Multiplatformní Instant Messanger Server+Client

Multiplatform Instant Messanger Server+Client

Bc. Libor Janota

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Libor JANOTA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Multiplatformní Instant Messager Server+Client**

Zásady pro vypracování:

1. **Vytvořte literární rešerši na téma Instant Messaging a zhodnoťte současný stav dostupných serverových a klientských řešení.**
2. **Navrhněte a vytvořte nový IM (server i klient) založený na multiplatformní softwarové knihovně wxWidgets demonstrující její vlastnosti a využívající XMPP (nebo jiný vhodný) komunikační protokol.**
3. **IM server by měl umožnit vícenásobné připojení stejného klienta.**
4. **IM server by měl umožnit archivaci historie.**
5. **Součástí práce bude specifikace a vytvoření knihovny wxXMPP (nebo podobné) integrující komunikační protokol do knihovny wxWidgets.**
6. **Vytvořte programovou dokumentaci Vašeho systému.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: UTB ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.**
2. **MASTERS, Jon. Linux profesionálně: programování aplikací. 1.vyd. Brno: Zoner Press, 2008. 539 s. ISBN 978-80-86815-71-8.**
3. **SMART, Julian, HOCK, Kevin. Cross-Platform GUI Programming with wxWidgets, Prentice Hall, 2006, ISBN 0-13-147381-6.**
4. **XMPP Standards Foundation na WWW (http://xmpp.org/).**
5. **wxWidgets Homepage na WWW (http://www.wxwidgets.org/).**

Vedoucí diplomové práce: **Ing. Michal Bližňák, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato diplomová práce pojednává o základních principech Instant Messagingu a jejich využití. Je zde uveden současný stav těchto technologií, jak ze strany aplikačních klientů, tak i serverových částí. Jsou zde popsány základní komunikační protokoly. Jedním z nich je protokol XMPP. Tento protokol je součástí praktické časti této diplomové práce. Tato část pojednává o návrhu a implementaci jednoduchého systému, který je na bázi Instant Messagingu. Jedná se o demo aplikace typu server a klient, které jsou založeny na multiplatformní knihovně wxWidgets. Tyto aplikace demonstrují základní použití této knihovny a implementují otevřený protokol pro síťovou komunikaci XMPP.

Klíčová slova:

Server, Klient, XMPP, Instant Messaging, wxWidgets, chat, XML, protokol, Jabber, ICQ, IM, síťová komunikace, sokety

ABSTRACT

This diploma thesis is about basic principles of Instant Messaging and its usage. There is described today's state of these technologies. In this thesis are client's and server's today's solutions. There are introduced basic communications protocols. One of them is XMPP protocol. This protocol is part of this thesis. This part is about suggestion and implementation simply communication system, which is a basic base of Instant Messaging. This application is a demo application of server and client type. Application is build on wxWidgets library. These applications demonstrate the basic usage of this library and there are implementing open XMPP protocol for net communication.

Keywords:

Server, Client, XMPP, Instant Messaging, wxWidgets, chat, XML, protocol, Jabber, ICQ, IM, net communication, sockets

Poděkování, motto

Děkuji panu Ing. Michalovi Bližňákovi, Ph.D., vedoucímu diplomové práce, za jeho rady, připomínky a všestrannou pomoc při řešení. Dále chci poděkovat své rodině, přítelkyni, kamarádům a známým za podporu při psaní této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Lidé a nejen lidé mezi sebou vždy měli a mají velkou potřebu komunikovat. Tato komunikace je dorozumívání předáváním informací. Mezi lidmi se provádí mluvenou nebo písemnou řečí a prostředky mimořečovými, např. mimika, gestikulace, apod. Dorozumívání je nezbytným předpokladem spolupráce, soupeření a výchovy, vytváření skupin a citových vztahů.

Již od počátků lidstva bylo dorozumívání velice důležité. Dříve tomu nebylo pomocí telefonů, internetu, či různých dnes vyspělých technologií, ale pomocí různých kouřových, akustických, či světelných signálů. K ukládání a předávání informací sloužily zářezy a uzly. Po objevení písma se lidé domnívali, že je zbaví povinnosti používat paměť. Znakové a hieroglyfické písmo představovalo obrazový výraz ústních významů a naučit se je bylo obtížné, proto čtení a psaní na sebe vázalo moc. Zásadní informační revolucí byl objev fonetické abecedy, který vedl k oddělení znaků i zvuků od jejich jazykového významu. Informace se dala převádět z jazyka do jazyka, z jedné kulturní oblasti do jiné.

Tyto a další pradávné komunikační techniky mohou pro nás být jen snem, dnes je přece jen známější telegraf, pošta nebo e-mail, telefon, rozhlas, radar, satelity a internet. Ten nepochybně již nyní zcela mění běh a rytmus našeho života, dává odumírat zažitou sociální inteligenci a rozmetá tradiční uspořádání pracovišť i bytů.

Velká část lidí v dnešní době bere internet tak, že si pomocí tohoto média zde mohou prohlížet různé stránky, číst různé knihy, nakupovat v internetových obchodech, či posílat emaily. Mnoho z nich však postupem času zjišťuje, že e-mail v mnoha případech není tak rychlý a bezprostřední, jak by bylo pro daného člověka v hodné. Proto je tu druh komunikace, využívající této sítě, který umožní komunikaci mezi více lidmi v reálném čase, přičemž účastníci této komunikace mohou být na druhém konci světa. Znamená to, že v dané chvíli bude sedět nějaký počet lidí u svých počítačů a budou spolu diskutovat tak, že budou psát zprávy, či například mluvit. Zároveň uvidí kompletní diskuzi a budou do ní moci kdykoliv vstoupit. Propojení mezi nimi bude zajišťovat internet či nějaká forma počítačové sítě.

Tato forma komunikace pak umožňuje vidět, kdo je připojen, neboli kdo je s vámi v danou chvíli schopen komunikovat, a poté můžete mezi sebou například posílat nejenom zprávy, ale i soubory, či si vyměňovat různé formy komunikace, co daná aplikace nabízí. Velkou výhodou takovéto konverzace oproti e-mailu je rychlost. Samozřejmě musí být

bráno v potaz, či daný člověk je schopen psát na klávesnici rychle, či naopak pomalu. To poté může být bráno jako určitý typ handicapu. Takováto forma komunikace je vlastně jistý druh tzv. chatování, jen k němu nemusíte používat internetový prohlížeč, nemusíte se připojovat na žádnou internetovou stránku a přihlašovat se do nějakých „chatovacích" místností. Jelikož zatím neexistuje český ekvivalent, tak se takovýto způsob komunikace nazývá **Instant Messaging** a o tomto druhu komunikace, využití, dnešního stavu a návrhu vlastního klienta, který umožní uživateli vícenásobné připojení, serveru, na kterém bude ukládána historie komunikace, je celá tato diplomová práce.

I. TEORETICKÁ ČÁST

1 INSTANT MESSAGING

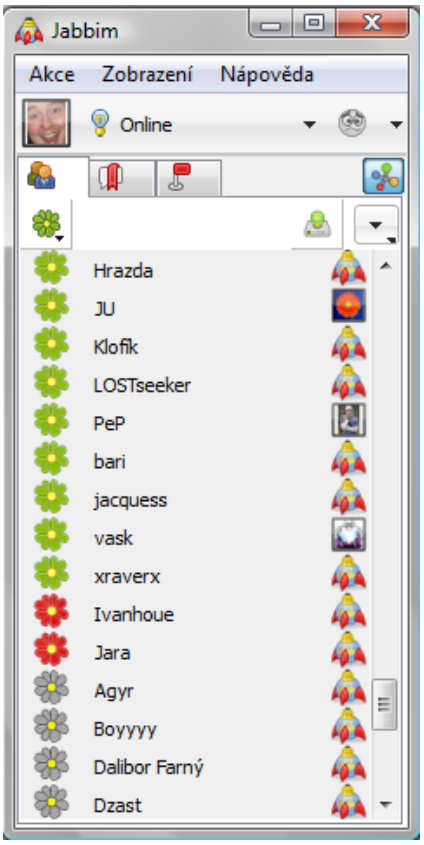
1.1 Co je Instant Messaging

Jak již bylo v úvodu zmíněno, tak Instant Messaging (dále IM) znamená, že jde o formu komunikace v **reálném čase**. V základu se jedná o textovou komunikaci mezi dvěma, či více osobami v počítačové síti, kterou může být internet, intranet, či jiný druh počítačové sítě. V dnešní době je však tato služba rozšiřována o přenos souborů, hlasu i obrazu a velmi často si můžeme psát či hovořit s více lidmi najednou. Jde o velmi efektivní metodu spolupráce, která má řadu výhod vůči obdobným metodám, jako je email nebo chat. Plně odpovídá heslu dnešní doby, kterým je "komunikace a spolupráce". IM systémů existuje celá řada, příkladem je ICQ, MS Messanger, Skype, AIM, Google Talk, Miranda či Jabber.

1.2 Princip

Princip a funkce takového systému spočívají v tom, že potřebujeme účet v daném systému umístěný na nějakém serveru, který schraňuje databázi uživatelů a zajišťuje komunikaci mezi vaším počítačem a ostatními účastníky diskuze. Jako první operaci si totiž musíte vytvořit své uživatelské jméno, pod kterým bude možné s vámi v rámci nějaké sítě komunikovat, nějaký klientský software, viz Obrázek 1. Většinou se jedná o software, který je napsán přímo pro danou aplikaci a pro konkrétní operační systém, ale může jít i například o webového klienta. Tento program zajistí, že vaše zprávy budou doručeny na zmiňovaný server, který je předá adresátovi. Komunikace probíhá pomocí tzv. komunikačního protokolu, který do detailů popisuje, jak budou formátovány příchozí a odchozí zprávy, jak bude vypadat komunikace při posílání souborů atd. To vše převádí do řeči protokolu právě klientská aplikace.

Zjednodušeně je tento princip v podstatě takový, že zasíláme textové zprávy, které druhá strana okamžitě obdrží, a může hned reagovat odpovědí. Jednoduše by se dal princip instant messagingu přirovnat k textovým zprávám (SMS) zprávám v mobilní telefonní síti.



Obrázek 1. Příklad klienta

1.3 Výhody

Jak je již zmíněno výše, tak IM má řadu výhod oproti jiné síťové komunikaci. Hlavní výhodou je to, že jde o komunikaci v reálném čase.

Hlavní výhodou oproti například emailu je, že můžeme vést víceméně online efektivní komunikaci, na což email není navržen. IM je optimalizován na zasílání krátkých zpráv, pomocí kterých vedeme konverzaci. Většinou si také můžeme ukládat historii.

Výhodou oproti klasickému chatu je, že máme seznam našich kontaktů a můžeme kdykoliv psát jednomu nebo více z nich. Ve většině systémů můžeme psát zprávy i kontaktu, který není připojen.

Dobrou vlastností většiny IM je tzv. presence, tedy informace o stavu přítomnosti a připravenosti k hovoru. Tento stav se nastavuje buď automaticky (pokud nějakou dobu nepracujeme na počítači) nebo ručně (např. nerušit).

Mnoho lidí však může navrhnout, že komunikace pomocí například pomocí telefonu je lepší. Z praxe se ale začíná potvrzovat, že IM je několika ohledech lepší. Buď můžeme odpovědět ihned, nebo si odpověď můžeme promyslet anebo máme možnost odpovědět až tehdy, kdy na to budeme mít čas. Můžeme jednoduše vést několik konverzací najednou. Navíc dnešní rozšíření IM dovolují zasílat soubory (třeba obrázky), přejít do hlasového nebo video hovoru. Jednoduše také můžeme zasílat odkazy, zdrojové kódy a obdobné věci, které se po telefonu těžko sdělují. Proto se IM stává čím dál více oblíbený, jak při firemní komunikaci, tak i při personálním použití.

1.4 Zneužití

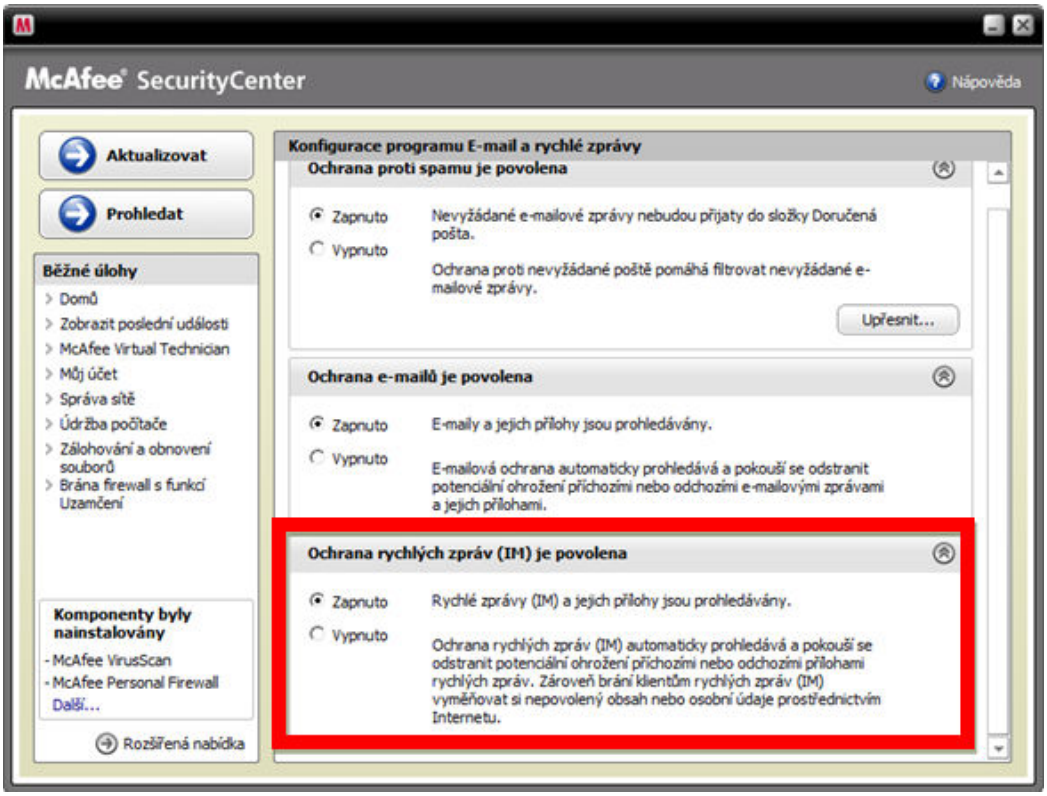
Jelikož si ale IM získává čím dál více popularity, tak díky své bezprostřednosti doplňuje či nahrazuje i oficiální firemní komunikaci, tudíž se stalo to, že začal výrazně lákat počítačové útočníky. Nejčastější zneužití IM představuje šíření klasické virové nákazy a z narůstajícího počtu takovýchto pokusů je vidět, že si v tomto útočníci našli zálibu.

Klasické šíření virů skrze IM víceméně stejný a jednoduchý scénář. V prvním kroku se dostupným kontaktům automaticky rozešle vybraná zpráva, která obsahuje především zahrnutý internetový odkaz. Ten může být ještě doplněn standardním textem, který apeluje na uživatele, ať klikne na tento odkaz. Vyzývavěji působí právě více strohá varianta. Působí více z tohoto důvodu, že neobsahuje výše zmíněný text, který může být například v anglickém, či ruském jazyce. A to člověka, který mluví jiným jazykem, již nějak upozorní na hrozící nebezpečí. Neznalý příjemce však na první pohled nic netuší, jelikož daný odkaz vypadá, jako by přišel od známého kontaktu, a tak není důvod, proč jej nenásledovat. Po tomto osudovém kliknutí se v nejlepším případě uživateli otevře okno prohlížeče, které nabídne nějaké zboží. V případě horším však dochází ke stažení škodlivého kódu, provedení tohoto kódu a dalšímu šíření opět v těchto kolejích. Tento škodlivý kód může mít za následek, že jsou poslána data o daném uživateli, jako jsou například jméno, příjmení, rodného čísla, emailu apod. Tato data, jsou pak nějak zneužita. Tento kód může také převzít kontrolu nad vaším účtem a účet vám ukrást.

