VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

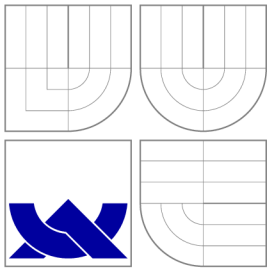
SPRÁVA STAŽENÝCH A ODESLANÝCH DAT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

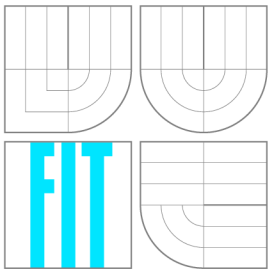
AUTOR PRÁCE
AUTHOR

TOMÁŠ KONEČNÝ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SPRÁVA STAŽENÝCH A ODESLANÝCH DAT

TRAFFIC STATISTICS IN LAN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ KONEČNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAEL KUNC

BRNO 2009

Abstrakt

Práce popisuje systém pro evidenci stažených a odeslaných dat v síti LAN, včetně evidence navštívených serverů. Dále se zabývá použitými technologiemi, analyzuje požadavky aplikace, popisuje etapy návrhu řešení a implementace a diskutuje další pokračování tohoto projektu.

Abstract

This task describes system for filing downloaded and uploaded data in the LAN, including filing of visited servers. In addition it deals with used technologies, analyses aplication requirements, describes phases of suggested solvings and implementation and debates on the next steps of this project.

Klíčová slova

Statistika, LAN, Data, PHP, JavaScript, CSS, AJAX, MySQL, Databáze, Web

Keywords

Statistics, LAN, Data, PHP, JavaScript, CSS, AJAX, MySQL, Database, Web

Citace

Tomáš Konečný: Správa stažených a odeslaných dat, bakalářská práce, Brno, FIT VUT v Brně, 2009

Správa stažených a odeslaných dat

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michaela Kunce.

.....
Tomáš Konečný
27. dubna 2009

Poděkování

Touto cestou bych chtěl poděkovat Ing. Michaelovi Kuncovi za poskytnutí odborné pomoci při konzultaci bakalářské práce. Rovněž bych chtěl poděkovat Ing. Karlu Zacharovi za možnost vytvořit aplikaci pro správu stažených a odeslaných dat v LAN síti VnorovyNet.

Obsah

Kapitola 1

Úvod

Již od pradávna měli lidé sklon k vytváření přehledů, statistik a jiných průzkumů, které by jim umožnily provést analýzu a ulehčily jim tak rozhodování. V informačních technologiích tomu není jinak. Právě naopak tyto technologie podporují sběr nejrůznějších dat, které je možné dále zpracovávat.

Jelikož pro potřeby dalšího rozvíjení poskytovatele internetového připojení společnosti VnorovyNet bylo zapotřebí vytvořit právě statistické přehledy přenesených dat, rozhodli se majitelé vyhledat aplikaci, která by jim tuto práci ulehčila. Volně dostupné aplikace v sobě však neobsahovaly všechny požadavky společnosti a tak vznikl nápad vytvořit aplikaci, která by tyto přehledy umožňovala a zároveň byla přístupná přes internetové prohlížeče.

Tento text tedy popisuje návrh, vývoj a testování aplikace, která by umožňovala přehledným způsobem zobrazovat stažená a odeslaná data v konkrétní síti LAN. Dále se text zabývá problematikou implementace aplikace, technologiemi pro vývoj webových aplikací a informačních systémů, návrhem aplikace, testováním aplikace a závěrečným zhodnocením.

Požadavkem plynoucím ze zadání práce je tedy tvorba aplikace, která pracuje v síti internet a bude dostupná prostřednictvím internetového prohlížeče. Aplikace musí umožňovat rozdělit uživatele na dvě skupiny a to podle přidělených práv:

- administrátor
- uživatel

Dále aplikace bude zaznamenávat a prezentovat množství přenesených dat v obou směrech, v různých časových intervalech, včetně grafického zpracování, umožňovat vyhledávání a řazení těchto dat, dále pak ukládání a zobrazení seznamu navštívených stránek pro každého uživatele včetně souhrnných statistik a rozlišovat, jestli se jedná o množství přenesených dat ze sítě www nebo p2p. Aplikace by také neměla příliš zatěžovat server.

1.1 Technologie pro vývoj

Aby bylo možné s daty dále pracovat, bude zapotřebí data vhodným způsobem ukládat. Pro ukládání dat bude sloužit databáze. Jako prostředek pro manipulaci s daty bude sloužit skriptovací jazyk.

Výhody a nevýhody jednotlivých technologií, jejich použitelnost, podrobnější popis a přehled technologií možných pro návrh aplikace je uveden v *kapitole 2.*

1.2 Požadavky na systém

Při návrhu aplikace bylo nutné také počítat s technickými možnostmi zadavatele a přizpůsobit tomu návrh aplikace. Požadavky na systém a volbou implementačního jazyka se zabývá *kapitola 3.*

1.3 Návrh aplikace

Další podmínkou bylo, aby aplikace pracovala po dobu 5-ti let a umožňovala správu dat pro 400 uživatelů lokální sítě. Dále aby byl kladen důraz na jednoduché a intuitivní uživatelské rozhraní a archivaci nekomprimovaných dat 6 měsíců zpětně. Podrobným návrhem aplikace se zabývá *kapitola 4.* V návrhu se také zabývám testovací verzí databáze. Vše je podrobně popsáno a přehledně zobrazeno pomocí ER diagramů a diagramu případů užití.

1.4 Implementace aplikace

V této kapitole je popsán postup při implementaci aplikace a úprava některých skriptů proti původnímu návrhu. Kapitola také popisuje testování a následné uvedení do ostrého provozu. Implementací aplikace se zabývá *kapitola 5.*

1.5 Závěr

Kapitola 6 se zabývá shrnutím dosažených výsledků a dalším možným pokračováním projektu.

1.6 Existující řešení

Nástrojů pro evidenci stažených a odeslaných dat existuje celá řada. Aplikace, která by však umožňovala zahrnout všechny požadavky zadavatele, není v době psaní této práce na trhu volně k dispozici.

