VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

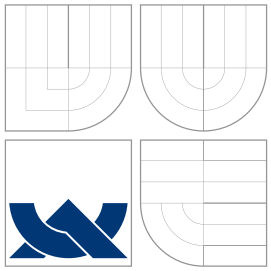
WEBOVÝ PROXY SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

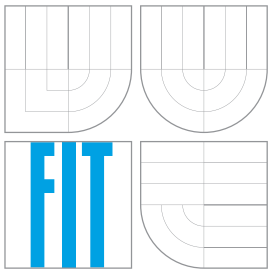
AUTOR PRÁCE
AUTHOR

JIŘÍ KUNČAR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ PROXY SERVER

WEB PROXY SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ KUNČAR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. FRANTIŠEK ŠČUGLÍK, Ph.D.

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2009/2010

Zadání bakalářské práce

Řešitel: **Kunčar Jiří**
Obor: Informační technologie
Téma: **Webový proxy server**
Web Proxy Server
Kategorie: Web

Pokyny:

1. Seznamte se s problematikou dnešního webu, tvorbou webových stránek a principem proxy serverů.
2. Seznamte se se způsoby optimalizace webu pro zobrazování v zařízeních s omezenou kapacitou připojení.
3. Navrhněte systém (proxy server), který bude optimalizovat webové stránky pro zobrazení v různých zařízeních. Zaměřte se na možnosti změny velikosti a formátu obrázků, filtrování obsahu (zkoumejte možnost importu pravidel z rozšíření Ad-block pro Mozilla Firefox), komprimaci textu. Navrhněte vhodné GUI pro snadné nastavení serveru.
4. Implementujte a otestujte navržený systém.
5. Navrhněte možná rozšíření (např. přizpůsobení nastavení serveru pro různé uživatele) a některá z nich implementujte.
6. Otestujte funkčnost systému v různých přohlížečích.
7. Diskutujte dosažené výsledky a možný další postup.

Literatura:

- Strebe, M., Perkins, C.: Firewally a proxy-servery: Praktický průvodce. Computer Press (2003), ISBN 80-722-6983-6
- Lee, J.: Open source: vývoj webových aplikací Linux, Apache, MySQL, Perl a PHP. Mobil Media (2003), ISBN 80-86593-43-6

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

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: **Ščuglík František, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2009
Datum odevzdání: 19. května 2010

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

Práce čtenáře seznámí s internetem a jeho historií, tvorbou webových stránek, principem proxy serverů a se souvisejícími protokoly TPC/IP a HTTP. Hlavním cílem této práce je návrh a následná implementace systému - webového proxy serveru, zaměřeného na optimalizaci webových stránek z hlediska redukce jejich velikosti, s možností individuálního nastavení parametrů optimalizace.

Abstract

The Bachelor's thesis makes reader acquainted with the Internet and its history, creation of Web pages, the principle of proxy servers and related TCP/IP and HTTP protocols. The main goal is a design and follow-up implementation of the system - the Web proxy server that focuses on optimalization of Web pages as regards their size reduction with the posibility of setting individual optimalization parameters.

Klíčová slova

Internet, WWW, webové stránky, TCP/IP, HTTP, proxy server, HTML, optimalizace, grafické formáty, komprimace, gzip, base64, Adblock Plus, ImageMagick, autentizace

Keywords

Internet, WWW, Web pages, TCP/IP, HTTP, proxy server, HTML, optimization, graphic formats, compression, gzip, base64, Adblock Plus, ImageMagick, authentication

Citace

Jiří Kunčar: Webový proxy server, bakalářská práce, Brno, FIT VUT v Brně, 2010

Webový proxy server

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Ščuglíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Kunčar
11. května 2010

Poděkování

Rád bych poděkoval panu Ing. Františku Ščuglíkovi, Ph.D. za jeho ochotu a trpělivost při vedení mé bakalářské práce.

Obsah

Kapitola 1

Úvod

V době širokopásmových připojení zapomínají někteří tvůrci webových stránek na uživatele využívající zařízení s omezenou kapacitou připojení. Současným obsahem webových stránek už není pouze text a několik obrázků, s příchodem nových technologií se rozšířil i obsah webu, který nyní tvoří nejrůznější multimediální elementy, od videa až po interaktivní aplikace. Velký rozmach internetu v posledních letech měl mimo jiné za následek jeho obrovskou medializaci, na webových stránkách se tak objevuje stále více druhů reklamních formátů, které zpravidla nešetří velikostí dat. Na druhou stranu je medializace současně důvodem zvýšení počtu informačně hodnotných stránek na internetu, jelikož se objevila reálná možnost monetizace obsahu.

Vedlejším efektem je zvyšování objemu přenášených dat, kde právě uživatelé bez širokopásmového připojení mají stále méně možností, jak tato data přijímat v alternativní optimalizované podobě. Řešením tohoto problému může být vužití *webového proxy serveru*, který funguje jako prostředník mezi uživatelem a cílovým serverem. Většina internetových prohlížečů podporuje přeposílání požadavků právě přes proxy server, čehož lze využít pro optimalizaci obsahu webových stránek.

Cílem této práce je návrh a následná implementace webového proxy serveru, který bude zaměřen na optimalizaci webových stránek, zejména pro zobrazení v zařízeních s omezenou kapacitou připojení. Redukce objemu dat bude mít za následek určitou ztrátu informačního charakteru, systém by měl být proto navržen tak, aby byl potenciální uživatel schopen ovlivnit míru této ztráty. Dalším cílem je tedy možnost individuálního nastavení parametrů optimalizace.

Obsahem práce je teoretické seznámení s internetem a jeho historií, webovými stránkami a principem proxy serverů, včetně souvisejících protokolů a algoritmů. K teoretickému přehledu patří rovněž informace o webových grafických formátech a programech využitých v tomto projektu. Práce se dále zabývá chronologickým návrhem systému, včetně možností optimalizace internetových stránek z hlediska redukce jejich velikosti a je popsána implementace nejdůležitějších částí systému. Pro vyhodnocení úspěšnosti projektu je důležitá kapitola zaměřená na testování systému, kde jsou diskutovány statistické výsledky optimalizace. V závěru je shrnut současný stav práce a jsou uvedeny možnosti dalšího rozvoje projektu.

Kapitola 2

Teoretický přehled

2.1 Internet

Internet je celosvětový systém navzájem propojených počítačových sítí (tzv. „síť sítí“), ve kterých mezi sebou počítače komunikují pomocí rodiny protokolů *TCP/IP*. Společným cílem všech lidí využívajících internet je komunikace, resp. výměna dat. Nejznámější službou poskytovanou v rámci internetu je *WWW* a e-mail, avšak nalezneme v něm i desítky dalších.

2.2 Historie internetu

Internet, který známe dnes, se zrodil již počátkem 90. let. Jeho historie je však mnohem složitější. První předchůdce dnešního internetu byl vytvořen v roce 1969 institucí Advanced Research Project Agency (ARPA) pod záštitou Ministerstva obrany USA. Síť byla nazvána ARPANET a zpočátku byla tvořena pouhými čtyřmi počítači. V roce 1972 k ní bylo připojeno 50 výzkumných a vojenských center. Později se rozdělila na dvě sítě: Arpanet a Milnet (armádní síť) a v roce 1981 přibyla síť Bitnet (využita k propojení amerických vysokých a středních škol).

Velkým problémem však byla komunikace na mnoha různých platformách, proto probíhal v této oblasti intenzivní výzkum a jeho výsledkem (v r. 1983) byl protokol *TCP/IP (Transmission Control Protocol / Internet Protocol)*, který je používán dodnes. I když v polovině 80. let existovalo několik sítí, stále ještě nebyly předmětem zájmu veřejnosti, protože nebyly volně přístupné.

Vznik dnešního internetu je datován do roku 1989, kdy Švýcar Tim Berners Lee vymyslel pro atomové fyziky ve švýcarském Bernu nový způsob výměny informací. Tento nový způsob, známý pod zkratkou *WWW (World Wide Web)*, znamenal start dnešního internetu. Krátce nato napsal Tim Berners Lee první webový prohlížeč.

2.3 Webové stránky, HTML

Webová stránka zobrazuje za použití webového prohlížeče informace poskytované v rámci *World Wide Webu*. Informace jsou prezentovány v podobě hypertextu, který je vytvořen např. použitím značkovacích jazyků *HTML* nebo *XHTML*. Samotné webové stránky se skládají z textu, obrázků, multimediálních dat a odkazů, které umožňují přechod na další webové stránky. Díky těmto propojením pomocí odkazů indexují fulltextové vyhledávače svůj obsah.

HTML (*HyperText Markup Language*), je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web. Předchůdcem HTML je rozsáhlý univerzální značkovací jazyk SGML (*Standard Generalized Markup Language*). Vývoj HTML byl samozřejmě ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. Vývoj jazyka HTML byl původně ukončen verzí 4.01. Dle W3C měl další vývoj psaní webových dokumentů patřit jazyku XHTML - následníkovi HTML, využívajícímu univerzální jazyk XML. 7. března 2007 W3C založilo novou pracovní skupinu HTML, jejímž cílem bylo uvolnit specifikaci nové verze HTML. Tato verze nese označení HTML5. „Specifikace by měla být hotova v letech 2010-2012, ukončení vývoje specifikace po vyřešení problémů a opravení všech chyb se odhaduje až na rok 2022.“ [2]

2.4 Grafické formáty obrázků na webových stránkách

Grafické formáty obrázků definují pravidla, podle kterých jsou obrázky uloženy v souborech. Formátů existuje celá řada, na webových stránkách se však nejčastěji objevují tři grafické formáty, konkrétně *JPEG*, *GIF* a *PNG*. Přehled těchto formátů je uveden v tabulce 2.1. V krajních případech lze využít i formát *BMP*, ten však zpravidla nevyužívá žádné komprese a je tak kvůli své nadbytečné velikosti pro použití na internetu zcela nevhodný, resp. je užitečné převádět formát BMP např. do formátu JPEG.

2.4.1 Formát JPEG/JFIF

Formát JPEG/JFIF (*Joint Photographics Experts Group/JPEG File Interchange Format*) je určen pro přenášení a ukládání fotografií v realistické podobě. Podporuje 24 bitovou hloubku barev. Formát není vhodný pro text nebo ikony, jelikož kompresní metoda JPEG vytváří v těchto případech rušivé elementy. Metoda komprese formátu JPEG je ztrátová, což znamená, že po provedení komprese už nelze původní data zrekonstruovat. Výhodou je velká míra komprese, avšak je zde absence průhlednosti a animace obrázků.

