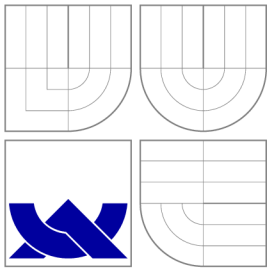
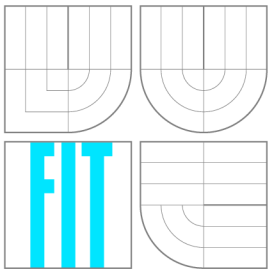


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLIENTSKÉ NÁSTROJE PRO PŘÍSTUP K CENTRÁLNÍMU REGISTRU DOMÉNOVÝCH JMEN

CLIENT TOOLS FOR ACCESSING DNS REGISTRY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VOJTĚCH ŠKORVAGA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2011

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Škorvaga Vojtěch**
Obor: Informační technologie
Téma: **Klientské nástroje pro přístup k centrálnímu registru doménových jmen**
Client Tools for Accessing DNS Registry
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s problematikou registrace DNS jmen u centrálního registrátora v ČR CZ.NIC. Popište současný stav registrace.
2. Seznamte se s protokolem EPP pro registraci doménových jmen.
3. Navrhněte a implementujte API pro další jazyky (C++, PHP, Java, apod.) pro překlad požadavků do protokolu EPP.
4. Otestujte vámi implementovaný systém. Zhodnoťte jeho použitelnost.

Literatura:

- S.Hollenbeck: Extensible Provisioning Protocol (EPP), RFC 4930. May 2007.
- P.Albitz: DNS and Bind, O'Reilly, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 - 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Matoušek Petr, Ing., Ph.D.**, UIFS FIT VUT
Konzultant: Surý Ondřej, CZ.NIC
Datum zadání: 1. listopadu 2010
Datum odevzdání: 18. května 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tento dokument popisuje práci řešící rozšíření sady registračních nástrojů pro registrování domén v aplikaci Fred (free registry for enum and domains). Nástroje jsem rozšířil o knihovnu a programy psané v jazyce C a Java, které zprostředkovávají komunikaci mezi centrálním registrem a klienty pomocí XML protokolu EPP (RFC 5730) a SSL/STL vrstvy. V implementaci pro jazyk C jsem pro zpracování XML použil knihovnou libxml2 a pro SSL/STL knihovnou OpenSSL. V Javě jsem v nástroji ANTLR navrhl postup automatizové tvorby XML souborů z XML schémat. Práce byla řešena ve spolupráci s organizací CZ.NIC (registrátoři domén).

Abstract

This document describes the work solving the extension of a set of registration utilities for registering domains in application Fred (free registry for enum and domains). I added library and programs written in Java and C language that enable communication between central register and clients via XML protocol EPP (RFC 5730) and SSL/STL layer. I used libxml2 library for processing XML and OpenSSL library for processing SSL/STL in C language. In Java I created system of automated creration XML files form XML schema. System is based on ANTLR. This work was solved in cooperation with organization CZ.NIC (domain registrators).

Klíčová slova

FRED, schéma XML, EPP, OpenSSL, ANTLR, DNS, registrace.

Keywords

FRED, schema XML, EPP, OpenSSL, ANTLR, DNS, registration.

Citace

Vojtěch Škorvaga: Klientské nástroje pro přístup k centrálnímu registru doménových jmen, bakalářská práce, Brno, FIT VUT v Brně, 2011

Klientské nástroje pro přístup k centrálnímu registru doménových jmen

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petr Matoušek, Ph.D.

.....
Vojtěch Škorvaga
17. května 2011

Poděkování

Tímto bych chtěl poděkovat Ing. Petr Matoušek, Ph.D. za trpělivost, pomoc a rady, které mi poskytl při řešení bakalářské práce. A Ondřeji Surému, technickému ředitelem CZ.NIC, z.s.p.o. za praktické rady a pomoc, které mi poskytl při řešení bakalářské práce.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá problematikou klientských nástrojů pro registraci záznamů v DNS registru systému Fred (free registry for enum and domains), který vytvořila organizace CZ.NIC. Práce byla řešena ve spolupráci s organizací CZ.NIC, pod kterou spadá správa internetové domény cz. Práce se zabývá možnostmi a postupem rozšíření klientských registračních nástrojů pro systém Fred.

Nástrojů vytvořených v dalším programovacích jazycích. Tím by registrátoři i tvůrci systému Fred měli širší možnosti při volbě svého řešení, tak aby systém Fred získal lepší pozici pro jeho nasazení v případě, že by správce domény chtěl změnit systém pro správu doménových jmen.

Cílem práce je navrhnout a vytvořit knohovnu funkcí a aplikace v jazycích Java a C pro komunikaci s centrálním registrem doménových jmen (Fred serverem) s využitím specificky upraveného EPP protokolu.

V tomto dokument je popsán současný stav systému Fred, protokolu EPP (kapitola 2), implementačního řešení klientské části protokolu EPP v programovacích jazycích C (kapitola 3) a Java (kapitola 4). Také je posán postup ověření správného fungování klientu.

V kapitole 3 je popisováno vytvoření knihovny libFredClient pro registraci domén v systému Fred. Knihovna pro vytváření EPP příkazů, jejich zasílání internetem a analyzování odpovědi. Výsledkem této práce je knihovna libFredClient a registrační nástroje (3.8) prezentující její použití. Práce popisuje metodiku testování knihovny libFredClient, kterou bylo ověřeno, že vytvořené registrační nástroje splňují základní funkcionalitu pro provádění registrací v systému Fred.

Řešení v Javě (kapitola 4) prezentuje mé řešení tvorby EPP příkazů, na základě vstupního řetězce a XML schémat popisujících určitou verzi protokolu. Kapitola popisuje princip načítání XML schémat a převod vstupního řetězce na EPP příkaz.

Kapitola 5 se zabývá postupem, kterým jsem ověřoval funkčnost knihoven. Knihovny byly prezentovány na jednoduchých ukázkových programech, které mají zakladní funkcionalitu pro přístup do centrálního registu doménových jmen.

Kapitola 2

Současný stav

Tato kapitola popisuje současný stav a strukturu nástroje Fred pro registrování domén, vytvořeného organizací CZ.NIC. Kapitola pojednává o protokolu EPP, kterým v systému Fred registrátoři komunikují s centrálním registrem. Protokol EPP je vystavěn nad jazykem XML, proto je v této kapitole rozebírán i jazyk XML.

2.1 Systém Fred

Fred z anglického „Free Registry for Enum and Domain“ je sada nástrojů pro správu domény. Je volně šiřitelný pod licencí GNU zkratka z General Public License[1] . Obrázek 2.1 zachycuje jeho třívrstvou architekturu, která odděluje komunikaci s registrem od databáze registru, aby ji neohrozili klienti v případě špatného vstupu. Skládá se z centrálního registru (dále jen CR), rozhraní pro administrátory, rozhraní pro registrátory a rozhraní pro veřejnost.

Obrázek 2.1: Tří vrstvá architektura systému Fred.

1. Příkaz je zaslán z klienta do centrálního registru.
2. Pokud se v žádosti nevyskytují chyby, centrální registr provede nad databází operace požadující příkaz. Jinak je klientovi zaslána chybová odpověď, bod 4.
3. Databáze vrátí výsledek operace centrálnímu registru.
4. Centrální registr zašle odpověď klientovi.

[1] Více na www: `<http://www.gnu.org/copyleft/gpl.html>`

2.1.1 Centrální registr

Centrální registr (CR) - Je umístěn mezi databázi a klienty (obrázek 2.1). Klienti posílají CR příkazy, které provede nad databází a zašle odpověď. Dále vykonává periodickou kontrolu nad databází, na jejímž základě provádí automatické akce (s blíží se exspirací domény zašle email s upozorněním atd..). Je implementován jako modul c v C++ pro serveru Apache 2.0.

2.1.2 Rozhraní pro administrátory

Rozhraní pro administrátory poskytuje nástroje pro zobrazování a manipulaci s daty a nástroje pro údržbu systému. Rozhraní je na serverové straně napsáno v C++ a je přímo začleněno do jádra centrálního registru. K administrátorské části serveru je možno přistupovat přes protokol CORBA. Klient administračního rozhraní je webovou aplikací napsanou v jazyku Python.

2.1.3 Rozhraní pro veřejnost

Rozhraní pro veřejnost realizuje přenos dat pro službu whois a statistiky. Rozhraním pro přenos je webová aplikace, která je chráněna proti strojovému vyhledávání metodou CAPTCHA nebo standardním protokolem whois. Protokolem whois se přenáší méně údajů, protože nemá podporu pro zamezení automatizovanému dotazování.

2.1.4 Rozhraní pro registrátory

Rozhraní pro registrátory se skládá z modulu v httpd serveru Apache 2.0 a sadou klientských konzolových nástrojů. Rozhraní je napsáno v Pythonu a vystavěno nad protokolem TCP/IP s využitím protokolu EPP. Pro bezpečný přenos dat používá SSL. Modul serverové části se jmenuje mod_eppd. SSL komunikace je na serveru rozluštěna modulem mod_ssl, který je vestavěnou součástí systému Apache.

Klientská část je tvořena několika klientskými nástroji:

Client
: Interaktivní správa CR. Vytvoří spojení se serverem. Spojení zůstává aktivní, dokud se příkazem pro odhlášení uživatel neodhlásí. Spojení je také ukončeno po dlouhé nečinnosti registrátora, nebo také po chybě.

Creator
: Tiskne na standardní výstup požadavky protokolu EPP.

Sender
: Pošle požadavek vytvořený creatorem do CR.

2.2 Protokol EPP

Extensible Provisioning Protocol byl navržen pro komunikaci mezi registrátory a doménovými registry. Specifikován k roku 2009 dle RFC 5730[2]. Většina implementací se jím striktně neřídí, ale stejně jako Fred používá RFC 5730[2] jako základ pro implementaci konkrétního řešení. Protokol EEP je textově orientovaný protokol, jenž má strukturu řetězců XML, jejichž formát je definovaných schématy XML. Zprávy jsou přenášeny mezi serverem a klientem přes internet. Zprávy jsou dvojího druhu:

Příkazy jsou zprávy zasílané klientem na server.

Odpovedi jsou zprávy zasílané serverem klientovi. XML dokument odpovědi obsahuje informace o úspěšnosti provedeného příkazu a chybovým hlašením v případě neúspěchu nebo požadovanými informacemi.