Aplikace, která se svými funkcemi nejvíce blíží zadání, je MRTG. Jedná se o volně šiřitelný nástroj napsaný ve skriptovacím jazyku Perl. Úvaha o úpravě tohoto již existujícího nástroje vedla k přesvědčení, že bude jednodušší a efektivnější vytvořit aplikaci novou. Tento nástroj totiž neumožňuje přidělení různých stupňů oprávnění a také nedovede zpracovat odděleně data ze sítí www a p2p, případně vést statistiku navštívených serverů.

Kapitola 2

Technologie pro vývoj webových aplikací

Pro vývoj webových aplikací existuje mnoho nástrojů. Tyto nástroje lze rozdělit do dvou kategorií:

- komerční
- volně šiřitelné

Mezi komerční nástroje pro vývoj webových aplikací patří skriptovací jazyky ASP, .NET, ASP.NET a další. Z databází jsou to například Oracle a MSSQL. Tyto nástroje budou zmíněny jen okrajově, jelikož se jedná o placený software a jsou zaměřeny zejména na rozsáhlé informační systémy.

Z volně šiřitelného software jsou to především skriptovací jazyky PHP, AJAX, JavaScript a z databází MySQL a PqSQL. Také je potřeba zmínit značkovací jazyky jako HTML, XHTML a kaskádové styly CSS, o kterých bude dále řeč.

2.1 HTML

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML. Jazyk HTML se z něj vyčlenil specifikací konkrétních značek.

Nejdříve se objevil jazyk HTML v roce 1991 ve verzi 0.9. Tato verze prozatím nepodporovala grafický režim. Vývoj jazyku se rychle posouval kupředu až k verzi 4.01, která se objevila v roce 1999. Podle původního předpokladu se mělo jednat o poslední verzi, po které by se přešlo na XHTML. V roce 2007 byla však založena nová pracovní skupina HTML, jejímž cílem je vývoj nové verze HTML. V květnu 2007 bylo odhlasováno, že základem nové specifikace se stanou Web Applications 1.0 a Web Forms 2.0. Částečně převzato z [1], [2] a [3].

2.2 XHTML

Značkovací jazyk XHTML je novější norma HTML. XHTML používá v době psaní tohoto textu verze:

- XHTML 1.0 přechodové (transitional)
- XHTML 1.0 striktní (strict)
- XHTML 1.1

Jazyk XHTML byl vytvořen z jazyků HTML 4.01 a XML 1.0 a z přidání typových definic dokumentu DTD, které určují strukturu dokumentu a elementy, které jsou v něm použité. Jazyk XHTML má mnohem přísnější syntaxi než jazyk HTML. Vše je psáno malými písmeny, parametry musí být vždy zapsány v závorkách. Všechny elementy i ty, které nemají v HTML ukončovací značku, musí končit tagem s lomítkem. Každý dokument napsaný v jazyku XHTML musí začínat hlavičkou *<?xml version="1.0" encoding="windows-1250"?>* a dále musí následovat typ dokumentu DTD.

Výhodou XHTML proti HTML je jednodušší implementace, rychlejší zpracování a přehlednější kód. Umožnuje definovat vzhled www stránky pomocí vnořených nebo externích kaskádových stylů. Více v [1], [2] a [4].

2.3 CSS

Kaskádové styly vznikly jako reakce na oddělení definice vzhledu dokumentu z HTML a XHTML. Umožňují tedy definovat vzhled dokumentu bez ohledu na jeho obsah.
Vzhled dokumentu je definován odděleně v souborech s příponou *.css. Díky tomu lze vzhled snadno modifikovat. Může být rovněž definován pro množinu stránek a zahrnuje jednotnou definici stylu pro všechny elementy. Snadno tak lze vytvořit například alternativní podobu dokumentu.

Výsledná podoba dokumentu je pak definována stylesheety ze tří zdrojů.

1. Styl připravený tvůrcem stránek
2. Uživatelský styl (vlastní definice uživatele)
3. Styl prohlížeče

Tyto styly se seřadí do kaskády a kompletně definují výsledný vzhled dokumentu. Více v [1].

2.4 PHP

PHP je volně šiřitelný skriptovací jazyk, který vznikl v roce 1994, kdy Rasmus Lerdorf vytvořil jednoduchý systém evidence přístupů ke svému webu nejdříve v jazyce Perl, poté v C. Ten se rozšířil mezi další uživatele, kteří přicházeli s požadavky na vylepšení. Vznikl tak systém Personal Home Page Tools, později Personal Home Page Construction Kit.

Skriptovací jazyk PHP se řadí mezi servlety. To znamená, že skript se neprovádí na straně uživatele (jako JavaScript), ale na straně serveru. Odpadá tedy nutnost kontrolovat podporu skriptu na straně uživatele. Jednou napsaný a odladěný skript se spouští a provádí stále ve stejném prostředí na straně serveru, uživateli se odesílá pouze zpracovaný výsledek, kterým může být například webová stránka ve formátu HTML, nebo součet čísel 5+3. Takový výsledek se pak zobrazí v internetovém prohlížeči. Komunikace tedy vypadá tak, že uživatel zažádá o zobrazení internetové stránky www.seznam.cz, požadavek se odešle na server, ten dotaz zpracuje a uživateli odešle požadovanou stránku.

Od verze 3.0 bylo možné v PHP používat SQL dotazy a tím pracovat s databázemi. PHP se stále vyvíjí a přizpůsobuje novým technologiím. V době psaní tohoto textu je nejaktulánější verze 5.2.9.

K tomu, abychom mohli provádět skripty napsané v jazyku PHP, potřebujeme server Apache. Jedná se o program, který běží na straně serveru a provádí příkazy obsažené v souboreh PHP.

Soubory v PHP mají nejčastěji příponu .php. Avšak je možné použít i .php3, php4, php5 a phtml.