Tento však případ není jedinou možností, kterou mohou současní útočníci zneužít ve světě instant messagingu. Příležitosti jim poskytují také samotní klienti pro různý

protokoly, jedná se totiž o aplikace jako každé jiné, které z principu nemohou být zcela zabezpečeny. Typickým případem zneužití zde může být hodně známé přetečení zásobníku (například zahrnutím speciálního řetězce do zprávy v konkrétní verzi vybrané aplikace) s následnou možností spuštění útočníkova vlastního kódu v rámci náchylného systému a s právy aktuálně přihlášeného uživatele, což může mít až katastrofální následky.

Naštěstí v současné době, kdy instant messaging se týká téměř každého, kdo pracuje s osobním počítačem a třeba internetem, postačuje použití zdravého rozumu při obdržení podezřelé zprávy, případně v kombinaci s kvalitnějším antivirem, viz Obrázek 2. Ty totiž často zahrnují komponentu pro sledování právě IM komunikace a dokážou zabránit případnému zneužití.



Obrázek 2. Příklad antivirového programu, umožňující filtraci IM

1.5 Historie

První zmínky o Instant Messagingu pocházejí z poloviny 60. let 20. století, kdy byly operační systémy jako CTSS nebo Multics, které používaly systémy k upozornění uživatelů, kteří byly připojeni na stejnou stanici, o tom, zda někdo používá tiskárnu, či nikoliv. Tyto systémy se celkem dobře ujaly a v sítích se začaly používat přímo mezi jednotlivými uživateli. Takovéto systémy používali peer-to-peer protokoly. Jednalo se například o programy talk, ntalk, či ytalk.

O pravém instant messagingu, se ale dá hovořit až od konce osmdesátých let, kdy ve Finsku v roce 1988 byla představena komunikační síť IRC. V době, kdy tato komunikace nebyla jen mezi nějakými výzkumnými sítěmi, ale začala se přemisťovat do sítě internet, tak se začaly tyto aplikace a sítě vyvíjet. Další a poměrně hlavní milník v této oblasti byl rok 1996, kdy byl uvolněn jeden z prvních volně dostupných programů pro instant messaging pro širokou veřejnost. Tento program byl ICQ, který patřil izraelské firmě Mirabilis. Díky jeho růstu, jej a celou firmu, zakoupila v roce 1998 firma AOL Instant Messenger za 407 miliónů amerických dolarů.

Během pár let bylo vyvinuto mnoho alternativních IM klientů, například Yahoo Messenger, MSN Messenger, Excite, Ubique, či bezpečnější program Lotus Sametime od firmy IBM, každý s vlastním protokolem, které byly pochopitelně navzájem nekompatibilní. Uživatelé proto museli provozovat několik klientů simultánně. Díky této skutečnosti se s velkým úspěchem proto setkali multiprotokoloví klienti jako Pidgin (dříve Gaim), Miranda, Trillian, SIM nebo Kopete. S ICQ je také kompatibilní nenáročný ruský klient QIP, který podporuje pouze tuto síť (v nové verzi QIP Infium už podporuje i síť Jabber a XIMMS), ale je stabilnější a snáze nastavitelný než Miranda, která ovšem podporuje mnoho rozšíření a daleko více protokolů. Oba dva tito klienti jsou pouze pro operační systém Windows. Rozdíly mezi protokoly stírá otevřený XMPP protokol pro IM (potažmo Jabber), který dokáže mimo jiné pomocí takzvaných transportů komunikovat i s ostatními sítěmi [2]

1.6 Dnešní stav

Jelikož je a bude tento způsob komunikace velmi populární, je zcela jasné, že musí existovat několik sítí, ve kterých se dá komunikovat. Tyto sítě může rozdělit do dvou hlavních skupin (ve smyslu, kdo tyto sítě provozuje a vyvíjí)

- **Komerční** (provozují je komerční firmy)
  - ICQ
  - MSN
  - AOL
  - SKYPE
  - …

- **Nekomerční** (rozuměj provozovány nadšenci či nevýdělečnými organizacemi nebo společnostmi, pro které není provozování této sítě hlavním zdrojem příjmů)
  - Jabber
  - IRC
  - …

Nevýhodou u těchto komunikačních sítí samozřejmě je, že tyto jednotlivé sítě nejsou mezi sebou kompatibilní, to znamená, že pokud si vytvoříte účet u jedné sítě, tak nebudete moci komunikovat s lidmi ze sítě druhé. Pokud potřebujete být v kontaktu i s lidmi ze sítě druhé, musíte si vytvořit registraci i tam a nakonec jich můžete mít několik.

Komunikační protokoly v rámci IM sítí mají stejná pravidla, která platí třeba pro software. Jsou zde protokoly plně uzavřené a neveřejně, např. Skype, nebo protokoly, které jsou sice otevřené, ale jejich použití se řídí speciálními podmínkami, např. protokol OSCAR v síti ICQ/AOL, a nakonec protokoly, které jsou plně svobodné, např. XMPP/Jabber.

Co se týče rozsáhlosti jednotlivých sítí, to nejde jednoduše říci. Nejrozšířenější síť totiž nelze jednoznačně změřit. Dalo by se to udělat podle počtu zaregistrovaných uživatelů? Určitě by to byla první otázka, co by každého napadla. Jak ale vyhodnotit lidi,

kteří mají účtů několik v rámci jedné sítě? Nebo podle počtu současně připojených uživatelů? Metodika měření se nedá tak snadno určit a vždy vychází z nějakých odhadů. Navíc komerční sítě někdy počty uživatelů nesdělují nebo jsou údaje zastaralé. Jednoznačně nemůžeme říci, která síť je největší. Tabulka 1 ukazuje celkovou statistiku používaných služeb ve světě.

Služba	Uživatelských účtů	Datum
AIM	53 milionů aktivních	Září 2006
	>100 milionů celkem	Leden 2006
eBuddy	35 milionů celkem	Říjen 2006, zahrnující 4 miliony mobilních uživatelů
Gizmo5	mobilů a PC	
Gadu-Gadu	6 milionů aktivních (hlavně v Polsku)	Červen 2008
IBM Lotus Sametime	17 milionů celkem (privátní, v podnicích)	Listopad 2007
ICQ	15 milionů aktivních	Červenec 2006
IMVU	1 milion celkem	Červen 2007
Jabber	40-50 milionů celkem	Leden 2007
Mail.ru Agent	1 milionů aktivních (denně)	Září 2006
Meebo	1 milionů celkem	Říjen 2006
mundu messenger	Přes 2 milionů celkem	2008
MXit	11 milionů celkem (9 milionů v Jižní Africe)	Leden 2009
Paltalk	3.3 miliony unikátních návštěvníků mesíčně	Srpen 2006
PSYC	1 milion aktivních (denně, hlavně v Brazílii)	Únor 2007. Část využívá i službu IRC
Skype	16 milionů online	Únor 2009
	309 milionů celkem	Duben 2008
Tencent QQ	40.3 milionů online (hlavně v Číně)	Březen 2008
	317.9 milionů "aktivních" (hlavně v Číně)	Březen 2008
	783 milionů aktivních účtů (hlavně v Číně)	Březen 2008
VZOchat	>550,000	Prosinec 2008
Windows Live Messenger (Dříve MSN Messenger)	300 milionů aktivních	
Xfire	11.5 milionů celkem	Říjen 2008
Yahoo! Messenger	248 milionů aktivních registrovaných Yahoo uživatelů	Leden 2008

Tabulka 1.: Světová statistika používaných služeb[2]

Čísla ve výš uvedené tabulce jsou někdy až 3 roky staré, tudíž tabulka je jen informativního charakteru, kolik zhruba daná síť má uživatelů.

1.7 Budoucnost

Blízká budoucnost instant messagingu je určitě taková, že tato služba bude zahrnuta ve více technologiích, než je doposud. Už nyní se ve velkém IM dostává do mobilních telefonů, buď jako přídavná aplikace a dokonce někteří výrobci už tuto službu začínají dodávat standardně. Tato služba má v mobilních telefonech určitě budoucnost, protože její provoz je několikrát levnější, než posílání krátkých textových zpráv.

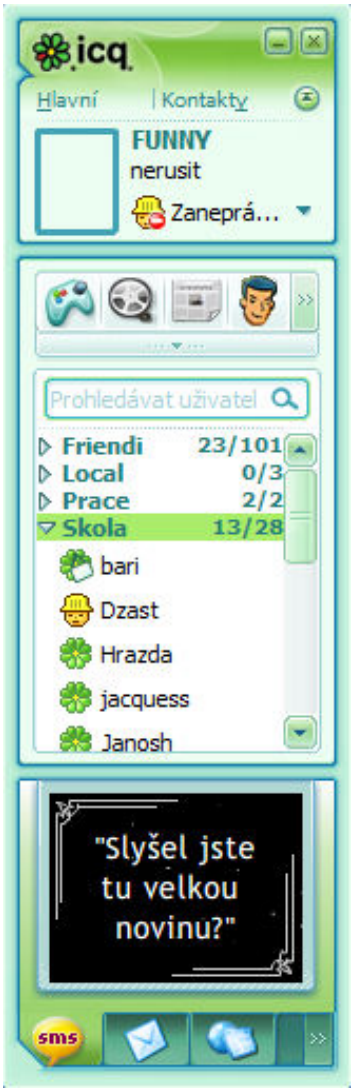
2 STÁVAJÍCÍ ŘEŠENÍ

Tato kapitola pojednává o základních sítí, klientských, případně serverových řešení, která jsou v dnešní době nejrozšířenější.

2.1 ICQ

ICQ je protokol pro instant messaging a zároveň komunikační software, viz Obrázek 3, vyrobený, jak je již zmíněno výše, izraelskou firmou Mirabilis a poprvé představený a uvedený v roce 1996. V roce 1998 firmu Mirabilis koupil nynější vlastník, firma AOL, za 407 milionů dolarů.

ICQ se vyslovuje jako „ajsíkjú“, což v angličtině zní jako „I seek you“, neboli „hledám tě“.



Obrázek 3.: Program ICQ

2.1.1 Funkce programu

Funkce programu ICQ zahrnují posílání:

- textových zpráv
- offline posílání zpráv
- chatování s více uživateli
- omezené odesílání SMS zpráv zdarma s denním limitem
- posílání souborů, pohlednic
- multiplayerové hry převážně v technologii Flash.
- vyhledávání uživatelů podle přezdívky (nicku), jména a ICQ UIN
- podpora protokolu POP3
- nově také uskutečňovat audio a video hovory
-

2.1.2 Základní informace

Velkou nevýhodou je, že komunikace není šifrována. Bez instalace oficiálního programu se lze na ICQ připojit také pomocí webového rozhraní ICQ2Go (ve Flashi a Javě), nebo pomocí alternativních klientů, např.: QIP, SIM, Pidgin, Miranda IM, meebo.com,…

ICQ uživatelé jsou číslování pomocí čísla UIN (Universal Internet Number nebo Unified Identification Number), jež je pro každého uživatele unikátní. Noví uživatelé nyní dostávají UIN přes 400-000-000. Nízká čísla z velké části vydražil eBay od uživatelů, jež se zaregistrovali velmi brzo. Menší část s nízkými čísly napadají hackeři za účelem dalšího prodeje.

2.1.3 Komunikace

ICQ síť je postavena na dnes neoficiálně otevřeném protokolu OSCAR, který je popsán v kapitole 2.1.4. Oproti Jabberu je však síť ICQ provozována trochu odlišně. Existuje jeden centrální server, který obsahuje kompletní databázi uživatelů. V posledních verzích ICQ klienta je možno se přihlásit pomocí UIN i pomocí e-mailu, který jste zadali při registraci svého uživatelského účtu.

2.1.4 Protokol OSCAR

OSCAR - Open System for CommunicAtion in Realtime.

Jde o binární protokol využívající TCP protokol. V předchozí kapitole jsem sice psal, že protokol je otevřený, ale oficiálně ne. Jeho specifikace je veřejnosti uzavřená i přesto, že název protokolu vypovídá o něčem jiném. V současné době tento protokol využívají dva klienti firmy AOL pro instant messaging: ICQ a AIM. Tato firma v minulosti vynaložila velké úsilí, aby se konkurentům jako je Microsoft nebo Jabber nepodařilo naimplementovat kompatibilní software pro posílání zpráv. V roce 2002 poskytla firma AOL firmě Apple Computer vlastní knihovnu, která umožnila kompatibilitu mezi produkty AOL a programem iChat firmy Apple Computer. Díky této skutečnosti klienti AOL mohli používat emailové účty na www.mac.com.

Další zpřístupnění protokolu třetím stranám se zatím nekonalo a dostupné specifikace protokolu na internetu vznikly jen po tzv. reverse-engineeringu (což v tomto případě ve zkratce znamená, že se odchytne komunikace originálního klienta a zpětně se dešifruje). Tato skutečnost způsobuje ne stoprocentní kompatibilitu s protokolem a tudíž nekorektní chování u alternativních klientů. Firmě AOL tak stačí menší změna protokolu a alternativní klienti přestávají fungovat. Firmě to však nevadí, protože používání jiných alternativních klientů je v rozporu s licencí.

2.1.5 Problémy

Pro mnoho lidí je však používání ICQ kontroverzní, protože používáním této sítě souhlasíte s podmínkami, které si AOL dává. V podstatě hlavními důvody jsou, že ICQ síť nesmí používat nikdo, kdo je mladší třinácti let, nesmíte ji používat ke komerčním účelům (tj. ICQ nesmí být používáno ke komunikaci se zákazníky, ani k firemní komunikaci) a navíc do sítě ICQ nesmíte přistupovat jiným než oficiálním klientem. Dále velký důvod, proč tuto síť nemá někdo rád je, že linuxová verze není k dispozici. Ovšem řada klientů pro Linux přesto existuje. Problém je v tom, že použitím těchto programů se dostáváte do konfliktu s podmínkami používání ICQ sítě. Jediná varianta, jak v souladu s licencí používat ICQ na linuxovém počítači, je využití ICQ2GO, což je klient, který se dá používat pomocí internetového prohlížeče a flashe. Navíc je oficiální klient zahlcen

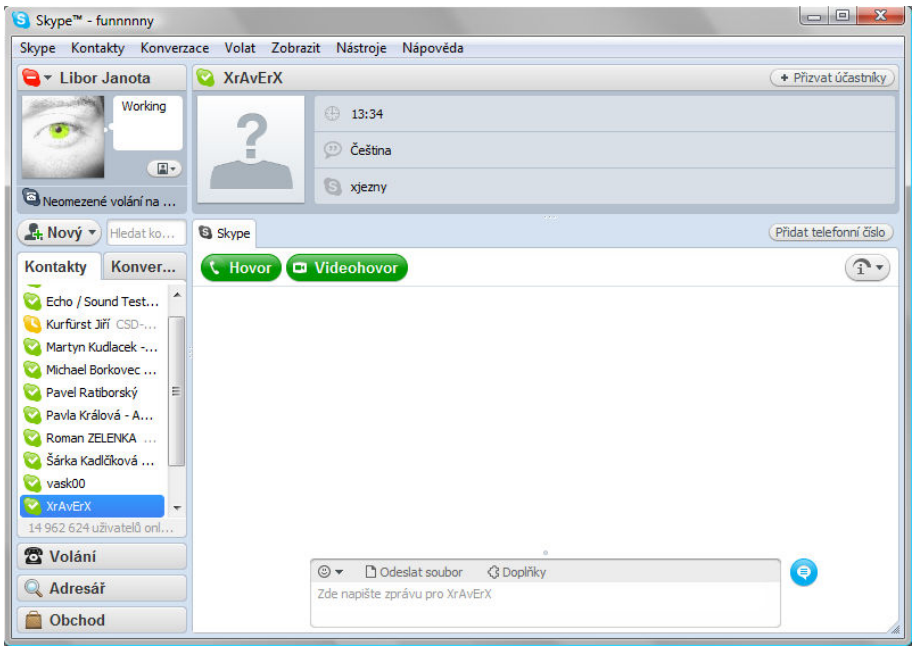
reklamou (různá okna s reklamou) a hodně uživatelů si na to stěžuje. Pokud se však nad tím zamyslíte, je to jeden ze způsobů, jak ponechat síť ICQ zdarma a vydělat si na provoz. V České Republice pokud se ale řekne instant messaging, tak si převážná většina lidí vybaví právě tuto sít.[3]

2.2 Skype

Skype je peer-to-peer program, umožňující internetovou telefonii (VoIP) a Instant messaging. Zaklad této aplikace byl napsán pomocí Estonských programátorů Ahti Heinla, Priit Kasesalu and Jaan Tallinn, kteří založili populární program Kazaa. Firma Skype byla založena Dány jménem Niklas Zennström a Janus Friis.