EPP definuje jen obecné operace nad objekty, a postup rozšíření protokolu, aby odpovídal požadavkům kladeným pro konkrétní implementaci. Registrátoři mohou protokolem EPP provádět operace v centrálním registru.

Dokument XML je uložen v čistém textu a formátem XML nebo EPP není definováno jakým způsobem zabezpečovat důvěrné údaje. Proto RFC 5734 [3] „EPP Transport over TCP“ sděluje, že pokud je SSL/TLS vrstva začleněna do EPP serveru a do EPP klienta, tak musí být použita při komunikaci. Ke každému odeslanému příkazu zasílá server EPP odpověď. Server EPP přijímá zprávy v textovém kódování UTF-8, RFC 5730 [3] umožňuje rozšíření na UTF-16.

2.2.1 Příkazy EPP

Základní sada příkazů EPP umožňuje v systému Fred správu spojení a operace nad třemi typy objektů. Tato sada příkazů je definována v RFC 5730. Jednotlivé objekty protokolu EPP:

Contact
Objekt pro uložení kontaktních údajů osoby nebo organizace.

Domain
Objekt reprezentující doménu, její jméno, správce atd.

Nsset
Objekt reprezentující skupinu name serverů.

Každý příkaz je odesílán se svým jedinečným id, které server vrací v odpovědi. Samotné příkazy se dělí do tří sekcí:

1. **Stavové (Management)** řídí spojení (login, logout, hello).
2. **Dotazovací (query)** nemění stav objektu (check, info, poll, transfer query).
3. **Výkonné (transform)** mění stav objektu (create, delete, renew, transfer, update).

Časový diagram komunikace protokolu EPP na obrázku 2.2 zobrazuje komunikaci klienta se serverem. Komunikace je zakreslena od okamžiku vytvoření zabezpečeného spojení. Komunikaci začíná server zasláním uvítací zprávy. Uživatel je po přijetí uvítací zprávy přihlášen na server a může posílat příkazy, na které server odpovídá. Server nemusí odpovídat chronologicky podle toho, jak příkazy přijal.

2.2.2 Rozšíření protokolu EPP v systému Fred

Rozšíření protokolu EPP v systému Fred je popsáno v RFC 3735. Rozšíření umožňuje registraci dalšího objektu pojmenovaného „keyset“. Objekt keyset je sadou klíčů pro technologii DNSSEC, která má zabraňovat podvržení DNS záznamů. Systém Fred rozšiřuje základní množinu operací protokolu EPP o příkazy:

Příkaz	Popis funkce příkazu
`login`	Zahájí relaci. Bez příslušnosti k objektu.
`logout`	Ukončí relaci. Bez příslušnosti k objektu.
`hello`	Udržení relace. Bez příslušnosti k objektu.
`check`	Zjišťuje, jestli objekt s daným identifikátorem může být vytvořen v CR.
`info`	Vypisuje atributy objektu s daným identifikátorem. Jaké množství informace se zveřejní, záleží na mnoha faktorech.
`poll`	Vyžádá si zprávu z fronty zpráv pro uživatele a informace o jejím stavu. Neváže se k žádnému objektu.
`transfer`	Provede změnu registrátora objektu.
`create`	Vytvoří v CR nový objekt.
`delete`	Vymaže z CR daný objekt.
`renew`	Prodlužuje platnost objektu. Definováno pouze pro objekt domény.
`update`	Změní požadovaný atribut daného objektu na danou hodnotu.

Tabulka 2.1: Přehled základních příkazů protokolu EPP.

Příkaz	Popis funkce příkazu
`test`	Požadavek na CR o test name serveru, objektu nsset.
`list`	Připraví seznam jednoho ze čtyř objektu. Seznam čtený příkazem `getResult`.
`prep`	Příkaz list se selekcí objektu podle vztahu k jinému objektu.
`getResult`	Vyzvedává položky ze seznamu.
`sendAuthInfo`	Zapříčiní zaslání emailu o objektu z CR na adresu příslušného kontaktu.
`creditInfo`	Vrací registrátorův kredit pro jednotlivé zóny. Tento příkaz není spojen s žádným objektem.

Tabulka 2.2: Přehled rozšiřujících příkazu systému Fred, pro protokol EPP.

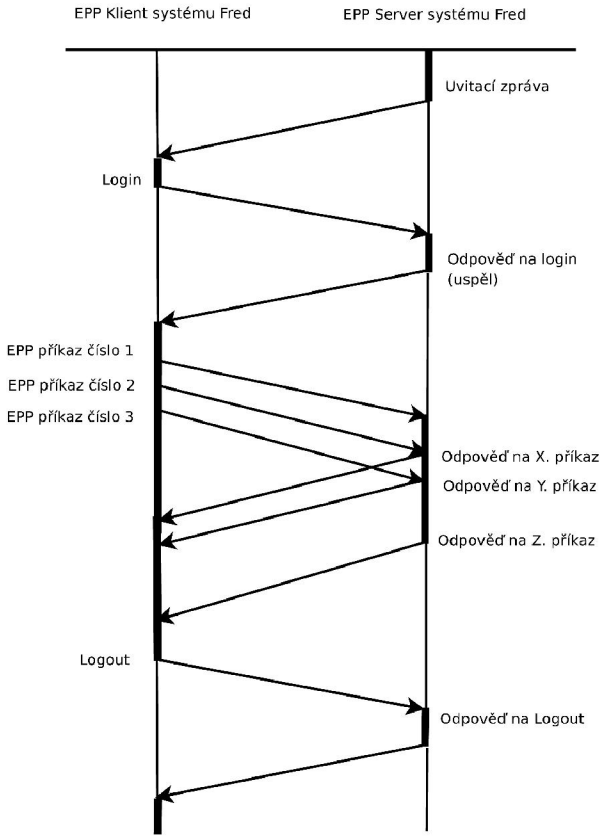
2.2.3 Formát EEP přes TCP

Je popisován v RFC 5734[3]. Přenášená zpráva je doplněna o hlavičku, která je umístěna před samotnou zprávu (obrázek 2.3). Její velikost je 32 bitů. Obsahem hlavičky je celková velikost zprávy v bajtech. Do velikosti zprávy se počítá i velikost samotné hlavičky, tedy počet bajtu EPP zprávy plus čtyři bajty hlavičky. Proti zneužití dat posílaných přes síť je v RFC 5734[3] napsáno, že musí dojít k použití šifrovaného spojení, pokud je podporováno klientem i serverem.

2.2.4 Znovu připojení uživatele k EPP serveru

Zajímavou vlastností protokolu EPP je neschopnost znovu připojení uživatele příkazem „login“ po vykonání příkazu „logout“ z aktivního spojení.

Dle specifikace protokolu stavovým automatem lze úspěšně provést příkaz login pouze jednou, ale před jeho úspěchem může proběhnout neurčený počet neúspěšných pokusů o přihlášení. Po odhlášení reaguje server Fred na pokusu o znovu-přihlášení uživatele zasláním 32-bitové odpovědi. V odpovědi je pouze binárně uloženo číslo čtyři, což znamená žádná odpověď od EPP serveru. Popisovaná situace nastane i po dlouhé nečinnosti klienta, při-



Obrázek 2.2: Časový diagram protokolu EPP.

čemž množství času označující pojem dlouho určí server. Nečinnost klienta nastává, pokud neposílá žádné zprávy na server.

Postupem, jak se znovu po odhlášení připojit na server systému Fred, je uzavření spojení a jeho znovuotevření před pokusem o přihlášení.

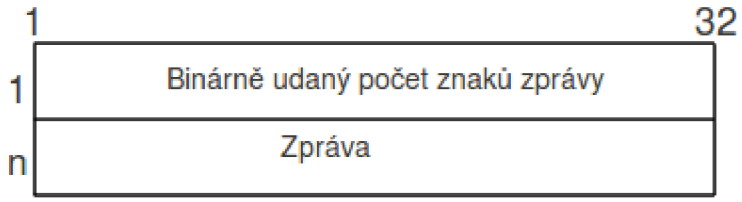
2.3 Extensible Markup Language

Extensible Markup Language zkráceně XML je značkovací jazyk vyvinut konsorciem W3C, pro přenos dat ve standardizovaném formátu [4]. Vychází z „Standard Generalized Markup Language“ (SGML) a mezi jeho hlavní vlastnosti patří flexibilita. Díky níž lze přidáním sémantiky některým značkám definovat nový jazyk. Protokoly vytvořeny nad XML jsou znakově orientovanými, například komunikační protokol Jabber.

Flexibilita XML je také komplikací, především při metodách formální definice nového jazyka. Problém se snaží řešit několik projektů. Jedním z nich, opět pod záštitou W3C, je projekt XML schémat.

XML rozeznává několik pojmů. „Tag“, taktéž značka je zapisován do lomených závorek, uvozující tag `<jmeno_uzlu>` a ukončovací tag `</jmeno_uzlu>` se zpětným lomítkem před jménem uzlu.

Tagy uvozují a ukončují uzly. Pokud má uzel atributy jsou uvedeny v uvozujícím tagu `<jmeno_uzlu jmeno_atributu="hodnota atributu">`, jeho hodnota je uzavřena v uvozovkách a od jména atributu je oddělena rovnítkem. Počet atributů není omezen a jsou od



Obrázek 2.3: Formát EPP přes TCP.

Soubor schématu XML	Určení souborů
all-2.2.xsd	Schéma importující všechny ostatní definiční soubory.
epp-1.0.xsd	Schéma pro základní příkazy EPP protokolu.
eppcom-1.0.xsd	Schéma pro základní datové typy EPP protokolu.
contact-1.6.xsd	Schéma s rozšířením pro správu kontaktů.
nsset-1.2.xsd	Schéma s rozšířením pro správu sad nameserverů.
keyset-1.3.xsd	Schéma s rozšířením pro správu sad klíčů.
domain-1.4.xsd	Schéma s rozšířením pro správu domén.
enumval-1.2.xsd	Schéma s rozšířením pro správu enum domén.
fred-1.4.xsd	Schéma s funkcemi přidanými nad rámec standardu EPP.
fredcom-1.1.xsd	Schéma se společnými strukturami nad rámec eppcom.

Tabulka 2.3: Popis rozdělení zodpovědnosti souborů schémat XML.

sebe odděleny bílými znaky. Obsahem uzlu jsou další uzly nebo řetězec. Pokud uzel nemá žádný obsah, může být zápis zkrácen na jeden tag s lomítkem před uzavírající hranatou závorkou `<jmeno_uzlu/>`.

