž existuje pouze jedna.

Rozšíření

Kapitola 5

Vyhodnocení výsledků

5.1 Manuální rozbor dat

–charakter dat, jaka data jsem nasbiral –popst jak jsem data prochael rucne, tedy manualni dataming, pocet zazanmu, delka behu programu, velikost datatabze, pocet uzivatelu...

Kapitola 6

Závěr

POKACOVANI =tridit lidi do podskupin v kotakt listu, napriklad podle zpravy zaslene s dotazem k autorizaci=: kazda skupina by mohla mit jin prava, jine dotazy an sluzby...mozne rozsirit o ruzne sluzby, ktere poskytuji server, podle dotazu....

Literatura

[1] Adams, D.: *Programming jabber*. Sebastopol: O'Reilly, první vydání, 2002, 455 s., iSBN 05-960-0202-5.

[2] Berka, P.: *Dobývání znalostí z databází*. Praha: Academia, první vydání, 2003, 366 s., iSBN 80-200-1062-9.

[3] Bramer, M.: *Principles of Data mining*. London: Springer, první vydání, 2007, 343 s., iSBN 18-462-8765-0.

[4] Fayyad, U. M.; Smyth, P.: *Advances in knowledge discovery and data mining*. California: MIT Press, první vydání, 1996, 611 s., iSBN 02-625-6097-6.

[5] Han, J.; Kamber, M.: *Data mining : concepts and techniques*. San Francisco: Morgan Kaufmann Publisher, druhé vydání, 2006, 770 s., iSBN 15-586-0901-6.

[6] Hildebrand, J.; Saint-Andre, P.: XEP-0080: User Location. [online], 15-09-2009, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0080.html

[7] Hildebrand, J.; Saint-Andre, P.; Tronçon, R.; aj.: XEP-0115: Entity Capabilities. [online], 26-02-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0115.html

[8] Hruška, T.: *Informační systémy : IIS/PIS*. Brno: Fakulta informačních technologií, 2008, 14733 s.

[9] Kolektiv autorů: Extensible Markup Language (XML) 1.0. [online], 26-11-2008, [cit. 4. května 2011].
URL http://www.w3.org/TR/2008/REC-xml-20081126/

[10] Kosek, J.: *XML pro každého : podrobný průvodce*. Praha: Grada, první vydání, 2000, 163 s., iSBN 80-716-9860-1.

[11] Meijer, R.; Saint-Andre, P.: XEP-0108: User Activity. [online], 29-10-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0108.html

[12] Millard, P.; Saint-Andre, P.; Meijer, R.: XEP-0060: Publish-Subscribe. [online], 12-07-2010, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0060.html

[13] Mizzi, S.; Saint-Andre, P.: XEP-0292: vCard4 Over XMPP. [online], 02-26-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0292.html

[14] Moore, D.; Wright, W.: *Jabber developer's handbook*. Indianapolis: Sams Publishing, první vydání, 2004, 487 s., iSBN 06-723-2536-5.

[15] Nemrava, M.; Pospíšil, J.: Dolování dat a jeho aplikace. [online], 2006, [cit. 4. května 2011].
URL http://www.spatial.cs.umn.edu/paper_ps/dmchap.pdf

[16] Saint-Andre, P.: XEP-0054: vcard-temp. [online], 07-16-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0054.html

[17] Saint-Andre, P.: XEP-0092: Software Version. [online], 02-15-2007, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0092.html

[18] Saint-Andre, P.: XEP-0118: User Tune. [online], 30-01-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0118.html

[19] Saint-Andre, P.: XEP-0153: vCard-Based Avatars. [online], 16-08-2006, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0153.html

[20] Saint-Andre, P.; Meijer, R.: XEP-0107: User Mood. [online], 29-10-2008, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0107.html

[21] Saint-Andre, P.; Smith, K.: XEP-0163: Personal Eventing Protocol. [online], 12-07-2010, [cit. 4. května 2011].
URL http://xmpp.org/extensions/xep-0163.html

[22] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Core. [online], 10-2004, [cit. 4. května 2011].
URL http://tools.ietf.org/html/rfc3920

[23] Saint-André, P.: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. [online], 10-2004, [cit. 4. května 2011].
URL http://tools.ietf.org/html/rfc3921

[24] Saint-André, P.; Smith, K.; Troncon, R.: *XMPP : the definitive guide : building real-time applications with jabber technologies*. Sebastopol: O'Reilly, první vydání, 2009, 287 s., iSBN 978-059-6521-264.