Původní název tohoto programu byl „Sky peer-to-peer“, který byl následně zkrácen na SKYPER. Jelikož ale doména skyper v několika zemích obsazena, tak název formy zkrátili na nynější SKYPE.

Oproti sítě ICQ má však tato aplikace značnou výhodu. Má oficiálního klienta, viz Obrázek 4, i pro operační systém Linux a Mac OS X. Vývoj aplikace pro operační systémy Windows, Linux a Mac OS sice neprobíhá synchronizovaně, ale i přesto, že verze mají úplně jiná čísla, tak jsou velmi podobné.



Obrázek 4.: Program Skype

2.2.1 Funkce programu

Funkce programu SKYPE zahrnují:

- Telefonování v rámci sítě Skype – Zdarma.
- Instant messaging – Zdarma. Zasílání zpráv a souborů mezi uživateli sítě.
- SkypeOut – Placená služba pro telefonování do tradičních telefonních sítí.
- SkypeIn – Placená služba, kdy je účastníkovi přiděleno telefonní číslo, na které se lze dovolat z tradičních telefonních sítí.
- VoiceMail – Placená služba. Poskytuje funkčnost hlasové schránky.
- Skype Video Calling – Videokonference mezi uživateli sítě Skype, dostupné od verze 2.0; zdarma.
- Skype SMS – Placená služba. Umožňuje posílat SMS na mobilní telefony.
- SkypeFind – Od verze 3.1 pro Windows. Služba umožňuje uživatelům vytvářet a hledat v databázi firem.
- Skype Prime – Od verze 3.1. Umožňuje nechat si platit za příchozí hovory (vyžaduje účet u PayPal).
- Call forwarding – Umožňuje přesměrovat hovory, když nejste online. Je zpoplatněno pouze na klasické telefony.
- Skype Extras – Doplňkové programy jako hry, nahrávání hovoru, sdílení pracovní plochy; za některé je nutno platit.
- Skypecasts – Velké hlasové konference (až 100 účastníků). Mohou být moderované (moderátor rozhoduje, kdo může právě mluvit).
-

2.2.2 Komunikace

Komunikace mezi jednotlivými klienty SKYPE probíhá decentralizovaně přes různé počítače zapojené v síti Skype, centrální server pouze ověřuje veřejný klíč uživatele při přihlášení do sítě. Komunikace je šifrována šifrou AES o délce klíče 256 bitů, provozovatel služby však může toto bez ohlášení, třeba i adresně, změnit.

Protokol komunikace ani zdrojové kódy programu nejsou veřejně dostupné, avšak povědomí o fungování protokolu je známé díky reverznímu inženýrství. K prolomení

komunikace však došlo polovině roku 2006. Tato skutečnost se povedla skupině čínských vědců.

Vlastní program pracuje jako klient i server. Případná bezpečnostní chyba může ohrozit celou síť Skype. Program může být také ovládán jinými programy přes zveřejněné API - pokud uživatel výslovně programu Skype povolí, aby byl přes toto API ovládán vnější aplikací, může to otevřít možnosti zneužití různými škodlivými programy (viry, spyware, malware,…)[4]

2.2.3 Problémy

Používat Skype prý není údajně úplně bezpečné, protože podle některých zdrojů někam odesílá sériové číslo základní desky. Dále je zde velká kritika ze strany lidí co sympatizují s OpenSource. Bohužel nejde jen o prostý fakt, že si můžete přečíst zdrojové kódy, ale především o další výhody, které z toho plynou. Běžného uživatele samozřejmě kód přímo nezajímá, ale jeho dostupnost má přímé důsledky, které se ho dotýkají, ať si je připouští, nebo ne. Přesně tohle je i případ Skype.

Níže jsou uvedeny základní nevýhody, které aplikace obnáší:

- Uzavřený protokol
- Centrální řízení služby

  Při technických potížích nebo výpadku v centrále je pak kompletně celá tato síť mimo provoz a nikdo nemůže například telefonovat.
- Kontrola nad tokem dat

  Cestu kudy data „potečou“ ovlivňuje server a vy jako uživatel nemáte vůbec ponětí, kudy data prochází a během hovoru se tato cesta může několikrát změnit.
- Funkce supernode

  Uživatelé, kteří sedí za proxy serverem, NAT nebo nějakým striktním firewallem, jsou *prosté nody*. Naopak ti, kteří jsou na slušné lince s veřejnou IP adresou, jsou povýšeni na takzvané *supernody*. Stanou se z nich uzly, které pak slouží jako ústředny pro přenos komunikačních dat a audio proudů. Což má za následek to, že aniž byste o tom věděli, tak je vaše linka

oslabována o tyto přenosy cizích. Pokud je člověk omezen nějakým datovým limitem, tak může nastat velký problém.

- Možnost odposlechu

  Firma provozující tuto službu může odposlouchávat vaši komunikaci, neboť to má v licenčních podmínkách dáno najevo, To mohou mít zařízeno dvěma způsoby. Mohou dát klientu příkaz, aby převedl komunikaci na konkrétní supernod a aby nešifroval nebo nějak oslabil šifrované spojení. Stačí k tomu použít omezenou množinu klíčů.

- Provoz Skype nelze jednoduše omezit

  Skype je od začátku navržen s cílem projít kudykoliv, jeho blokování je tudíž více než obtížné. To sice nahrává uživatelům, ale zároveň komplikuje život správcům sítí.

2.3 MSN

MSN, neboli také Microsoft Network, je sbírka služeb pro internet, kterou vyvíjí společnost Microsoft. Původně bylo MSN spuštěno 24. 8. 1995 při příležitosti prvního vydání operačního systému Microsoft Windows 95. Od té doby se rozsah poskytovaných služeb několikanásobně rozrostl. První poskytovanou služnou byla e-mailová schránka Hotmail následovaná instant messangerem MSN Messenger. Ten byl ovšem nedávno přejmenován na Windows Live Messenger.

MSN prošlo v roce 2006 velkou transformací na Windows Live, hlavně díky snaze Microsoftu zvýšit atraktivitu poskytovaných služeb pro širší masy obyvatel. Reagoval tak na velký zájem o instant messaging a služeb jemu podobných. V USA není MSN pouze poskytovatelem obsahu a provozovatelem vyhledávače. Poskytuje zde také internetové služby. Postupně jsou všechny současné služby MSN transformovány do nového rozhraní, kde zatím probíhá jejich betatestování.

Slovo MSN se vžilo hlavně jako synonymum k MSN Messenger, dnes Windows Live Messanger, viz Obrázek 5. Je to klient pro instant messaging od společnosti Microsoft, který je nyní k dispozici pro Windows XP, Windows Vista, Windows 7,

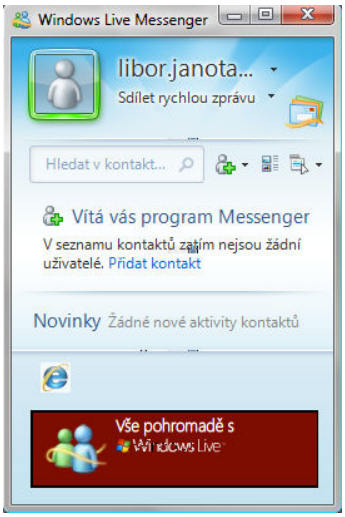
Windows Server 2003, Windows Server 2008 a Windows Mobile. Je k dispozici zdarma v licenci freeware.

Velkou výhodou tohoto klienta je synchronizace s Microsoft Outlook a plná integrace v systému Windows. Dále také umožňuje komunikaci prostřednictvím textových zpráv, video přenosů, chat, sdílení fotek, sdílení plochy a přenosu hlasu. Samozřejmostí jsou smajlíci. Specialitou je tzv. „šťouchnutí“ (způsobí zatřesení uživatelova okna) a mrknutí (animace v okně pro posílání zpráv). Komunikovat lze i s uživateli Yahoo! Messenger, který používá 25 milionů uživatelů.

Windows Live Messenger používá aktivně 291 milionů uživatelů po celém světě (2008). Celkový počet registrovaných uživatelů je 452 milionů (2008).[5]

2.3.1 MSN protokol

Windows Live Messenger pracuje na protokolu MSNP, který byl k těmto účelům navrhnut firmou Microsoft. Microsoft Notification Protocol funguje přes TCP/IP spojení, v některých případech používá i HTTP, k připojení na NET Messenger Service. Nyní je k dispozici verze protokolu s číslem 15. Tento protokol využívají i klienti třetích stran, jako je například Pidgin nebo Trillian. V této nové verzi je zaveden nový a bezpečnější druh autorizace. Protokol taktéž není zcela otevřený veřejnosti.



Obrázek 5.:Windows Live Messanger

.

2.4 Jabber

Jedním z IM systémů je Jabber Instant Messaging and Presence technology (Jabber). Jabber je založen na protokolu Extensible Messaging and Presence Protocol (XMPP), což je otevřený standardizovaný protokol založený na XML. Tento protokol je popsán v kapitole 3.3 XMPP protokol.

Díky tomu, že se jedná o otevřený standard, může kdokoliv vytvářet aplikace, které využívají XMPP. Situace je tedy obdobná jako v případě emailu a protokolu SMTP. Navíc princip funkce i použití je obdobné jako u elektronické pošty. Tyto výhody vedou k tomu, že Jabber je dnes (podle některých studií) nejpoužívanější IM systém. Navíc je asi jediný založený na standardizovaném protokolu.[6]

Velkou výhodou Jabberu je jeho otevřenost a univerzálnost. Můžeme ho jednoduše používat pro soukromé účely jako jiné IM systémy (a nahradit například ICQ). Prostě si najdeme nějakého veřejného poskytovatele (server), kde si zřídíme (zdarma) účet. Ale stejně dobře může být Jabber nasazen ve firemním prostředí a můžeme tak vybudovat komunikační infrastrukturu, která může být uzavřená nebo připojená do veřejné sítě (a komunikovat s ostatními účty v Jabberu).

Protože síť není nijak centralizovaná a jedná se vlastně jen o protokol, neexistuje ani žádný oficiální klient. U této sítě to však není zapotřebí, její používání není nijak omezeno na jeden program, můžete si dokonce napsat i svou vlastní aplikaci a používat ji. Z toho pramení jeden drobný problém, protože tím, že je možno si vytvořit aplikaci sám, nemusíte implementovat všechny vlastnosti komunikačního protokolu, a proto se občas stává, že ne vše je v daném programu implementováno. S většinou nejpoužívanějších klientů byste však neměli narazit na nějaké větší problémy. Navíc je také podstatné, zda danou službu podporuje i váš Jabber server, ke kterému se připojujete. Týká se to především tzv. Transportů (popsány níže), což je sada funkcí, která vám umožní komunikovat v rámci různých jiných IM sítí.

2.4.1 Funkce a princip Jabberu

Základní princip tohoto systému je trošku jiný, než jsou výše uvedené IM systémy, které jsou napojeny centralizovaně na hlavní server. Zapojení sítě Jabber je obdobné jako u emailu. Máme řadu serverů a při komunikaci je kontaktován server, na kterém je cílový učet.

Jak je výše uvedeno, princip komunikace v Jabberu je podobný principu přenosu elektronické pošty. Důležitým termínem je Jabber ID (JID), jednoznačný identifikátor v síti Jabber. Je stejná jako emailová adresa. Její syntaxe je jméno@server. Příklad tedy může vypadat takto FUNNNY@jabbim.cz.

Popis syntaxe:

- *Jméno*

  Jméno nebo přezdívka uživatele, unikátní na daném serveru

- server

  Jméno serveru (domény), kde má uživatel vedený účet nebo transport. Jinak řečeno adresa, kam je směrována komunikace server - server.

Celý systém je založen na distribuovaných serverech po internetu, ke kterým jsou připojeni klienti, a které mezi sebou komunikují. Je však možno použít i pouze jeden server uvnitř organizace a komunikovat pouze v rámci této organizace. Výhodou je, že výpadek serveru ovlivní pouze uživatele, kteří mají účet na tomto serveru. Oproti tomu v případě centrálních serverů, kde výpadek ovlivní celou část klientů.

Organizace mají svůj vlastní server, kde mohou nastavit požadované politiky a bezpečnost pro komunikaci klientů v rámci tohoto serveru. Dále existují veřejné servery, kde je možno si vytvořit účet. Příkladem je Google Talk, který využívá XMPP a je tedy kompatibilní jak s klienty, tak s celou sítí, i když přináší některá vlastní rozšíření. V Čechách jsou veřejné servery například jabber.cz či jabbim.cz.

Jak probíhá komunikace, když chci zaslat zprávu jednomu příjemci:

- klient se přihlásí k serveru, kde se ověří
- pokud pošlu zprávu, tak ji můj klient zašle na můj server
- ten provede směrování
- spojí se s cílovým serverem (pokud není cílový on sám), dle adresy serveru v JID
- odešle zprávu na cílový server
- cílový server předá zprávu klientovi (ve chvíli, kdy bude přihlášen)

2.4.2 Transporty v Jabberu

Transporty v síti Jabber jsou zjednodušeně řečeno od toho, aby spojili více sítí dohromady. Například když někdo používal delší dobu ICQ a rozhodne se přejít na Jabber, tak nechce ztratit kontakty, které měl v ICQ. Pro současné používání Jabberu a dalších IM máme dvě možnosti:

- **multiklient**

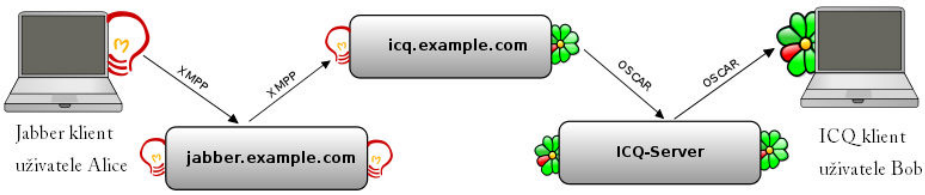
  Máme jednoho klienta, který se umí připojit do několika sítí najednou. Příkladem je Miranda IM, Trillian či Pidgin. U ICQ však porušujeme licenční ujednání.

- **transport**

  Na serveru můžeme mít třeba ICQ transport, což je nějaká brána do jiné sítě. Klient se připojuje k našemu serveru přes Jabber a ten teprve zprostředkovává komunikaci do jiné sítě. Tyto transporty jsou ale dle mě taky v rozporu s licencí ICQ. Bohužel se u těchto transportů sem tam objeví nějaká chyba.

Výše uvedené transporty jsou ale mnohem univerzálnější. Jsou to určitá rozšíření serveru, která umožňují komunikaci s nějakým jiným systémem. Existují například transporty pro informace o počasí, novinkách či posílání SMS.

Na níže uvedeném obrázku (Obrázek 6.) jde vidět, jak fungují u jabberu. Uživatel Alice vyšle zprávu Jabber serveru. Ten pošle díky transportu zprávu na bránu icq.example.com a tato brána se postará o doručení na ICQ server. Pak už je jen v jeho kompetencích doručit zprávu uživateli Bob.



Obrázek 6.: Princip transportů v síti Jabber

2.4.3 Jabber klient – Jabbim

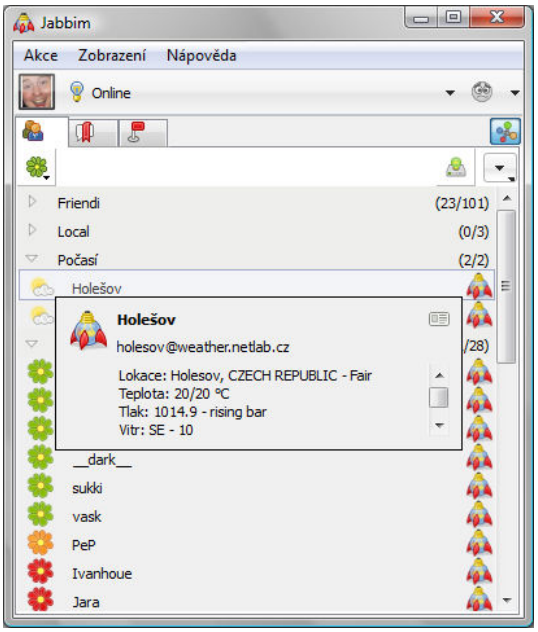
Jabbim je český klient pro Jabber, jehož vývoj je podporován provozovatelem stejnojmenného Jabber serveru. Jeho cílem je přiblížit běžným uživatelům PC moderní a užitečné funkce, které nabízí Jabber. Proto také podporuje přidávání nových funkcí v podobě modulů (pluginů). Je to klient, který je napsán v Pythonu a je multiplatformní – jde pod Linuxem a Microsoft Windows. Je zaměřen v prvé řadě na běžné uživatele, proto se také plánuje podpora zábavních her. Tento program je napsán pod licencí GNU GPL.