2.4.1 Značky ohraničující PHP skript

Pro vložení PHP kódu do HTML elementu existuje několik možností:

- <SCRIPT language="php"> úsek kódu </SCRIPT>
- <?php úsek kódu ?>
- <? úsek kódu ?>
- <? include 'externiscript.php'; ?>

2.4.2 Komentáře

Ke každému skriptu patří pro lepší přehlednost komentáře. V PHP se zapisuje komentář takto:

- *// jednořádkový komentář*
- */*
  víceřádkový komentář
  **/*

Všechny příkazy a deklarace musí v PHP končit středníkem. Proměnnou deklarujeme znakem dolar $. Jednoduchý příklad, který by sečetl dvě čísla, by vypadal takto: *$soucet = 5 + 3;*

Název proměnné se uvádí hned za značku $. Pokud proměnnou tvoří číslo, se kterým se bude později počítat (sčítat, dělit..), musí být zapsáno bez uvozovek, jinak by jej server pokládal za text. Typ proměnné se neudává, určí se podle typu přiřazené hodnoty.

2.4.3 Základní datové typy

Hodnoty jednotlivých datových typů můžeme zapsat přímo literálem. V případě celých čísel (integer) můžeme hodnotu zapsat v desítkové soustavě, v osmičkové soustavě s prefixem 0 nebo v šestnáctkové s prefixem 0x. Datové typy:

Skalární	Float	String	Pole	NULL
Integer	Double	Boolean	Speciální	

PHP je určeno svým zaměřením pro širokou veřejnost. Při srovnání s komerčním softwarem jako je např. ASP.NET může však v některých případech dojít k pomalejšímu zpracování. Ve výsledku se však skriptovací jazyk PHP používá pro nejrůznější aplikace, od jednoduchých prezentací až po internetové obchody. Částečně převzato z [5].

2.5 ASP

ASP, neboli Active Server Pages, je technologie určená zejména pro webové stránky, která je vykonávána na straně serveru. Jedná se tady o servlet. Byla vyvíjena společností Microsoft, která jej později nahradila systémem ASP.NET. Pro programování v ASP se používaly programovací jazyky jako VBscript nebo JScript. Pro ASP.NET se již používá Visual Basic, .NET, Jazyk C# a Jazyk J#. Ke svému provozu potřebuje ASP webový server, např. Microsoft ISS.

ASP je svým charakterem a postavením na trhu zaměřeno spíše na rozsáhlejší webové aplikace. Výhodou je profesionální podpora ze strany Microsoftu a rychlost zpracování samotných skriptů oproti PHP.

2.6 JSP

JSP byla vyvinutá společností Sun Microsystems za účelem vývoje aplikací na straně serveru. JSP soubory jsou vlastně HTML stránky, do kterých jsou vložené speciální tagy obsahující zdrojový kód napsaný v Jave. Tento kód potom vytváří dynamický obsah stránky. Použít JSP je vhodné hlavně tehdy, kdy většinu částí stránky tvoří statický obsah. V opačném případě je vhodné použít na vytváření stránek technologii servletů. Převzato z [6].

2.7 JavaScript

JavaScript je skriptovací jazyk vyvinutý firmou Netscape. Programy napsané v JavaScriptu jsou nezávislé na platformě počítače, na kterém jsou provozovány. To znamená, že pokud bude webový prohlížeč podporovat JavaScript a bude mít tuto podporu zapnutou, lze skript spustit na všech dostupných operačních systémech. Skripty se zpracovávají na straně klienta. JavaScrit můžeme tedy použít jako vhodný nástroj pro oživení www stránek. Umožnuje nám reagovat na různé události, které na stránce nastanou. Například pohyb myši, stlačení tlačítka a podobně. Částečně převzato z [7].

2.8 Ajax

Ajax vznikl jako reakce na další interaktivitu webových aplikací. Pro svou funkci využívá především JavaScript, CSS, DOM. Ajax nám umožňuje úplně nový náhled na práci s webovými aplikacemi a to především asynchronně komunikovat se serverem. To znamená, že můžeme provádět mnoho úkolů, které byly dříve vyhrazeny klasickým klientským aplikacím. Když například uživatel zadá poštovní směrovací číslo, můžeme ho ověřit a případně s jeho pomocí automaticky vyplnit další položky formuláře jako jsou město a stát.

Ajax je tedy technika pracující na straně klienta a funguje bez ohlednu na to, jakou technologii používáme na straně serveru. Klíčovou komponentou Ajaxu je objekt XMLHttpReguest. Více v [8].

2.9 MySQL

Tento volně šiřitelný systém řízení báze dat se nasazuje hlavně u webových aplikací menšího rozsahu. Lze jej provozovat na stanicích s operačním systémem Unix, Linux, Solaris, OS/2 i Windows. MySQL bylo od počátku optimalizováno především na rychlost a to i za cenu

některých zjednodušení. Od verze 5.0 však již podporuje i uložené procedury, triggery a pohledy. Více v [9].

2.10 Oracle

Databáze od firmy Oracle je komerční software špičkové úrovně. Databáze od této firmy je možné použít pro rozsáhlé databázové systémy i pro menší projekty. Společnost Oracle uvolnila s určitými omezeními i volně dostupnou verzi. Databáze se vyznačuje vysokou bezpečností, rychlostí a podporou samotné společnosti. Nad databází Oracle lze vyvíjet aplikace v prakticky libovolném vývojovém prostředí jako je Java, PHP, Python, C++ nebo .NET a mnohé další.

Kapitola 3

Požadavky na systém

Při návrhu aplikace pro správu stažených a odeslaných dat v síti LAN bylo nutné počítat s technickými možnostmi zadavetele a aplikaci tomu přizpůsobit. Aby však byla zachována požadovaná funkčnost a použitelnost aplikace, bylo zapotřebí upravit softwarové nastavení některých komponent.