2.3.1 Schéma XML

XML schéma je standardizovaný značkovací jazyk pro tvorbu jiných značkovacích XML jazyků. Schéma XML je dokument dokumentem XML definující podobu jiného dokumentu XML. Standard XML schémat určuje sémantiku některým značkám, datovým typům. Ty jsou rozděleny na jednoduché a složené, přičemž nový složený datový typ vznikne spojením minimálně dvou značek již existujících typů, ať jednoduchých či složených. Soubory schématu XML mohou mít příponu ve tvaru xsd.

2.3.2 Schémata XML v aplikaci Fred

Podoba XML zprávy v protokolu EPP je definovaná schématem XML[4]. V implementaci Fredu to jsou následující soubory[2] schémat XML:

Obsah, jméno a počet schématech XML protokolu EPP pro projekt Fred se s vysokou pravděpodobností změní, protože je stále aktivně vyvíjen a upravován.

[2]Více na www: `<http://dsdng.nic.cz/schema/production/>`

2.4 Shrnutí kapitoly

Tato kapitola vystihuje princip fůnkce registračního systému Fred. Byl popsán textově orientovaný protokol EPP, kterým komunikuje registrátor domén s centrálním registu. Aktuálně (18.května 2011) nejnovější verze EPP protokolu v systému Fred odpovídá XML schématum uvedených v tabulce 2.3. Pro správné implementování klienta EPP je dále důležité znát způsob přenosu EPP přes internet, který byl popsán v sekci 2.2.3.

Kapitola 3

Implementace knihovny v jazyku C

Knihovna, která je vytvořena na základě zadání této bakalářské práce poskytuje funkce a procedury pro tvorbu klientských aplikací protokolu EPP v systému Fred. Knihovna libFredClient je napsána jazykem C pomocí knihovny libxml2. Knihovna libxml2 umožňuje vytváření příkazů XML v protokolu EPP, jejich validaci a analyzovaní odpovědí na zaslané příkazy. Modul pro bezpečnou síťovou komunikaci je vystavěn nad knihovnou OpenSSL podle článku [1] a podporuje IPv4 i IPv6 adresaci. Knihovna obsahuje modul pro načtení konfigurace z konfiguračního souboru.

3.1 Implementované lineární seznamy

Lineární seznamy byly implementovány pro uchovávání dat z uživatelského vstupu nebo z konfiguračního souboru. Implementované lineární seznamy se liší uchovávanými informacemi nebo jejich sémantikou. Jejích výčet je v tabulce 3.1.

Jméno seznamu	**Určení**
`simply_string_list`	Seznam obsahující řetězce, v C tedy ukazatel na datový typ `char`.
`keyset_ds_list`	Seznam obsahující uživatelský vstup záznamu Delegation Signer (DS) při vytváření objektu keyset pro DNSSEC.
`keyset_dnskey_list`	Seznam obsahující uživatelský vstup záznamu dnskey při vytváření objektu keyset pro DNSSEC.
`VarList`	Seznam určený pro přechovávaní dat z konfiguračního souboru.
`UserList`	Seznam určený k ukládání loginu a hesel uživatelů definovaných v konfiguračním souboru.

Tabulka 3.1: Přehled lineárních seznamů knihovny libFredClient.

Všechny lineární seznamy mají stejnou hlavičku.

```
struct jméno_linearního_seznamu {
     datova_struktura_ln First;
     datova_struktura_ln Act;
     datova_struktura_ln End;
                              };
```

3.2 Analýza souboru pro DNSSEC

Pro analýzu souboru pro DNSSEC k objektu keyset jsem sestavil regulární výrazy. Analyzovány jsou soubory reprezentující záznamy DNSKEY a DS z konfiguračního souboru DNS. Jazyk C nemá standardní podporu regulárních výrazů, proto jsem použil standardní linuxovou knihovnu Regex (regex.h), která dodává podporu regulárních výrazů.

Delegation Signer (ds) záznam určí podpis k delegované zóně.	
vzor záznamu DS:	”cz. 3600 IN DS 20487 5 1 1ff4b01e82cd41f2edb65b925d3f4b2ab68a4467”
Regulární výraz pro C	`"[ \t]*[a-zA-Z0-9]+\.[ \t]+([0-9]*)[ \t]+IN[ \t]+DS[ \t]+([0-9]1,5) +([0-9]1,3) +([0-9]1,3) +([0-9a-fA-F]+)"`

Tabulka 3.2: Regulární výraz pro záznam DS.

Záznam DNSKEY.	
vzor souboru se záznamem DNSKEY	”cz. IN DNSKEY 256 3 5 AwEAAddt2AkL-fYGKgiEZB5SmIF8EvrjxNMH6HtxWEA4RJ9Ao6LCWheg8 TSoH4+jPNwiWmT3+PQVbL5TD90KVw6S09Ae9cYU8A7xnZW-kfzq8q2pX6 7yVvshlQqJnuSV6uMBEMziIGu3NZEJb9eTl1T5q1-cli7Fk+xTt5GVvZR 3BJhtRAf”
Výraz záznamu DNSKEY	`"[ \t]*[a-zA-Z0-9]+ \.[ \t]+IN[ \t]+DNSKEY[ \t]+([0-9]1,5) +([0-9]1,3) +([0-9]1,3) +(\.*)"`

Tabulka 3.3: Regulární výraz pro záznam DNSKEY.

3.3 Vytváření a analyzování dokumentu XML

Vytváření a analyzování dokumentu XML je vystavěno nad knihovnou libxml2, která má rozhraní pro jazyk C. Obsahuje moduly pro tvorbu a zpracování dokumentů XML. Syntaxe zasílaných zpráv a odpovědí je definovaná v jednom ze schémat XML.

3.3.1 Datové struktury libxml2 pro tvorbu XML

EPP příkazy jsou dokumenty XML. Při tvorbě dokumentu XML využívá knihovna libFredClient několika datových struktur knihovny libxml2. Výčet použitých struktur je v tabulce 3.4.

3.3.2 Tvorba EPP příkazů

Kód pro tvorbu příkazů umožňuje pouze jejich statickou tvorbu. Funkce knihovny nezvládnou změnu schémat XML samy bez úpravy jejich vnitřních kódů.

Knihovnu jsem se pokoušel navrhnout s ohledem na uživatelské rozhraní již existujících nástrojů vytvořených v jazyce Python. A to tak, aby bylo zachováno uživatelské rozhraní, na které si uživatelé systému Fred zvykli. Zachování rozhraní jsem se pokoušel dosáhnout rozdělením uživatelského vstupu na seznam textových řetězců nazývaných tokenů. Stavovou

Struktura	Ukazatel	Popis
`xmlDoc`	`xmlDocPtr`	Struktura dokumentu XML, v níž je strom uzlů. V knihovně vytvoří dokument funkce `FredXmlDocInit()` a maže `FredXmlDocFree()`
`xmlChar`	`xmlCharPtr`	Je datový typ bezznaménkového charu. Je naformátován v UTF-8 kódování. Pro převod z typu `char` na `xmlChar` definuje libxml makro `BAD_CAST`, které je vytvořeno kódem `(unsigned char*)`.
`xmlNode`	`xmlNodePtr`	Je uzlem XML, na nějž se napojují další.
`xmlAttr`	`xmlAttrPtr`	Atribut XML tagu
`xmlNs`	`xmlNsPtr`	Struktura jmenného prostoru anglicky ”namespace”, pro schémata XML

Tabulka 3.4: Přehled datových struktur z knihovny libxml2 použitých v knihovně libFredClient.

analýzu vytvořeného seznamu pro každý tvořený příkaz EPP provádím samostatně. Uživatelský vstup se funkcí `argumentsToList()` převede na seznam tokenů. Seznam je datového typu `simply_string_list`. Hodnota záznamů obsahuje pouze řetězce. Token v seznamu může nabývat tří stavů:

1. ”(”- levá závorka, Pythonem používána pro označení začátku seznam.
2. ”)”- pravá závorka, Pythonem používána pro označení konce seznam.
3. řetězec – řetězec, který není pravou ani levou závorkou, nese data.

Samostatný znak pravé nebo levé závorky nesmí byt použit ve smyslu dat pro centrální registr. Navíc uživatelé Fredu jsou zvyklí na použití závorek jako řídící informace pro klientské nástroje. Pokud by chtěl uživatel zadat znaky `( )` jako součást nějakého řetězce, musí tento řetězec být uzavřen do znaků `""` nebo `' '` . Příkladem muže byt uživatelský vstup pro vytvoření domény v registru:

```
create_domain domain.cz cid:regid NULL NULL NULL (3 y) (cid:admin1,
'cid:admin(2)')
```

Tento uživatelský vstup vytvoří příkaz create nad objektem domain. Uživatelský vstup bude funkcí `argumentsToList()` převeden na list řetězců tohoto tvaru:

```
 {"domain.cz","cid:regid","NULL","NULL","NULL","(","3","y",")", "(",
"cid:admin1","cid:admin(2)",")"}
```

Dokument XML každého Fred příkazu je uložen ve struktuře `xmlDoc`. Dokument je vytvářen samostatnou funkcí s prefixem `command_`. Pro každý příkaz je jedna funkce. A každá taková funkce tvořící příkaz má přidruženou funkci s prefixem `listToCommand_`.

Funkce s prefixem `listToCommand_` zpracovávají list tokenů vytvořený z uživatelského vstupu. Ve svém těle volá svou přidruženou funkci s prefixem `command_` pro tvorbu XML dokumentu příkazu. Na rozdíl od funkcí pro tvoření příkazů mají funkce pro zpracování uživatelského vstupu totožné rozhraní.

Ke každému příkazu v modulu jsou dvě funkce, první s prefixem `command_` a druhá s prefixem `listToCcommand_`. Dále moduly mohou obsahovat funkce pro tvorbu částí příkazů umístěných v jiných modulech.

Jméno modulu	schéma XML	Popis modulu
`commands_epp`	epp-1.0.xsd	Tvorba základních EPP příkazu. Příkazy Login, logout, hello a pool.
`commands_contact`	contact-1.5.xsd	Tvorba příkazů pro objekt contact. Příkazy Create_contact, delete_contact atd.
`commands_nsset`	nsset-1.2.xsd	Tvorbu příkazů pro objekt nsset. Příkazy Check_nsset, update_nsset atd.
`commands_keyset`	keyset-1.2.xsd	Tvorba příkazů pro objekt keyset.
`commands_domain`	domain-1.4.xsd	Tvorbu příkazů pro objekt domain.
`commands_enumval`	enumval-1.1.xsd	Rozšíření protokolu EPP o enum domény, funkce z něj jsou volány v modulu commands_domain. Enum část pro příkazy create_domain, update_domain a renew_domain.
`commands_fred`	fred-1.4.xsd	Rozšíření ze systému Fred v protokolu EPP. Příkazy: credit_info, get_result, list_, prep_, senauthinfo_ atd.