2.4.2 Formát GIF

Formát GIF (*Graphics Interchange Format*) byl vyvinut společností CompuServe pro usnadnění přenosu obrázků pomocí online služeb. Formát GIF se dále rozděluje na dvě varianty, původní verze GIF87a a novější varianta GIF89a, která definuje průhlednost, umožňuje jednoduché animace a uložení dalších metadat.

GIF používá bezeztrátovou kompresi LZW, která zmenšuje velikost souboru beze ztráty důležitých grafických dat. Nevýhodou je omezení na ukládání pouze 8bitových obrázků, což je maximálně 256 barev.

2.4.3 Formát PNG

PNG (*Portable Network Graphics*) je relativně nový grafický formát primárně vytvořený pro zobrazení na webových stránkách. Od formátu JPEG se liší tím, že nepoužívá ztrátovou kompresi a podporuje průhlednost, od formátu GIF naopak tím, že není omezen na 256 barev a nepodporuje animace. Hlavní výhodou tohoto formátu je lepší kvalita obrazu, nevýhodou, že jsou soubory objemově větší.

	JPEG	GIF	PNG
Maximální barevná hloubka	24 bitů	8 bitů	48 bitů
Podpora prokládaných obrázků	ano	ano	ano
Podpora průhlednosti	ne	ano	ano
Animace	ne	ano	ne
Komprese	ztrátová	bezztrátová	bezztrátová

Tabulka 2.1: Přehled grafických formátů na webových stránkách

2.4.4 Formát BMP

BMP (*Microsoft Windows Bitmap*) je grafický formát pro rastrovou grafiku vyvinutý firmou Microsoft. Soubory ve formátu BMP nejčastěji nepoužívají žádnou kompresi (existují ale i varianty s kompresí RLE), což je hlavní důvod, proč jsou obvykle BMP soubory mnohem větší než obrázky stejného rozměru, uložené ve formátech využívajících kompresi. Formát BMP proto není vhodný pro použití na internetu.

2.5 Proxy server a jeho funkčnost

Původně proxy servery poskytovaly počítačům s běžným připojením k internetu služby ukládání často navštěvovaných internetových stránek do vyrovnávací paměti. V začátcích internetu byla propojení v síti WAN velmi pomalá, internet byl poměrně malý a stránky na něm byly statické. Celou síť tvořilo jen několik tisíc stránek pro vědecké pracovníky a akademiky. Kdykoliv se na nějaké stránce objevila nějaká nová zpráva, navštívil uvedenou stránku v rámci jedné organizace velký počet vědců. Uložením stránky do vyrovnávací paměti na lokálním serveru minimalizovaly proxy servery zbytečné připojování k internetu a opakované stahování stejné stránky. (zdroj [6])

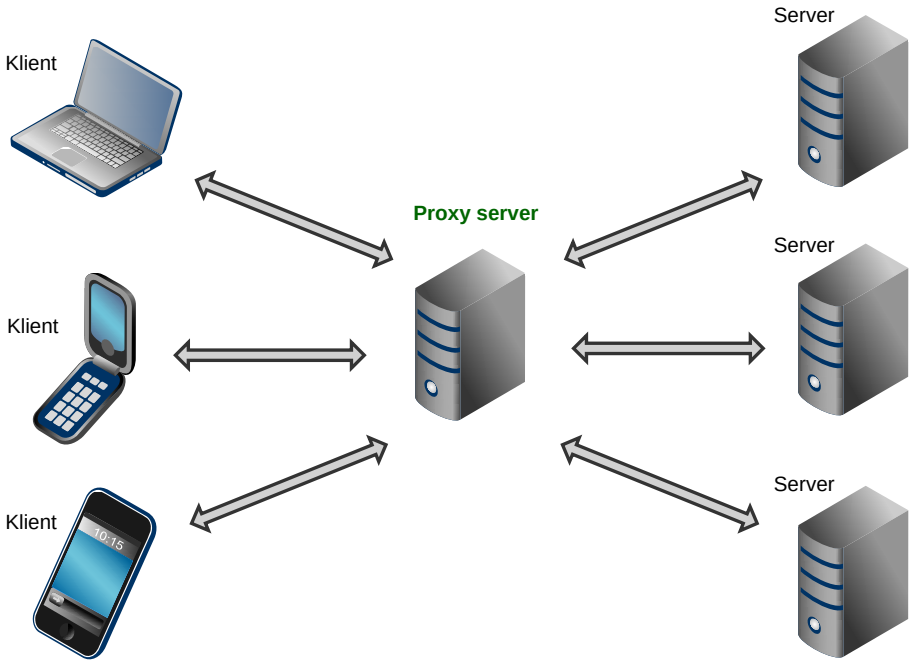
V dnešní době funkce ukládání do vyrovnávací paměti, kterou proxy zajišťovaly, výrazně ustoupila. Internet je rozsáhlý, stránky jsou často dynamické a uživatelé v jedné organizaci si mohou mezi návštěvou jedné a té samé stránky prohlížet miliony dalších stránek. Všechny tyto faktory byly samozřejmě pro ukládání do vyrovnávací paměti negativní. U proxy serverů se ale objevil neočekávaný a užitečný postranní účinek: umí skrýt všechny uživatelé sítě za jediné zařízení, umí filtrovat URL adresy nebo optimalizovat obsah.

Proxy server funguje jako prostředník mezi klientem a cílovým serverem. Jedná se zpravidla o program pracující na aplikační úrovni. Proxy server očekává klientské požadavky, které následně předává na cílový server, jako kdyby byl sám klient. Jakmile obdrží proxy server odpověď od vzdáleného serveru, odešle ji klientovi např. v upravené podobě. Na obrázku 2.1 je tento proces znázorněn.

2.6 Síťové protokoly, protokol TCP/IP

Počítače v počítačových sítích používají pro vzájemnou komunikaci síťové protokoly. Síťových protokolů existuje celá řada, v internetu se používají síťové protokoly TCP/IP.

Síťový protokol je norma. V internetu se používají normy nazývané *Request For Comments - RFC*, které se číslují průběžně od 1. V současné době jich jsou necelé tři tisíce, avšak postupem času zastaraly, takže z první tisícovky jich je aktuálních jen několik. Vzhledem ke složitosti problémů je síťová komunikace rozdělena do tzv. vrstev, které znázorňují



Obrázek 2.1: Funkčnost proxy serveru

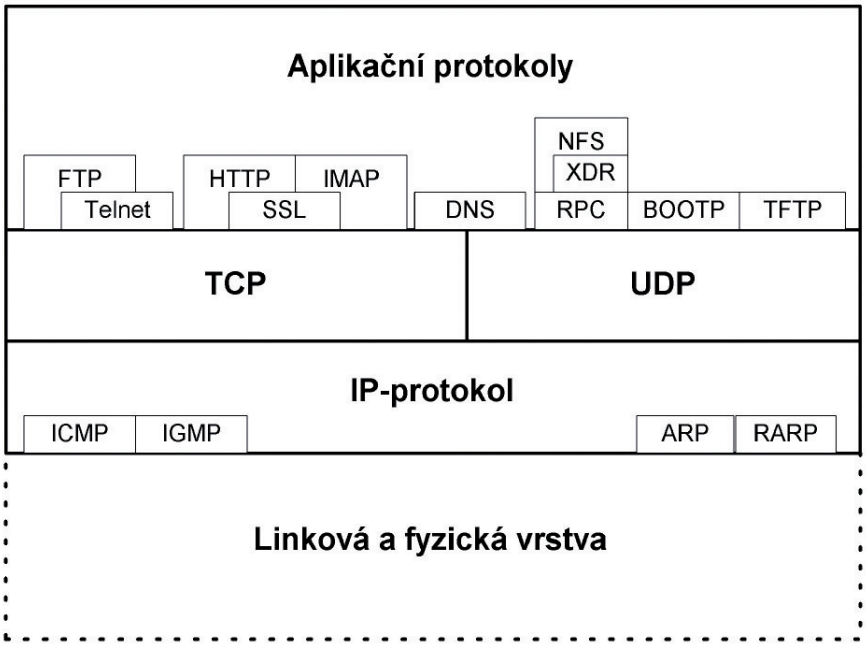
hierarchii činností. Výměna informací mezi vrstvami je přesně definována, každá vrstva využívá služeb vrstvy nižší a poskytuje své služby vrstvě vyšší.

Rodina protokolů TCP/IP (*Transmission Control Protocol/Internet Protocol*) se nezabývá (až na výjimky) fyzickou a linkovou vrstvou. V praxi se i v internetu používají pro fyzickou a linkovou vrstvu často protokoly vyhovující normám ISO OSI, které standardizoval ITU (organizace vydávající normy v oblasti komunikací). Některé protokoly z rodiny protokolů TCP/IP jsou uvedeny na obrázku 2.2. Internet Protokol (*IP*) prakticky odpovídá síťové vrstvě. IP přenáší tzv. IP datagramy mezi vzdálenými počítači. Každý IP-datagram ve svém záhlaví nese adresu příjemce, což je úplná směrovací informace pro dopravu IP-datagramu k adresátovi. Takže síť může přenášet každý IP datagram samostatně. IP-datagramy tak mohou k adresátovi dorazit v jiném pořadí, než byly odeslány. Každé síťové rozhraní v rozsáhlé síti internet má svou celosvětově jednoznačnou IP-adresu.

Protokoly *TCP* a *UDP* odpovídají transportní vrstvě. Protokol TCP dopravuje data pomocí TCP segmentů, které jsou adresovány jednotlivým aplikacím. Protokol UDP dopravuje data pomocí tzv. UDP datagramů. Protokoly TCP a UDP zajišťují spojení mezi aplikacemi běžícími na vzdálených počítačích. Rozdíl mezi protokoly TCP a UDP spočívá v tom, že protokol TCP je tzv. spojovanou službou, tj. příjemce potvrzuje přijímaná data. V případě ztráty dat (ztráty TCP segmentu) si příjemce vyžádá zopakování přenosu. Protokol UDP přenáší data pomocí datagramů, tj. odesílatel odešle datagram a už se nezajímá o to, jestli byl doručen.

2.7 Protokol HTTP

HTTP (*Hypertext Transfer Protocol*) je protokol aplikační vrstvy, určený původně pro výměnu hypertextových dokumentů ve formátu HTML. HTTP protokol verze 1.0 je definován v *RFC 1945* a verze 1.1 protokolu je definována v *RFC 2616* [1]. Tento protokol je spolu



Obrázek 2.2: Některé protokoly z rodiny protokolů TCP/IP (zdroj [3])

s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech.

2.7.1 Princip protokolu

Protokol funguje způsobem dotaz-odpověď. Klient pošle serveru dotaz ve formě zprávy, obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server poté odpoví pomocí hlavičky (*header*), popisující výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.), za kterou následují data samotného požadovaného dokumentu.