3.1 Microtik RouterOS

Klíčovou komponentou pro běh aplikace je server s nainstalovaným systémem Mikrotik verze 2.9.48. Zařízení umožňuje pasivní monitorování provozu na síti - sběr dat z čítačů. Ke komunikaci využívá Simple Network Managment Protokol, zkráceně SNMP. Zařízení podporuje pouze první verzi protokolu SNMPv1. Komunikace běží nad UDP, port 161. Technické parametry serveru:

- CPU - INTEL Core 2 Duo E4600 2,40GHz (2MB/800) BOX LGA775
- HDD - 80GB
- RAM - 512MB
- OS - Debian
- Velikost čítačů: 2^{32}

Informace o jednotlivých zařízeních na síti se ukládají ve virtuální databázi všech objektů, zkráceně MIB. Každý takovýto objekt má své jméno a identifikační číslo tvořené sekvencí čísel od kořene, zkráceně OID. OID využívaná na serveru Microtik:

- .1.3.6.1.2.1.2.2.1.10.7 - celkový WAN přenos dat do sítě
- .1.3.6.1.2.1.2.2.1.16.7 - celkový WAN přenost dat ze sítě
- .1.3.6.1.2.1.2.2.1.11.7 - celkový WAN přenost paketů do sítě
- .1.3.6.1.2.1.2.2.1.17.7 - celkový WAN přenost paketů ze sítě
- .1.3.6.1.4.1.14988.1.1.2.2.1.2 - obsahuje IP adresy ve zkrácené formě všech uživatelů sítě + jejich OID

Pro zajištění potřebných dat pro aplikaci jsou každému uživateli přidělena 4 OID (x1 až x4).

- .1.3.6.1.4.1.14988.1.1.2.2.1.5.x1 - celkový upload ze sítě www
- .1.3.6.1.4.1.14988.1.1.2.2.1.5.x2 - celkový upload ze sítě p2p
- .1.3.6.1.4.1.14988.1.1.2.2.1.5.x3 - celkový download ze sítě www
- .1.3.6.1.4.1.14988.1.1.2.2.1.5.x4 - celkový download ze sítě p2p
- .1.3.6.1.4.1.14988.1.1.2.2.1.6.x1 - celkový download paketů ze sítě www
- .1.3.6.1.4.1.14988.1.1.2.2.1.6.x2 - celkový download paketů ze sítě p2p
- .1.3.6.1.4.1.14988.1.1.2.2.1.6.x3 - celkový upload paketů ze sítě www
- .1.3.6.1.4.1.14988.1.1.2.2.1.6.x4 - celkový upload paketů ze sítě p2p

Aby bylo možné jednotlivá OID pro aplikaci rozlišit a přidělit je tak ke správným datům, jsou pojmenovány pro každého jednotlivého uživatele pomocí tohoto klíče:

- net-xyz.xyz.xyz.xyz
- net-p2p-xyz.xyz.xyz.xyz
- xyz.xyz.xyz.xyz-net-p2p
- xyz.xyz.xyz.xyz-net

kde x, y a z představují číselně vyjádřenou IP adresu.

3.2 Hardwarová konfigurace serveru

Pro běh aplikace slouží jednojádrový procesor Celeron od společnosti Intel pracující na frekvenci 1.2GHz. Základní deska je použita Intel Little Valley D201GLY2A SiS664. Server je osazený 1GB paměti RAM a pevným diskem o velikosti 250GB připojeným přes rozhraní SATA II. Do sítě je připojen přes 100Mbit síťovou kartu. Jako záloha napájení je použita UPS s výkonem 480VA. Pro grafický výstup slouží integrovaná grafická karta na základní desce serveru a 17" CRT monitor.

3.3 Softwarová konfigurace serveru

Při volbě operačního systému se už od začátku počítalo s distribucí Linuxu. Linux byl vybrán pro svoji stabilitu (je zapotřebí provoz 24 hodin denně, 7 dní v týdnu) a cenu. Jako operační systém byl nejprve použit 64bit Linux Mandriva 2007.1. Tento systém však musel být pro svoji nekompatibilitu odinstalován a nahrazen 32bit operačním systémem verze Linux Mandriva 2009 i586. V předchozím operačním systému se vyskytla chyba při zpracování hodnot z čítačů a docházelo k přetečení.

Jako souborový systém je použit ext3. V operačním systému je nainstalovaný daemon Cron, který v pravidelných 5-ti minutových intervalech spouští skript traffic.php.

Aby bylo možné skript traffic.php spouštět, má nastavena práva 777. Soubor traffic.php je uložený v adresáři traffic. Stejné nastavení práv má i soubor provoz.log uložený v adresáři log.

Dalším daemonem je syslogd, který vytváří soubor provoz.log. Daemon naslouchá na portu UDP/514. Každý den ve 4:00 je daemon ukončen programem logrotate. Logrotate následně provede archivaci souboru. Archiv je označen provozX.gz (kde X je pořadí) a po dokončení archivace program logrotate opět spustí daemona syslogd. Soubor provoz.log obsahuje přístupy jednotlivých uživatelů sítě k webovým serverům. Příklad záznamu v souboru provoz.log: *Mar 4 17:20:34 172.20.10.17 firewall,info provoz: provoz-syn forward: in:LAN out:WAN, src-mac 00:0c:42:20:b2:c3, proto TCP (SYN), 192.168.5.32:1049– >77.75.72.2:80, len 52*

Z technologií pro vývoj webových aplikací byl vybrán skriptovací jazyk PHP. Pro svoji dostupnost a širokou podporu ze strany vývojářů se jedná o naprosto dostačující nástroj poskytující relativně rychlé zpracování skriptů a dostatečnou podporu pro SNMP komunikaci. Na server byla nainstalována verze 5.2.6. K tomu, aby bylo možné používat skriptovací jazyk PHP, musel být nainstalován také server Apache. Použita byla verze 2.0 Handler. Pro správnou funkčnost aplikace musejí být oproti výchozímu nastavení povoleny v konfiguračním souboru php.ini balíčky libnet-snmp15, net-snmp, net-snmp-mibs, net-snmp-utils a php-snmp.

Jako databázový systém byl zvolen volně dostupný MySQL. Tento systém vyniká především u databází s omezeným počtem záznamů. Jelikož aplikace pro správu dat má plnit pouze informační charakter a bude využívána v průměru jednou denně (požadavky zadavatele), můžeme si tedy dovolit pomalejší načítání aplikace při vybraných operacích. Pro uživatele s přístupovými právy "uživatel" bude rychlost načítání dat dostačující pro pohodlné prohlížení obsahu. Uživateli s těmito právy se bude z databáze vybírat pouze omezené množství dat. Nainstalována byla verze 5.0.67.