Tabulka 3.5: Přehled modulů knihovny libFredClient pro tvorbu příkazů EPP.

3.3.3 Validace XML dokumentu proti schématu

Validace je proces kontroly dokumentu XML. Pokud validujeme proti schématu XML, tak kontrolujeme, zda mu XML odpovídá. Zda má schématem předepsanou strukturu, názvy uzlů a zda odpovídají rozsahy datových typů.

3.3.4 Validace dokumentu XML

Validací dokumentu XML, který obsahuje příkaz EPP, kontroluji jeho správné sestavení a uživatelský vstup. Schéma XML je šablonou, které musí odpovídat správně sestavený dokument XML. Validací se kontrolují příkazy protokolu EPP na straně klienta. Příkaz může být zaslán na server, přestože validace hlásí chybu. Pokud je na server zaslán nevalidní příkaz, je skoro jisté, že příkaz bude na severu odmítnut při kontrole vstupů a sever zašle informaci o chybném formátu. Funkce `xml_Validate_print()` provádí validaci vytvořeného XML dokumentu. Funkce využívá nástrojů knihovny libxml2 a objevené chyby tiskne na standardní chybový výstup.

Rozhraní funkce: `int xml_Validate_print (xmlDocPtr Doc, const char * schema_File_Name)` Vrací `OK` pokud validace proběhla úspěšně nebo jednu ze dvou chyb `"NON_VALID"`, `"ERROR_VALIDATE_INTERNAL"`. Vstupem do ní je ukazatel na XML dokument a jméno souboru XML schématu, proti kterému se bude validace provádět.

3.3.5 Metody zpracování dokumentu XML

W3C definovalo pro tento účel metodu s anglickým názvem "Document Object Module" zkráceně DOM. DOM byl navržen pro objektový přístup k dokumentu XML. API metody DOM umožňuje volný pohyb po stromě dokumentu XML a přepisování elementu uzlů v dokumentu. Avšak jeho výhody jsou vykoupeny vysokou paměťovou náročností, protože během zpracování se dokument několikrát nahraje do paměti. Z tohoto důvodu je vhodnější využít metodu DOM pro malé dokumenty, jako například EPP příkazy. Ovšem nelze zaručit,

že individuální specifikace protokolu EPP nebude obsahovat velký dokument, už jen proto, že velikost XML dokumentu je relativní k závislosti na systémových zdrojích.

Jako alternativa k metodě DOM byla vyvinuta metoda SAX, zkratka z "Simply Api for Xml". Taktéž pro objektový přístup. Díky proudovému zpracování dokumentu je rychlejší a má výrazně nižší nároky na paměť, ale nelze dokument měnit nebo se po něm pohybovat jinak, nežli vpřed. Proudové zpracování znamená postupné čtení dokumentu. Při něm se volají obslužné rutiny, definované programátorem, vždy po výskytu události. Příkladem události, která může nastat, je začátek dokumentu, jeho konec, načtení jména nového uzlu, načtení řetězce v uzlu, načtení atributů, atd.

3.3.6 Analýza serverové odpovědi

Pro zpracování přijatých odpovědí jsem zvolil metodu analyzování dokumentu XML jménem SAX1, protože analyzovaný dokument budu pouze číst. Metodu SAX jsem vybral také pro její nízké nároky na paměť a jednoduché použití v implementaci knihovny libxml2. Použití SAXu vyžaduje napsání obslužných funkcí, které se připojí k analyzátoru pomocí ukazatele. Struktura analyzátoru SAX udržující seznam obslužných funkci se jmenuje xmlSAXHandl. Pokud nastane událost, je zavolána její obslužná funkce. Proto je nutné nastavit zpětné volání neobsluhovaných událostí na hodnotu NULL, aby nedošlo k chybám za běhu programu. Aby mohl být analyzátor SAX použit, musí být deklarován jako globální proměnná. Používání globálních proměnných není z hlediska jazyka bez jmenných prostorů nejvhodnějším řešením, kvůli možným konfliktům identifikátorů.

Pro zpracování odpovědi jsem vytvořil dvě fůnkce:

- Funkci `start_element()` kterou připojuji na volání startElement, jenž se aktivuje po načtení jména uzlu i s jeho atributy.
- Funkci `chars_found()` připojenou k volání characters, která se zavolá pro načtení hodnoty uzlu.

Pro omezené upravení výstupního formátu používám ve funkci `start_element` filtr vytvořený z pole řetězců reprezentujících názvy uzlů, které se nemají tisknout. V těle funkce je pouze globální odkaz na implicitní pole řetězců, což dává možnost uživateli knihovny vytvořit vlastní filtr podle svého uvážení. Vlastní filtr má formát pole `char *filtr[]` a připojuje se k ukazateli `no_printable_arg_xml_node` procedurou:

`void fred_set_non_printed_nodes(char **string_array)`

Kde `string_array` je ukazatelem na vlastní filtr.

Další metodou úpravy zobrazeného výstupu je překlad. Pokud nalezený řetězec ve funkci `start_element` odpovídá některému z prvního řetězce, je nahrazen druhým řetězcem. Oba řetězce jsou v poli ukazatelů na dvojice ukazatelů na char. Formátu: `char *_global_transla te_element_for_print[][2]`. Obdobně jako při vynechávání tisku vybraných uzlů, může uživatel nadefinovat vlastní pole pro překlad a připojit ho na globální ukazatel procedurou `fred_set_translate_element(char *string_array[][2])`, která se chová stejně jako `fred_set_non_printed_nodes()`.

3.4 Konfigurační modul

Konfigurační modul obsahuje funkce a struktury pro načtení a uchování informací z konfiguračního souboru. Je pojmenován `"fred_setting"`. Funkcemi se nastavují a čtou informace ze struktury `"FredSetting"`, jenž uchovává základní údaje o spojení a nastavení klienta.

Obrázek 3.1 zachycuje příklad konfiguračního souboru.

```
;; Příklad konfiguračního souboru
; slouží jako poznámka

;; Připojení
dir = ssl
host = 147.229.XXX.XXX
port = 22351
ssl_cert = %(dir)/test-cert.pem
ssl_key  = %(dir)/test-key.pem
username = REG-FRED_A
password = passwd
username2 = REG-FRED_B
password2 = passwd

;; timeout spojení v sekundách
timeout = 10.0

;; Typ socketu. Platné hodnoty: IPv4 nebo IPv6.
;socket = IPv6

;; Vypnout proces automatického přihlášení na server po startu aplikace.
;nologin = y

;; Soubor se schématy XML pro validaci příkazu EPP protokolu
schema = schemas/all.xsd

;; Pokud není jazyk nastaven, použije se hodnota en
;; lang = cs
lang = en

;; řetězec pro prazdnů argumet příkazu, implicitně "NULL"
;null_value = None
```

Obrázek 3.1: Příklad konfiguračního souboru.

Struktura FredSetting:

```
typedef struct fred_setting{
     // spojeni
    UserList * users; // list uživatelů (login,password)
    char *host;         // jméno nebo ip adresa serveru v IPv4 nebo IPv6
    char *port; // port Fred serveru
    long timeout;     // množství času pro timeout
    char * socket;    // [ "IPv4" | "4" ] nebo  [ "IPv6" | "6" ]
    char * ssl_key_file;   // název souboru, soukromý klíč ve
                            //                      formátu pem
    char * ssl_cert_file;  // název souborů, certifikát ve formátu pem
    char nologin;          //  'y' nebo 'n'

      // nastavení klienta
    char * nullParam;     // vzorový řetězec pro přeskočení vstupního
                          //        argumentu při tvorbě EPP příkazu
    char * actual_user;   // actual username je nastaveno i funkci
```

```
                        //              Fred_Login_by_Setting();
  char * language;   // jazyk
  char * schema;   // název souboru pro, schémata XML
}FredSetting;
```

Při zpracovávání souboru jsou přeskočeny znaky od znaku středníku (;) a levé hranaté závorky ([) až do konce řádku. Pravidlo se nevztahuje na znaky uzavřeny do páru znaků uvozovek nebo apostrofů. Nejprve se načtou jména proměnných a jejich hodnoty, kde jména jsou odděleny od hodnoty znakem rovnítka.

Pokud se vyskytuje v hodnotě proměnné řetězec ve formátu "%(jméno)s"(bez uvozovek), je jméno v načteném listě proměnných hledáno jako jméno proměnné. Pokud je jméno nalezeno, dojde k nahrazení %(jméno)s za jeho hodnotu. Z čehož plyne, že proměnná pro specifikování hodnoty jiné proměnné musí být definovaná dříve, než je použita.

Pokud se jméno proměnné v listu proměnných shoduje s klíčovým slovem, je její hodnota uložena do struktury FredSetting. Proměnné odpovídajícím klíčovým slovům nemusejí být v souboru zapsány. Pak se v ukazateli určeném pro proměnnou vyskytuje hodnota "NULL".

Klíčové slovo	Popis
`host`	Klíčové slovo pro položku host.
`port`	Klíčové slovo pro položku port.
`ssl_cert`	Klíčové slovo pro jméno souboru ve formátu pem pro ssl certifikát.
`ssl_key`	Klíčové slovo pro jméno souboru ve formátu pem pro ssl klíč.
`schema`	Klíčové slovo pro jméno souboru schématu XML protokolu EPP.
`username`	Klíčové slovo pro přihlašovací jméno (login) uživatele.
`password`	Klíčové slovo pro přihlašovací heslo uživatele.
`timeout`	Klíčové slovo pro řetězcovou reprezentaci množství času pro timeout.
`socket`	Klíčové slovo pro typu socketu.
`nologin`	Klíčové slovo pro automatické přihlášení uživatele na server.
`language`	Klíčové slovo pro jazyk.

Tabulka 3.6: Přehled klíčových slov pro nastavení klienta protokolu EPP v systému Fred.

Klíčová slova username a password tvoří pár pro strukturu _user uloženou v listu users pro FredSetting. Konfigurační soubor takovýchto dvojic může obsahovat více. Dvojce se identifikuje řetězcem bezprostředně za klíčovým slovem, pokud je tedy v souboru napsáno "usernameID4"pak identifikátorem je ID4 a někde v souboru musí být i "passwordID4". Funkce vytvářející seznam loginů vyžaduje ke každému klíčí username i klíč password.