HTTP je *bezestavový protokol*, tzn. neumí uchovávat stav komunikace, dotazy spolu nemají souvislost. Tato vlastnost samozřejmě není vhodná pro implementaci složitějších informačních systémů přes HTTP (např. internetový obchod potřebuje uchovávat informaci o identitě klienta, o obsahu jeho „nákupního košíku“ apod.). K těmto účelům byl protokol rozšířen o tzv. *HTTP cookies*, které umožňují serveru uchovávat data uložená u klienta, což tuto nepříjemnou vlastnost elegantně eliminuje. Cookies jsou po té součástí každé hlavičky zasílané na daný server.

Jako ukázka komunikace poslouží HTTP dotaz na server cs.wikipedia.org a jeho následující odpověď. (zdroj [2])

Dotaz klienta:

```
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
User-Agent: Mozilla/5.0 Gecko/20040803 Firefox/0.9.3
Accept-Charset: UTF-8,*
```

Odpověď serveru:

```
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8
```

Za touto hlavičkou následuje jeden prázdný řádek (označující její konec) a pak požadovaný HTML dokument. Hlavička obsahuje informaci o stavu odpovědi (první řádek: „200 OK“), datum a čas vyřízení dotazu, popis serveru, který odpovídá, informace o typu vráceného dokumentu (textový soubor s obsahem HTML v kódování UTF-8) a další informace.

2.7.2 Dotazovací metody

HTTP definuje několik dotazovacích metod, které se mají provést nad uvedeným objektem (dokumentem), konkrétně jsou uvedeny v tabulce 2.2.

Metoda	Popis
GET	Nejčastěji používaný HTTP požadavek, který slouží k získání dokumentu.
HEAD	Podobné metodě GET, ale už nepředává data. Poskytne pouze metadata o požadovaném cíli.
POST	Odesílá uživatelská data na server. Používá se např. při odesílání formuláře na webu.
PUT	Nahraje data na server. Objekt je jméno vytvářeného souboru. Používá se zřídka.
DELETE	Smaže objekt ze serveru. Jsou zapotřebí jistá oprávnění, stejně jako u PUT.
TRACE	Odešle kopii obdrženého požadavku zpět odesílateli, takže klient může zjistit, co na požadavku mění nebo přidávají servery, kterými požadavek prochází.
OPTIONS	Dotaz na server, jaké podporuje metody.
CONNECT	Používá se při průchodu skrze proxy pro ustanovení kanálu SSL.

Tabulka 2.2: HTTP dotazovací metody

2.7.3 Stavové kódy odpovědí

Formát HTTP odpovědi je rozdělen na dvě části, konkrétně hlavička a obsah. První řádek odpovědi se nazývá status line a skládá se ze tří částí. První část slouží k identifikaci verze HTTP protokolu, druhá část obsahuje status code - číslo značící úspěch/neúspěch požadované operace - a v třetí části je slovní reprezentace stavového kódu. Stavový kód je trojciferné číslo, kde nejvyšší řád rozděluje stavové kódy do 5 skupin, viz tabulka 2.3.

Stavový kód	Popis
1xx	kódy informačního charakteru
2xx	kódy označující úspěšné vykonání požadavku
3xx	od klienta je požadovaná další akce, např. přesměrování
4xx	během zpracování požadavku došlo k nějaké chybě
5xx	došlo k chybě na straně serveru

Tabulka 2.3: HTTP stavové kódy odpovědí

2.7.4 Konec HTTP zprávy

HTTP protokol definuje několik možností určení konce požadavku, resp. odpovědi. Při aplikaci tohoto protokolu je nutné implementovat všechny tyto možnosti.

1. Hlavička `Content-Length`, která určuje přesnou délku těla zprávy. Hlavička na příkladu říká, že má tělo odpovědi přesně 6754 bajtů:

   ```
   Content-Length: 6754
   ```

2. Tělo zprávy je prázdné, ukončení požadavku tedy určuje konec HTTP hlavičky, např. požadavek GET:

   ```
   GET /test.html HTTP/1.1
   Host: www.example.com
   ```

3. Hlavička `Transfer-Encoding` s parametrem „chunked“ je nejčastěji aplikována v případech, kdy není předem známá velikost odesílané odpovědi. Tělo zprávy je rozděleno do oddělených bloků („chunků“). Délka těchto bloků v hexadecimálním tvaru je pak definována na prvním řádku každého z chunků. Poslední blok obsahuje prázdné tělo, je proto definován nulovou délkou:

   ```
   HTTP/1.1 200 OK
   Content-Type: text/plain
   Transfer-Encoding: chunked

   25
   Ukazka prvniho chunku s koncem radku

   1F
   a toto je ukazka druheho chunku
   0
   ```

4. Uzavření komunikace ze strany odesílatele po dokončení požadavku (pomalé pro klienta)

2.8 Algoritmus Base64

Algoritmus Base64 je datový formát, který umožňuje zakódovat vstupní řetězec (např. binární data) do řetězce složeného z tisknutelných ASCII znaků. K jeho hlavním výhodám

patří binární převod vstupního řetězce. Při implementaci se tak nemusí řešit např. znakové kódování, Base64 je tedy binárně bezpečný algoritmus. Nevýhodou tohoto algoritmu je navýšení velikosti kódované zprávy přibližně o 33% a absence kontrolního mechanismu. Tento datový formát je např. použit u Basic šifrování při autentizaci s použitím HTTP protokolu, nejčastěji se však pomocí něj kódují binární data, např. přílohy emailů, multimédia, apod.

Algoritmus převede každé 3 bajty vstupního řetězce nejprve na číselný binární zápis podle ASCII tabulky (např. znak „M“ je ve dvojkové soustavě „01001101“). Vznikne tak 24 bitů za sebou, které jsou následně rozděleny po 6 bitech a tyto šestice bitů tvoří v dekadickém tvaru indexy pro převodní tabulku znaků (tabulka 2.4), jsou tak opět převedeny zpět na ASCII znaky. Z každých 3 bajtů vstupního řetězce tedy vzniknou 4 šifrované znaky ASCII. V případě, že není počet bajtů vstupního řetězce dělitelný 3, je výstup doplněn rovnítky, jejichž počet je dán zbytkem počtu bajtů vstupního řetězce po dělení třemi. V tabulce 2.5 je několik příkladů.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ASCII znak	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d
	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	t	u	v	w	x	y	z	0	1	2	3	4	5	6	7
	60	61	62	63											
	8	9	+	/											

Tabulka 2.4: Převodní tabulka 6-bitových indexů Base64

Vstupní data	**Base64 zakódovaná data**
FIT	RklU
test	dGVzdA==
test2	dGVzdDI=
123456789	MTIzNDU2Nzg5

Tabulka 2.5: Příklady zakódovaných řetězců v Base64

2.9 Operační systém GNU/Linux

GNU/Linux je svobodný operační systém tvořený velkým množstvím volně šířitelných programů, výkonostně a funkčně srovnatelný s jinými operačními systémy. Systém je tvořen z linuxového jádra, z knihoven a nástrojů z projektu GNU, ale i z dalších zdrojů. Linuxové jádro a mnoho dalších GNU komponent je licencováno pod *GNU General Public License (GPL)*. Zdrojové kódy software pod GPL mohou být svobodně upravovány a používány, šířeny však musí být opět pod GPL.

2.10 Gzip (GNU zip)

Gzip (*GNU zip*) je program užívaný pro kompresi dat. Gzip je volně šiřitelný software určený pro projekt GNU, který je tak i součástí linuxových distribucí. Gzip byl vytvořen Jean-loup Gaillyem a Markem Adlerem. Verze 0.1 byla poprvé veřejně představena 31. října 1992. Verze 1.0 následovala v únoru 1993.

> Komprese dat (také komprimace dat) je speciální postup při ukládání nebo transportu dat. Úkolem komprese dat je zmenšit datový tok nebo zmenšit potřebu zdrojů při ukládání informací. Obecně se jedná o snahu zmenšit velikost datových souborů, což je výhodné např. pro jejich archivaci nebo při přenosu přes síť s omezenou rychlostí (snížení doby nutné pro přenos). Komprese může být nutná při omezené datové propustnosti, např. mobilní telefon komprimuje hovor pro přenos GSM sítí. [2]

Program Gzip vznikl jako náhrada za zastaralý program „compress“, přičemž ve většině případů dosahuje Gzip lepších kompresních poměrů a je navíc zbaven všech problémů s patenty. Gzip ztělesňuje samotný kompresní algoritmus založený na algoritmu *DEFLATE*, který je kombinací *LZ77* a *Huffmanova kódování*. DEFLATE byl určený k nahrazení patentem zatížených algoritmů pro kompresi dat, který měl v té době omezenou použitelnost komprese.

Souborový formát programu gzip obsahuje (zdroj [2]):

- 10bajtová hlavička, obsahující magic number, číslo verze a datum poslední změny
- nepovinné extra hlavičky, jako například originální jméno souboru
- tělo obsahující DEFLATE-kompresi
- 8bajtové zápatí, obsahující CRC-32 součet a délku originálních nekomprimovaných dat

Protokol HTTP/1.1 umožňuje klientům, aby volitelně žádali o kompresi obsahu na straně serveru. Standardně sám specifikuje dvě kompresní metody: „gzip“ (obsah zabalený v gzip proudu) a „deflate“ (obsah v bezhlavičkovém DEFLATE proudu). Oba kompresní postupy jsou podporovány mnoha HTTP klientskými knihovnami a většinou moderních prohlížečů, nejvíce užívanou metodou je kompresní metoda Gzip. Na kompresi obsahu webových stránek je podrobněji zaměřena kapitola 3.5.4.

2.11 ImageMagick

ImageMagick je balík volně šiřitelných programů sloužících k neinteraktivnímu zpracování obrázků. K hlavní funkcím patří konverze a úpravy široké škály formátů obrázků, např. otáčení, převracení, ořezávání, změna velikosti, zaostřování, animování atd. Balík je distribuován pod licencí GPL.

Téměř žádný z programů v balíku nedisponuje grafickým uživatelským rozhraním, vyžadují tedy terminálový přístup, díky tomu jsou ideální k použití ve skriptech a k dávkovému zpracování souborů. V tabulce 2.6 je uveden výčet programů z balíku ImageMagick. Programy jsou součástí většiny distribucí operačního systému Linux, jsou však dostupné i pro ostatní operační systémy.