2.4.3.1 Základní funkce

- Sdružování konverzací do jediného okna s panely (taby)
- Formátování zpráv (XHTML-IM)
- Grafické emotikony
- Přenos souborů
- Kontrola soukromí, blokování kontaktů (Privacy Lists)
- Obousměrná indikace aktivity diskutujících (Chat State Notifications)
- Konference (Multi-User Chat, groupchat), uzpůsobené rozhraní pro jejich hledání, pokročilé moderování
- Záložky pro oblíbené konference a možnost automatického připojování do nich
- Možnost konvertovat běžnou konverzaci (1-to-1 chat) na konferenci
- Ovládání služeb a jiných klientů pomocí speciálních příkazů (Ad-Hoc Commands[1])
- Neviditelnost
- Rozhraní pro hledání a registraci služeb a transportů (Service Discovery)
- Šifrování pomocí TLS
- Rozšířené statusy (User Tune, User Mood, User Activity, User Chatting)

Na obrázku č. 6 je ukázka programu Jabbim. Jak je možno vidět, je zde jako příklad uveden detail u „kontaktu“ Holešov. Jabbim díky transportům dokáže číst a zobrazovat aktuální počasí ve vybraných městech.

[1] umožňují nechat si posílat soubory z Jabbimu na jiného klienta s podporou ad-hoc příkazů nebo například vzdálenou změnu svého stavu. Jabbim také do jisté míry zvládá remote controlling, takže se dá klient ovládat na dálku.



Obrázek 7.: Program Jabbim

2.4.4 Jabber server – OpenFire

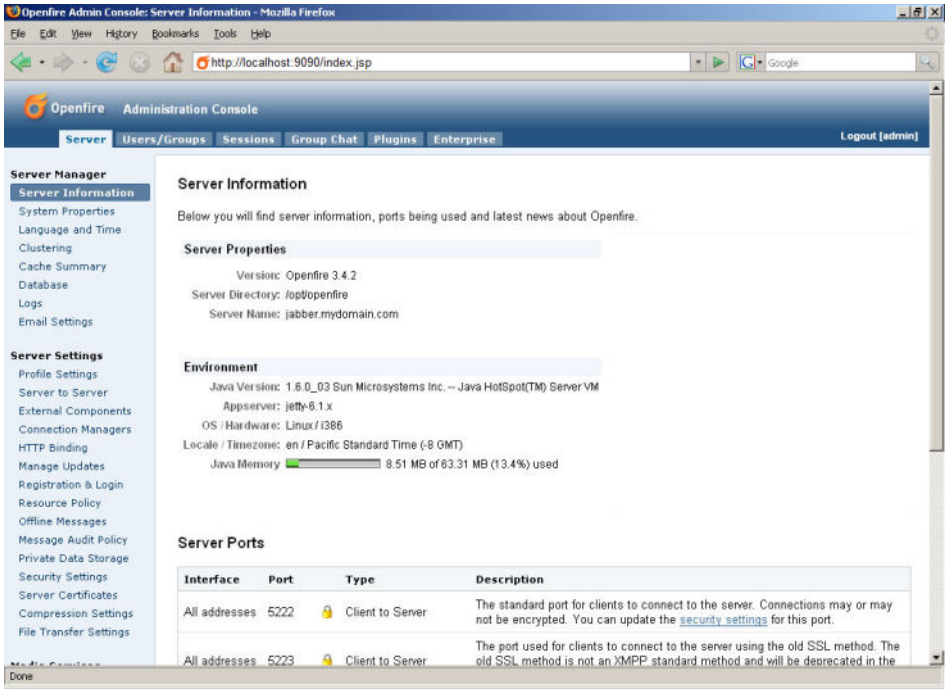
Serverů pro jabber není mnoho, ale existují například.: jabberd2, ejabberd, OpenFire, jaberd14, atd… Podle mě za zmínku stojí určitě vcelku jednoduchý server Openfire. OpenFire je XMPP/Jabber (dříve Wildfire) vyvíjený společně s klientem Spark společností Jive Software. Základní verze serveru je k dispozici pod svobodnou licencí a se zdrojovými kódy. Navíc je možné i licencování komerční, které vám zajistí přístup k celé řadě zajímavých uzavřených rozšíření, jako je Clustering Plugin nebo Openfire Enterprise.

Tento server je velmi snadný na instalaci i na provoz, proto se hodí jako snadná a plně funkční alternativa instant messaging pro firemní prostředí. Má snadné a přehledné webové rozhraní, které je lokalizované do češtiny.

Funkčnost serveru:

- Admin rozhraní
- Statistika serveru
- Authentizace
    - Certifikáty
    - Kerberos
    - LDAP (AD nebo OpenLDAP)
    - PAM

  - Radius
- Datové úložiště
  - Active Directory
  - LDAP
  - MS SQL
  - MySQL
  - Oracle
  - PostgreSQL
- SASL
  - ANONYMOUS Mechanism
  - DIGEST-MD5 Mechanism
  - PLAIN Mechanism



Obrázek 8.: Server OpenFire

2.4.5 Jabber server – Jabberd14

Jedná se o originální implementaci 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.

2.4.6 Jabber server – Ejabberd

Jedná se v současné době o jabber servere 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 noho 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).[7]

2.5 Alternativní klienti

2.5.1 Miranda

Miranda je multi-protokolový instant messenger. Podporuje všechny známé a nejvíce rozšířené IM. Z počátku na uživatele působí velmi stroze a zastarale. Nebarevný, hranatý, žádné ovládací prvky atd. Na druhou stranu je tím pádem nenáročný na hardware. Její jednoduchý vzhled je spíše výhodou než nevýhodou. Zobrazuje jen podstatné věci a neotravuje uživatele zbytečnostmi, jako jsou např. reklamy. Její filozofií je rozbalit a používat - což je výhodné např. na počítačích, kde uživatel nemá práva k instalaci programů či chce mít své povídátko vždy u sebe, např. na USB flash disku (pokud se nepoužívá mnoho pluginů, tak dokonce i na běžné disketě). Další důvod proč je miranda tak oblíbená, je, že podporuje paginy a těch může být nespočet. Uživatel si tak může svoji instalaci přetvořit k obrazu svému, od přidání smajlíků, odesílaní sms, budíku, psaní poznámek, po nejrůznější statistiky. Jak již bylo uvedeno, Miranda umí spoustu nadstandardních věcí, které si ale uživatel musí doinstalovat a nakonfigurovat. Klient je však pouze ve verzi pro Microsoft Windows.

2.5.2 Kopete

Kopete, bylo navrženo pro integraci do prostředí KDE. Je to multiprotokolový klient (dohromady podporuje asi 10 protokolů – např. Jabber, ICQ, AIM, IRC či Yahoo Messenger). Je častou volbou lidí, kteří potřebují komunikovat v síti ICQ (která je u nás velmi rozšířena) a zároveň mají existující Jabber účet.

Obrázek 9.:Klient Kopete

2.5.3 SIM

Tento klient pochází z Ruska. Tvůrci si zřejmě řekli, že v jednoduchosti je krása a celý design tomuto podřídili. Klient má hranaté tvary, nepříliš dobře zvolené barvy, ikony z prvních verzí originálního klienta. I tento klient podporuje pluginy (je jich ale menší množství než pro Mirandu). Největší přednost tohoto klienta však spočívá v možnosti být přihlášen k více účtům na stejném protokolu. Mohu být tedy přihlášen jak já, tak třeba i můj bratr, pokud zrovna oba potřebujeme být ve stejnou chvíli přihlášeni.

Existuje verze pro Windows i pro Linux.

2.5.4 Pidgin

Dříve jako Gaim. Opět je to multiprotokolový klient, který podporuje řadu používaných IM sítí. Funkce jsou opět standardní, funkčnost lze navíc rozšiřovat pomocí zásuvných modulů. Z netradičních funkcí bych zmínil třeba gesta myší (mouse gestures), šifrování, integrace s programem Evolution či správcem souborů Nautilus.

Pidgin běží na MS Windows, Linuxu a na BSD.

2.5.5 Google Talk

Jedná se o jednoduchý instant messenger a zároveň i komunikační služba společnosti Google. Tato služba je založena na protokolu XMPP/Jabber. Nabízí klasické služby instant messagingu a k tomu ještě službu VoIP, která je postave na protokolu Jingle. V souvislosti s těmito službami pak poskytuje i možnost videochatu, sdílení souborů a hlasové schránky.

Služba je zcela zdarma. Podmínkou však je založit si účet u jmenované společnosti, čímž získáte email. Tato podmínka je proto, že služba je vázána na tento email a tyto dvě služby úzce spolupracují (sdílení kontaktů, oznamování nové pošty, integrovaný klient ve webovém rozhraní)

Aplikaci lze používat v operačních systémech Windows 2000, XP a Vista.

3 ZÁKLADNÍ TECHNOLOGIE A PRINCIPY

Tato kapitola pojednává o základních prvcích, pomocí kterých byl program, který je přílohou této diplomové práce, vytvořen. Program je napsán pomocí multiplatformních knihoven wxWidgets. Jeho komunikaci zajišťuje protokol XMPP, který je postaven na XML technologii. Všechny tyto základní části jsou zde popsány.

3.1 wxWidgets

Vývoj začal v roce 1992 díky Julianu Smartovi, který je dodnes vývojářem jádra. Dříve se knihovna jmenovala wxWindows, ale díky tlaku Microsoftu se v roce 2004 knihovna přejmenovala na wxWidget. Knihovna wxWidgets je zajímavá a vcelku jednoduchá (po stránce aplikace a využívání) knihovna nabízející vytváření grafického uživatelského rozhraní k vašim programům. Je to programovací nástroj pro vývoj aplikací s grafickým uživatelským rozhraním, jak pro stolní počítače, tak i pro mobilní zařízení. Tato knihovna je framework, který za uživatele udělá spoustu rutinní práce a postará se o standardní chování vašich aplikací. Knihovna wxWidgets obsahuje velký počet tříd a metod ať už pro přímé použití či pro další úpravy. Typické aplikace zobrazují okno se standardními ovládacími prvky, často zobrazují speciální obrázky a grafiku a reagují na vstup z klávesnice, od myši či z jiných zdrojů. Taktéž mohou komunikovat s dalšími procesy nebo je přímo ovládat. Jinak řečeno, wxWidgets dělají programátorům vývoj standardních aplikací využívajících moderních technologií relativně snazší.

Ačkoliv je wxWidgets často označován jako vývojový nástroj pro tvorbu GUI aplikací, ve skutečnosti je mnohem více a má mnoho užitečných vlastností pro různá stádia vývoje aplikací. Což je velmi vhodné, neboť veškeré wxWidgets aplikace musí být schopné běhu na různých platformách a to se netýká pouze GUI. wxWidgets poskytuje třídy pro práci se soubory, s proudy, více vlákny, nastavením aplikace, pro komunikaci mezi procesy, online nápovědu, přístup k databázi a mnoho dalšího.[8]

wxWidgets se liší od mnoha frameworků, například MFC či OWL, především tím, že je již primárně vyvíjen multiplatformě. Aplikační programové rozhraní je v případě wxWidgets totožné, nebo alespoň velmi podobné na všech podporovaných platformách. Důsledkem toho je, že můžete svoji aplikaci vyvinout v prostředí MS Windows a s minimálními, a nebo vůbec žádnými, úpravami ji pak úspěšně přeložit a spustit v Linuxu či pod Mac OS X. Toto je veliká výhoda oproti přepisování aplikace pro každý další

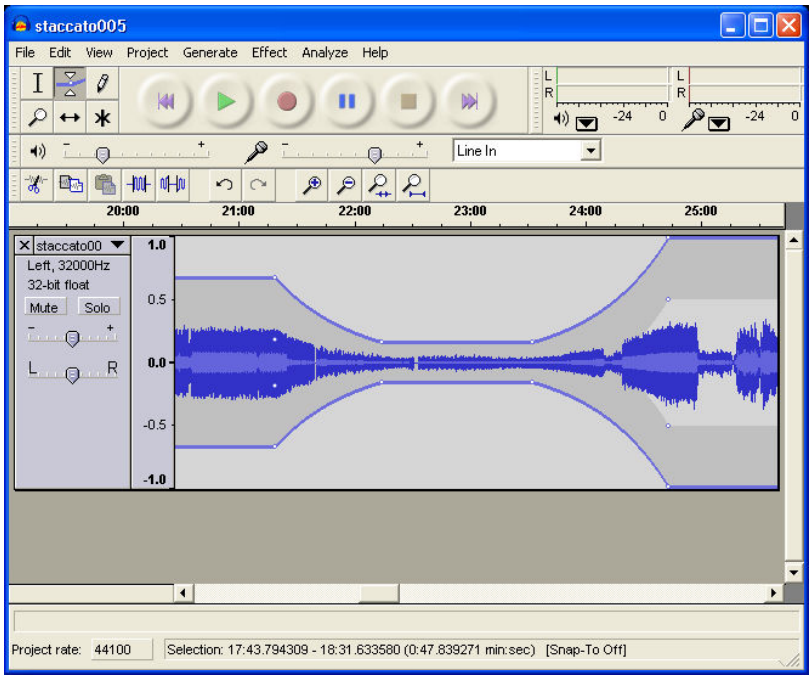
podporovaný systém. Také to znamená, že se nepotřebujete učit rozdílná API různých platforem. Navíc vaše aplikace nezastarají. Stejně tak jako se vyvíjí programátorské prostředí a technologie, mění se i wxWidgets. Tímto je umožněno portování vašich aplikací na nejnovější a nejlepší systémy při využití jejich nejnovějších vlastností.

Další vlastností, kterou se wxWidgets odlišuje je to, že poskytuje aplikacím nativní vzhled a chování. Některé frameworky využívají pro ovládací prvky stejný kód na všech platformách. Nativního vzhledu pak dosahují za pomoci témat pro konkrétní platformu. Oproti tomu wxWidgets používá kdekoliv je to jen možné nativních ovládacích prvků a vlastní až v případě že to není možné. wxWidgets tedy nejen poskytuje nativní vzhled, wxWidgets opravdu nativní je. Z hlediska přijetí uživatelem je toto nesmírně důležité. Jakýkoliv, i malý a téměř nepostřehnutelný rozdíl v chování aplikace od standardu pro danou platformu, může pro uživatele znamenat nepříjemnou zkušenost v podobě pocitu něčeho nepřirozeného.

Knihovna wxWidgets je použita pro C++, kromě tohoto existuje podpora řady dalších programovacích jazyků, jako například:

- Python (alternativa wxPython)
- Perl (alternativa wxPerl)
- Ruby (alternativa wxRuby)
- Java (alternativa wx4j)
- C#

Na níže uvedeném obrázku (Obrázek 10) je příklad, jak takováto aplikace ve wxWidgets může vypadat. Na každé platformě bude její funkce téměř stejná.



Obrázek 10.: Příklad aplikace napsané pomocí wxWidgets

Zde je ukázka pár nejznámějších programů, které jsou založeny na wxWidgets

- aMule - 'All-platform' P2P klient založený na eMule
- Audacity - svobodný/open source editor zvuků
- Audiobook Cutter - Uživatelsky příjemný separátor MP3 audioknih.
- Chandler - Personal Information Manager (PIM) zahrnuje kalendář, emaily, kontakty, úlohy a instant messaging
- Code::Blocks - svobodný/open source, multiplatformní C++ IDE
- e - Textový editor pro Windows
- Fityk - open source a multiplatformní vědecký nástroj pro kování křivek
- Mahogany - open source a multiplatformní emailový klient
- Monolith - nástroj k distribuci copyrightových dat
- MUTE - anonymní sdílení souborů
- TreeMaker - designovací program Roberta Langa na tvorbu origami.
- WASTE - bezpečný P2P software pro malé sítě
- wxCommunicator - multiplatformní SIP softphone

3.2 XML

3.2.1 Historie

Koncem 60. let řešila firma IBM problém, jak ukládat velké množství dat, právních dokumentů. Bylo potřeba vymyslet formát, který by měl dlouhou životnost, nebyl závislý na používaných programech atd. Vytvořili obecný značkovací jazyk, který se neustále vyvíjel a v roce 1986 z něj vyšel jazyk, který se jmenoval SGML. Tento jazyk byl přijat za ISO normu. SGML bylo velice flexibilní, aby vyhovělo různým požadavkům. To byla jeho velká přednost, ale zároveň i největší slabina. Vývoj aplikací, které by plně podporovaly SGML, byl velice nákladný. Proto se SGML používalo jen u velkých projektů, například ve vojenském, či leteckém průmyslu.