3.5 Stavba síťového modulu

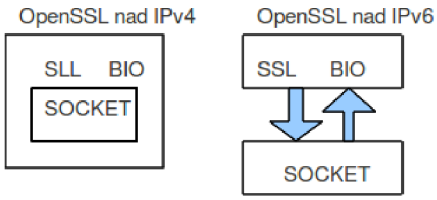
Síťový modul je nazván "libsender"a je napsán s využitím knihovny OpenSSL, kterou jsem zvolil pro bezpečnou komunikaci po sítí. OpenSSL jsem vybral proto, že je plnohodnotnou kryptografickou knihovnou s implementací ve většině používaných operačních systémech. Šifrované spojení bývá také nazýváno vrstva SSL/STL. Modul podporuje IPv4 i IPv6 adresaci.

3.5.1 Vrstva SSL/STL

Vrstva SSL/STL je umístěna mezi daty na sítí a zpracovávanými daty. Slouží pro bezpečnější přenos informací, slouží k snížení možnosti podvržení dat nebo zmírnění následků odcizení dat. Data jsou před odesláním zašifrována, přenesena a u příjemce před jejich zpracováním dešifrována.

3.5.2 Podpora IPv4 a IPv6 adresace v OpenSSL

Knihovna OpenSSL poskytuje programové rozhraní pro bezpečný přenos dat protokoly SSLv2/v3 a TSLv1. Konkrétně verze 0.9.8m. Struktura BIO je abstrakcí vstupu a výstupu. Funkce a metody s ní pracující zastřešují detaily V/V. Zvládá operace se soubory, síťový přenos nešifrovaný i šifrovaný. Struktura `SSL_CTX` je kontextem SSL spojení, do něhož jsou nahrávaný certifikáty a z ní je vytvořena struktura `SSL`. Struktura `SSL` je generována funkcí `SSL_new()`. Po jejím vytvoření se do ní nepropagují žádné změny provedené ve zdrojovém kontextu. OpenSSL dokáže pro spojení adresované pomocí IPv4 automaticky při vytváření struktury BIO vytvořit i soket, avšak pro IPv6 ne. To je zapříčiněno především malým objemem struktur uchovávajících adresu cíle která je v IPv6 4krát větší než pro adresu IPv4. Knihovna podporuje práci se soubory a sockety, tedy nepřímo i IPv6 adresaci. To znamená, že je nutné vytvořit socket a navázat spojení a až poté k socket funkcí `BIO_new_socket()` vytvořit strukturu `BIO`. K existující struktuře BIO lze připojit procedurou `SSL_set_bio()` vrstvu SSL. Logickou reprezentaci práce s IPv4 a IPv6 adresací je vidět na obrázku 3.2.



Obrázek 3.2: Princip OpenSSL v programu pro ipv4 a ip6.

3.5.3 Použití síťového modulu

Pro použití síťového modulu je zapotřebí zavolat proceduru `Fred_Ssl_Init()`. Data nejsou modulem přímo posílána a čtena, ale jdou přes SSL/STL vrstvu. Struktura pro uchovávání socketu a kontextu celého spojení:

```
typedef struct fred_ssl_socket{
    struct addrinfo hints; // informace o spojení
    SSL_CTX *ctx;          // SSL kontext
    SSL *sslSock;          // SSL mezivrstva mezi socketem a aplikací
    BIO *bio;        //
    int sock;              // socket
    long timeout;          // množství sekund pro timeout
} FredSslSocket ;
```

Struktura je naplněna během volání funkce vytvářející síťové spojení. Takovou funkcí je `Fred_Ssl_Socket_Init()` nebo `Fred_Ssl_Socket_Init_From_Setting()`. Druhá z jmenovaných funkcí přejímá informace pro vytvoření spojení ze struktury `FredSetting` modulu `fred_setting`. Pokud byl při volání funkcí `Fred_Read_Message()`, `Fred_Login()` a `Fred_Logout()` nastaven ukazatel na pole znaků pro odpověď serveru na neprázdnou hodnotu (ne NULL), je zapotřebí uvolnit paměť procedurou `Fred_Message_Char_Free()`. Funkce totiž alokují paměťové zdroje pro odpověď.

3.6 Reprezentace a překlad chyb

Reprezentace a překlad chyb se v knihovně libFredClient řeší moduly `error_defines` a `fred_language_string`. Pokud není určeno jinak, funkce vrací hodnotu datového typu int reprezentující chybový stav, který nastal během jejího vykonávání. Hodnota reprezentující správné dokončení funkce je 0 definovaná jako "OK"nebo také `NO_ERROR`. V opačném případě nastala chyba. Hodnoty chyb jsou definovány v souboru `error_defines.h`. Číslo chyby je určeno preprocesorovou klauzulí jazyka C `#define`. Číselnou reprezentaci nastalé chyby lze přeložit na chybové hlášení funkcí `Fred_Get_Err_String(int)`, která se vyskytuje v modulu `fred_language_string`. Funkce vrací ukazatel na řetězec, v němž je chybové hlášení vybrané podle vstupního parametru.

3.7 Přehled modulů knihovny libFredClient

Moduly pro tvorbu příkazů EPP byly již jmenovány. V následující tabulce jsou zobrazeny zbývající moduly. Diagram zobrazuje závislosti modulů v knihovně libFredClient a přípojení

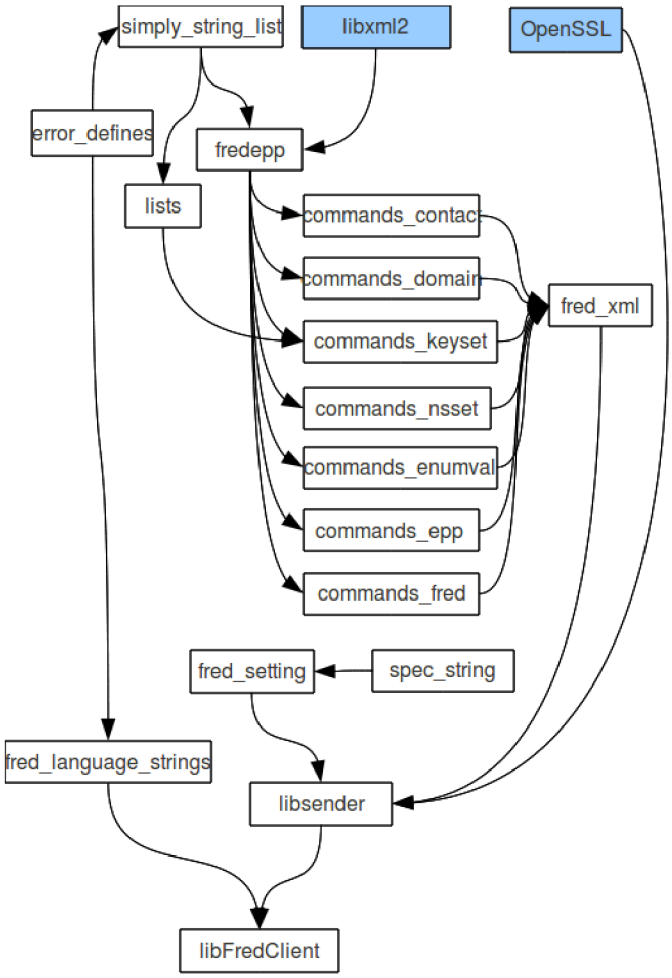
Jméno modulu	Popis modulu
error_defines	Definice chybových stavů.
fredepp	Definice základních hodnot pro moduly tvořící příkaz EPP.
fred_language_strings	Řetězce pro uživatelský výstup (chybové hlášení).
fred_setting	Uchovávaní nastavení klienta a načítaní konfiguračního souboru.
fred_xml	Zastřešuje moduly pro tvorbu příkazů EPP a analýzu dokumentu XML.
libsender	Síťová komunikace.
lists	Lineární seznamy pro práci s objektem keyset.
simply_string_list	Lineární seznam s ukazatelem na char jako daty.
spec_string	Knihovna pro nekonečné řetezce.

Tabulka 3.7: Popis modulů knihovny libFredClient.

knihoven libxml2 a OpenSSL ke knihovně libFredClient.

3.8 Ukázkové programy nad knihovnou libFredClient

Součást mé práce bylo vytvořit programy prezentující použití knihovny libFredClient. Programy mají splňovat základní funkcionalitu klientské části implementace Fred protokolu EPP. Programy pro příkazový řádek implementují vytvoření EPP příkazu a komunikaci se



Obrázek 3.3: Modulové závislosti.

serverem. Nástroj vytvářející XML dokumenty se jmenuje `fred-creator`. Příkazy vytvořené programem `fred-creat` lze zaslat na server pomocí programu `fred-sender`. Program `fred-client` umožňuje interaktivní registraci, tedy vytváření příkazů EPP a komunikaci se serverem.

3.8.1 Nástroj fred-creator

Program `fread-creator` je jednoduchou alternativou Pythonovského skriptu `creator.py`, jenž se snaží zachovávat uživatelské rozhraní jeho předlohy. Umožňuje vytvořit příkaz z parametru programu i z definovaného souboru. Soubor obsahuje vstup pro tvorbu příkazu EPP, na jednom řádku vždy vstup pro jeden příkaz. Výstup je zobrazen na standardní výstup. Přesto, že se fred-creator snaží zachovat uživatelské rozhraní, nedovoluje definovat hodnotu uzlu XML se jménem `cltrID` jako jeho protějšek napsaný v pythonu `creator.py`. Hodnotu uzlu `cltrID` program určí samostatně na základě času vytváření a vytvářeného příkazů.

Obrázek 3.4: Program fred-creator při tvorbě příkazu EPP.

3.8.2 Nástroj fred-sender

Program fred-sender je jednoduchým nástrojem pro komunikaci se serverem EPP postaveným nad síťovým modulem knihovny libFredClient. Program je určený pro hromadné zaslání několika příkazů EPP na server.