Program	Popis
`animate`	animace obrázků
`compare`	matematický a vizuální popis rozdílů mezi obrázky
`composite`	překládá obrázek přes jiný obrázek
`conjure`	interpretuje a spouští skripty napsané v *Magick Scripting Language (MSL)*
`convert`	převádí mezi formáty obázků a mění velikost, ořezává, překlápí atd.
`display`	zobrazuje obrázky (grafické rozhraní)
`identify`	popíše formát a vlastnosti obrázkových souborů
`import`	uloží jakékoliv viditelné okno jako obrázkový soubor
`mogrify`	téměř shodné s `convert`, ale přepisuje vstupní soubory
`montage`	kombinuje samostatné obrázky do složených
`stream`	nástroj pro streamování pixelových složek obrázku do vybraného formátu

Tabulka 2.6: Programy z balíku ImageMagick

2.12 Adblock Plus

Adblock Plus je rozšíření (*Plug-in*) webových prohlížečů Mozilla Firefox a Google Chrome. Rozšíření jsou malé balíčky, které po instalaci přidají do prohlížeče nové funkce. Adblock Plus je plug-in, díky kterému je umožněno blokování vymezených částí stránek, nejčastěji reklam. Do značné míry se tím snižuje objem přenášených dat, což v důsledku zrychluje načítání stránky. Webová stránka je také bez rušících prvků a je přehlednější pro čtení.

Adblock Plus funguje na principu filtrovacích pravidel, které lze rozdělit do dvou skupin. Základní pravidla fungují na principu blokování požadavků na cílový server podle URL adresy, druhá skupina pravidel umožňuje upřesnit typ souboru, popř. lze blokovat konkrétní HTML elementy na webové stránce. V každé z obou skupin pravidel lze definovat výjimky, abychom mohli např. omezit aplikaci některých příliš obecných pravidel. V tabulce 2.7 je uvedena syntaxe základních pravidel včetně příkladů.

Prvek	Význam	Příklad
`*`	nahrazuje libovolný počet znaků	`*.com/*ad*`
`|`	označuje začátek/konec adresy (pouze na začátku/konci pravidla)	`|http://ad*swf|`
`||`	nahrazuje verzi protokolu a připadnou subdoménu www (pouze na začátku pravidla)	`||example.cz`
`@@`	definuje výjimku (pouze na začátku pravidla před vším výše uvedeným)	`@@||reklama.cz`
`^`	nahrazuje 1 z těchto znaků: / : ? = &	`.com^ad.php^id`

Tabulka 2.7: Syntaxe základních pravidel Adblock Plus

Důležitá vlastnost tohoto rozšíření je možnost exportu/importu filtrovacích pravidel, popř. je možné využít online seznamy pravidel. Tím je umožněno snadné sdílení mezi uživateli a není nutné vytvářet vlastní pravidla, popř. lze tato pravidla aplikovat při implementaci nových programů.

Kapitola 3

Návrh systému

Tato kapitola obsahuje chronologický návrh systému proxy serveru. Konceptuální návrh přiblíží základní funkčnost budoucího systému, díky modelu případu užití se ujasní role uživatelů a jejich oprávnění, stavový diagram obecně definuje průchod požadavku systémem. Dále je popsán způsob autentizace uživatelů a metody optimalizace webových stránek. Závěr kapitoly je věnován návrhu uživatelského rozhraní a statistikám.

3.1 Konceptuální popis

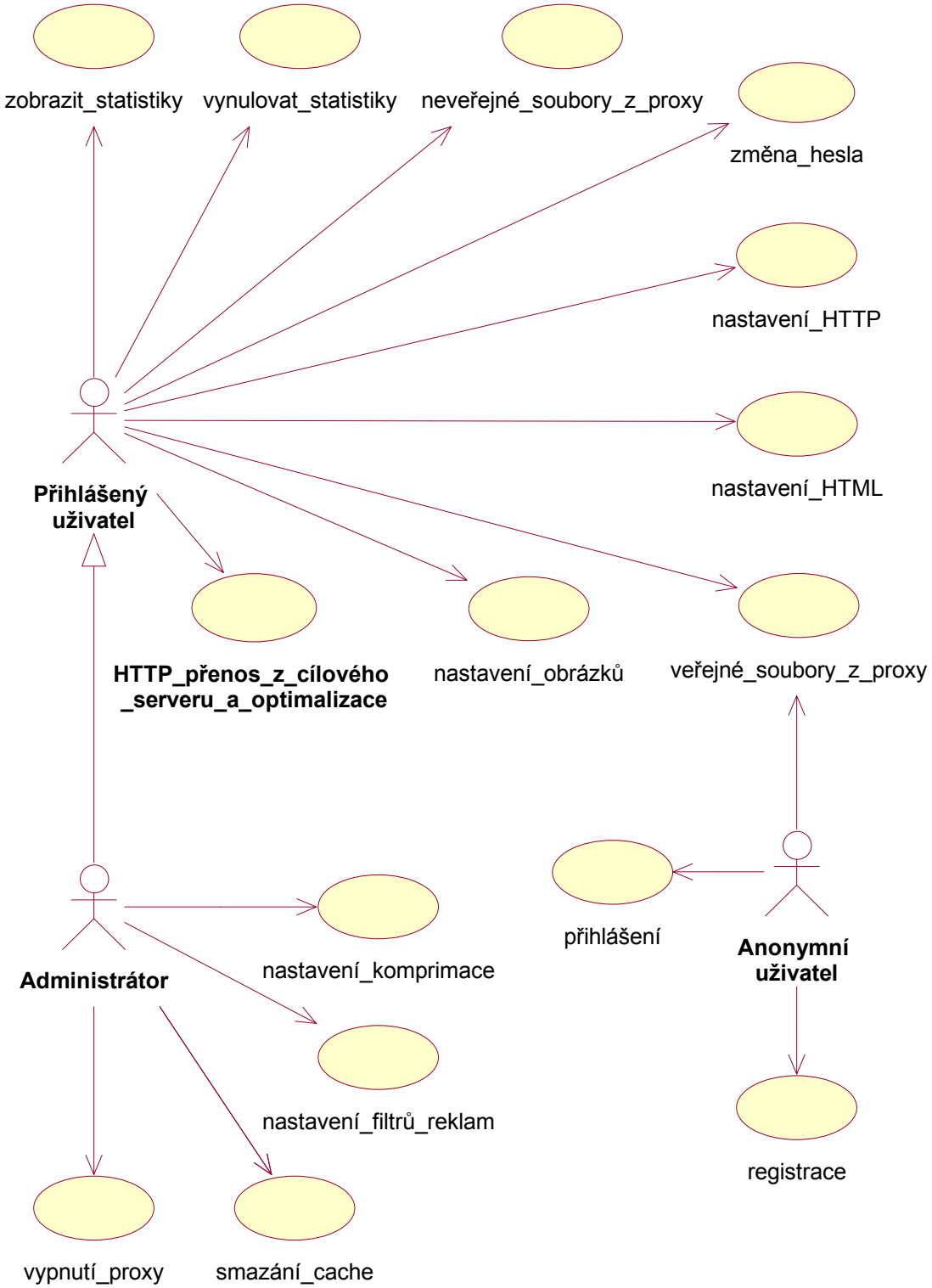
Proxy server bude očekávat na zvoleném portu HTTP požadavek od klienta. Tento požadavek bude po analyzování proxy serverem přeposlán na cílový server. Po přijetí odpovědi s obsahem proxy serverem a následném zoptimalizování obsahu dle předem nastavených pravidel, přepošle proxy server optimalizovanou odpověď klientovi. Možnosti optimalizace webových stránek budou popsány v kapitole 3.5. Nastavení pravidel optimalizace bude umožněno pouze přihlášeným uživatelům (přesněji vysvětluje kapitola 3.4), každý uživatel bude mít možnost různých nastavení optimalizace.

GUI (*graphical user interface*) pro nastavení pravidel bude klientovi přístupné na předem definované URL adrese, např. `http://webproxy/`. Z této skutečnosti plyne, že některé požadavky budou žádat obsah přímo z proxy serveru, což budou HTTP požadavky, které bude muset zpracovat sám proxy server. K těmto požadavkům budou patřit dotazy na soubory s obsahem uloženým v cache proxy serveru, žádost o registraci, zobrazení statistik, atd.

3.2 Model případů užití

Model případů užití (*Use case diagram*) jednoznačně definuje funkční strukturu systému z pohledu uživatele a uživatelských oprávnění. Je primárně určen k definici chování systému vůči uživatelům, aniž by odhaloval jeho vnitřní strukturu.

Model webového proxy serveru je znázorněn na obrázku 3.1, ze kterého je patrná existence 3 uživatelských rolí v systému, konkrétně nepřihlášený (anonymní) uživatel, přihlášený uživatel a administrátor systému. Nejdůležitejším případem užití tohoto systému bude HTTP přenos souborů z cílových serverů a jejich následná optimalizace z hlediska omezeného připojení, proto je tato typová činnost v modelu označena tučně. Z modelu jasně plyne skutečnost, že přenos souborů z cílového serveru bude umožněn pouze přihlášenému uživateli, resp. administrátorovi systému.



Obrázek 3.1: Model případů užití - Webový proxy server

3.3 Stavový diagram

Stavový diagram (*State diagram*) názorně přiblíží průchod klientského požadavku systémem. Z diagramu na obrázku 3.2 jsou zřejmé 4 obecné odpovědi na požadavek (koncové stavy), konkrétně:

- upravená odpověď z cílového serveru,
- interní obsah (odpověď z proxy serveru),
- odpověď s žádostí o přihlášení a
- odpověď z cache proxy serveru.

3.4 Přihlášení uživatele

HTTP protokol zavádí mechanismus prokazování totožnosti - *autentizace*. Server bude odesílat data zpravidla jen oprávněným žadatelům, kteří se prokáží správným jménem a heslem. Takto chráněný přístup nebude definován k celému serveru, k některým částem (adresářům) bude umožněn i veřejný přístup. Pro proxy server bude použito jednoduché autentizační schéma Basic. Jméno a heslo bude klient posílat v hlavičce `Proxy-Authorization`.