Pro vykreslování grafů byla použita knihovna JpGraph. Tato knihovna umožňuje v PHP kódu vytvářet grafy ve formě obrázků s potřebnými vlastnostmi. Lze tak vytvářet grafy sloupcové, spojnicové, 3D a mnohé jiné. Knihovna funguje od verze PHP 4.3.1. a jejím autorem je švédská firma Aditus. Podmínkou pro použití neplacené verze je nekomerční využití.

Kapitola 4

Návrh aplikace

Při návrhu aplikace jsem vycházel z požadavků zadání bakalářské práce. Pro co nejlepší použitelnost aplikace bylo zapotřebí navrhnout optimální podobu databáze, skriptů a uživatelského rozhraní. Postup při návrhu a jeho konečnou podobu popíši v následujících kapitolách.

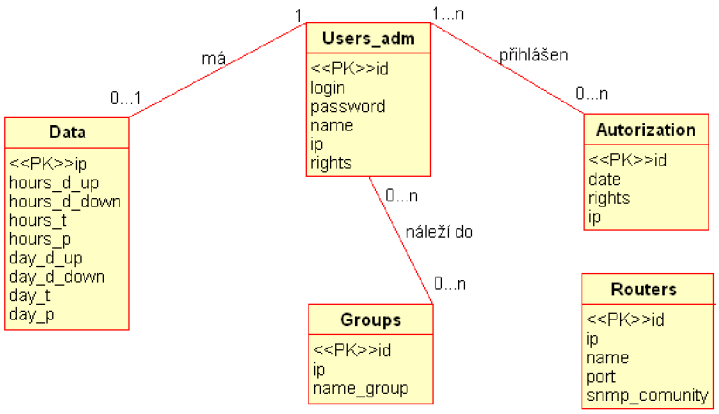
4.1 Návrh databáze

Při návrhu databáze jsem se ubíral dvěma směry. Mým prvním záměrem bylo vytvořit databázi, která by i po 5-ti letech provozu nezabírala mnoho místa na disku, měla rychlou odezvu při výpisu požadovaných dat a zpracování výsledků nezatěžovalo významně procesor. Tímto návrhem se zabývá kapitola *Testovací verze databáze*. Kapitola *Konečná podoba databáze* popisuje finální návrh databáze použité v aplikaci.

V databázi jsou data aktualizována každých 5 minut pomocí skriptu traffic.php.

4.1.1 Testovací verze databáze

Tato testovací verze nepočítá se všemi funkcemi požadovanými v zadání. Jedná se pouze o testovací návrh, který by měl pomoci s konečnou podobou databáze. Požadavek tedy je, aby data mohla v databázi být uložena po dobu 5-ti let a to v denních intervalech.



Obrázek 4.1: Obrázek ER diagramu znázorňující testovací verzi databáze.

Údaje zaznamenávající 24 hodinový průběh stažených a odeslaných dat budou po 24 hodinách mazány. Ukládány budou pouze údaje o celkových stažených a odeslaných datech a celkových paketech. Databáze bude pro účely testování naplněna daty jako v případě 2 letého provozu.

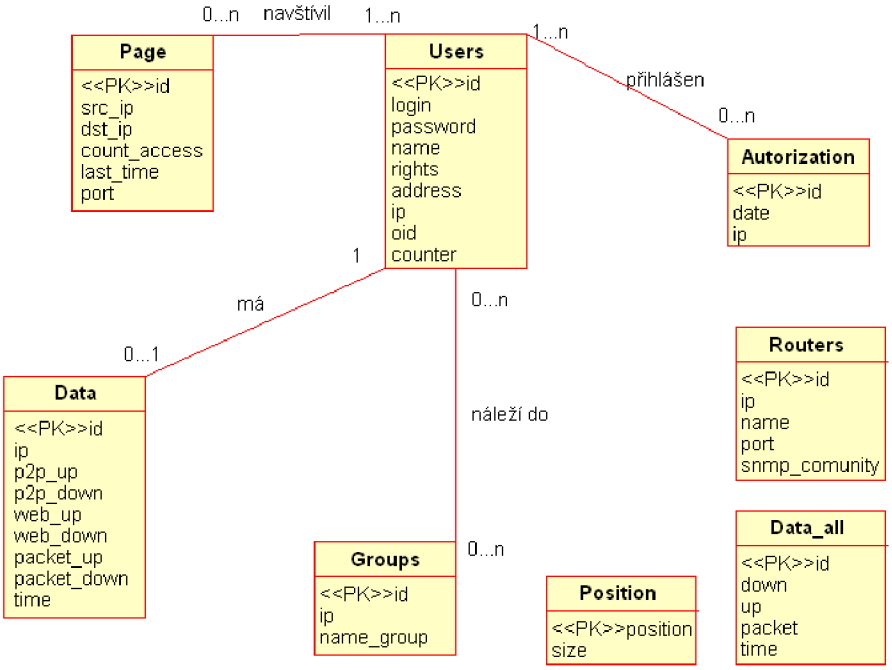
Při návrhu jsem nejdříve vytvořil konceptuální model v podobě ER diagramu. Odpovídající ER diagram je na obrázku 4.1. Návrh vychází z požadavku na data. V tomto návrhu databáze jsem zamýšlel přidělit každému uživateli maximálně jeden řádek v tabulce data. Jednotlivé sloupce v tabulce (hours_d_up, hours_d_down,...) jsou datového typu text. Díky tomu mohou být jednotlivé hodnoty v buňce odděleny čárkami. V tomto uspořádání má tedy tabulka data při přidělení všech IP adres počet řádků stejný, jako je počet uživatelů v síti i po 5-ti letech provozu. Narůstají pouze textové hodnoty v jednotlivých buňkách. Příklad záznamu v buňce hours_d_up by po třech cyklech aktualizace dat vypadal takto: 7684136,7661817,7816809,11245824. Jednotlivá čísla představují množství přenesených dat do sítě v 5-ti minutovém intervalu u konkrétního uživatele.