3.8.3 Nástroj fred-client

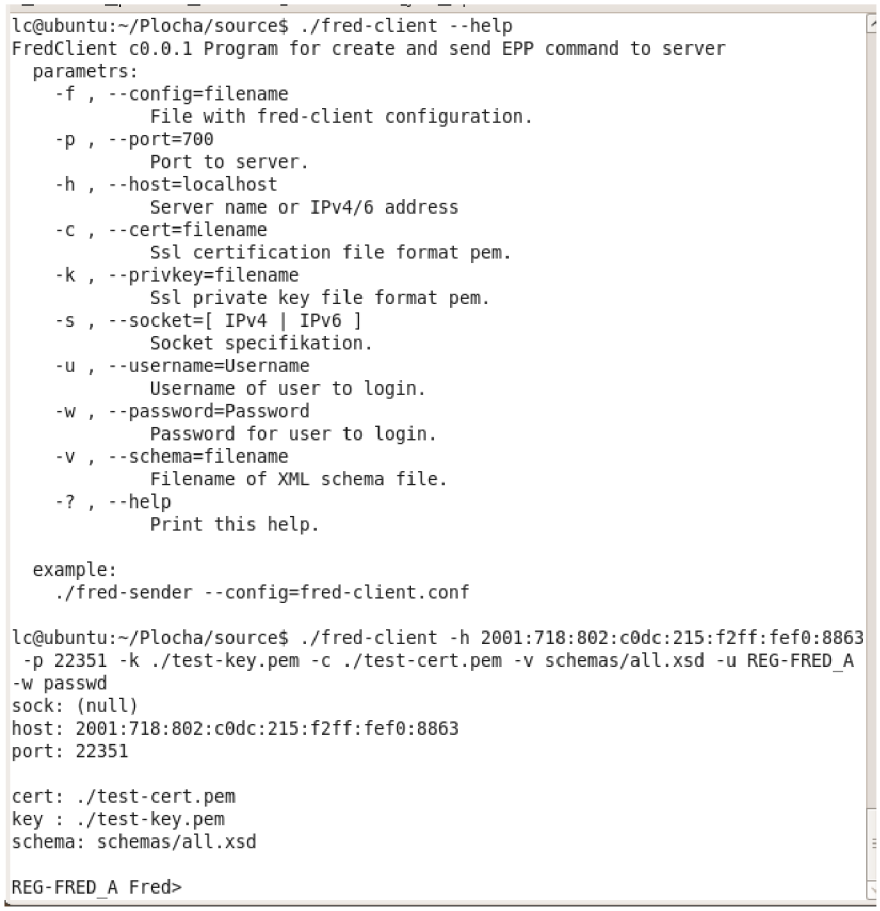
Program fred-client je jednoduchý interaktivní program pro správu CR z příkazové řádky. Kombinuje vlastnosti tvoření příkazu a jeho zaslání. Obrázek 4.1 znázorňuje stavy a přechody programu fred-client. Program je řešen s návazností na knihovnu "libedit", která slouží pro ukládání historie uživatelského vstupu v programu.

Pro inicializování ssl vrstvy může klient inicializovat socket ze struktury FredSetting, která je naplněna buď z konfiguračního souboru, nebo z parametrů spuštěného programu. Jako parametr host akceptuje DNS jméno nebo adresu ve formátu IPv4 či IPv6. Pro šifrované spojení je zapotřebí načíst šifrovací soubory, tzn. zadaná jména certifikátu soukromého klíče jsou jmény souboru s platným certifikátem a klíčem formátu pem.

Po navázání spojení fred-client přečte uvítací zprávu serveru a může se pokusit o přihlášení. Pokud se pokus nezdaří, smí ho opakovat. Po přihlášení může registrátor zadávat do konzole EPP příkazy. Tato situace je zachycena na obrázku 3.5. Jeho validace proběhne, pouze pokud byla zadána cesta k souboru s XML schématem EPP příkazů. A až po úspěšné validaci bude příkaz zaslán do centrálního registru. Pokud ale XML schéma pro validaci zadáno nebylo, bude příkaz zaslán bez validace.

Po odhlášení, vypršení časového limitu spojení a nebo po přijetí EPP zprávy s návratovým kódem, který odpovídá vzoru X5XX (druhá číslice je 5), dojde k odhlášení z Fred serveru.

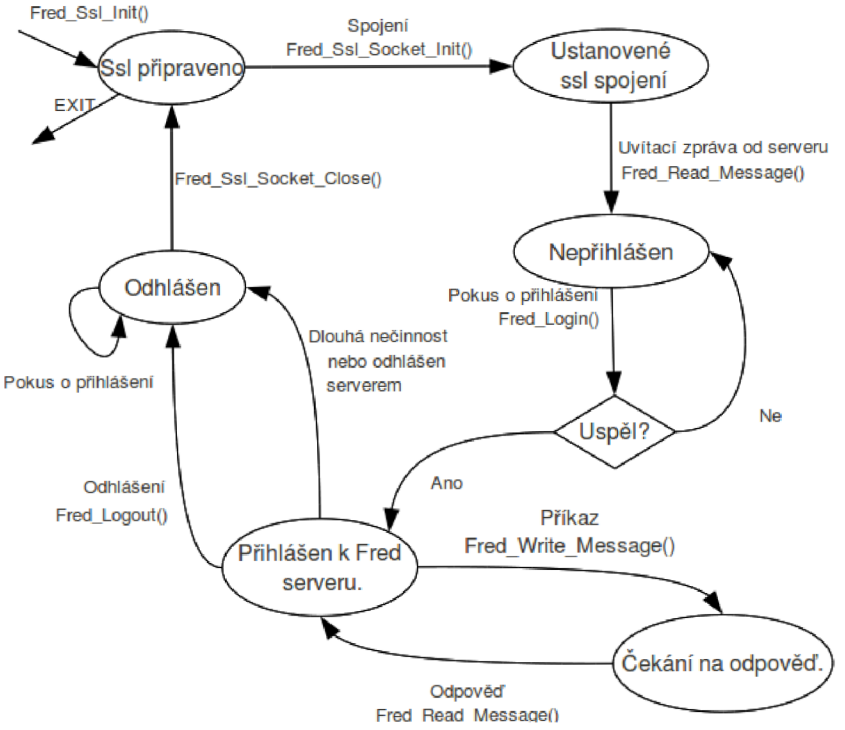
Obrázek 3.6 zobrazuje diagram stavů možné implementace síťového spojení protokolu EPP knihovnou libFredClient. Aplikace fred-client uzavírá socket ihned po provedení odhlášení uživatele od serveru.



Obrázek 3.5: Aplikace fred-client po prihlášení na server.

3.9 Shrnutí kapitoly

Tato kapitola popsala imlementaci klienta protokolu EPP v jazyku C. Jakyk C byl zvolen po domluvě s konzultantem z organizace NIC.CZ. Během implementace bylo zapotřebí vyřešít tvorbu XML dokumentů EPP příkazů, parsrování, odpovědi a přenášení XML dokumentů přes internet ve formátu pro EPP. Celá implementace je rozdělena do modulů jejíchž závislost je zobrazena v obrázku 3.3 a jsou spojeny do knihovny `libFredClient`. Klientské nástorje nad těmito moduly se jmenují `fred-creator`, `fred-sender` a `fred-client`. Jejich podrobný popis je se nalézá v sekci 3.8.



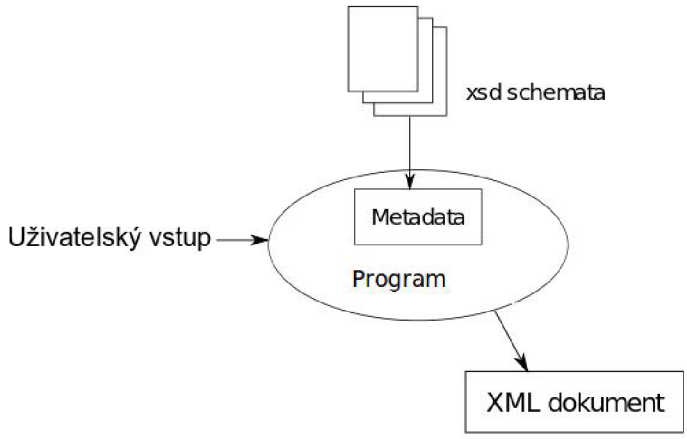
Obrázek 3.6: Diagram stavů síťové komunikace protokolu EPP z pohledu klienta.

Kapitola 4

Implementace v Javě

Implementace knihovny pro Fred v Javě řeší tvorbu XML zpráv protokolu EPP. Zprávy jsou generovány podle XML schémat a řetězce popisujícího EPP zprávu. Pro načítaní XML schémat a zpracování řetězce s uživatelským vstupem je použito nástroje ANTLR[1] (ANother Tool for Language Recognition). Jako reference pro ANTLR poslouží kniha [5]. Knihovna se snaží k uživatelskému vstupu nalézt struktury v načteném XML schématu. Tím si ověří správnost vstupů a vygeneruje XML soubor příkazu EPP. Pokud vstupů není kompatibilní s načteným schématem, nastane chyba. Běhovým prostředím pro Javu bylo *JavaSE-1.6 (java-6-sun-1.6.0.22)*

Obrázek 4.1 zachycuje princip knihovny.



Obrázek 4.1: Princip.

4.1 Načitaní xml schémat

XML schémata jsou načítána do stromové struktury objektů reprezentující chování jednotlivých klíčových slov jazyku XML schémat. K lexikální analýze schémat v ANTLR jsem použil XML lexikální analyzátor z [6], protože XML schémata jsou zapsána v XML dokumentu. Pro syntaktickou analýzu jsem rozšířil pravidla z obecného XML analyzátoru [7].

[1] více na webu `<http://www.antlr.org>`

Vstupem pro načtení schémat EPP protokolu je jeden soubor osahující všechny dílčí schémata. Prvním klíčové slovo "element"v načítaném souboru reprezentuje nejvyšší element XML dokumentu, který lze podle schémat vytvořit. V protokolu EPP to je element "epp"

4.1.1 Podporované klíčové slova xlm schémat

Pro načítání protokolu EPP jsem implementoval podporu pouze nezbytných klíčových slov XML schémat. Pokud schéma rozšíření EPP protokolu bude obsahovat nepodporované klíčové slovo, nedojde k chybě, protože syntaktická analýza navrhnuta pro obecný XML dokument, ale dojde k ignoraci elementu s klíčovým slovem. Tento fakt může způsobit problém ve fázi převodu uživatelského vstupu na XML.

- **Podporovaná klíčová slova:** `schema`, `import`, `element`, `simpleType`, `simpleContent`, `complexType`, `choice`, `sequence`, `maxOccurs`, `minOccurs`, `attribute`, `any`, `anyAttribute`
- **Nepodporovaná klíčová slova:** `unique`, `key`, `keyref`, `include`, `group`, `attributeGroup`, `complexContent`, `enumeration`, `restriction`, `all`, `list`, `union`

4.2 Gramatika pro převod příkazů na XML dokument

Je vytvořena v nástroji ANTLR. Vstupem do ní je načtené XML schéma a řetězec popisující XML soubor (dále také jmenován jako EPP řetězec), jedná se tedy o kontextový jazyk, kde kontext specifikuje XML schéma. Jeden EPP řetězec může vytvořit různé XML soubory podle různých schémat. Podle načteného schématu se rozhoduje, jaký význam bude mít uživatelský vstup, zda to bude element, hodnota elementu, atd.