Příklad komunikace klienta a proxy serveru při přístupu do chráněné oblasti (v příkladu uvedeny jen hlavičky týkající se autentizace):

1. Klient vyšle běžný požadavek na proxy server:

   ```
   GET http://www.example.cz/private.html HTTP/1.1
   ```

2. Proxy server vrátí odpověď (za touto HTTP hlavičkou může v odpovědi následovat např. HTML formulář pro registraci nového uživatele, popř. chybová zpráva):

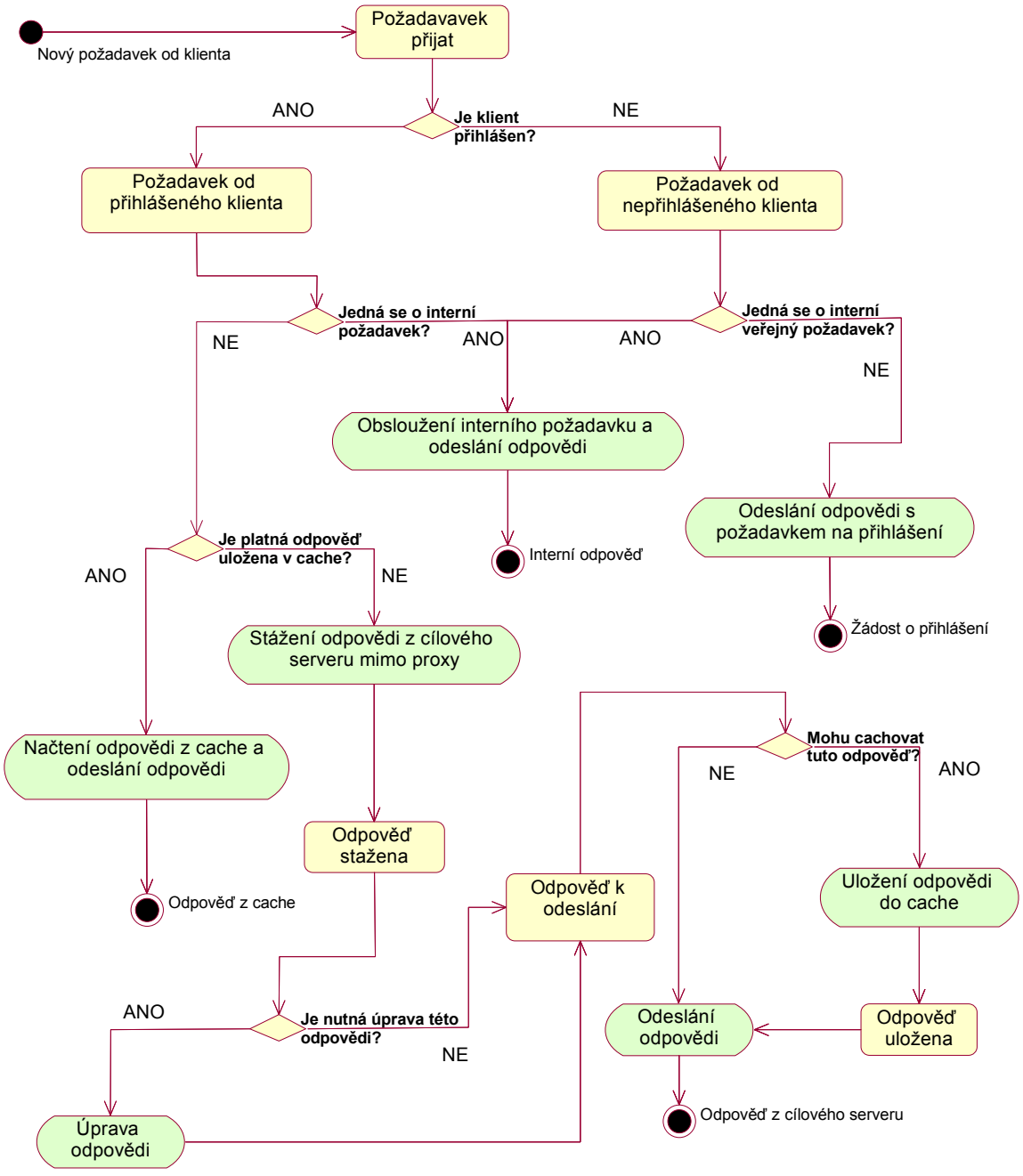
   ```
   HTTP/1.1 407 Proxy Authentication Required
   Proxy-Authenticate: Basic realm='Proxy server'
   ```

3. Webový prohlížeč zobrazí uživateli dialogové okno pro zadání jména a hesla. Dialogové okno obsahuje i název oblasti „realm“, v předchozím příkladě to byl řetězec „Proxy server“. Pokud klient operaci přihlášení stornuje, zobrazí se mu obsah HTTP odpovědi z předchozího bodu. Po zadání jména a hesla opakuje klient původní dotaz, do kterého přidá hlavičku `Proxy-Authorization`. Jméno a heslo se kóduje standardním base64 algoritmem ve tvaru „uživatel:heslo“:

   ```
   GET http://www.example.cz/private.html HTTP/1.1
   Proxy-Authorization: Basic dXppdmF0ZWw6aGVzbG8=
   ```

4. Pokud je jméno a heslo správné, vrátí proxy server pro zpracování požadavku standardní odpověď:

   ```
   HTTP/1.1 200 OK
   ```



Obrázek 3.2: State diagram - Webový proxy server

3.5 Optimalizace webových stránek

V této kapitole jsou popsány způsoby optimalizace webových stránek pro zobrazování v zařízeních s omezenou kapacitou připojení. Každý uživatel bude mít umožněno individuální nastavení konkrétních vlastností optimalizace.

3.5.1 Změna velikosti a formátu obrázků

Změna velikosti rozlišení a formátu obrázků ušetří velké množství přenášených dat mezi klientem a proxy serverem, ve většině případů to však bude na úkor ztráty kvality zobrazení webové stránky.

Úroveň změny velikosti rozlišení obrázků bude udávána v procentech v rozsahu 1 až 100, uživatel tak nebude muset řešit problémy s poměrem stran, obrázky budou zmenšeny se zachováním proporcí. Snížení kvality je v tomto případě zcela zřejmé, zároveň ale dochází k výrazné úspoře dat při zachování informačního charakteru obrázků.

Při změně formátu obrázků se bude projevovat problém ztráty některých vlastností výchozího formátu, např. ztráta průhlednosti (vlastnosti webových grafických formátů byly podrobně popsány v kapitole 2.4). Aby bylo dosaženo největší úspory přenášených dat, bude nutné zvolit ztrátovou kompresi formátu JPEG. Formát JPEG umožňuje nastavení míry komprese a je tak vhodným kandidátem pro nastavení kvality cílových obrázků a zároveň dosahuje nejlepších výsledků redukce objemu dat. Je důležité podotknout, že tomu tak nemusí být vždy, proto se očekává porovnání velikosti souborů před a po optimalizaci. V případě, že bude velikost obrázku po optimalizaci větší, proxy server odešle originální data. V systému webového proxy serveru se tedy očekává implementace převodu všech webových formátů do formátu JPEG.

3.5.2 Blokování HTTP provozu

Blokováním HTTP provozu se u systému webového proxy serveru myslí zaslání odpovědi s prázdným obsahem, avšak s původní HTTP hlavičkou (pouze se opraví informace o délce obsahu). Většina webových prohlížečů tak odpověď uloží a při opětovném načtení stránky neposílá požadavek znovu až do vypršení platnosti původní odpovědi. Je to tak výhodné z hlediska minimalizace datového toku.

V dnešní době internetu se nedá spolehnout na přípony souborů, protože je zcela běžné, že obrázek JPEG má místo `jpg` přípony např. příponu `php`. HTTP hlavička odpovědi proto obsahuje důležité informace o typu obsahu, je tak možné ihned po stažení hlavičky zjistit, o jaký typ dokumentu se jedná. Konkrétně je typ obsahu uveden v hlavičce `Content-Type`, přesněji se jedná *MIME* typ.

MIME je standard zavádějící dvouúrovňovou klasifikaci datového obsahu, zapisovanou jako dvojice „typ/podtyp“. Rozeznává tak několik hlavních typů, které jsou dále upřesněny, např. image/gif říká, že se jedná obrázek formátu GIF. Díky informaci o typu obsahu bude možné zcela konkrétně blokovat určité dokumenty, např. obrázky, Flash, Javascript, CSS, atd. Podobně bude umožněno u určitého typu dokumentu omezit jeho maximální velikost, bude tak možné např. blokovat obrázky, která mají před optimalizací více než 200kB apod.

Další metodou blokování HTTP provozu bude blokování na základě URL adresy. Této vlastnosti bude využito při aplikaci pravidel z rozšíření Adblock a budou tak blokovány reklamy podle tvaru internetové adresy. Na stejném principu funguje první skupina základních filtrovacích pravidel rozšíření Adblock (viz kapitola 2.12).

3.5.3 Úprava HTML dokumentů

Jakékoliv rozsáhlejší úpravy větších HTML dokumentů budou mít za následek zvýšení odezvy serveru, nicméně bude možné dosáhnout vysoké redukce přenesených dat a to nejen díky snížení velikosti HTML dokumentu, ale i díky nenačtení příslušných souborů, které by byly dále ze serveru vyžadovány, např. obrázky, multimediální soubory, apod. Ze zdrojového kódu webové stránky bude nutné odstranit požadované HTML tagy. Uživatelům bude umožněno z HTML dokumentů odstranit např.:

- obrázky (<img>),
- Javascript (<script>) nebo
- HTML komentáře (<!-- -->).

Možností je samozřejmě mnohem více, bylo by možné filtrovat i konkrétní textový obsah atd. Z hlediska optimalizace webových stránek pro zařízení s omezenou kapacitou připojení budou však výše uvedené HTML tagy vhodné.

3.5.4 HTTP komprese

Většina prohlížečů vydaných po roce 1999 podporuje standard HTTP 1.1, který umožňuje definovat způsob komprese obsahu („content-encoding“). HTTP komprese je veřejně definovaný způsob, jak redukovat velikost obsahu webových stránek při přenášení dat mezi serverem a klientem. Komprese probíhá pomocí veřejně přístupných kompresních algoritmů, např. pomocí algoritmu *gzip*.

Klient bude informovat proxy server pomocí hlavičky v HTTP požadavku, že preferuje komprimovaný obsah, např. `Accept-Encoding: gzip`. Server pak bude přenášet data v komprimované podobě, která následně klient (resp. prohlížeč) dekomprimuje. Data budou uložena na serveru (např. v cache) již komprimovaná nebo bude komprese probíhat v reálném čase.

Výhodou komprese bude dramaticky nižší velikost souborů, rychlejší reakce webových stránek a menší nároky na přenosovou šířku pásma. „Kódování obsahu umožňuje snížit zatížení šířky pásma o 30 až 50 procent. Kompresní poměr závisí na míře redundance obsahu webových stránek a poměru text/multimédia.“ (zdroj [4]) Komprimované stránky se tak budou zobrazovat mnohem rychleji, protože budou klienti stahovat méně dat.