Z jazyku SGML pak vznikl jazyk HTML, který z počátku obsahoval jen malou množinu tagů, které se rozšiřovaly. Tento jazyk se používal a stále používá při tvorbě hypertextových dokumentů. Jak je již zmíněno, tak se jazyk HTML rozšiřoval, aby vyhovoval více a více uživatelům a programátorům webových aplikací, stránek. Jelikož ale narůstaly požadavky a vznikaly různé webové prohlížeče, tak začal být problém s kompatibilitou a dané webové stránky se musely dělat ve více verzích, aby byly uspokojeny všechny prohlížeče. Konsorcium W3C se snažilo tomuto trendu zabránit tím, že by právě pomocí SGML přesně definovalo, které tagy a kde mohou webové stránky obsahovat. Vznikly tak postupně jazyky HTML 2.0, HTML 3.2 a HTML 4.0. Výrobci prohlížečů však stále přidávali nové tagy a působili tím značné komplikace především autorům stránek.

Z tohoto všeho vyplývá, že tento stavy byl neudržitelný a web potřebuje nové technologie, která je hodně striktní avšak mnohem flexibilnější než je jazyk HTML. Proto vznikl jazyk XML, jehož finální verze byla na svět uvedena v roce 1998. XML je vlastně jednodušší verze příliš složitého a komplikovaného jazyka SGML. Flexibilita však zůstala zachována, a tak XML na rozdíl od HTML umožňuje tvorbu dokumentů s vlastními tagy a s možností validace dokumentů oproti DTD.

3.2.2 Základní syntaxe

Syntaxe je poměrně jednoduchá, avšak velmi přísná.

- Celý dokument musí být uzavřen do nějakého párového kořenového tagu. V příkladě níže je to tag <dokument>

  <dokument>
  ...tělo dokumentu
  </dokument>

- Všechny tagy musí být párové, to znamená, že pro počáteční tag <něco> musí vždy existovat i ukončovací tag </něco>. Vyjma elementů bez obsahu. Ty musí mít tag však ukončen znaky `/>' místo pouhého `>'. To umožní parseru snadné rozpoznání prázdných elementů. Příkladem prázdného elementu může být například nový řádek v jazyce HTML

  *
*

- V XML může mít tag atributy podobně, jak je známe z HTML. Tyto atributy musí být ohraničeny do jednoduchých nebo dvojitých uvozovek. Jednoduchá uvozovka musí být uzavřena jednoduchou a dvojitá dvojitou. Opačný pár uvozovek může být použit uvnitř hodnot.

  <param security="public">...veřejně přístupný text...</param>

- Jména elementů v XML rozlišují malá a velká písmena: např. „<Příklad>“ a „</Příklad>“ je pár, který vyhovuje správně strukturovanému dokumentu, pár „<Příklad>“ a „</příklad>“ je chybný.
- Elementy mohou být vnořeny, ale nemohou se překrývat. To znamená, že každý (ne kořenový) element musí být celý obsažen v jiném elementu.
- XML je standardně psáno v kódování Unicode, což znamená, že názvy parametrů je možno psát v jakémkoliv jazyce a vždy by výsledný kód měl být čitelný. Pokud však chceme použít jiné kódování, je na to vyhrazeno prvních 128 znaků, které musí být psány v ASCII. Pokud v dokumentu teda použijeme jiné kódování, musíme to uvést v XML deklaraci:

  <?xml version="1.0" encoding="iso-8859-2"?>
  <dokument> ... </dokument>

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Poznamka je nutné přidat více receptů. -->
<recept jméno="chleba" čas_přípravy="5 minut" čas_vaření="3 hodiny">
        <titulek>Jednoduchý chleba</titulek>
        <přísada množství="3" jednotka="šálky">Mouka</přísada>
        <přísada množství="0,25" jednotka="unce">Kvasnice</přísada>
        <přísada množství="1,5" jednotka="šálku">Horká voda</přísada>
        <přísada množství="1" jednotka="kávová lžička">Sůl</přísada>
        <instrukce>
                <krok>Smíchejte všechny přísady dohromady</krok>
                <krok>Zakryjte tkaninou a nechejte hodinu odležet</krok>
                <krok>Prohněťte, umístěte na plech a pečte v troubě</krok>
        </instrukce>
</recept>[9]
```

Výše je uveden příklad, jak jednoduchý XML kód může vypadat.

Popis XML kódu je zde uveden, protože v přiloženém programech, klient a server, je zabezpečena komunikace mezi těmito programy protokolem XMPP (popsaným v další kapitole), který je formulován v XML jazyce.

3.3 XMPP protokol

Extensible Messaging and Presence Protocol (XMPP), je otevřený standardizovaný protokol pro posílání zpráv a zjištění stavu, který je založený na XML(popsáno v kapitole 3.2). Protokol vytvořil v roce 1998 Jeremie Miller jako protokol pro svůj projekt otevřeného komunikačního systému Jabber XMPP. Protokol je již od základu plánovaný pro rozšiřování (o funkce jako je VoIP apod.). Základ XMPP je specifikován v RFC 3920 a RFC 3921, ale řada vlastností je popsána v dalších RFC.

IETF (Internet Engineering Task Force) vytvořilo několik pracovních skupin, které se věnují XMPP. Význačná je XMPP Standards Foundation jejímiž hlavními sponzory jsou Google, Hewlet Packard a Jabber inc.

Specifikace jsou zcela otevřené a dostupné všem, kdo mají zájem o implementaci software s podporou XMPP. Servery XMPP protokolu běží standardně na TCP portu 5222. Pro vzájemnou komunikaci serverů je pak vyhrazen port 5269.

Tento protokol byl původně založen pro IM, avšak postupem času se ukázalo, že je vhodný i pro komunikaci programů nebo pro ovládání různých automatických služeb. Situace a funkce je velmi podobná v případě emailu a protokolu SMTP. Tato výhoda vede k tomu, že Jabber je dnes (podle některých studií) nejpoužívanější IM systém. Navíc je asi jediný založený na standardizovaném protokolu.

3.3.1 Základní prvky protokolu

Jak již bylo zmíněno, tak tento protokol je založený na XML. Protokol má 3 základní druhy elementu[2]:

- <presence/>
- <message/>
- <iq/>

3.3.1.1 presence

Tímto elementem jsou vyřizovány záležitosti týkající se připojení, statusu a autorizací. Vyznačují se tím, že pokud nemají určeného příjemce, tak fungují způsobem "vysílání" (broadcast), tedy jsou směrovány všem lidem, kteří jsou zaregistrování k odběru presencí od daného uživatele (tedy "mají autorizaci").

Příklad:

<presence to='romeo@montague.net' type='unsubscribe'/>

[2] V terminologii XMPP standardu se tomuto elementu říká „Stanza“

3.3.1.2 message

Tímto elementem jsou posílány zprávy všeho druhu. Slouží k odesílání dat způsobem „push“, tedy odeslat a dál se nestarat.

Základní elementy prvku message

- **subject** - obsahuje titulek dané zprávy v lidsky čitelném formátu.
- **body** - obsahující vlastní tělo zprávy, také v lidsky čitelné formě.
- **thread** - tento element obsahuje jednoznačnou identifikaci dané komunikace. Používá se pro stopování "session" dané konverzace.

Příklad:

```
<message to='romeo@montague.net'>
        <subject>Test Message</subject>
        <body>Here is a test message.</body>
</message>
```

3.3.1.3 Iq

Info/Query ~ Požadavky, informace a odpovědi na ně

Tímto elementem jsou v podstatě zprávy všechna data, která nejsou ani presence, ani message. Fungují způsobem dotaz-odpověď. Odeslání odpovědi je povinné. IQ stanzy musejí mít nastaveny typ a id.

Základní typy IQ stanz:

- **get** - žádost o data. V odpovědi jsou odeslána požadovaná data.
- **set** - žádost o nastavení dat.
- **result** - tento typ se pouužívá jakožto odpověď na úspěšnou IQ get či set.
- **error** - chybová hláška. Obsahuje informace o chybě.

Příklad:

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@montague.net'/>
  </query>
</iq>
```

3.3.2 Základní zprávy

3.3.2.1 Zpráva

```
<message to='FUNNNY@jabbim.cz'  from='funny2@jabbim.cz'>
 <subject>Test</subject>
 <body>Testovací zpráva</body>
</message>
```

Jedná se o základní element message, který nese informace o zprávě a její data. Určujě komu je zpráva poslána, od koho zpráva je, předmět zprávy a nakonec tělo vlastní zprávy. Zpráva je poslána na server, ten se podívá, komu je zpráva poslána a přepošle danému účastníku. Pokud element neobsahuje atribut pro koho zpráva je, tak server posle zprávu všem uživatelům.

3.3.2.2 Přidání kontaktu

```
<iq type='set'>
 <query xmlns='jabber:iq:roster'>
  <item jid='romeo@montague.net' name='Romeo Montague'/>
 </query>
</iq>
```

Tento typ požadavku je vyslán na server, tehdy kdy uživatel přidává do svého seznamu kontaktů dalšího uživatele. Element musí obsahovatatribut JID, čili id uživatele k přidání.

3.3.2.3 Smazání kontaktu

```
<iq type='set'>
 <query xmlns='jabber:iq:roster'>
  <item jid='romeo@montague.net' subscription='remove'/>
 </query>
</iq>
```

Element je naprosto stejný jako element pro přidání kontaktu, jen s tím rozdílem, že obsahuje atribut subscription='remove', což má za následek, to že se kontakt v seznamu kontaktů odebere.

3.3.2.4 Stažení seznamu kontaktů

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

Klient vyšle požadavek na server a server následně pošle seznam kontaktů zpět klientovi.

3.3.2.5 Změna stavu

```
<presence from="'romeo@montague.net " xml:lang="en"
        to="'romeo2@montague.net ">
        <priority>0</priority>
        <show>dnd</show>
        <status>Pracuji</status>
</presence>
```

Požadavek je vyslán a server a ten následně odešle na daný kontakt informaci a novém stavu. Stav je poté v klientské části aplikace nastaven na nový.

Značka <show> je nepovinná, její obsah může nabývat jedné z hodnot: away, chat, dnd, n/a, které označují stavy klienta. Jestliže tato značka není v presence stanze přítomna, považuje se stav klienta za on-line.

3.3.2.6 Registrace uživatele

```
<iq type='set'
   from='juliet@capulet.com/balcony'
   to='contests.shakespeare.lit'
   id='reg2'>
  <query xmlns='jabber:iq:register'>
   <x xmlns='jabber:x:data' type='submit'>
    <field type='hidden' var='FORM_TYPE'>
     <value>jabber:iq:register</value>
```

```
    </field>
    <field var='auth-token'>
     <value>
      md5(heslo)
     </value>
    </field>
   </x>
  </query>
</iq>
```

Dále je v elementech field možno posílat ještě položky typu jméno, příjmení, pohlaví, email, atd. Místo položky md5 (heslo), se posílá heslo v šifrované podobě. Já heslo posílám zašifrované pomocí algoritmu md5. Heslo není nijak ukládáno v klientské části. Jen při začátku sezení, aby nebylo možno nějak zneužít.

3.3.2.7 Stažení historie ze serveru

```
<presence from='funnny@jabbim.cz'
        to='funnny2@jabbim.cz'>
        <x xmlns='http://jabber.org/protocol/muc'>
        <history since='1970-01-01T00:00:00Z'/>
        </x>
</presence>
```

Po poslání této stanzy na server je klientu poslána zpráva obsahující historii komunikace mezi danými účastníky

3.3.2.8 Příklad komunikace

k: klient, s: server

```
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>
```

Zde je uvedena modelová situace komunikace mezi klientem a serverem. Na prvních řádcích se vytvoří XML proudy (každý v jednom směru). Nastaví se šifrování, autentifikace a pak už samotná komunikace. Zde se odešle zpráva z klienta jinému, který následně odpovídá. Na závěr je komunikace uzavřena.

3.4 Práce se sokety

Soket je mechanizmus pro komunikaci. Poprvé se objevil v operačním systému BSD. Soket je velice obecný nástroj. Stejné funkce můžeme používat pro komunikaci pomocí různých protokolů. V mém případě se jedná o TCP/IP protokol.

Pomocí soketů můžeme číst a zapisovat prostřednictvím připojení na další počítač a to bez nutnosti znalostí aktuálního síťového software. Aplikace serveru nebo klienta je obvykle vyhrazena k jedné službě jako je Hypertext Transfer Protocol (HTTP) nebo File Transfer Protocol (FTP). Pomocí soketů serveru, aplikace poskytující jednu z těchto služeb, se může spojit s klientskou aplikací, která chce používat tuto službu. Klientské sokety umožňují aplikaci používající jednu z těchto služeb, spojení se serverovou aplikací, která službu poskytuje.

3.4.1 Typy soketových připojení

Soketová připojení můžeme rozdělit na tři základní typy, které určují, jak připojení bude inicializováno a k čemu se připojujeme. Jsou to:

- Klientské připojení
- Naslouchací připojení
- Serverové připojení

3.4.1.1 Klientské připojení

Klientské připojení připojuje klientský soket na lokálním systému k serverovému soketu na vzdáleném systému. Klientská připojení jsou inicializována klientským soketem. Klientský soket musí popisovat serverový soket, ke kterému se chceme připojit. Klientský soket pak hledá serverový soket, a když server nalezne, vyžaduje připojení. Serverový soket udržuje frontu klientských požadavků a kompletuje připojení. Když serverový soket akceptuje klientské připojení, pak serverový soket, ke kterému se připojujeme, zasílá svůj plný popis klientskému soketu a připojení je klientem kompletováno.

3.4.1.2 Naslouchací připojení

Serverové sokety nelokalizují klienty. Tyto sokety pouze naslouchají a vyřizují klientské požadavky. Serverové sokety přiřazují frontu ke svým naslouchacím připojením.

Tato fronta zaznamenává požadavky klientů na připojení, jak přicházejí. Když serverový soket akceptuje požadavek na klientské připojení, provede to formou nového soketu pro připojení klienta a naslouchací připojení zůstává otevřeno pro akceptování ostatních klientských požadavků.

3.4.1.3 Serverové připojení

Serverové připojení je formováno serverovými sokety, když naslouchací soket akceptuje klientský požadavek. Popis serverového soketu, který kompletuje připojení ke klientu je zaslán klientu, když server připojení akceptuje. Připojení je zřízeno, když klientský soket přijme tento popis a kompletuje připojení.

3.4.2 Čtení a zápis soketů

Jaké informace čteme nebo zapisujeme nebo když jsou čteny nebo zapisovány, závisí na službách přiřazených k soketovému propojení.

Čtení a zápis prostřednictvím soketů může probíhat asynchronně, a tak neblokujeme provádění jiného kódu v naší síťové aplikaci. Toto propojení se nazývá **neblokující propojení**. Můžeme také formovat **blokující propojení**, kde naše aplikace čeká na dokončení čtení nebo zápisu před provedením následujícího řádku kódu.

3.4.2.1 Neblokující spojení

Pokud čtení či zápis dat ze/do socketu probíhá asynchronně, tedy pokud odesílatel a příjemce dat nejsou nějak domluveni, jedná se o neblokující spojení. Lze jej přirovnat ke komunikaci formou SMS zpráv: aplikace (uživatel mobilního telefonu) pošle data (SMS zprávu) přes soket (mobilní telefon) a dál pokračuje ve svém běhu (jde se sprchovat) bez ohledu na to, jak dopadlo doručení dat (SMS zprávy). Jinak řečeno – aplikace po odeslání zprávy ihned běží dále a nečeká na dokončení čtení (odesílání) dat. Není tedy nijak blokována, ani GUI.