4.2.1 Uživatelská reakce na element

Element je základním prvkem XML. Každý element má své jméno, které je uzavřeno ve znacích složených závorek. Příklad: `<jmeno_elementu>` V XML schématech je nový element popsán XML elementem se jménem `element`, jenž má povinný atribut `name`, kterým je určeno jeho jméno. Gramatika analyzující řetězec popisující EPP příkaz se musí vypořádat s různými typy elementů.

- **Prázdný element** nemá ve schématu zadaný typ. Je zapsán: `<element name= "hello"/>`. A uživatelský vstup pro tento element je očekáván shodný s jeho jménem, tedy `name`
- **Element s typem** je ve schématech zapsán jako `<element name="command" type= "epp:commandType"/>`. pro zpracovaní vstupu je důležité zda je typ `epp:commandType` jednoduchý (simply) nebo složený (complex).
- **Element se složeným typem** je rozebírán dále v dokumentu. Reakce na jednotlivé typy složeného typu jsou v podkapitolách 4.2.2 (typ `choice`) a 4.2.3 (typ `sequence`).
- **Element s jednoduchým typem** je zpracován jako text. Uživatelský vstup se nastaví jako text do těla elementu mezi jeho otevírací a uzavírací tag. Vstupem může

být třeba `"uživatelský vstup"`. Důležité je, že gramatika nekontroluje rozsah datového typu, takto zadaný vstup může vytvořit XML soubor, který nebude validní k načtenému XML schématu.

- **Opakující se element** jako `<element name="hellou"maxOccurs="5"/>`, kde hodnota maxOccurs určuje maximální počet opakování. Jednotlivé elementy jsou od sebe odděleny středníkem (;) a všechny výskytu elementu jsou uzavřeny v závorkách. Vstup: `(hello; hello; hello)` nebo `hello`.

- **Nekonečně se opakující element** je načítan dokud parser nenarazí na znak lomítka nebo dokud nedošlo k ukončení bloku sekvence nebo bloku volby. Ukončení nekoněčného elementu lomítkem vypadá následovně " `; / ;` ". Jednotlivě opakování jsou odděleny středníky (;). V XML schématu je zapsán jako `<element name="hello" maxOccurs="unbounded"/>`.

4.2.2 Uživatelská reakce na element choice ve schématech

Element `choice`, z XML schémat, se vyskytuje ve složeném typu (complexType). Jeho významem je zvolení jednoho elementu z uvedeného výběru. Přiklad elementu choice z XML schémat FREDu:

```
<choice>
  <element name="greeting" type="epp:greetingType"/>
  <element name="hello"/>
  <element name="command" type="epp:commandType"/>
  <element name="response" type="epp:responseType"/>
  <element name="extension" type="epp:extAnyType"/>
</choice>
```

Výběr se provede tak že se z uživatelského vstupu načte jméno jednoho z elementu. Při volbě mohou nastat tři situace.

- **Výběr elementu** se provede podle uživatelkého vstupu `command ( ... )` , kde obsah v závorkách reprezentuje typ vybraného elementu.

- **Výběr prázdného elementu** se provede napsáním jeho jména. Za elementem s typem nemusí následovat pár závorek a pak je sním zacházeno jako s elementem bez typu. XML se dále nezanořuje hlouběji.

- **Vynechání volby** se provede zápisem páru závorek `()`.

Gramatika pro zpracování vstupů zvládá v elementu "choice" zpracovat pouze "element", přestože XML schémata dovolují rekurzivní výskyt elementů "choice"a "sequence".

4.2.3 Uživatelská reakce na element sequence

Sequence je dalším elementem pro složený typ. Reprezentuje posloupnost elementů. Přiklad ze schémat CZ.NIC (je zkrácen):

```
<sequence>
  <element name="id" type="fredcom:objIDType"/>
  <element name="postalInfo" type="contact:postalInfoType"/>
```

```
  <element name="voice" type="contact:e164StringType" minOccurs="0"/>
  <element name="fax" type="contact:e164StringType" minOccurs="0"/>
  <element name="email" type="contact:emailType"/>
</sequence>
```

Uživatelský vstup k elementu sequence je uzavřen ve složených závorkách () a jeho jednotlivé elementy jsou odděleny středníkem. Příklad: `{ reakce_na_element_id ; postalInfo ; () ; () ; g@g.g }`. Pokud je místo reakce na element vstupem pouze dvojice znaků `()` je element vynechán. Tak jak se to stalo v případě elementu fax. Gramatika pro zpracování vstupů zvládá v elementu "sequence"zpracovat pouze "element"a element "choice", přestože XML schémata dovolují rekurzivní výskyt elementů "choice"a "sequence".

4.2.4 Reakce na kombinaci choice a sequence

Gramatika pro zpracování vstupů zvládá v sekvenci zpracovat elementy "choice", ale elementy "choice""nesmí být zanořen do jiného elementu choice. Zkrácený příklad složeného typu pro element "command"ze schémat CZ.NIC:

```
<sequence>
  <choice>
    <element name="check" type="epp:readWriteType"/>
    <element name="create" type="epp:readWriteType"/>
    <element name="delete" type="epp:readWriteType"/>
    <element name="info" type="epp:readWriteType"/>
    <element name="login" type="epp:loginType"/>
    <element name="logout"/>
  </choice>
  <element name="extension" type="epp:extAnyType" minOccurs="0"/>
  <element name="clTRID" type="epp:trIDStringType" minOccurs="0"/>
</sequence>
```

- **Volba s prázdným elementem.** Uživatelský vstup: `{ (logout) ; () ; id}`
- **Volba elementu s typem.** Uživatelský vstup: `{ (login ( ... ) ) ; () ; id}`

4.2.5 Uživatelská reakce na element any

Ze schématu se lze odkazovat na jine chéma přes element `any` a `namespace`. Kde hodnota atributu `namespace` je jménem jmenného prostoru konkrétního XML schéma. Příklad:

```
<sequence>
  <any namespace="##other"/>
</sequence>
```

Uživatelsky vstup je uzavřen ve složených závorkách (pro sekvenci) a vyžaduje zadání jména `namespace`, které je spjato se schématem. Dále je třeba určit element kterým má zpracování pokračovat. Příklad s namespace `contact` a pokračujícím elementem `create`, pokud typ elementu `create` byl choice nebo sequence:

- **sequence** `{ contact create { ... }}`
- **choice** `{ contact create ( ... )}`

4.2.6 Uživatelská reakce na element attribute

Atributy v XML schématech může obsahovat složený typu, ale jednoduchý ne.

```
<complexType name="discloseType">
  <sequence>
    <element name="voice" minOccurs="0"/>
    <element name="fax" minOccurs="0"/>
    <element name="email" minOccurs="0"/>
    <element name="vat" minOccurs="0"/>
    <element name="ident" minOccurs="0"/>
    <element name="notifyEmail" minOccurs="0"/>
  </sequence>
  <attribute name="flag" type="boolean" use="required"/>
</complexType>
```

Uživatelsky vstup reprezentující atribut je uzavřen do hranatých závorek: `[ flag = true ]`. Kde `flag` je jméno atributu a `true` je jeho hodnota. Možnosti připojení atributu k element:

- **k elementu** - `element [ flag = true ]`
- **k elementu v sekvenci, který je volbou** - `; [ flag = true ] ( ... ) ;`
- **k elementu v sekvenci, který je sekvenci** - `; [ flag = true ] { ... } ;`

4.2.7 Seznam pravidel pro tvorbu zpracovatelných XML schémat

Automatické vytváření XML dokumentů, které jsem vytvořil, nepodporuje celý rozsah XML schémat. Proto je při tvorbě nového schématu dodržovat několik pravidel.

- **Kořenový element** bude první element, který se vyskytuje ve schématu. Pro protokol EPP to je element `epp`.
- **Element choice** nesmí obsahovat jiné elementy než element `element`.
- **Element sequence** může obsahovat pouze elementy `element` a `choice`.
- **Element any** byl diskutován v odstavci 4.2.5. Musí být v elementu `sequence` jako jediný.

4.3 Ukázkový EPP klient s využitím automatické tvorby XML dokumentu

Ukázkový klient v Javě slouží pouze pro demonstraci práce s automatickou tvorbou EPP příkazů. Po konzultaci se zástupci CZ.NIC bylo rozhodnuto, že podoba řetězce popisujícího XML soubor (dále jen EPP řetězec), tak jak je popisována v kapitole 4.2 je pro uživatele komplikovaná, proto je v programu, kromě přímého zadávání EPP řetězce, implementována logická mezivrstva. Mezivrstva je umístěna mezi uživatele a EPP řetězec, stará se o správnou tvorbu EPP řetězce. Hlavní výhoda představovaného řešení by mněla spočívat v rychlejší úpravě fred klienta pro novou verzi schémat. EPP příkaz lze vytvořit zadaním jednoho

ze vstupních příkazů, které se zobrazí po zadání `help` v běžícím programu. Pro vytvoření šifrované spojení se servrem, klient načítá certifikáty ve formátu der. Klient načítá certifikát a klíč ve formátu der, protože implementace načítaní tohoto formátu je v Javě jednoduší než načítaní formátu pem. Formát pem používá implementace v C a původní klent systému fred. Převod certifikátu a soukromého klíče z formátu pem na formát der lze provést pomocí OpenSSL:

```
openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
openssl pkcs8 -topk8 -nocrypt -in key.pem -inform PEM -out key.der
-outform DER
```

Odpovědi od serveru jsou XML dokumenty, které klient parsruje metodou SAX (více o saxu v odstavci 3.3.5). Každý EPP příkaz má vlasní SAX parser.

4.4 Shrnutí kapitoly

Tato kapitola popsala imlementaci klienta protokolu EPP v jazyku Java. Dále byla diskutována možnost automatické tvorby XML dokumentu. Pro automatickou tvorbu XML dokumentu je zapotřebí XML schéma, podle kterého má byt vytvořen a jeho popis v podobě EPP řetězce. Ten je nevhodný pro uživatele, ale ve spojení s popisem příkazů EPP protokolu, který by uživatelský vstup převáděl na EPP řetězec, by urychlil úpravu klienta na novou verzi.