3.6 Statistiky

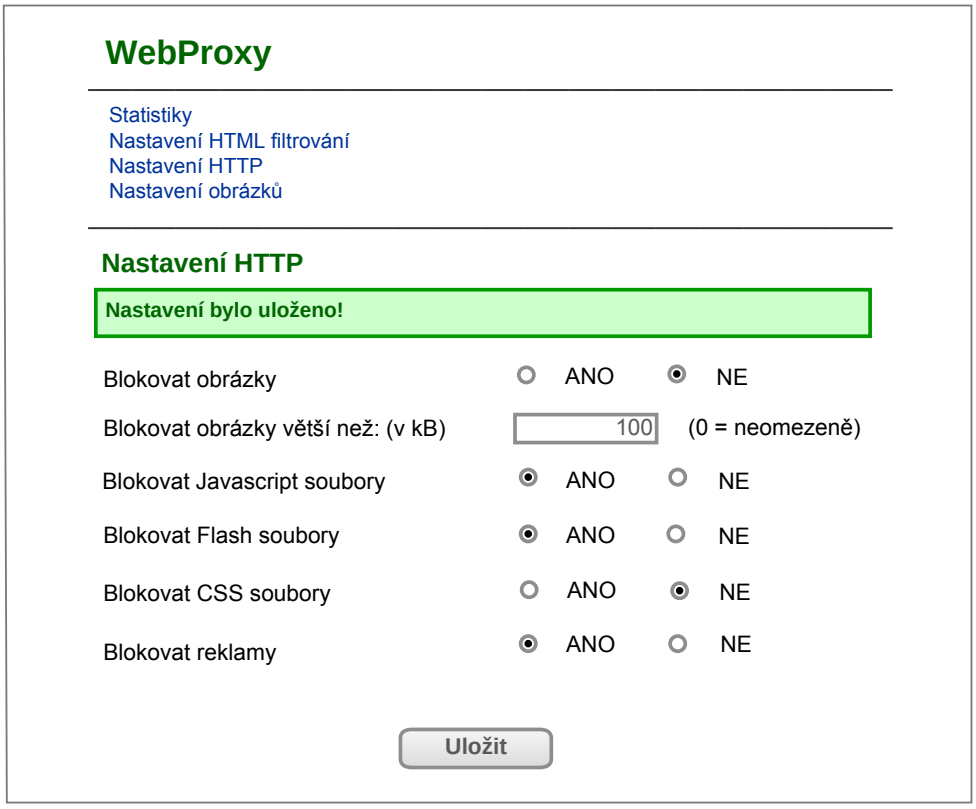
Jelikož bude proxy server zaměřen na optimalizaci z hlediska omezené kapacity připojení, bude užitečné ukládat určitá data o tom, jak bude optimalizace úspěšná, resp. uchovávat informace o množství přenesených dat. Cíl statistik bude úspěšnost optimalizace v procentech, proto budou důležité tyto 3 údaje:

- velikost originálních dat před optimalizací určených k odeslání klientovi,
- objem dat z proxy serveru ke klientovi - „download“ (optimalizovaná data) a
- objem dat od klienta na server - „upload“ (beze změny).

3.7 Návrh uživatelského rozhraní

Uživatelské rozhraní, resp. GUI (Graphical User Interface) umožňuje ovládat systém pomocí interaktivních ovládacích prvků. V případě webového proxy serveru se zcela jasně nabízí možnost webového rozhraní. Uživatel tak bude používat standardní vstupní prvky, konkrétně prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře apod. Výstup systému bude zobrazován přímo v okně prohlížeče.

Jelikož bude možností nastavení proxy serveru více, bude vhodné rozdělit jednotlivé varianty do skupin, uživatelské rozhraní tím bude pro uživatele jednoznačně přehlednější. Skupiny budou zpřístupněny v uživatelském menu, které umožňuje snadný přechod mezi jednotlivými sekcemi. Návrh uživatelského rozhraní webového proxy serveru je na obrázku 3.3.



Obrázek 3.3: Návrh grafického uživatelského rozhraní

Kapitola 4

Implementace systému

Webový proxy server je implementován pro operační systém GNU/Linux v programovacím jazyce C++ pro protokol HTTP s hlavním zaměřením na optimalizaci webových stránek pro zobrazování v zařízeních s omezenou kapacitou připojení. Odborné znalosti k implementaci proxy serveru byly částečně čerpány z knihy [5].

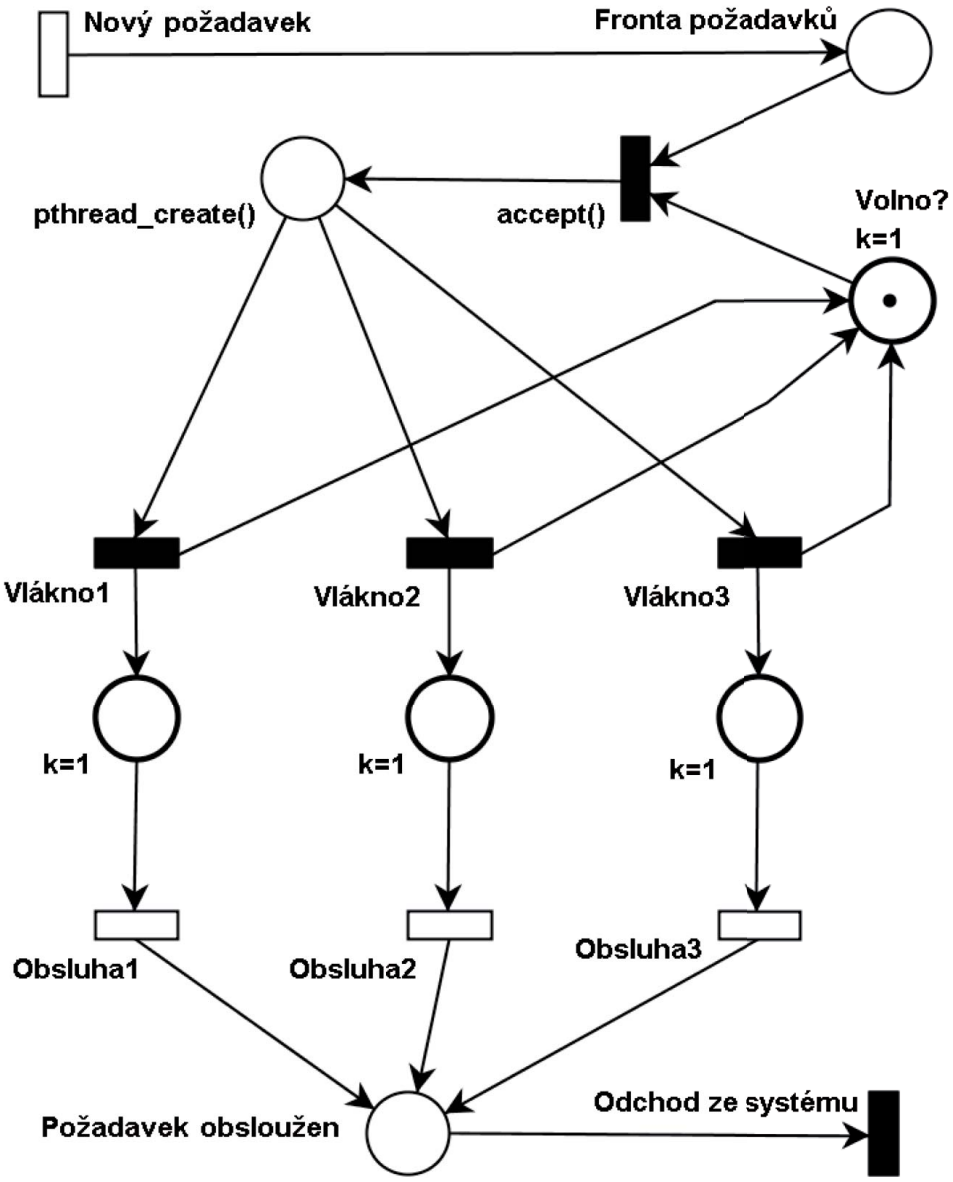
4.1 Vícevláknový proxy server

V případě webového serveru si je nutné uvědomit, že požadavky od klientů budou s největší pravděpodobností přicházet paralelně a každý požadavek bude mít rozdílnou dobu obsluhy. Zpracování požadavků na serveru musí rovněž probíhat současně a připojení klienta musí být „neblokující“, aby klienti nemuseli čekat na dokončení obsluhy dříve připojených klientů.

Paralelnost zpracování je implementována pomocí vláken (*threads*). Vlákna jsou v podstatě paralelně prováděné funkce v rámci jednoho procesu. Vlákna se do jisté míry podobají procesům (jsou prováděna paralelně), vlákna však sdílí téměř všechny prostředky procesu a jsou tak výhodné např. pro sdílení globálních dat. Je však potřeba zajistit, aby se např. jedno vlákno nesnažilo zapisovat do proměnné a jiné vlákno se nepokoušelo současně z té stejné proměnné načítat data. Vlákna spolu tedy musí nějakým způsobem komunikovat a právě k tomu slouží synchonizační prostředky, tzv. *mutexy*.

Mutex je jednoduchý prostředek pro synchronizaci vláken. Lze pomocí něj „zamykat“ globální proměnné a struktury, ke kterým se přistupuje z více vláken. Zamknutí obstará funkce `pthread_mutex_lock()`, odemknutí naopak funkce `pthread_mutex_unlock()`. Mutex může být uzamknut pouze jedním vláknem, proto nemůže nastat situace, kdy budou dvě a více vláken požadovat přístup ke sdíleným proměnným. Pokud je už mutex uzamčen, vlákno čeká na jeho odemknutí, přístup ke sdíleným proměnným je tak výlučný.

Komunikace mezi klienty a proxy serverem je realizována pomocí socketů. Po vyjmutí prvního připojení z fronty nevyřízených spojení pomocí funkce `accept()`, která přidělí souborový deskriptor pro nový socket (původní socket je opět použit pro čekání na další spojení), je volána funkce `pthread_create()`, která vytvoří nové vlákno. Klienti jsou tak odbavování současně, omezení je zde pouze v počtu vláken (resp. volné paměti). Realizace je modelově znázorněna pomocí Petriho sítě na obrázku 4.1, kde je zvoleno omezení na 3 vlákna.



Obrázek 4.1: Petriho síť - vícevláknový server

4.2 Databáze uživatelů

Systém využívá vlastní databázový systém, který ukládá uživatelské údaje a statistiky do souboru. Aplikace tak umožňuje snadnou instalaci, jelikož nepotřebuje instalaci nebo nastavení jiných databázových systémů. Z databázového souboru jsou data načtena při spuštění systému, za běhu jsou do něj data pouze ukládána a to jen v případech, kdy je to naprosto nezbytné. Veškeré informace jsou načteny stále v paměti, aby nebylo nutné s každým příchozím požadavkem přistupovat k databázovému souboru (např. ověřovat autentizační údaje uživatele).