3.4.2.2 Blokující spojení

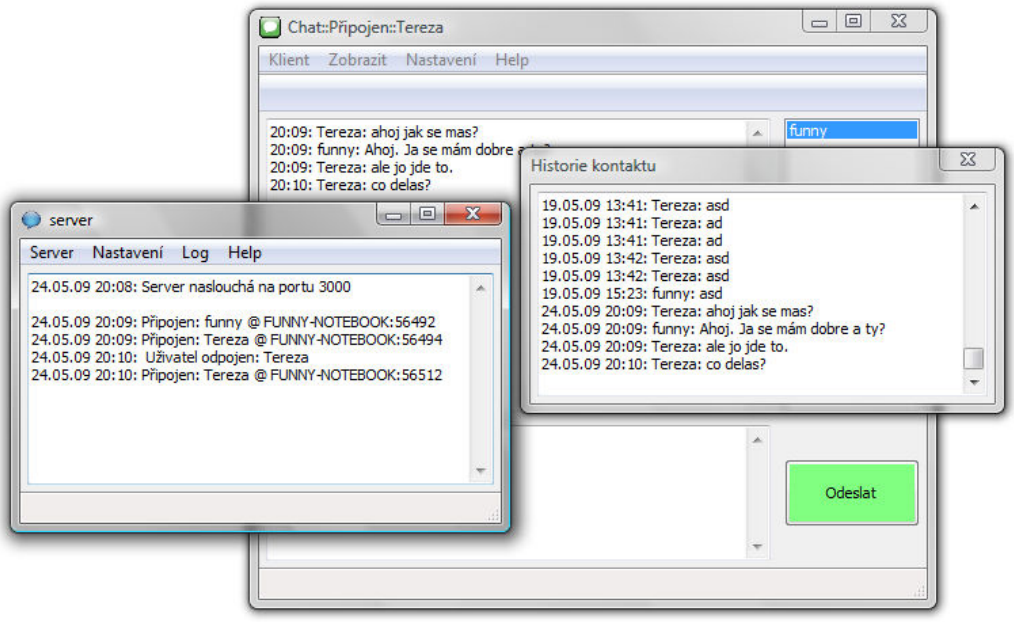
Blokující spojení, je takové, které znamená pravý opak od výše uvedeného: aplikace je po odeslání zprávy blokována až do okamžiku, kdy je operace dokončena. Tento způsob spojení si lze představit jako telefonát: jedna aplikace (volající telefonista) je zablokovaná (čeká na lince), dokud se druhá aplikace (volaný telefonista) neozve (nepřijme hovor). Proto je vhodné k takovémuto spojení používat vlákna.

II. PRAKTICKÁ ČÁST

4 VYPRACOVÁNÍ

Tato kapitola pojednává a popisuje základní implementaci dané aplikace, která byla navrhnuta k této diplomové práci. Aplikace (viz Obrázek 11) se skládá ze dvou částí. Klientská část a serverová část. Obě tyto části jsou naprogramovány pomocí multiplatformní knihovny wxWidgets, jejíž základy jsou popsány v kapitole 3.1. Obě dvě aplikace byly testovány a vyvíjeny v operačním systému Microsoft Windows Vista ve vývojovém prostředí Microsoft Visual Studio 2009 s integrovaným balíkem obsahující knihovnu wxWidget. Obě tyto části byly poté odzkoušeny na operačním systému OpenSuse, tudíž by měly být multiplatformní.

Jedná se o software, který umožňuje vícenásobné přihlášení uživatele k serveru s tím, že historie zpráv se bude ukládat na serveru. Jedná se pouze o základní verzi, která je vhodná například pro komunikaci do menší firmy, jelikož v seznamu kontaktů se zobrazují všichni připojení účastníci.



Obrázek 11.: Klient a server aplikace

4.1 Základní princip

Základní princip aplikace je ten, že se klientem připojíte na server a následně si můžete psát s dalšími klienty, kteří jsou taktéž připojeni k serveru. K serveru je možno se připojit pod stejným uživatelským jménem vícekrát, aniž by to mělo nějaký vliv na funkcionalitu aplikace. Na serveru se uchovává historie zpráv, tuto historii si klient může vyžádat. Celá aplikace používá základní elementy z protokolu XMPP. K dispozici je i knihovna, která umožňuje tyto základní elementy skládat a číst.

4.2 Klient

4.2.1 Princip

Po spuštění klienta se zobrazí okno, které čeká na pokyn uživatele k připojení. Aplikace vyzve uživatele k zadání uživatelského jména a jeho hesla. Pokud se autorizace na serveru potvrdí, tak se klient připojí. Pokud ne, tak je klient o této skutečnosti informován a odpojen. Pokud je uživatel připojen, tak si může psát s kontakty, kteří jsou právě připojeni. Aplikace je navrhnuta tak, že se všem objevují jen připojení uživatelé a nikdo nějaký privátní seznam kontaktů nemá, což by mohlo být v budoucnu o tuto vlastnost rozšířeno. Uživatel může nahlédnout na historii psaní s daným kontaktem. Po skončení konverzace je možno se odpojit a vypnout program.

Tato část popisuje funkčnost trochu praktičtěji. Po spuštění klienta se po jeho inicializaci pokusí aplikace načíst konfigurační soubor, kde jsou uloženy základní informace o uživateli. Pokud je to první spuštění a soubor neexistuje, je vytvořen nový. Název tohoto souboru je *config.ini.*

Po zaznamenání události o připojení k serveru (fce KlientFrame::OnOpenConnection) je vygenerován nový soket, který je třídy *wxSocketClient*, a ten se následně pokusí připojit k serveru na uživatelem daném portu. Toto spojení je blokované, aby byla všechna data přenesena od serveru, který následně posílá odpovědi. Pokud se připojení podaří, tak je zkontrolována autentizace. Heslo je zašifrováno pomocí md5 algoritmu, aby nebylo možno jeho odposlechu. Po úspěšném připojení pak při každé události, která se uskuteční na soketu, volána funkce, která tyto

události ošetřuje. Události na soketu mohou být trojího druhu, které jsem stanovil při inicializaci soketu.

- wxSOCKET_INPUT
  Na soketu přišla nějaká data
- wxSOCKET_ LOST
  Spojení bylo ukončeno
- wxSOCKET_CONNECTION
  Spojení bylo vytvořeno

Nejvíce je prováděn stav první, wxSOCKET_INPUT, pomocí tohoto stavu jsou přijata nějaká data, která jsou následně zpracována funkcí, která rozliší, o jaký typ dat jde. Mohou to být:

- MESSAGE
  Jedná se o normální zprávu, která přišla od uživatele.
- ROSTER_ADD
  Server poslal informaci o tom, že se připojil nějaký kontakt
- ROSTER_REMOVE
  Server poslal informaci o tom, že se odpojil nějaký kontakt
- OTHER
  Pokud dojde jiný, neznámy typ zprávy, tak je zobrazena do okna se zprávami

Pokud je zavolána událost, která se snaží **stáhnout historii ze serveru**, tak je vygenerována z práva v xmpp formátu a poslána na server. Pro správné odeslání informace je ale potřeba mít vybrán nějaký kontakt, pro který bude historie zobrazena. Jedná se opět o typ blokovaného spojení. Následně přijdou data, která se zobrazí v nově otevřeném okně.

Odeslání zpráv (KlentFrame::OnOdeslatClick**)** funguje obdobným způsobem. Je vybrán kontakt a tomu je poslána zpráva. Klient vygeneruje zprávu, která je poslána na server a ten pak odešle tuto zprávu všem uživatelům, se stejnou přezdívkou. Samozřejmě se jedná o vícenásobné připojení. Není možno mít dva uživatele se stejnou přezdívkou a jiným heslem. Pokud přijde zpráva od neoznačeného kontaktu, tak se kontakt zvýrazní postfixem [*]. Pokud se kontakt označí nebo již je, postfix se zruší.

Seznam kontaktů funguje tak, že pokud se připojí uživatel na server, je zobrazen v tomto seznamu. Tento seznam je uložen v poli prvků. Tento prvek je struktura, třída obsahující přezdívku uživatele a zprávy od daného kontaktu v daném sezení. Pokud je vyvolána událost, která překlekává mezi jednotlivými uživateli, tak do prvku s danými zprávami pro daného uživatele jsou zobrazeny vždy zprávy uložené v této struktuře pro daného uživatele. Tato struktura, v mém případě třída je zobrazena níže.

```
class MyArrayM;
class Messages
{
public:
        Messages(void);
  ~Messages(void);
        wxString* Message;
        wxString* Nick;
};
WX_DEFINE_ARRAY(Messages*, MyArrayM);
```

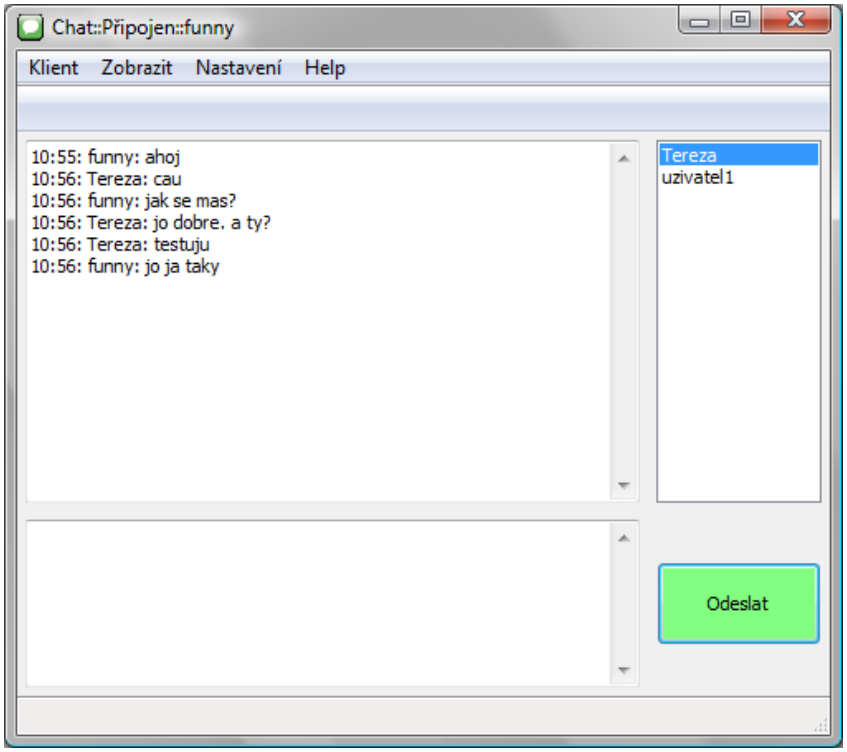
Registrace nového uživatele (KlientFrame::OnRegistration), vás vyzve k zadání nového účtu. Jedná se jen o přezdívku a heslo. Pokud se uživatele podaří založit je možno se pod novým účtem přihlásit. Pokud se uživatele nepodaří založit je o této skutečnosti uživatel informován. K této situaci mlže dojít pokud například je uživatel již registrován, či jiná chyba.

4.2.2 Struktura

Zdrojové kódy jsou rozděleny do základních 10 *.cpp souborů, které k sobě mají příslušný hlavičkový soubor. O jednotlivých funkcích je vygenerována projektová dokumentace v programu DOXYGEN, která je přílohou diplomové práce nebo také můžete nahlédnout na druhou přílohu, což jsou diagramy tříd v jazyce UML.

4.2.3 Program a jeho funkce

Na níže uvedeném obrázku je uveden příklad, jak tento klient vypadá.:



Obrázek 12.: Klientská část aplikace

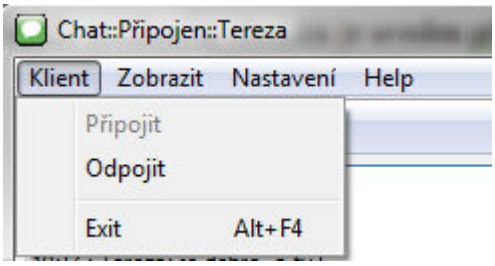
4.2.3.1 Menu Klient

Připojit, Odpojit

Připojí nebo odpojí klienta se serverem. Pokud program byl jen spuštěn a následně kliknuto na „připojit“, tak se zobrazí dialogové okno k zadání potřebných údajů k přihlášení.

Exit

Zobrazí dialogové okno, zda chce uživatel opustit aplikaci, či nikoliv

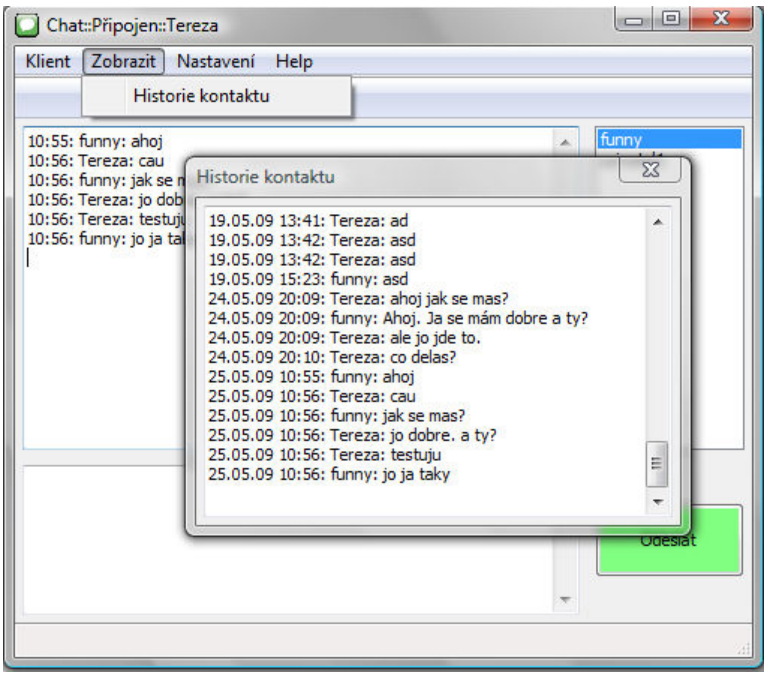


Obrázek 13.: Menu Klient

4.2.3.2 Menu Zobrazit

Zobrazit historii

Zobrazí historii (obrázek níže) u vybraného kontaktu. Kontakt musí být vybrán v seznamu připojených kontaktů. Historie je uchovávána na serveru. Jediná historie ze serveru na klienta není posílána ve formátu xmpp. Je poslán holý text, ve kterém jsou poté upraveny konce řádků (nahrazeny znaky
 na nový řádek) a poté zobrazeny v novém okně.

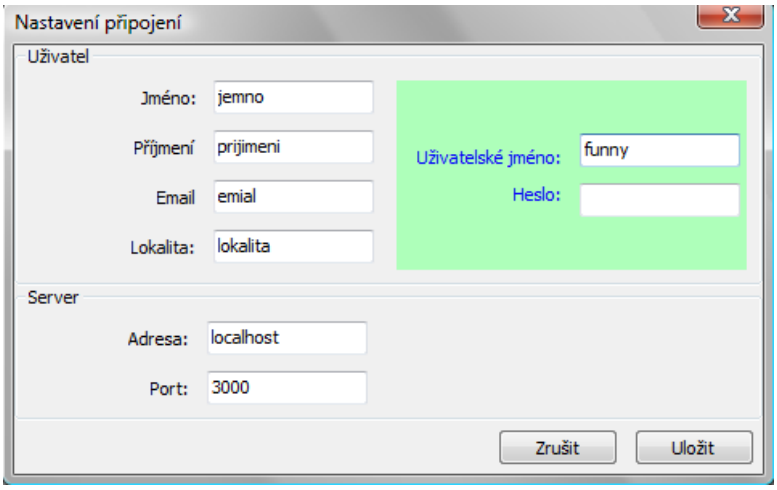


Obrázek 14.: Menu zobrazit a okno s historií

4.2.3.3 Menu nastavení

Nastavení připojení

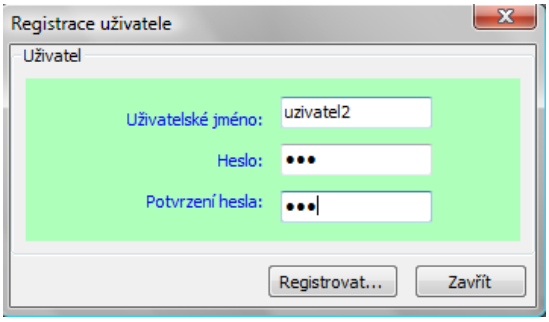
Po kliknutí na tuto položku se uživateli objeví nové okno (viz Obrázek 15), kde uživatel vyplní přihlašovací údaje, tj. uživ. jméno a heslo. A také musí zadat adresu serveru a jeho port. Jinak se uživatel nepřipojí. Informace o uživateli jsou nepovinný údaj. Tyto změny jdou buď uložit anebo stornovat. Po uložení se data uloží do souboru s názvem config.ini, který je pak načten po zavření okna. Toto okno se objeví taktéž při prvním přihlášení konkrétního sezení aplikace.