Toto řešení se však v praxi osvědčilo jen částečně. Velký důraz se musel klást na správnou synchronizaci dat, aby k vybranému časovému horizontu náležela správná data. Další nevýhodou bylo velké zatížení procesoru při hledání konkrétních dat nebo při vypsání určité množiny dat. Bylo nutné nejdříve celý řetězec projít, než se našla požadovaná data.

Naproti tomu velikost databáze by po 5-ti letech provozu při 400 uživatelích sítě nepřekročila velikost 30MiB.

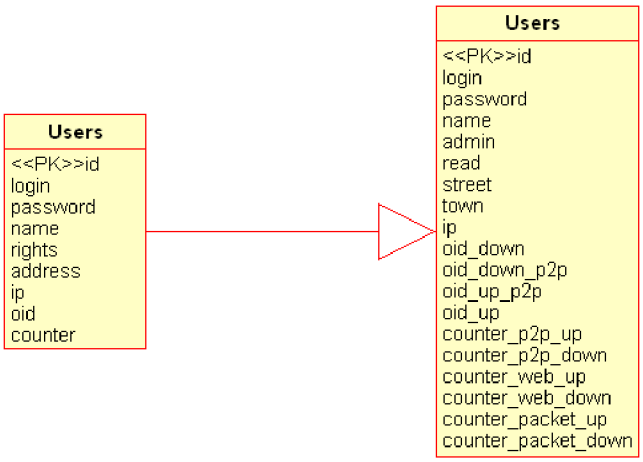
4.1.2 Konečná podoba databáze

Tento návrh databáze počítá již se všemi požadovanými funkcemi aplikace a zásadně se liší v návrhu od testovací verze databáze. Při návrhu jsem nejdříve vytvořil konceptuální model v podobě ER diagramu. Odpovídajíci ER diagram je na obrázku 4.2.



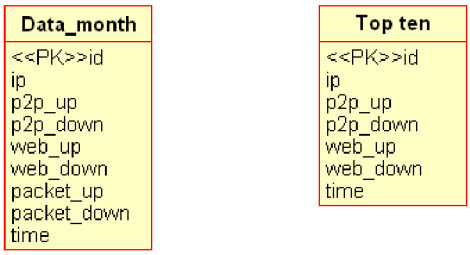
Obrázek 4.2: Obrázek ER diagramu znázorňující návrh databáze.

Dalším krokem bylo odstranění složených atributů. Odstranění je na obrázku 4.3 Při



Obrázek 4.3: Odstranění vícehodnotových atributů

návrhu jsem také vycházel z poznatků testovací databáze a přizpůsobil tak díky tomu i konečnou podobu databáze. Data v databázi jsou aktualizována v pětiminutových intervalech. Tedy u každého uživatele je to 288 nových záznamů za den. Při testech databáze vyšlo najevo, že zpracování dat z tabulky data do požadované formy výrazně zatěžuje diskové pole. Při výpisu měsíčního toku dat je to 288 záznamů * 31 dnů = 8928 záznamů u jediného uživatele za měsíc. Kvůli zvýšení rychlosti načítání dat bylo nutné k tabulce data přidat ještě tabulky top_ten a data_month. Tyto tabulky jsou na obrázku 4.4. Tabulky



Obrázek 4.4: Tabulka TopTen a Data month

uchovávají komprimované hodnoty z tabulky data. v případě použití tabulky data_month je to maximálně 31 záznamů za měsíc. Tabulky data_month a top_ten zvětšují velikost celé databáze, ale pomáhají rychlejšímu načítání aplikace. Data v těchto tabulkách jsou aktualizována průběžně.

Posledním krokem bylo převedení ER diagramu na relační databázi. Skript pro vytvoření databátových tabulek je uložen v souboru statistika2.sql.

Počet záznamů v databázi

Při výpočtu množství záznamů v databázi jsem vycházel z požadavků zadání. Tedy počet uživatelů sítě bude 400, doba provozu 5 let.

V databázi data by po 5-ti letech provozu tedy bylo 288(aktualizací) * 400(uživatelů) * 365(dní) * 5(let) = 210 240 000 záznamů. Takovéto množství záznamů by představovalo 1.6GiB místa v diskovém poli (změřeno při testovacím plnění).

V tabulce data all by to bylo 288(aktualizací) * 365(dní) * 5(let) = 525 600 záznamů. Tedy 23,8MiB (změřeno při testovacím plnění).

Tabulka data month by obsahovala 400(uživatelů) * 365(dní) * 5(let) = 730 000 záznamů. Tedy 51,6MiB (změřeno při testovacím plnění).

Tabulka top ten by obsahovala 400(uživatelů) * 12(měsíců) * 5(let) = 24 600 záznamů. Tedy 1MiB (změřeno při testovacím plnění).

Záznamy v tabulce page nelze přesně spočítat, jelikož množsví nově navštívených stránek je závislé na uživatelích. Vypočtená hodnota je tedy statistickým průměrem. Tabulka page by tedy obsahovala 400(uživatelů) * 365(dní) * 5(let) * 39,907(serverů/den/uživatele) = 29 132 143 záznamů. Tedy 2870MiB (změřeno při testovacím plnění).