Data jsou v paměti uchována ve formě lineárního jednosměrně vázaného seznamu, kde každá položka seznamu obsahuje strukturu s uživatelskými daty. Seznam uživatelů patří ke sdíleným prostředkům a přístup k němu je nutné synchronizovat pomocí mutexů. Obsah datové struktury uživatele definuje tabulka 4.1.

Název položky	**Datový typ**	**Popis**
`username`	string	uživatelské jméno, jednoznačný identifikátor
`auth`	string	base64 zakódovaný autentizační řetězec
`load_images`	boolean	blokuje přenos obrázků
`max_image_size`	integer	maximální velikost obrázku
`load_javascript`	boolean	blokuje přenos Javascript souborů
`load_css`	boolean	blokuje přenos CSS souborů
`load_ad`	boolean	blokuje přenos reklam - import Adblock
`load_flash`	boolean	blokuje přenos Flash souborů
`downloaded`	integer	objem stažených optimalizovaných dat
`original_downloaded`	integer	objem originálních dat k odeslání
`uploaded`	integer	objem dat přenesených od klienta na server
`remove_spaces`	boolean	odstraňuje z HTML dokumentů komentáře
`remove_javascript`	boolean	odstraňuje z HTML dokumentů tagy <script>
`remove_images`	boolean	odstraňuje z HTML dokumentů tagy <img>
`image_resize`	integer	změní velikost rozlišení obrázků (v procentech)
`image_compression`	integer	úroveň JPEG komprese (v procentech)
`image_jpg_jpg`	boolean	zvyšuje úroveň komprese u JPEG souborů
`image_png_jpg`	boolean	konvertuje formát PNG na JPEG
`image_bmp_jpg`	boolean	konvertuje formát BMP na JPEG
`image_gif_jpg`	boolean	konvertuje formát GIF na JPEG

Tabulka 4.1: Informace o uživateli - datová struktura a obsah databáze

4.3 Implementace optimalizačních metod

Návrh optimalizačních metod byl podrobně rozebrán v kapitole 3.5. V následujících několika kapitolách je stručně popsána implementace těchto metod.

4.3.1 Úpravy obrázků a HTTP komprese

Úprava obrázků byla kompletně implementovaná pomocí programu `convert` z balíku programů ImageMagick (viz kapitola 2.11), k jehož hlavním výhodám patří jednoduchost pou-

žití. Navíc program umí pracovat s Gzip komprimovanými soubory, není tak nutné odpovědi z cílových serverů nejprve dekomprimovat. K HTTP kompresi a dekompresi byl použit program Gzip (viz kapitola 2.10). Pro práci s těmito programy v C++ byla zvolena možnost systémového volání funkcí `system()`, je proto nutné mít výše zmíněné programy nainstalované v operačním systému (součástí GNU/Linux).

4.3.2 Úprava HTML dokumentů

HTML dokumenty jsou upravovány za využití regulárních výrazů. Soubor je po načtení z cílového serveru postupně procházen po jednotlivých HTML elementech (tag po tagu) a podle uživatelského nastavení některé HTML tagy a jejich obsah proxy server filtruje. Implementováno bylo filtrování komentářů (<!-- -->), obrázků (<img>) a Javascriptu (<script>).

4.3.3 Blokování HTTP provozu

HTTP provoz je blokován na základě MIME typu odpovědi z cílového serveru. V tomto případě proxy server nejprve stáhne hlavičku a pokud má být obsah blokován, proxy server již nepokračuje v načítání odpovědi. Pokud se tedy jedná o typ, který požaduje uživatel zablokovat, je uživateli vrácena HTTP hlavička z originální odpovědi, ale s prázdným tělem. Webové prohlížeče v tomto případě prázdnou odpověď uloží do cache a nevysílají při návratu na stránku opětovný požadavek, dokud jim to „nedovolí“ cílový server. Druhou metodou blokování HTTP provozu je blokování podle tvaru požadované adresy, o tom pojednává podrobněji následující kapitola.

4.4 Import pravidel z Adblock Plus

Webový proxy server umožňuje správci serveru importovat základní filtrovací pravidla z rozšíření Adblock Plus (viz kapitola 2.12), uživatelé pak mohou využít volby „Blokovat reklamy“. Blokování požadavků na reklamní formáty probíhá na základě obsahu URL adresy na úrovní HTTP.

Filtrovací pravidla rozšíření Adblock Plus mohou definovat nejen pravidla pro blokování, lze definovat i výjimky, které naopak pravidla pro blokování ruší. Syntaxe pravidel je poněkud odlišná od regulárních výrazů aplikovatelných v C/C++, je proto potřebná určitá konverze. Tato konverze probíhá v systému pouze interně, správce serveru spravuje pravidla stále ve formátu Adblock Plus. Převod na regulární výrazy je znázorněn v tabulce 4.2 na příkladech.

Adblock Plus pravidlo	Regulární výraz	Typ
`*/ads/*`	`.*\/ads\/.*`	blokování
`@@*/ads/*`	`.*\/ads\/.*`	výjimka
`@@||vut*.jpg|`	`^(http:\/\/www\.|http:\/\/)vut.*\.jpg$`	výjimka
`*.swf?clickthru=*`	`.*\.swf?clickthru=.*`	blokování
`*^click^*`	`.*[/:?=&$]click[/:?=&$].*`	blokování
`|http://web.cz/ad.jpg|`	`^http:\/\/web.cz\/ad\.jpg$`	blokování

Tabulka 4.2: Příklady konverze Adblock Plus pravidel na regulární výrazy

Regulární výrazy jsou v systému načteny opět v podobě lineárního jednosměrně vázaného seznamu, kde je v každé položce seznamu uchována informace o tom, zda se jedná o výjimku

či blokovací pravidlo. Výjimky jsou po té procházeny jako první a pokud adresa výjimce vyhoví, blokovací pravidla se už nekontrolují. Při inicializaci interního seznamu pravidel jsou ukládány již zkompilované regulární výrazy.

4.5 Cache

Cache proxy serveru je způsob, jak uchovávat odpovědi z cílových serverů, není tak nutné požadavky na stejné dokumenty opakovat a znovu optimalizovat. V případě souborů, které nevyžadují optimalizaci je to snadné, odpověď proxy server uloží do souboru do předem vytvořeného adresáře. Složitější situace nastává v momentě, kdy má každý uživatel jiná nastavení optimalizace daného typu souborů, např. obrázků nebo HTML dokumentů. Je tak nutné rozlišit, jaká nastavení byla na soubor aplikována, aby klienti s odlišným nastavením nedostali odpověď s nevhodně optimalizovaným obsahem. Soubor je pak uložen s předponou značící nastavení, např. originální soubor s názvem `index.html` bude uložen jako `1_0_1_index.html`, kde čísla značí v pořadí zleva doprava, že byly odstraněny HTML komentáře, Javascript byl v kódu zanechán v nezměněné podobě a byly odstraněny obrázky, resp. HTML tagy <img>. Dále je nutné uchovat informaci o době pořízení odpovědi, aby bylo možné zjistit, zda je uložená odpověď platná. Posledním parametrem je originální velikost odpovědi před optimalizací sloužící k záznamu statistik i u souborů uložených v cache. Poslední 2 informace jsou uloženy vždy na prvním řádku v ukládaném souboru.

4.6 Uživatelské rozhraní

Uživatelské rozhraní je implementováno v XHTML a CSS, realizováno pomocí šablon pro snadnou následnou úpravu správcem systému. Proxy server umožňuje administrátorovi nahrát vlastní soubory (např. obrázky, Javascript) nebo lze vytvořit např. statický web. K nastavení jednotlivých uživatelkých voleb jsou použity formuláře. Uživatelské rozhraní je přístupné z jakéhokoliv klientského webového prohlížeče umožňující nastavení HTTP proxy serveru. Uživatelské rozhraní je proto nezávislé na operačním systému.

Kapitola 5

Testování systému

5.1 Testování v různých prohlížečích

Webový proxy server byl testován v nejrozšířenějších webových prohlížečích a to ve všech případech s pozitivním výsledkem. Celá implementace je nezávislá na volbě prohlížeče, takže je zde pouze omezení na podporu proxy serverů (protokol HTTP). Otestovány byly prohlížeče Mozilla Firefox, Microsoft Internet Explorer, Google Chrome, Apple Safari a Konqueror, s největší pravděpodobností by nenastala žádná komplikace ani v jiných prohlížečích.

5.2 Dosažené výsledky

Každá webová stránka má jiný charakter obsahu, což je nutné brát v úvahu při vyhodnocování výsledků. Testovací vzorek webových stránek byl zvolen s ohledem na obsah stránek, aby bylo možné získat průměrný výsledek. Je zcela zřejmé, že není možné optimalizovat naprosto všechna data, stejně tak není možné přesně definovat chování potencionálního uživatele proxy serveru. Webové stránky zahrnuté ve vzorku však obsahují většinu elementů, které vykazují dnešní internetové stránky.

Testování za účelem zisku výsledků probíhalo ve webovém prohlížeči Mozilla Firefox 3.5 za využití rozšíření Firebug, serverová aplikace byla nainstalována na počítači `merlin.fit.vutbr.cz`.

5.2.1 Doba přenosu

Vzorek webových stránek byl podroben nejen testu redukce objemu dat, ale měřena byla i doba načítání webových stránek. Z tohoto testu je patrné, že uživatelé s omezenou kapacitou připojení ušetří poměrně dost času, čímž byl jednoznačně splněn jeden z cílů práce. Naopak uživatelé s rychlejším připojením budou kvůli redukci objemu přenesených dat čekat v podstatě stejně dlouho, což v kombinaci se ztrátou informací nepřináší cílený efekt. Konkrétní hodnoty jsou uvedeny v tabulce 5.1. Hodnotou „před“ se v tabulce myslí doba, která je potřebná ke stažení webové stránky bez použití proxy serveru, projeví se tedy i čas, který je potřebný ke komunikaci mezi proxy serverem a cílovým serverem.