Obrázek 15.: Nastavení klienta a připojení

Registrace uživatele

Při výběru této volby se uživateli objeví okno, kde je vyzván pro jméno a heslo nového uživatele. Pokud je uživatel na serveru již vytvořen, tak se uživatel nevytvoří a uživatel je o této skutečnosti patřičně informován. Jméno a heslo nesmí být prázdné. Dole je uvedený příklad registračního formuláře.



Obrázek 16.: Registrační formulář

4.2.3.4 Menu Help

O programu

Vypíše krátké informace o programu.

4.3 Server

Tato aplikace je druhou částí. Jedná se o server, který se stará o komunikaci mezi jednotlivými připojenými klienty. Stará se o uchovávání historie, autorizaci jednotlivých klientů apod.

4.3.1 Princip

Základní princip aplikace je, že po spuštění aplikace, se spustí server, který naslouchá na uživatelem uvedeném portu a čeká na příchozí spojení. Pokud přijde nové spojení, tak se jedná buď o registraci nového uživatele, nebo se jedná o žádost připojení k serveru. Obě dvě situace jsou vyhodnoceny a podle toho provedeny. Pokud přijde na server zpráva je zjištěn uživatel, kterému je zpráva posílána a následně uložena do historie na serveru a poslána na konkrétní určení.

Tato část popisuje funkčnost trochu praktičtěji. Po spuštění aplikace dojde k vytvoření a inicializaci serveru, který je odvozen od třídy wxSocketServer. Je vytvořen na uživatelem stanoveném portu. Já aplikaci testoval, tudíž bych i doporučoval port 3000, který na běžném počítači není ničím používán. Po spuštění jsou hlídány 2 základní události. Událost, kdy se stane něco na serveru a událost, kdy se stane něco na soketu. Starají se o to dvě funkce.

- **serverFrame::OnServerEvent**

  Může dojít zpráva typu **wxSOCKET_CONNECTION** a nebo neznámá.

  Pokud je to první případ, tak se jedná o inicializaci nového spojení, kdy se provede i autentizace a nebo registrece. Záleží co poslaná data obsahují. Jsou poslány xmpp zprávy(autentizace, nebo registrece), které jsou popsány v kapitole výše.

  Hesla a databáze uživatelů jsou uložena v souboru *password.dat* ve formátu:

  uzivatel:md5 hash

  Příklad:

  uzivatel2:7815696ecbf1c96e6894b779456d330e

  Nakaždém novém řádku je jiný uživatel. Není možno stejné uživatele

  Ve druhém případě jde o neznámou inicializaci, která je ignorována.

Pokud je připojen nový uživatel, je jeho soket uložen do pole, třídy, která je následující struktury:

```
class MyArray;
class ContactList
{
public:
        ContactList(void);
        ~ContactList(void);

        wxSocketBase* sock;
        wxString* nick;
};
WX_DEFINE_ARRAY(ContactList*, MyArray);
```

Pole má prvek, kde je uložena přezdívka, popisovač soketu a pokud je zpráva poslána na konkrétní jméno, tak je v tomto poli nalezeno a poslána na příslušný soket.

- **serverFrame::OnSocketEvent**

  Mohou nastat tři situace takové, že dojdou události s následujícími ID. **wxSOCKET_INPUT**, **wxSOCKET_LOST** nebo **neznámá**.

  Pokud je ID prvního typu, tak se jedná buď o klasickou zprávu, nebo o žádost o historii. **V prvním případě** je zjištěno, komu zpráva je, a na daný soket je zaslána.

  V druhém případě se jedná o žádost na **historii**. Aplikace se podívá, zda daná historie existuje a pošle historii kontaktu zpět. Historie se posílá v čistém formátu, bez jakéhokoliv použití nějakého protokolu, či standardu. Je ukládána v adresáři HISTORY. Vždy ve tvaru:

  history_uzivatel1_uzivatel2.dat

  historie je ukládána ve formátu:

  *18.05.09 18:32: uzivatel: text
text*

  Na každém řádku je jedna zpráva. Znaky *
* znamenají, že ve zprávě uživatel napsal enter. Jelikož je ale každá zpráva ukládána na nový řádek, tak znaky musely být nahrazeny.

Pokud je ID hodnoty **wxSOCKET_LOST**, tak se z výše uvedeného pole soket a přezdívka odstraní. Tato přezdívka pak následně je poslána všem kontaktům v poli, ale jen v případě, zda byl uživatel, co se odpojil přihlášen jen z jednoho místa.

Aplikace vypisuje informace o různých stavech či činnostech serveru:

- Start serveru
- Stav serveru
- Restart serveru
- Registrace uživatele, její stav
- Připojení uživatele
- Odpojení uživatele
- Špatné heslo uživatele

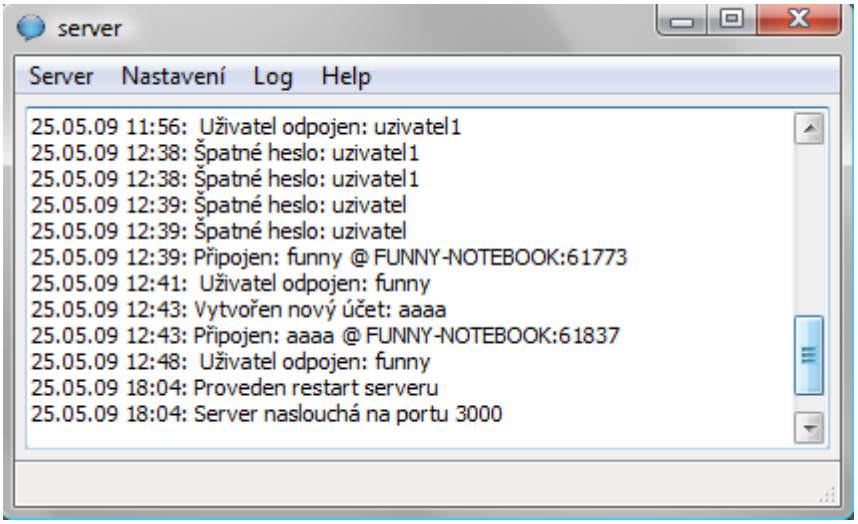
Ostatní funkce jsou popsány v přiložených programových dokumentacích

4.3.2 Struktura

Zdrojové kódy jsou rozděleny do základních 7 *.cpp souborů, které k sobě mají příslušný hlavičkový soubor. O jednotlivých funkcích je vygenerována projektová dokumentace v programu DOXYGEN, která je přílohou diplomové práce nebo také můžete nahlédnout na druhou přílohu, což jsou diagramy tříd v jazyce UML.

4.3.3 Funkce

Na níže uvedeném obrázku je uveden příklad, jak tento klient vypadá.:



Obrázek 17.: Serverová část aplikace

4.3.3.1 Menu Server

Restart serveru

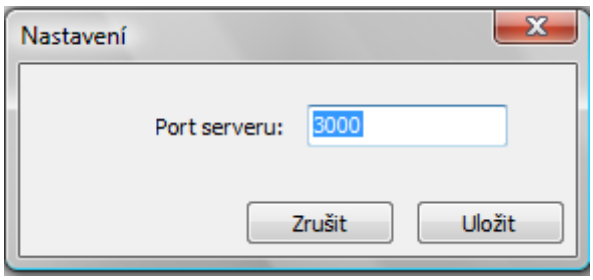
Restartuje server na daném portu.

Exit

Zobrazí dialogové okno, zda chce uživatel opustit aplikaci, či nikoliv

4.3.3.2 Menu Nastavení

Vyzve uživatele k nastavení portu, na kterém má server naslouchat.



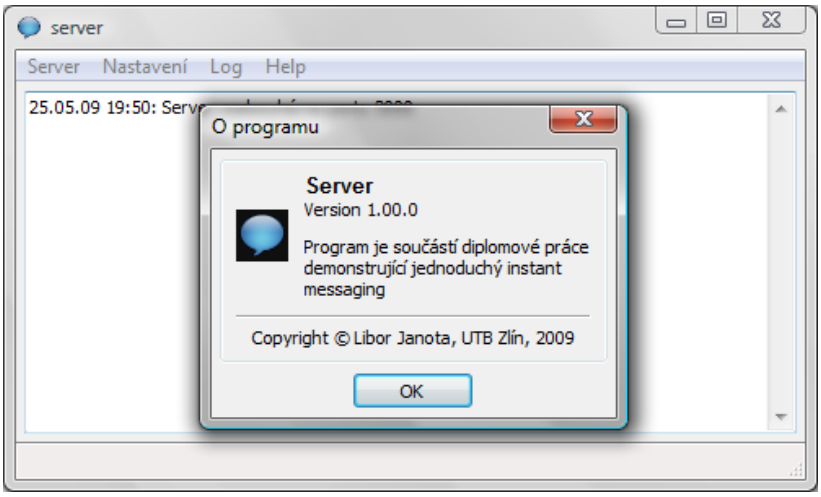
Obrázek 18.: Nastavení serveru

4.3.3.3 Menu Log

Vyzve uživatele k zadání souboru, kam bude log uložen

4.3.3.4 Menu Help

Vypíše krátké informace o programu.



Obrázek 19.: Dialogové okno zobrazující krátké informace

4.4 Rozšíření

Obě aplikace by si v budoucnu, pokud by bylo potřeba, zasloužily nějaké rozšíření. Teď jsou aplikace jen v základech, ale hlavní myšlenka je splněna. Software byl navrhován pro použití do menší firmy, či menší komunitu lidí. Další části vývoje by mohly mimo jiné být:

- **Klient**
  - Použití „smajlíků“
  - Barevně odlišené kontakty při komunikaci
  - Posílání souborů
  - Přenos hlasu
  - Mít „vlastní“ seznam kontaktů
  - Integrace pro aplikační doplňky (pluginy)
  - Lepší upozorňování na události

- Možnost změny stavu
- Možnost použití vlastních vzhledů
- …
-

- **Server**
  - Nastavení platnosti historie
  - Editace, administrace uživatelů
  - Volba vlastních vzhledů
  - Integrace pro aplikační doplňky (pluginy)
  - Statistiky serveru
  - …

4.5 XMPP knihovna

Součástí práce bylo také navrhnout knihovnu, která by využívala protokol XMPP. Knihovna byla navrhnuta pro základní typy stanz, které byly uvedeny v kapitole o XMPP protokolu. Práce s těmito zprávami umožňuje zjištění od koho, či komu je zpráva poslána a podobně. Umí také tyto základní typy zpráv generovat. Použití je velmi jednoduché. Stačí si vytvořit objekt, který bude této třídy a poté její funkce, při použití správných parametrů vracejí výsledné zprávy. Struktura této třídy je uvedena v přiložené programové dokumentaci vygenerované pomocí programu DOXYGEN, případně pomocí přiloženého diagramu tříd.

Zde uvádím příklad použití a základní členy této knihovny:

Funkce, která zjišťuje typ zprávy.

int GetTypeOfMessage (wxString message)

Funkce, která zjišťuje od koho je zpráva.

wxString GetMessageFrom (wxString message)

Funkce, která zjišťuje komu zpáva je poslána.

wxString GetMessageTo (wxString message)

Funkce, která zjišťuje tělo zprávy.

wxString GetMessageBody (wxString message)

Funkce, která zjišťuje 1. člena historie.

wxString GetHistoryTo (wxString message)

Funkce, která zjišťuje 2. člena historie.

wxString GetHistoryFrom (wxString message)

Funkce, která zjišťuje jak jméno je potřeba přidat.

wxString GetRosterNameToAdd (wxString message)

Funkce, která zjišťuje jaké jméno je potřeba odebrat.

wxString GetRosterNameToRemove (wxString message)

```
Xmpp* xmpp=new Xmpp();
int i=xmpp->GetTypeOfMessage(message2);
switch(i)
{
  case MESSAGE:
  …
  case ROSTER_ADD:
if(GetContactID(xmpp->GetRosterNameToAdd(message2))==-1
&& wxStrcmp(xmpp->GetRosterNameToAdd(message2),Nick)!=0 )
{
  m_listBox2->Append(xmpp->GetRosterNameToAdd(message2));
  Messages* CurrentMessage= new Messages();
  CurrentMessage->Message= new wxString(_(""));
  CurrentMessage->Nick=new wxString(
  xmpp->GetRosterNameToAdd(message2));
  MyArrayOfMessages->Add(CurrentMessage);

}
```

Tento zdrojový kód vytvoří nový objekt třídy Xmpp. Poté otestuje, o jaký druh zprávy se jedná. V příkladu se jednalo třeba o přidání Uživatele to seznamu kontaktů. Zjistí, koho má přidat. Přidá uživatele do pole.

Knihovna rozeznává tyto druhy zpráv:

- MESSAGE

  Jedná se o normální zprávu, popsanou v kapitole o protokolu XMPP
- ROSTER_ADD

  Jedná se o přidání kontaktu, popsáno v kapitole o protokolu XMPP

- ROSTER_REMOVE

  Jedná se o smazání kontaktu, popsáno v kapitole o protokolu XMPP

- GET_HISTORY

  Jedná získání historie ze serveru, popsáno v kapitole o protokolu XMPP

- OTHER

  Neznámý druh zprávy

ZÁVĚR

Tato diplomová práce shrnuje základní informace o tzv. „Instant Messagingu“. Popisuji zde vývoj a trendy tohoto typu komunikace. Dále jsou v první části popsány základní principy o tom, co to Instant Messaging je a dále jeho použití. Uvádím zde dosavadní celosvětovou situaci, jaká nejběžnější klientská a serverová řešení se nyní používají. Ty hlavní, které jsou pro Českou Republiku důležitější (ICQ, Jabber), jsou v této práci popsány více. Jsou zde ale například i různí alternativní klienti a podobně.

Další část se věnuje technologiím a principům, pomocí kterých je praktická část diplomové práce vytvořena. Uvádím zde jednouchý přehled o multiplatformní knihovně wxWigets, kterou jsem použil pro návrh aplikací k této práci. Po rychlém shrnutí této knihovny se práce zabývá otevřeným a celosvětově používaným, pro dnes ne již jen síťovou komunikaci, protokolem XMPP.

Třetí a poslední hlavní část je část praktická, která se zabývá návrhem demo aplikací, které demonstrují vlastnosti již výše uvedených wxWidgets a XMPP protokolu. Jedná se o dvě aplikace, které slouží pro komunikaci mezi menší skupinou uživatelů, například pro firemní komunikaci. Jde o aplikace typu klient a server. Klientská část umožňuje uživateli komunikovat s dalšími uživateli, kteří jsou k serveru připojeni. Server umožňuje vícenásobné připojení jednoho uživatele, s tím že historie komunikace je zachovávána na serveru. Jedná se o řešení, které není běžné mezi dosavadními řešeními.

Aplikace má jen základní funkcionalitu a do budoucna, pokud by pokrčoval vývoj, je zde mnoho možností rozšíření, například o možnost měnit si motivy vzhledu, používání „smajliků“, či například přenos audio, či visuálního hovoru.

K těmto aplikacím byla vytvořena také projektová dokumentace. Ta i veškeré zdrojové kódy jsou na přiložených CD nosičích.

Práce čtenáře obohatí o problematiku internetové komunikace mezi lidmi, kteří potřebují komunikovat živě a při tom mít čas na rozmyšlenou. O tomto a všem uvedeném v této práci je dnešní Instant Messaging

ZÁVĚR V ANGLIČTINĚ

This diploma thesis summaries the basic information about thing what is Instant Messaging. I describe a progress and trends of this type of communication here. Further there are described a basic principles about what is Instant Messaging and its usage in the first part of the thesis . I mention actual worldwide situation of it. There are described today's client's and server's solutions, which we use today. The basic one which is more important for the Czech Republic (ICQ, Jabber) are described more than another one. There are another alternative client's applications too in the thesis.

