Vysoké učení technické v Brně
Fakulta informačných technológií

Programování síťové služby

Dokumentace k projektu předmětu ISA

XMPP/Jabber klient (Ing. Pluskal)

Vypracoval: Mark Birger (xbirge00)

30. listopadu 2014

Obsah

1 Cílová aplikace

Cílem je implementace jednoduchého klientů pracujícího s protokolem XMPP/Jabber. Budoucí aplikaci musí poskytovat několik různých parametrů spuštění, který umožňují odesílání a příjem zpráv, práce s kontakty. Taky aplikace musí poskytovat interaktivní režim.

2 Uvedení do problematiky

Zprávy protokolu XMPP jsou XML zprávy. Server a klient komunikují pomoci schránek. Pro implementace takové aplikace musím zajistit správnou práce ze schránkami, odesílaný správných XML zprav a kontrolu očekávaných odpovědi. Protokol XMPP nezahrnuje odesíláni rozsahu zprav. Komunikace serverů a klienta není přesně časové definovaná (například některé verze XMPP serveru odesílají `<?xml>` a `<stream>` zprávy dohromady, některé samostatné). Tento princip vyžaduje složitější kontrolu zprav, než v ostatních protokolech. Druhy problém, se kterým budu se setkávat, je interaktivní režim, ve kterém aplikace musí asynchronně očekávat vstup a očekávat zprávy ze serveru. V tomto případě je potřeba pracovat s vicevláknovým/viceprocesovým programem.

3 Informací z literatury

Pro úvod do protokolu XMPP a pochopeni základních druhu zprav XMPP používal jsem knihu *XMPP: The Definitive Guide*. Ale, praktickým příkladem aplikace je *Psi* instant messenger. Ve virtuálním strojů s nainstalovaným operačním systémem Windows jsem zkoušel základní funkce aplikace. Program *Wireshark* poskytuje moznost filtrace XMPP zprav, takže pro každou potřebnou funkci budoucího programu jsem prováděl podobné řešení v IM Psi a sledoval odesílaný zprávy. Tento přístup je velmi užitečný, protože XMPP je docela rozlehly protokol a jsem dostal moznost definovat množinu potřebných pro své implementace zprav. Dalším krokem studia protokolu XMPP byl XMPP RFC. Používal jsem většinou RFC XMPP *Core*, protože aplikace využívá jen základy XMPP. Pro každou ze zapsaných zprav jsem zjistil, co znamenají různý parametry a jaký mohou byt očekávaný odpovědi. Později jsem použil modifikovaný zprávy ve finální aplikace.

4 Návrh aplikace

Jako programovací jazyk jsem zvolil *Python 3*. Python jako skriptovací jazyk poskytuje vice moznosti práce s řetězci. Taky kladem je to, ze nemusím se zabývat alokace paměti. Podle návrhu na začátku musím načíst vstupní parametry spuštěni programu. Níže v teto dokumentaci je vzájemné chovaní různých parametru aplikace. Druhým korkem je otevření soketů na definovanou adresu a port. Od teto doby mužů komunikovat ze serverem. Dále podle zjištěných parametru provádím operace XMPP. Pro každou operace vytvářím novou zprávu a přijímám odpověď (pokud podle protokolu musím očekávat tuto odpověď). Podle zjištěných znalosti o protokolu XMPP bude několik různých části komunikace se serverem: vytvořeni streamu, dotaz na registrace, autorizace, bind a session, dotazy pro práce s rosterem a odesílaní zprav, zavření streamu. Očekávaní příchozích zprav bude prováděno v jinem procesu/vlákně, abych neblokovat základní chovaní aplikace.

5 Popis implementace

Během vývoje používal jsem objektové orientovaný přístup. Aplikace obsahuje dvě třídy.