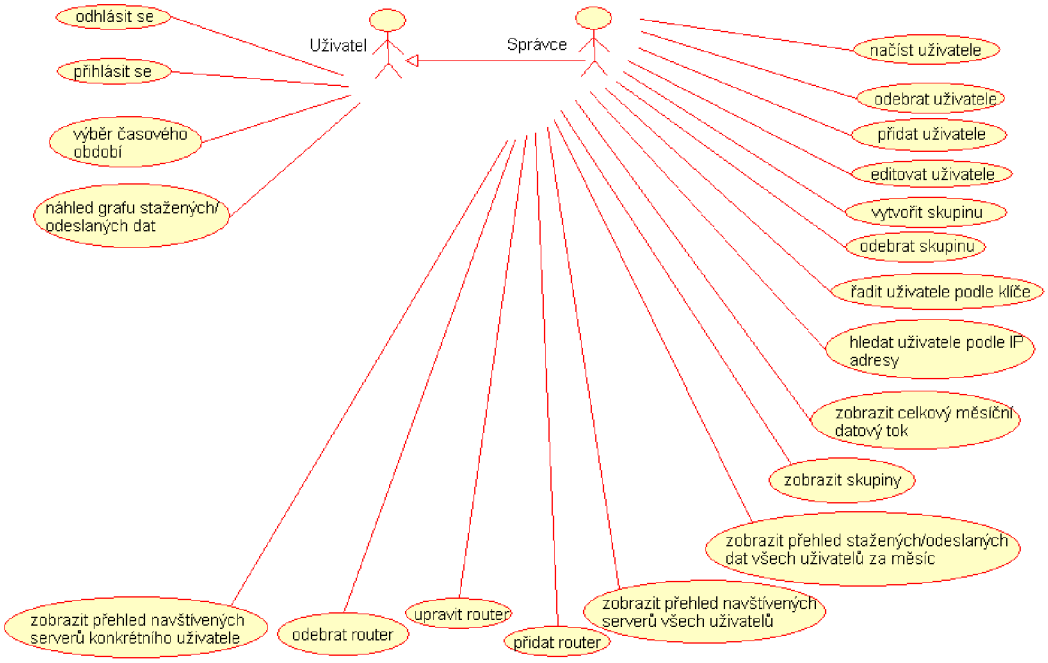
Celkově by tedy databáze po 5-ti letech provozu zabírala na disku místo přibližně 4546,4MiB.

Integritní omezení

Při realizaci databáze jsem použil datové typy s ohledem na jejich maximální hodnoty a vhodně zvolil jejich omezení tak, aby nedocházelo k nechtěnému zpomalení databáze.

4.2 Případ užití

Diagram případů užití popisuje požadavky zadání s ohledem na použitelnost aplikace.
V diagramu na obrázku 4.5 nalezneme dva účty. Uživatel a správce a jejich možnosti práce s aplikací.



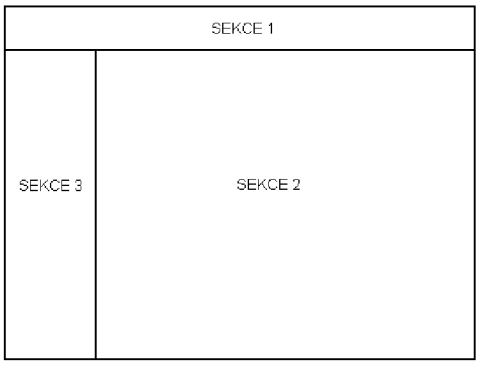
Obrázek 4.5: Diagram případů užití

Z diagramu je patrné rozdělení práv pro dva typy uživatelů. Na uživatele připadá jen malé množství funkcí celé aplikace. Je to z toho důvodu, že za uživatele je považována osoba, která má mít přístup jen ke svému učtu. Tento omezený přístup uživateli umožňuje prohlížet jen svá data. Tedy množství stažených a odeslaných dat ve vybraném časovém období. Díky tomu má uživatel přehled o množství přenesených dat a může tak ovlivnit přenášení dat v dalších obdobích. Poskytovatel připojení tedy může podle množství přenesených dat určovat cenu připojení a uživatel si může množství přenesených dat zkontrolovat.

Správce aplikace má neomezený přístup ke všem účtům. Aplikace mu umožňuje manipulovat se všemi účty, vytvářet statistické přehledy a provádět údržbu aplikace. Na základě těchto informací pak může změnit ceník služeb, posílit páteřní prvky sítě nebo případně omezit poskytovanou službu vybraným uživatelům.

4.3 Návrh uživatelského rozhraní

Uživatelské rozhraní jsem navrhoval s maximálním ohledem na jednoduchost, přehlednost a použitelnost. Při návrhu jsem také vycházel z požadavků ovládání alikace přes nejpoužívanější internetové prohlížeče (IE6, IE7, Firefox, Opera). Rozhraní je tedy navrženo jako stránka s šířkou 1024px a délkou přizpůsobující se obsahu stránky. Jako podkladovou barvu jsem zvolil bílou kvůli její neutralitě. Aby byl zachován standard zobrazení pro všechny stránky, vytvořil jsem základní layout. Náhled layoutu je na obrázku 4.6 Stránka je



Obrázek 4.6: Návrh layoutu

tedy rozdělena na 3 základní sekce. První sekcí je horní lišta, která zůstává nezměněna bez ohledu na vybranou funkci. V této liště jsou umístěny základní funkce aplikace oddělené od sebe modrými rámy, fungujícími jako hypertextové odkazy. Výběrem některé z funkcí dojde k načtení požadovaných dat v pravé části stránky. To je tedy druhá sekce stránky. Třetí a poslední sekcí je levé menu obsahující skupiny adres. Skupiny adres jsou však přístupné pouze uživateli s právy administrátor, uživateli s omezenými právy se v této sekci zobrazí výběr časového období. Třetí sekce je rovněž přístupná ze všech stránek aplikace.

Do první sekce jsem k ovládacím prvkům umístil vyhledávací pole. Toto vyhledávací pole umožňuje ze všech stavů aplikace pohodlně vyhledat hledaného klienta a zobrazit informace o něm v případě, že známe jeho IP adresu. Do pole se zadává IP adresa ve formátu: xxx.xxx.xxx.xxx, kde hodnoty X představují číslice. Příklad hledané IP adresy: 192.168.2.1 v případě, že IP adresa v systému není nebo má chybný formát, aplikace nahlásí chybu.

Dalším pomocným prvkem v první sekci je rolovací nabídka obsahující IP adresy všech

uživatelů. Jestliže neznáme přesně IP adresu klienta, můžeme ji nalézt pomocí tohoto pole. IP adresy jsou pro lepší orientaci seřazeny vzestupně. Výběrem některé z adres a kliknutím na tlačítko *Zobrazit* dojde k vyhledání a zobrazení informací o klientovi ve druhé sekci.

V první sekci jsem rovněž sjednotil vzhled hypertextových odkazů a tlačítek. Všechny ovládací prvky tedy mají vzhled tlačítka. Díky tomu je menu přehledné a není narušeno rozdílným vzhledem ovládacích prvků. V ostatních částech aplikace jsou již hypertextové odkazy "klasického vzhledu". U všech tlačítek a hypertextových odkazů jsem rovněž přidal pro lepší popis funkcí popisek *title*.