[25] WWW Stránky: Database Systems. [online], 2007, [cit. 4. května 2011].
URL
http://www.cs.uct.ac.za/mit_notes_devel/Database/Lates%t/index.html

[26] WWW Stránky: RapidMiner. [online], 2011, [cit. 4. května 2011].
URL http://rapid-i.com/content/view/181/190/

[27] Řezánková, H.; Húsek, D.; Snášel, V.: *Shluková analýza dat*. Praha: Professional Publishing, druhé vydání, 2009, 218 s., iSBN 978-808-6946-818.

Příloha A

Obsah CD

Příloha B

Manual

Příloha C

Konfigrační soubor

Příloha D

Slovník zkratek

DNS — Domain Name System

GPG — dkshckdsjvlsdjvodsvjdfokj

IM služby — dkshckdsjvlsdjvodsvjdfokj

IP — Internet Protocol

JEP — dkshckdsjvlsdjvodsvjdfokj

JID — dkshckdsjvlsdjvodsvjdfokj

SASL — dkshckdsjvlsdjvodsvjdfokj

TCP — dkshckdsjvlsdjvodsvjdfokj

TLS — dkshckdsjvlsdjvodsvjdfokj

WWW — dkshckdsjvlsdjvodsvjdfokj

XEP — dkshckdsjvlsdjvodsvjdfokj

XML — dkshckdsjvlsdjvodsvjdfokj

XMPP — dkshckdsjvlsdjvodsvjdfokj

e-mail — dkshckdsjvlsdjvodsvjdfokj

jabber — dkshckdsjvlsdjvodsvjdfokj

klient — dkshckdsjvlsdjvodsvjdfokj

presence — dkshckdsjvlsdjvodsvjdfokj

server — dkshckdsjvlsdjvodsvjdfokj

stanza — dkshckdsjvlsdjvodsvjdfokj

vCard — dkshckdsjvlsdjvodsvjdfokj

ID3v1 — dkshckdsjvlsdjvodsvjdfokj

ID3v2 — dkshckdsjvlsdjvodsvjdfokj

ASCII — dkshckdsjvlsdjvodsvjdfokj

utf-8 — dkshckdsjvlsdjvodsvjdfokj

Base64 — dkshckdsjvlsdjvodsvjdfokj

Příloha E

Stanza - základní schéma

Přehled základních elementů, které jsou využívány při Jabber komunikaci. Struktura jednotlivých částí stanzy ukazuje pouze prvky relativní k této práci. Pomocí hranatých závorek je znázorněna množina, ze které musí být vybrán právě jeden prvek. Na místě uvozovek se očekává jakákoliv povolená hodnota.

E.1 Iq

```
1       <iq from=""
2           to=""
3           type="[get,set,result,error]"
4           id=""
5         Namespace
6       </iq>
```

Algoritmus E.1: Popis elementu *iq*.

E.2 Message

```
1   <message from=""
2            to=""
3            type="[normal,chat,groupchat, headline, error]"
4            id=""
5     <body>  </body>
6     <x xmlns="jabber:x:event">
7       [Offline, Delivered, Displayed, Composing]
8     <subject>  </subject>
9     <thread>   </thread>
10    <error>    </error>
11    <x>        </x>
12  </message>
```

Algoritmus E.2: Popis elementu *message*.

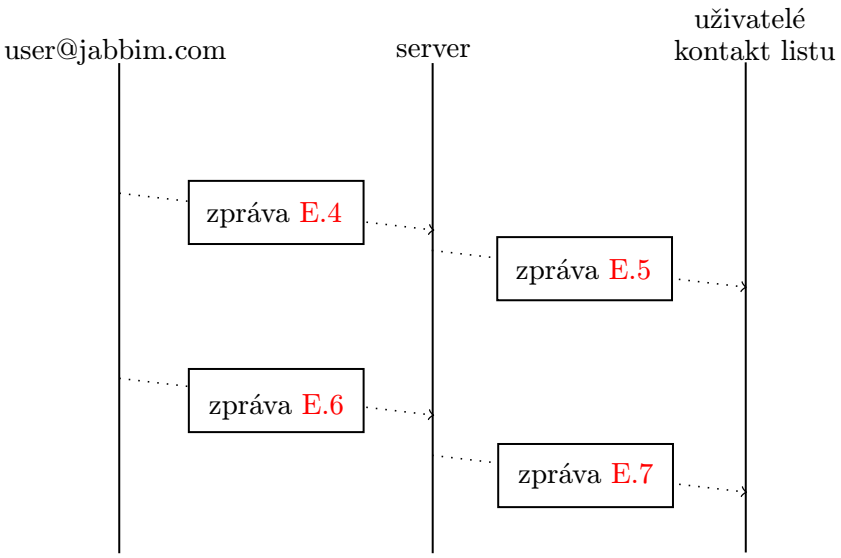
E.3 Presence

```
<presence from=""
          to=""
          type="[available, unavailable, probe, subscribe,
                 unsubscribe, subscribed, unsubscribed, error]"
          id=""
  <show>
    [away, chat, dnd, normal, xa]
  </show>
  <status>      </status>
  <priority>    </priority>
  <error>       </error>
</presence>
```

Algoritmus E.3: Popis elementu *presence*.