5.1 Parser argumentů

První třida slouží pro načtení vstupních parametru a využívá standartní knihovnu Python *argparse*. Načítá vstupní parametry, zobrazuje nápovědu. Výsledkem teto části je objekt, který obsahuje všechny potřebný údaje pro provádění aktivity aplikace. Tento objekt je přijímán druhou třídou.
Spolu použití různých parametru v tomto programu je složitější, a nelze ho implementovat pomoci knihovny argparse, takže rozhoduji o chovaní aplikace už uvnitř třídy Klient. Rozhodnul jsem takové vzájemné chovaní parametru:

- Pokud registrace provedla úspěšné, a nedefinován parametr autorizace, tak proběhne autorizace ze stejnými parametry z registrace.
- Pokud například není při spuštěni definovaná autorizace, zapisuju chybu jenom v případě pokud jsou jiny parametry, který vyžaduji autorizace
- Nelze spolu použít interaktivní režim a očekávaní zprávy před ukončením
- Očekávaní zprávy nulu sekund je ok, aplikace jen vypisuje warning
- Pro interaktivní režim je potřeba definovat adresáta pomoci parametru *-u*
- Podle priority výpis rosteru probíhá před přidávaným nového JID do rosteru, takže přidaný JID se nevypisuje

5.2 XMPP klient

Druha třida implementuje XMPP klient. Během inicializace zjišťuje vstupní hodnoty a vola potřebný metody. Většinou metody odpovídají provádění nějaké aktivity XMPP nebo zahrnuje práce ze sokety. Základem komunikace XMPP je jazyk XML. Pro Python jazyk XML není tak přirozený v porovnávaní z JSON. Zprávy pro odesílaní vytvářím pomoci postaveni hodnot do předem přepravených řetězců. Pro parsing příchozích XML zprav na začátku chtěl jsem využít *xml* knihovnu Python, nebo parser *BeautifulSoup*, ale jistil jsem, že oba tito dvě nejlepší parsery většinou spatně zpracovávají XMPP zprávy. XMPP má modifikovaný atributy XML, který jsou spatně zpracovávaný tradičními parsery. Rozhodl jsem pro zjišťovaní správnosti příchozích zprav využít regulární výrazy.

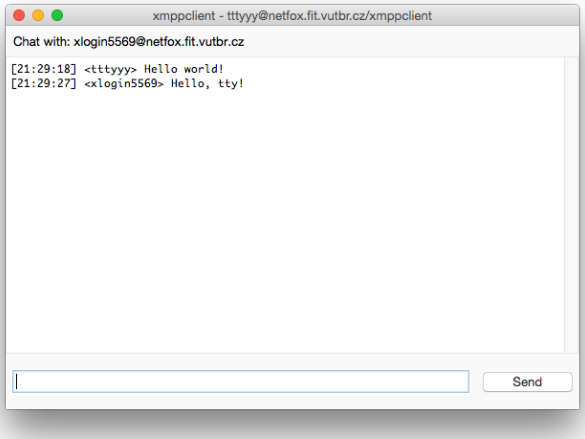
Pro autorizace jsem využil mechanismus PLAIN, jako nejjednodušší variantu poskytovanou referenčním serverem. Z těchto důvodu není finální aplikace moc použitelná, je potřeba využít lepší a složitější metodu autorizace. Očekávaní příchozích zprav a interaktivní režim musí podporovat komunikace ze serverem, abych ten ne zavřel stream, proto oni využívají trochu zmíněny kód příjemce zprav ze soketů. Pro vetší použitelnost metod, wait taky očekává novy zprávy v novem procesu.

5.3 Interaktivní režim

Interaktivní režim na začátku implementoval pomoci vice vláknové aplikace. Jedné vlákno očekávalo vstup od uživatele, druhé příchozí zprávy ze serveru. Třetí vlákno vypisovalo všechno na obrazovku. Narazil jsem na velký problém během implementace. Python využívá buffer pro vstup, přičemž vstup pomoci knihovny sys taky využívá buffer, a prakticky neexistuje moznost cist tlačení kláves bez tlačítka *"Enter"* Existuje moznost využít modul tty, ale to potřebovalo bych zmínit už napsaný kód, protože má jiny přístup k výpisu čehokoliv do obrazovky. Mimo rozsah standartních knihoven existuje knihovny, který načítají jeden symbol, ale oni blokuji tlačeni *Ctrl-C*, což je velkým záporem. Rozhodnul jsem implementovat GUI.