Kapitola 5

Ověření funkcionality knihoven

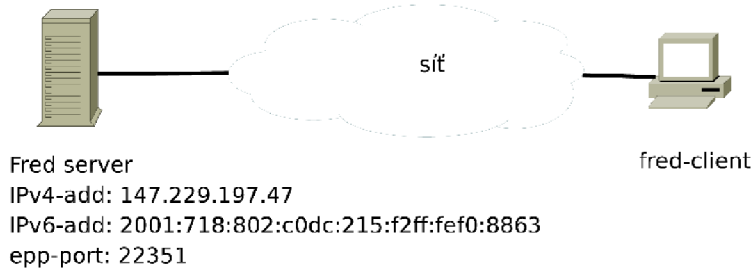
Nedílnou součástí každé implementace je její testování. V případě klientu FRED to bude ověření vytvořených příkazů ke schématům a síťového modulu s šifrovaným přenosem v síti adresované IPv4/IPv6. Na konci kapitoly je popsán test reálným provozem.

5.1 Tvorba EPP příkazů ve správném formátu

Ověření vytvořeného příkazu EPP validací proti schématu XML jsem vybral, protože je začleněna do XML knihoven jako ověřovací mechanizmus pro standardní kontrolu XML dokumentu. Výhodou validace je automatizování kontroly obsahu elementu. Aby program `fred-client` napsán v jazyce C validoval vytvářené dokumenty, musí být spuštěn s volbou `schema` s názvem souboru schématu XML.

5.2 Ověření síťového modulu

Ověření chování tohoto modulu spočívá v testování funkce, která vytváří šifrované spojení přes internet. Testování proběhlo v síti třídy B(147.229.0.0). Obrázek 5.1 zobrazuje topologii, do které byly zapojeny testovací stroje, na kterých běží Fred server a mnou vytvořené klientské nástroje. Předpokladem pro provedení testů je správné nastavení vstupů pro klienta systému Fred, především jmen souborů s SSL klíčem a certifikátem a uživatelské jméno s heslem. Prvním předpokladem je ovšem funkční spojení mezi systémem serveru a klientu.



Obrázek 5.1: Zobrazení testovací sestavy.

#	host	socket	wireshark pozorování	výstup aplikace fred-client
1.	147.229.197.47	NULL	Komunikace na portu 22351	Conected to server.
2.	2001:718:802:c0dc:215:f2ff:fef0:8863	NULL	Komunikace na portu 22351	Conected to server.
3.	b07-602b.kn.vutbr.cz	IPv4	Komunikace na portu 22351	Conected to server.
4.	147.229.197.47	IPv6	—	error: 103 : Cannot create socket, unable connect to server.

Tabulka 5.1: Zobrazuje průběh testů síťového modulu. Reakce programu fred-client na určité vstupy.

5.2.1 Testování spojení

Pro ověření spojení jsem použil nástroj Wireshark. Předpokladem provedení testů je správné nastavení vstupů pro program klienty systému fred, především jmen souborů s ssl klíčem a certifikátem, uživatelské jméno a heslo. Bezproblémové připojení Fred serveru do sítě a stejně tak i PC s programem klientem systému Fred. Tabulka 5.1 zachycuje různá nastavení položek host a socket v konfiguračním souboru. Položka host v konfiguračním souboru byla naplněna DNS jménem, IPv4 adresou a IPv6 adresou.

5.3 Správná funkce klientů

Správnou funkci klientů lze nejlépe otestovat jejich reálným použitím v komunikaci se serverem. Toto testování ale závisí na uspěšně vytvořeném spojení mezi klientem a sevrem. Server zasílá v odpovědi návratovou hodnotu a textovou zprávu informující o vykonání operace v CR. Jak je vidět na obrázku 5.2 server odpověděl hodnotou 1000 a textem "Command completed successfully", který znamená že příkaz dopadl správně. Tato kontrola za běhu byla provedena na všech příkazech, ale ne na všech možných kombinacích vstupů.

5.4 Shrnutí kapitoly

Registrátor může registrovat záznamy v CR jen pokud jeho klient správně funguje. Tato kapitola popisovala, jakým způsobem bylo ověřováno tvoření EPP příkazů ve správném formátu a postup testování vytvoření spojení mezi klientem a serverem. Každý příkaz byl samostatně testován za běhu aplikace. Bohužel nelze z časových důvodu otestovat naprosto vše.

```
Connecting to server: 192.168.56.101
                port: 22351
Logging as: REG-FRED_A

REG-FRED_A@192.168.56.101> list_contacts
Do you really want to send this command to the server? (y/N):y

result   code: 1000
resutl reason: Command completed successfully


CID:ID01
CID:ID02
CID:ID03
CID:ID04
CID:ID09
III

REG-FRED_A@192.168.56.101>
```

Obrázek 5.2: Výstup testovacího programu napsaného v Javě.

Kapitola 6

Závěr

Předmětem bakalářské práce bylo vytvoření knihovny pro protokol EPP v systému Fred. Práce knihovny měla být prezentována ukázkovými aplikacemi.

V této práci je popsán současný stav registrování DNS objektu v systému Fred. Prostudoval jsem protokol EPP, kterým se v systému Fred komunikuje mezi klientem a serverem. Z těchto znalostí jsem v jazyce C vytvořil knihovnu libFredClient, která slouží pro tvorbu příkazů protokolu EPP a komunikaci mezi klientem a servrem. Modul pro síťovou komunikaci jsem vytvořil podle příkladu [1]. Nad knihovnou libFredClient jsem vytvořil programy prezentující její funkčnost. Testy prokázaly, že implementace knihovny odpovídá zadání, tedy vytvoření knihovny v jazyce C pro klienta protokolu EPP aplikace Fred, se zachováním formátu uživatelského vstupu nástrojů již používaných pro tento účel.

Dále jsem v Javě a nástroji ANTLR vytvořil knihovnu pro tvorbu XML dokumentu z řetězce popisujicího EPP příkaz. Použití knihovny jsem demonstroval na jednoduchém klientu pro registrátory v systému Fred. Knihovna plně nepodporuje specifikaci XML schémat, což omezuje její použití. Ale největším omezením knihovny je uživatelské rozhraní, které je nepříjemné pro přímé zadávání řetězce pro popis EPP do knihovny samotnými uživateli (registrátory). Pro snadnější uživatelské rozhraní by bylo vhodné vytvořit postup převodu sady konkrétních uživatelských vstupů na vstupy do knihovny pro automatickou tvorbu XML dokumentů.

Literatura

[1] BALLARD, K.: *Secure programming with the OpenSSL API, Part 1: Overview of the API.* [cit. 2011-05-16], [online].
Dostupné z WWW: <http://www.ibm.com/developerworks/linux/library/l-openssl.html>

[2] HOLLENBECK, S.: *Extensible Provisioning Protocol (EPP).* 2009 [cit. 2011-05-16], [online].
Dostupné z WWW: <http://tools.ietf.org/html/rfc5730>

[3] HOLLENBECK, S.: *Extensible Provisioning Protocol (EPP) Transport over TCP.* 2009 [cit. 2011-05-16], [online].
Dostupné z WWW: <http://tools.ietf.org/html/rfc5734>

[4] HRUŠKA, Tomáš; KROULÍK, Jan; TEHET, Jiří: Internetové aplikace (WAP) III: část XML, XML schémata, XPath, XSLT Studijní opora. *Language*, 2007 [cit. 2011-05-16]: s. 1–122.

[5] PARR, T.: *The definitive ANTLR reference: building domain-specific languages [online].* Pragmatic Bookshelf, 2007 [cit. 2011-05-16], ISBN 0978739256, 384 s.
Dostupné z WWW: <http://www.pragprog.com/titles/tpantlr/the-definitive-antlr-reference>

[6] ZEIGERMANN, O.: *Lexing XML with ANTLR 3.* 2007 [cit. 2011-05-16], [online].
Dostupné z WWW: <http://www.antlr.org/wiki/display/ANTLR3/1.+Lexer>

[7] ZEIGERMANN, O.: *Parsing XML with ANTLR3.* [cit. 2011-05-16], [online].
Dostupné z WWW: <http://www.antlr.org/wiki/display/ANTLR3/2.+Parser>

Příloha A

Instalace knihoven pro implementaci v jazyce C

Knihovny s kterými pracuje knihovna libFredClient. Knihovny jsou instalovány ve vývojových verzích. Popis instalace na operační systém Ububtu 8.10 – 32bit , Linux 2.6.27-15-generic.

Sada nástrojů Fred.
Instalace popisována na http. http://fred.nic.cz/wiki/download

OpenSSL.

```
$ wget http://www.openssl.org/source/openssl-0.9.8m.tar.gz
$ tar -xf openssl-0.9.8m.tar.gz
$ cd openssl-0.9.8m
$ ./configure
$ make
$ make test
$ make install
```

Více informací k instalaci naleznete v souboru INSALL ve složce openssl-0.9.8m.

Knihovna libxml2.
Více na webu `<http://xmlsoft.org/FAQ.html#Installati>`

libedit.

```
$ wget http://www.thrysoee.dk/editline/libedit-20100424-3.0.tar.gz
$ tar -xf ibedit-20100424-3.0
$ cd libedit-20100424-3.0
$ ./configure
$ make
$ make instal
```

Více informaci k instalaci naleznete v souboru INSALL ve složce libedit-20100424-3.0.

Příloha B

Ukázka vstupu pro automatickou tvorbu XML

EPP příkaz logout

```
command ( { ( logout ) ; () ; id } )
```

EPP příkaz check domain

```
command ( { ( check ({ domain check { prvnidomena.cz } }) ) ; () ; id })

command ( { ( check ({ domain check { prvnidomena.cz ; druhadomena.cz ;
tretidomena.cz} }) ) ; () ; id })
```

EPP příkaz listContacts je v systému Fred rozšířujícím příkazem protokolu EPP.

```
extension ( { fred extcommand {( listContacts ) ; id } } )
```

EPP příkaz domains by nsset

```
extension ( { fred extcommand {( domainsByNsset ({ nssetId }) ) ; idgetID
} } )
```

EPP příkaz create contact vytvoří v databázi na serveru objekt kontatk. Příklad zobrazuje použití argumetu pro XML element (znaky []).

```
command ( {( create ({ contact create { contactID; { name; organization
; { street1 ; city ; sp ; pc ; cc } }; voice ; fax; email ; password ; ()
; () ; [type = op] 88888 ; notifyEmail } }) ) ; () ; id })
```

EPP přikaz create domain vytvoří v databázi na serveru objekt domain. Příklad zobrazuje ukončení nekonečného elementu znakem lomítka (/)

```
command ( { ( create ({ domain create {name ; [ unit=y ] peridNumber; nssetId
; keysetId; registrant; admin1 ; admin2 ; / ; mypassword } } ); {enumval
create { date ; true}} ; id }) )
```