Next part is about technologies and principles which I'm using in this part of thesis. I introduce simply summary about wxWidgets library, which I used to design an applications for this thesis. The thesis consider about open and worldwide using of XMPP protocol after quick summary of this library.

The third and last basic part is functional. This one is about designing demo application, which is demonstrating properties of library wxWidgets and protocol XMPP. There are two applications for this thesis, which is servant for communication in smaller group of users, for example company group communication. There are two applications, client and server. Client's part allows communication to user with another one user, which is connected to the server in the same time. Server's part allows multi connection of the one user. History of the instant messaging is saved in the server. This solution is not common solution worldwide.

Application's functionality is very simple, so there are many solutions for extension in the future time. For example changing skins, using smiles, and do conversation by audio or video calls.

The project documentation was made for these applications. This one and every source codes are include on CD.

This thesis richens the reader with problems of internet communication between people, which want to communicate quickly, but they need time for reflection. Everything what is introduced in this thesis is today's Instant Messaging.

SEZNAM POUŽITÉ LITERATURY

[1] Přispěvatelé wikipedie. *Instant Messaging* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Instant_messaging>.

[2] Přispěvatelé wikipedie. *Instant Messaging* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Instant_messaging>.

[3] VÍT, Svatopluk. *Instant messaging pro začátečníky aneb klienti pro IM sítě* [online]. c2008 [cit. 2009-05-01]. Dostupný z WWW: <http://www.root.cz/clanky/instant-messaging-zacatecniky-klienti-pro-im-site/>.

[4] Přispěvatelé wikipedie. *Skype* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Skype>.

[5] Přispěvatelé wikipedie. *MSN Messanger* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MSN_Messenger >.

[6] SAMURAJ. *Popis Instant Messagingu Jabber se zaměřením na firemní nasazení* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://www.samuraj-cz.com/clanek/popis-instant-messagingu-jabber-se-zamerenim-na-firemni-nasazeni/>.

[7] HRAZDIL, Jiří. *JABBER/XMPP ROBOT PRO VYHLEDÁVÁNÍ POMOCÍ GOOGLE*. [s.l.], 2007. 44 s. Bakalářská práce.

[8] SMART, Julian, HOCK, Kevin, CSOMOR, Stefan. *Cross-Platform GUI Programming with wxWidgets*. [s.l.] : [s.n.], 2006. 705 s.

[9] Přispěvatelé wikipedie. *XML* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Xml>.

[10] *XMPP standard foundation* [online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://xmpp.org/>.

[11] *wxWidgets Homepage*[online]. c2009 [cit. 2009-05-01]. Dostupný z WWW: <http://wxWidgets.org/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

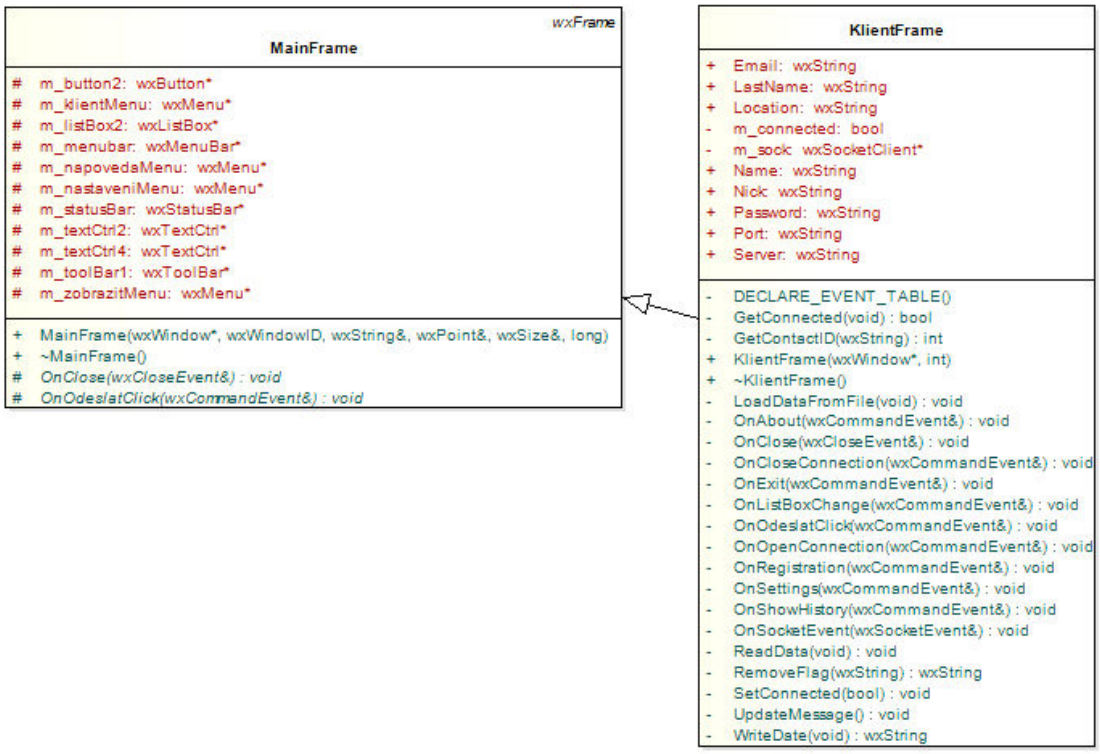
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ID	Identification
IM	Instant Messaging
IRC	Internet Relay Chat
ODBC	Open Database Connectivity
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

SEZNAM OBRÁZKŮ

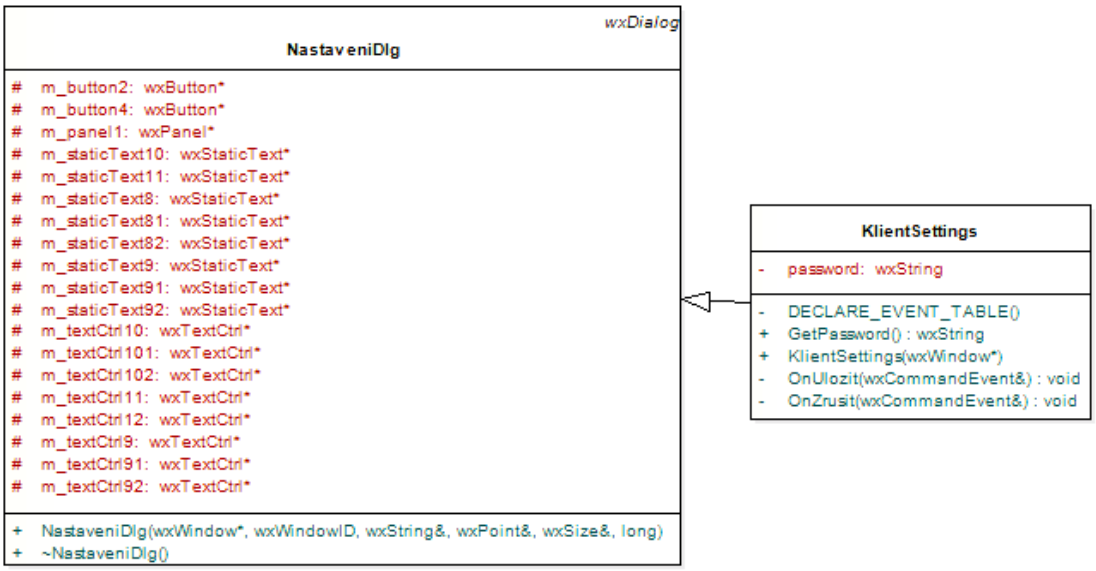
SEZNAM TABULEK

SEZNAM PŘÍLOH

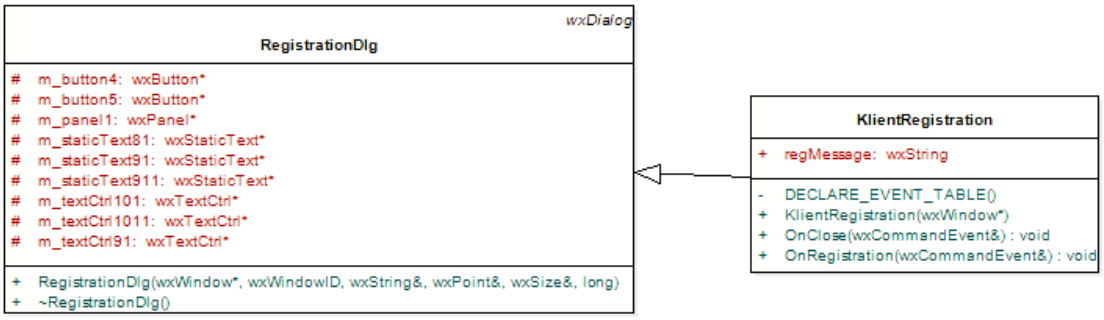
PŘÍLOHA P I: DIAGRAM TŘÍD – KLIENT



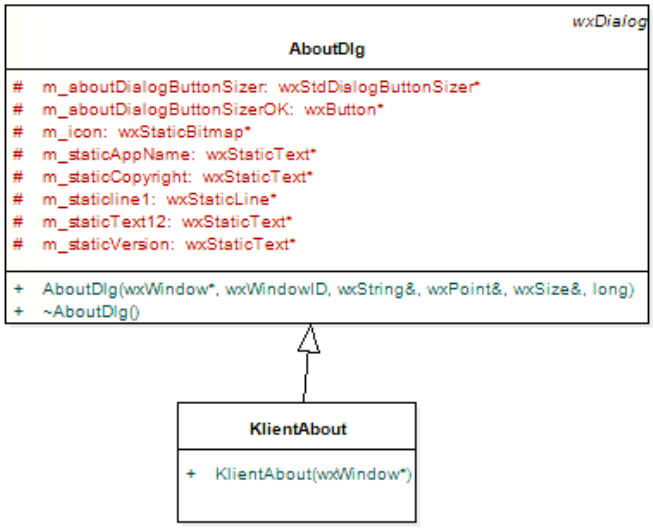
Hlavní třída klienta KlientFrame, která se stará o hlavní chod aplikace. Jsou zde vyřizovány veškeré události, komunikace a zprávy.



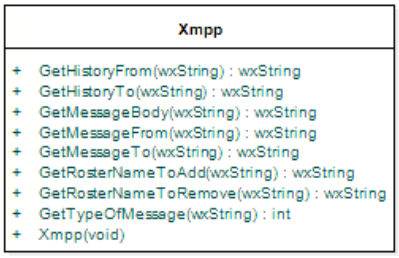
Třída KlientSettings má na starosti zobrazení okna pro nastavení a jeho správu.



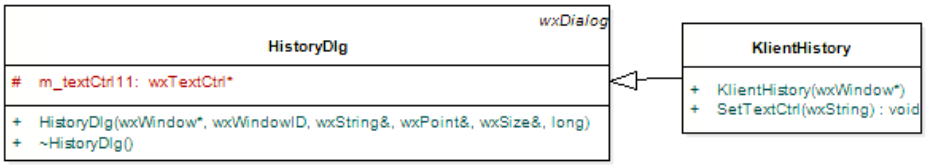
Třída KlientRegistration zobrazuje a spravuje okno pro registraci uživatele



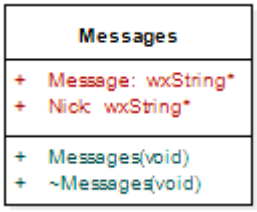
Třída KlientAbout zobrazuje okno „o aplikaci“



Třída Xmpp, která se stará o získání základní informací o přijaté zprávě

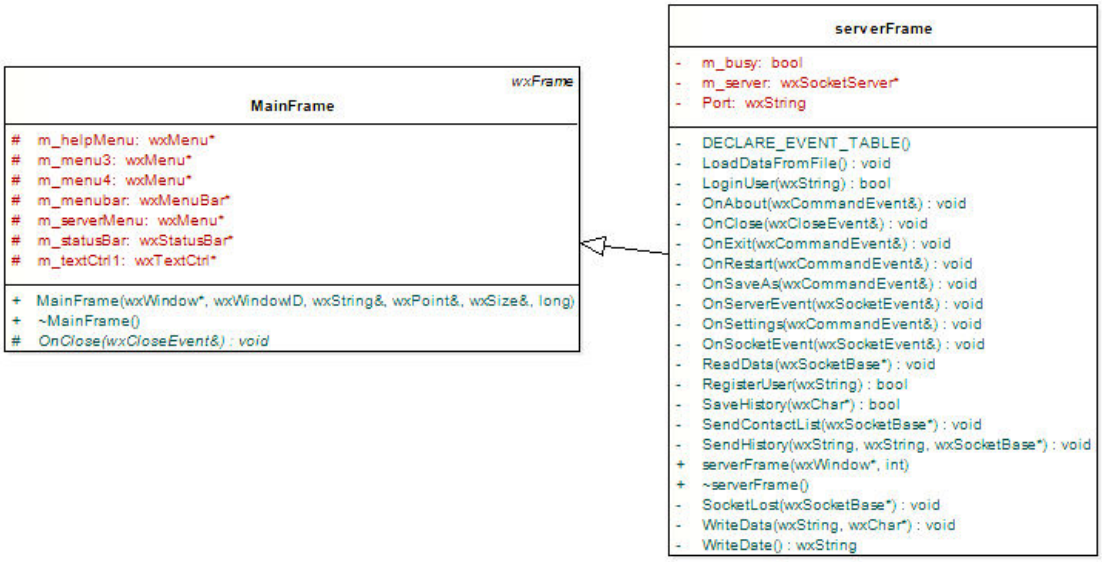


KlientHistory je třída zabezpečující zobrazování historie u daného kontaktu

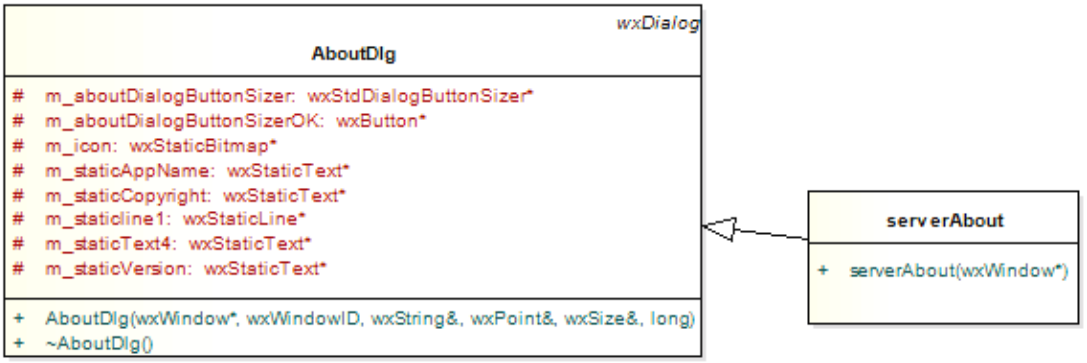


Messages je třída, která uchovává pro daný kontakt danou zprávu

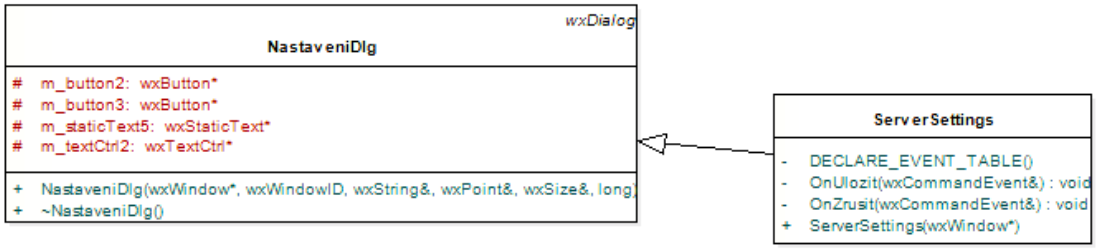
PŘÍLOHA P II: DIAGRAM TŘÍD – SERVER



Hlavní třída serveru ServerFrame, která se stará o hlavní chod aplikace. Jsou zde vyřizovány veškeré události, komunikace a zprávy.



Třída serverAbout zobrazuje okno „o aplikaci“



Třída ServerSettings má na starosti zobrazení okna pro nastavení a jeho správu.

Třída ContactList slouží k uchovávání právě připojených kontaktů