Jako knihovnu pro grafické uživatelské rozhraní jsem zavolil *tkinter*. Není tato knihovna součásti standartních knihoven Python. Ale je součásti oficiální distribuce Python. Zjistil jsme, že není nainstalovaná na referenčním virtuálním strojů, ale nelze ji využít mimo balík Python, takže jediný validní způsob je instalace balíku *python3-tk*. Tato knihovna poskytuje jednoduchý vývoj uživatelského rozhraní pomoci widgetu. Odesílaní zprav realizováno pomoci callbacku. Příjem nových zprav je realizovan pomoci sheduled volaní funkce, která kontroluje frontu mezi procesové komunikace.

Později jsem změnil vice vláknový přístup na vice procesový. Hlavním důvodem je moznost zavřít child proces pomoci signálu ukončeni. V Python vlákna a procesy mají docela stejný syntaktické konstrukce, takže tato změna proběhla bez problémů. Jiny procesy, pracující ze soketem vyžadují zápis zprav do log souboru. Řešil jsem to správným způsobem, pomoci druhy fronty pro zprávy (zahrnuje i presence). Do souboru zapisuje jen jeden proces. Zapisuje se do souboru na začátku informace o tom, ze proces začal logovat. Přístup *try/except* je velmi užitečný pro vývoj síťových aplikace. V případě nějakého výpadku, jednoduše mužů to zjistit a převzat řízeni programu.



Obrázek 1: Přiklad GUI interaktivního režimu.

5.4 Využity knihovny

- ***argparse*** pro načtení vstupních parametru
- ***socket*** pro komunikace ze serverem
- ***base64*** pro encoding autorizace mechanismem PLAIN
- ***re*** pro kontrolu a parsing příchozích odpovědi
- ***time*** pro kontrolu casu mimo timeout, protože ten vynuluje v případě ping/presence zprávy
- ***multiprocessing*** pro implementace vice procesové aplikace a front pro mezi procesové komunikace
- ***tkinter*** pro implementace GUI

6 Základní informace o programu

6.1 Metriky kódu

Počet souborů: 1

Počet řádků zdrojového textu: 688

Velikost zdrojového souboru: 24 121 B

Zdrojové soubory formátovaný podle návodu *PEP8*.

7 Návod na použití

Před použitím je potřeba spustit inicializační skript *init.sh*. Ten nainstaluje balík *python3-tk* a nastaví pravá na soubor *xmppclient*.

```
$ chmod +x init.sh
$ ./init.sh
```

Pote je možně spouštět aplikace ze všemi dostupnými parametry:

```
$ ./xmppclient -s netfox.fit.vutbr.cz:5222 -r xlogin00:pass
$ ./xmppclient -s netfox.fit.vutbr.cz:5222 -l xlogin00:pass -c -a xlogin01@netfox.fit.vutbr.cz
$ ./xmppclient -s netfox.fit.vutbr.cz:5222 -l xlogin00:pass -m "testovaci_zprava" -u xlogin01@netfox.fit.vutbr.cz
$ ./xmppclient -s netfox.fit.vutbr.cz:5222 -r xlogin00:pass -l xlogin00:pass -w 60
$ ./xmppclient -s netfox.fit.vutbr.cz:5222 -l xlogin00:pass  -u xlogin01@netfox.fit.vutbr.cz -i
```

Pro výpis nápovědy je potřeba spustit program s parametrem *-h* nebo *–help*.

8 Použita literatura

- The XMPP Standards Foundation. RFC 6120, 2011
- Peter Saint-Andre , Kevin Smith, Remko Tronçon: XMPP: The Definitive Guide, 2009
- Python Software Foundation. Python Language Reference, version 3.4