E.4 Přehled průběhu rozšíření

Ukázka celého příkladu šíření statusu pomocí rozšíření *User Tune*. Uživatel *user* poslouchá hudbu a informuje server zasláním zprávy zobrazené v příkladu E.4.



Obrázek E.1: Ukázka „šíření“ *User Tune*.

```
<iq from="user@jabbim.com" type="set" id="pub1">
   <pubsub xmlns="http://jabber.org/protocol/pubsub">
      <publish node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
               <track>5</track>
            </tune>
         </item>
      </publish>
   </pubsub>
</iq>
```

Algoritmus E.4: Informování serveru o právě přehrávající hudbě.

Server obdrží informace od klienta *user* zprávu o přehrávající hudbě. Pomocí elementu *message* ji přepošle všem uživatelům z kontakt listu uživatele *user*, kteří jsou pro odběr těchto typů zpráv zaregistrováni. Tato struktura zprávy je prezentována na příkladu E.5.

```
<message from="user@jabbim.com" type="set"
         to="jabinfo@jabbim.com/bot" id="pub1">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
              <track>5</track>
            </tune>
         </item>
      </items>
   </event>
</message>
```

Algoritmus E.5: Server informuje uživatele podporující rozšíření o stavu *user@jabbim.com*.

Zpráva o přehrávané hudbě je také přeposlána všem otevřeným spojením uživatele *user*, ukázáno na příkladě E.6.

```
<message from="user@jabbim.com" type="set"
         to="user@jabbim.com/doma" id="pub2">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune">
               <artist>Daniel Landa</artist>
               <length>255</length>
               <source>Nigredo</source>
               <title>1968</title>
               <track>5</track>
            </tune>
         </item>
      </items>
   </event>
</message>
```

Algoritmus E.6: Server přepošle informace o přehrávané hudbě všem otevřeným spojením uživatele *user@jabbim.com*.

Přestane–li uživatel *user* poslouchat/vysílat informace o přehrávané hudbě, provede to pomocí zprávy ukázané na příkladu E.7. Zpráva typu *iq*, ve které je položka *tune* nesoucí informace o skladbě prázdná.

```
<iq from="user@jabbim.com/prace" type="set" id="pub1">
   <pubsub xmlns="http://jabber.org/protocol/pubsub">
      <publish node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune"/>
         </item>
      </publish>
   </pubsub>
</iq>
```

Algoritmus E.7: Uživatel ukončil „vysílaní“ rozšířených zpráv o svém stavu.