Připojení/čas stažení	**ADSL 3 Mbit/s**		**EDGE 236 kbit/s**		**GPRS 64 kbit/s**	
Webová stránka	Před	Po	Před	Po	Před	Po
`www.fit.vutbr.cz`	0,323 s	0,294 s	4,081 s	0,394 s	15,611 s	0,700 s
`www.seznam.cz`	1,720 s	1,510 s	8,032 s	1,893 s	27,397 s	3,069 s
`www.novinky.cz`	2,560 s	0,860 s	35,480 s	1,251 s	136,475 s	2,453 s
`www.youtube.com`	1,890 s	0,946 s	9,719 s	1,379 s	33,737 s	2,708 s
Průměr	1,623 s	0,903 s	14,328 s	1,229 s	53,305 s	2,233 s
Úspora času	**44,36%** (0,72 s)		**91,42%** (13,099 s)		**95,81%** (51,072 s)	

Tabulka 5.1: Srovnání výsledků po maximální optimalizaci podle připojení

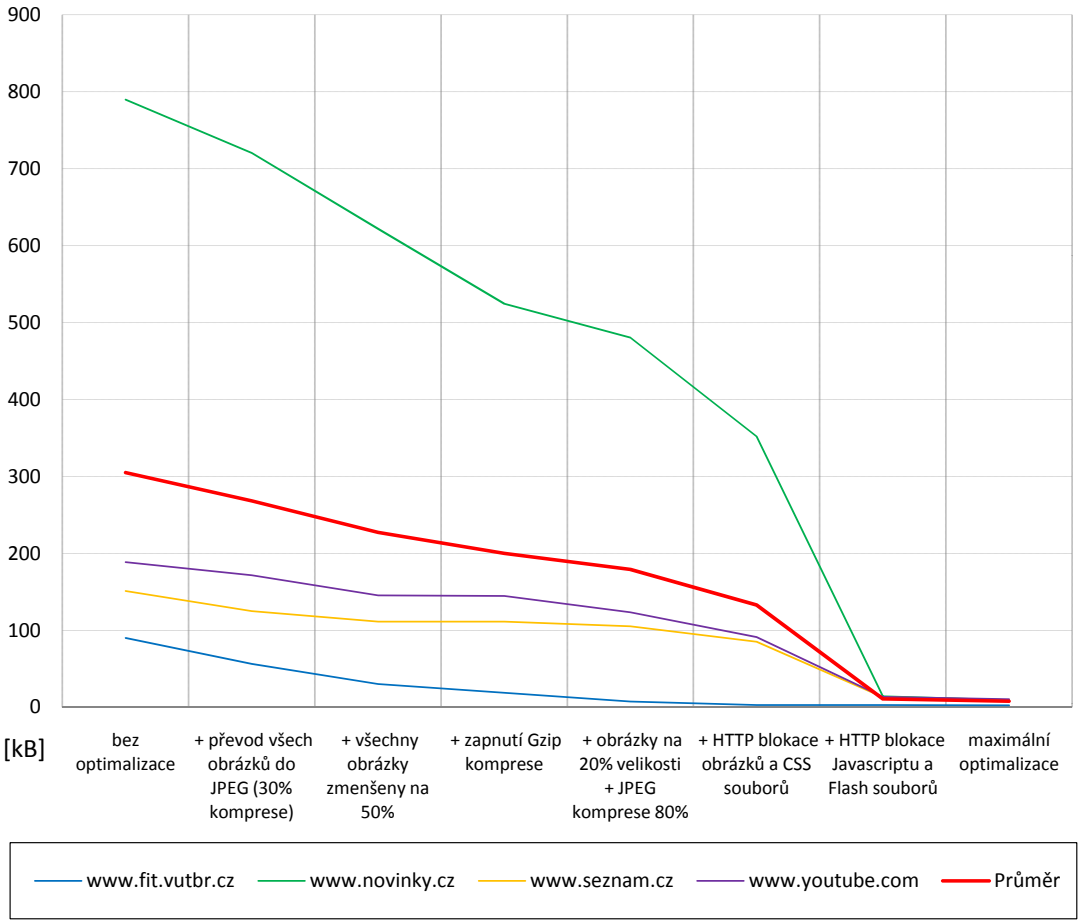
5.2.2 Redukce objemu dat

K názorné ukázce úspěšnosti optimalizačních metod je na obrázku 5.1 graf objemu přenesených dat v závislosti na optimalizačních metodách. V testu byly jednotlivé optimalizační techniky aktivovány postupně, je tak dokázáno, jakých výsledků tyto metody dosahují. Z grafu jsou průkazné webové stránky komprimované již před optimalizací. Přesné velikosti webových stránek po maximální optimalizaci, jsou uvedeny v tabulce 5.2.

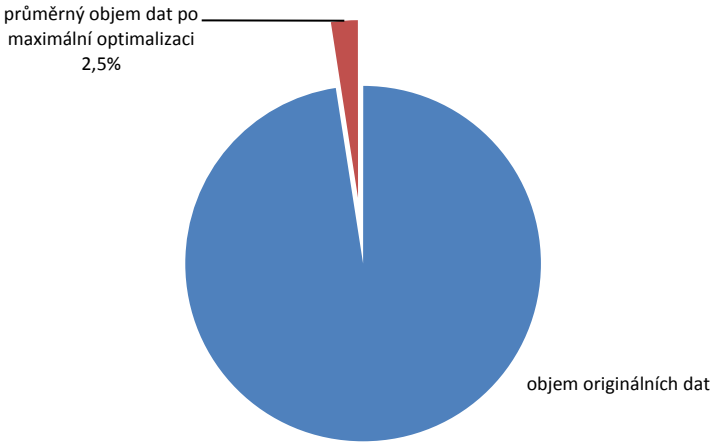
Webová stránka	**Bez optimalizace**	**Po optimalizaci**	**Úspora dat**
`www.fit.vutbr.cz`	90,2 kB	2,4 kB	87,8 kB
`www.seznam.cz`	151,2 kB	9,2 kB	142,0 kB
`www.novinky.cz`	789,3 kB	9,4 kB	779,8 kB
`www.youtube.com`	188,5 kB	10,4 kB	178,1 kB
Průměr	**304,875 kB**	**7,85 kB**	**97,5 %**

Tabulka 5.2: Redukce velikosti webových stránek po maximální optimalizaci

Testováním bylo zjištěno, že průměrná redukce velikosti webových stránek je při maximální optimalizaci průměrně čtyřicetinásobná, což je velice pozitivní výsledek. Pro reálnou představu úspory přenesených dat je na obrázku 5.2 poměr originálních a optimalizovaných dat. Je nutné poznamenat, že úroveň optimalizace radikálně snižuje informační kvalitu multimediálních stránek, naopak např. úvodní strana `www.fit.vutbr.cz` si i přes maximální optimalizaci zachovává téměř shodný informační charakter a je tak názorným příkladem, jak má vypadat přístupná webová stránka bez bariér.



Obrázek 5.1: Graf objemu přenesených dat v závislosti na optimalizačních metodách



Obrázek 5.2: Poměr objemu originálních a optimalizovaných dat

Kapitola 6

Závěr

Hlavním cílem této bakalářské práce byl návrh a následná implementace webového proxy serveru, který na základě uživatelských nastavení optimalizace redukuje velikost webových stránek. Proxy server jsem implementoval pro protokol HTTP včetně uživatelské databáze. Výsledky testování systému jsem podrobně zpracoval v předchozí kapitole, ve které jsem statisticky prokázal až čtyřicetinásobnou úsporu objemu dat, přičemž uživatelé s omezenou kapacitou připojení mohou průměrně ušetřit až 95 procent z původního času potřebného ke stažení webových stránek. Mohu tedy konstatovat, že jsem splnil všechny v úvodu formulované cíle.

Proxy server umožňuje import základních filtrovacích pravidel z rozšíření Adblock Plus, čímž jsem docílil blokování reklamních souborů. Budoucí uživatelé proxy serveru mohou této volby využít a omezit velké množství přenášených dat. Systém je tedy vhodný nejen k optimalizaci obrázků, HTML dokumentů a multimediálního obsahu, lze ho využít např. pro rozšíření možnosti blokování reklam na prohlížeče, které nepodporují Adblock Plus, např. Microsoft Internet Explorer, Opera nebo Apple Safari.

Pro reálné nasazení systému doporučuji více zabezpečit administrátorskou část systému, jelikož algoritmus base64 a základní autentizační HTTP schéma neposkytují dostatečné zabezpečení. Pro další rozvoj systému bych tedy použil kombinaci kontroly IP adresy uživatele a náhodně vygenerovaného a pravidelně obměňovaného řetězce uloženého u uživatele (např. pomocí HTTP cookies) a v databázi systému. Do systému by bylo možné v budoucnu implementovat rozšířená filtrovací pravidla z Adblock Plus a dosáhnout tak filtrování konkrétních HTML elementů, čímž by mohly být odfiltrovány např. textové reklamní bloky.

Literatura

[1] Dostupné z WWW: RFC2616 - Hypertext Transfer Protocol - HTTP/1.1. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[2] Dostupné z WWW: Wikipedie, otevřená encyklopedie. http://cs.wikipedia.org.

[3] Dostálek, L.; Kabelová, A.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Computer Press, Praha, 2000, ISBN 80-7226-323-4.

[4] King, A. B.: *Zrychlete své WWW stránky!* Zoner Press, Brno, 2004, ISBN 80-86815-02-1.

[5] Lee, J.; Ware, B.: *Open Source - vývoj webových aplikací Linux, Apache, MySQL, Perl a PHP*. Mobil Media a.s., 2003, ISBN 80-86593-43-6.

[6] Strese, M.; Perkins, C.: *Firewally a proxy-servery: Praktický průvodce*. Computer Press, Brno, 2003, ISBN 80-7226-983-6.

Dodatek A

Obsah CD

Adresářová sktruktura přiloženého CD je popsána v následující tabulce:

Cesta	**Popis**
`/program/zdroj/`	Zdrojový kód webového proxy serveru (přeložitelný pomocí `Makefile`).
`/program/merlin_compiled/`	Spustitelná verze programu přeložená na `merlin.fit.vutbr.cz`.
`/dokumentace/`	Úplná programová dokumentace.
`/BP/zdroj/`	Zdrojový kód této technické zprávy. LaTeX
`/BP/print/`	Tato technická zpráva ve verzi pro tisk.
`/BP/online/`	Elektronická verze této technické zprávy.

Dodatek B

Manuál

Tento manuál popisuje instalaci a použití webového proxy serveru.

1. Program nejprve přeložit pomocí přiloženého `Makefile` příkazem `make`.
2. Spustit ve tvaru `./webproxy PORT &` (PORT nahradit zvoleným portem).
3. V internetovém prohlížeči nastavit připojení přes proxy server (pouze pro HTTP protokol).
4. Po zadání jakékoliv URL adresy je uživatel vyzván k přihlášení. V programu je ve výchozím stavu přidán uživatel „admin“ s heslem „admin“. Tento jediný účet má nejvyšší oprávnění v systému (je mu umožněna správa), je proto doporučeno ihned po přihlášení změnit heslo. Pokud uživatel přihlášení v prohlížeči stornuje, je mu nabídnuta registrace. Každá další registrace uživatele v systému má již nižší oprávnění.
5. Na adrese `http://webproxy/` se nachází grafické uživatelské rozhraní, kde je možné nastavit parametry optimalizace a sledovat statistiky. Správce systému (uživatel „admin“) má v uživatelském rozhraní možnost správy proxy serveru.