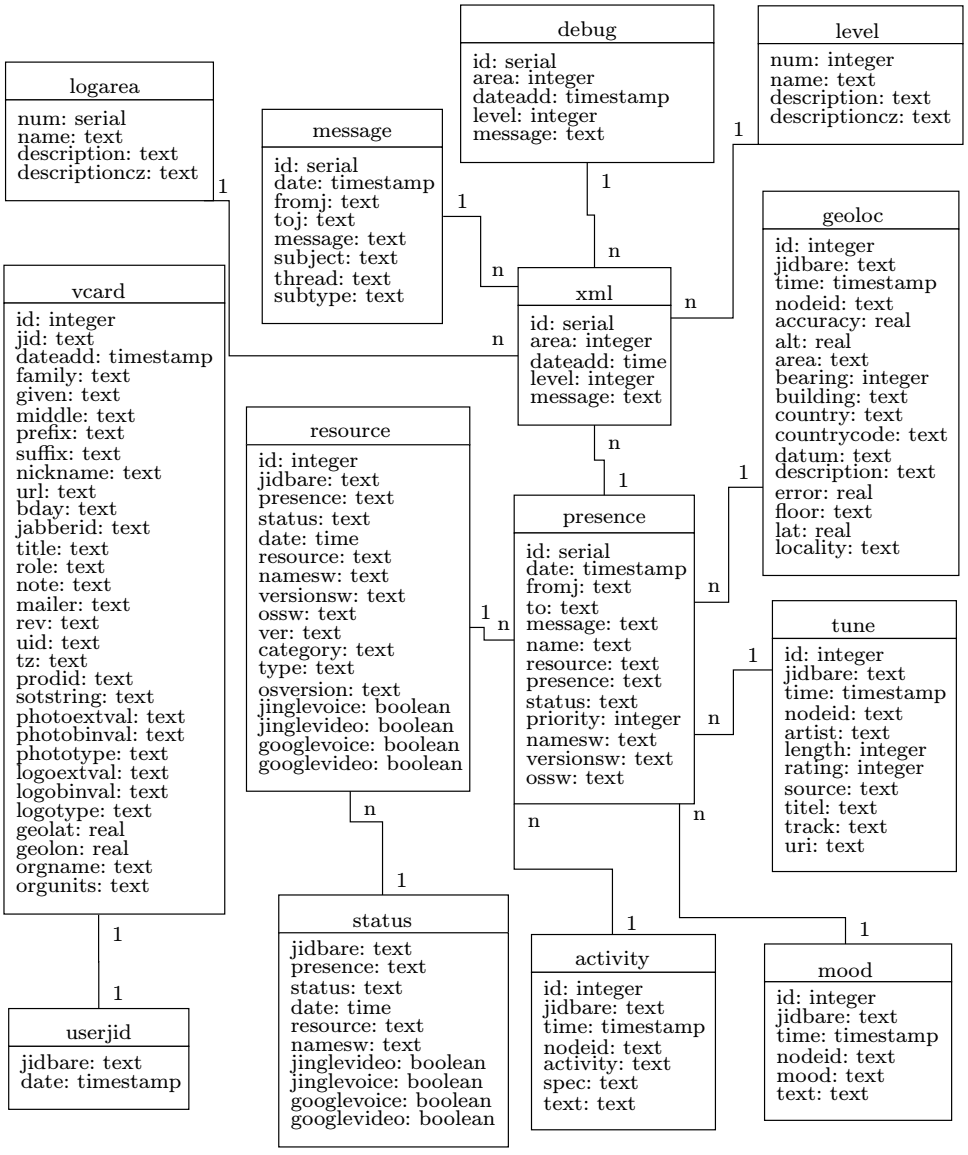
Server informuje všechny účastníky odběru zprávou, která má položku *tune* prázdnou. Tak jak to prezentuje příklad E.8.

```
<message from="user@jabbim.com"
         to="jabinfo@jabbim.com/bot">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="http://jabber.org/protocol/tune">
         <item>
            <tune xmlns="http://jabber.org/protocol/tune"/>
         </item>
      </items>
   </event>
</message>
```

Algoritmus E.8: Server informuje klienty o ukončení šíření rozšířeného statusu uživatele *user@jabbim.com*.

Příloha F

Návrh databáze



Obrázek F.1: Struktura databáze

Příloha G

Přehled klientů a jejich rozšíření

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
Adium										
Agile Messenger										
AQQ										
Ayttm										
beejive										
Beem										
BitlBee										
Bombus										
BuddyMob										
Chatopus										
Citron										
Claros Chat										
climm										
Coccinella										
Crosstalk										
Digsby										
eM Client										
emite										
Empathy										
Exodus										
Finch										
Gajim										
Galaxium										
glu										
GNU Freetalk										
Gossip										
iChat										
iJab										
IM+										
imov Messenger										
irssi-xmpp										
Jabbear										
Jabber Mix Client										
jabber.el										
Jabbim										
Jabbim for Android										
Jabiru										
JAJC										
Jappix										

Klient	OS	XEP--60	XEP--163	XEP--80	XEP--92	XEP--107	XEP--108	XEP--118	XEP--	XEP--
JBuddy Messenger										
Jeti										
Jitsi (SIP Communicator)										
JWChat										
Kadu										
Kopete										
Lampiro										
m-im										
mcabber										
mChat										
Miranda IM										
Monal IM										
OctroTalk										
OneTeam										
OneTeam for iPhone										
Oyo										
Pandion										
Poezio										
Pidgin										
Prodromus										
Psi										
Psi+										
Quiet Internet Pager (QIP)										
qutIM										
saje										
SamePlace										
Sim-IM										
Slimster										
SoapBox Communicator										
Spark										
SparkWeb										
Synapse										
Talkonaut										
Tigase Messenger										
Tigase Minichat										
Tkabber										
Tlen										
Trillian										
TrophyIM										
V&V Messenger										
Vacuum-IM										
Vayusphere										
WTW										
Xabber										
xmppchat										
Yambi										
Yaxim										

Tabulka G.1: Přehled podporovaných rozšíření u jednotlivých klientů.