V třetí sekci jsou umístěny v horní části prvky *radio button*. Slouží k přepínání stavu mezi měsíčním a 24 hodinovým grafem. Tento ovládací prvek jsem zvolil pro své jednoduché a intuitivní ovládání. Tento ovládací prvek jsem rovněž použil v kategorii *TopTen*. Zde slouží jako přepínač pro seřazení dat.

Jiným použitým ovládacím prvkem je *checkbox*. Tento prvek jsem umístil do kategorie *Nová skupina*, kde umožňuje vybrat více IP adres a vytvořit tak novou skupinu. Jednolivé IP adresy jsou pro přehlednost seřazeny vzestupně a vypisovány na nový řádek. Tlačítko *Vložit* jsem umístil na konec stránky záměrně, jelikož při procházení jednotlivých IP adres dochází k rolování stránky směrem dolů.

V kategoriích *Uživatelé* a *Router* jsem použil prvek *textové pole*. Zvláště v kategorii *Uživatelé* jsem tyto prvky rozmístil do dvou sloupců a to z důvodu lepšího využití prostoru na stránce. V těchto dvou kategoriích jsem taky využil stejného rozhraní pro vkládání a editaci uživatelů, případně routerů. Například sekce uživatelé tak v sobě kombinuje několik možností práce s uživateli. Můžeme vložit nového uživatele do systému, smazat uživatele a zároveň můžeme procházet již vložené uživatele. To vše je tak přehledně vloženo v jedné kategorii a omezeno pouze na jedno tlačítko *uživatelé*. V případě, že bychom chtěli uživatele editovat, klikneme na odkaz *Prohlédnout* u konkrétního uživatele a informace o uživateli se načtou do textových polí. V tom okamžiku dojde i ke změně tlačítek a jejich funkcí. Pro přehlednost se taky zobrazí informace, se kterým uživatelem právě pracujeme. Jednotliví uživatelé jsou vypsáni v tabulce na jendotlivé řádky a seřazeni podle toho, jestli obsahují vyplněné informace. Uživatelé s vyplněnými informacemi jsou ve výpisu upřednostněni.

Při zobrazení požadovaných dat jsem zvolil dvojí výpis. Prvním je grafické znázornění v podobě grafu. Graf jsem zvolil proto, že umožňuje jednoduše odhalit minimální a maximální hodnoty, opakující se hodnoty a přináší i rychlý přehled o uživatelově aktivitě. Pro výpis dat uživatele slouží 6 grafů. První tři zobrazují celkový měsíční přehled ze sítě www a ze sítě p2p. Zbylé tři grafy zobrazují množství přenesených dat ve vybraném dnu. Opět je to celkový přenos, přenos ze sítě www a ze sítě p2p. Výběr časového období jsem umístil nad tyto grafy. Předpokládám, že pokud uživatel bude chtít zobrazit data za konkrétní měsíc, bude chtít nejdříve vidět celkový přehled ve vybraném měsíci. Jednotlivé dny ho budou zajímat až po celkovém přehledu. Proto je výběr jednotlivých dnů umístěn až po celkovém měsíčním přehledu. Po vybrání zvoleného dne navíc dojde k přesunu stránky s odkazy na dny k hornímu okraji prohlížeče. To aby nebylo potřeba ručně rolovat se stránkou a ulehčilo to tak uživateli porovnávání jednotlivých dnů. Ve spodní části stránky je pak umístěna tabulka s přesnými údaji o množství přenesených dat. Z grafu se totiž nedají odečíst přesné hodnoty, ale pouze přibližné.

Odkaz *Navštívené stránky* u výpisu konkrétního uživatele otevírá nové okno s požadovanými informacemi. Jedná se o rozšiřující informace související s množstvím přenesených dat, proto jsem zvolil nové okno. Díky tomu je tedy možné porovnávat množství navštívených serverů s množstvím přenesených dat.

4.3.1 JavaScript pro lepší uživatelské rozhraní

Pro zjednodušení práce s aplikací a pro omezení nechtěných nevratných zásahů do aplikace jsem použil JavaScript. Pomocí JavaScriptu dochází k automatickému načtení stránky při zvolení denního nebo měsíčního výpisu dat ve druhé sekci. Jiným použitím JavaSriptu jsou dotazovací okna při mazání libovolných dat z aplikace. Touto funkcí jsem tak chtěl zabránit nechtěnému smazání dat. Po stisku tlačítka *odstranit* se tedy aplikace uživatele zeptá, jestli chce danou položku skutečně odstranit.

4.3.2 Ajax pro lepší uživatelské rozhraní

Ajax jsem použil při překladu navštívených IP adres na jejich doménová jména. U některých IP adres tento překlad trvá velmi dlouho a proto by hromadný překlad všech navštívených IP adres omezoval použití aplikace. V tabulce obsahující navštívené IP adresy pak stačí najet na libovolný řádek kurzorem myši a ve volném sloupci se objeví požadovaný překlad. V některých případech však může pod jednou IP adresou být i více doménových jmen a proto při opětovném najetí kurzorem myši na daný řádek může dojít ke změně doménového jména.

4.3.3 Kaskádové styly

Kaskádové styly zahrnuji také do návrhu uživatelského rozhraní, jelikož jsou zodpovědné za zobrazení stránky korektním způsobem v jednotlivých prohlížečích. Kaskádové styly jsou umístěny v externích souborech a pro správné zobrazení v některých typech prohlížečů bylo zapotřebí vytvořit styly dva. Při načítání stránky tak dochází ke kontrole typu internetového prohlížeče a následně k vybrání správného kaskádového stylu.

4.3.4 Výpis provedených operací

Výpis provedených operací považuji za velmi užitečnou funkci aplikace. Zprostředkovává tak uživateli zpětnou vazbu po provedených operacích. Dojde-li například k vložení nového uživatele, aplikace po úspěšném vložení vypíše tuto informaci v horním menu. Tedy například: “Uživatel Test byl vložen do databáze”. Stejně tak pokud nedojde k vložení uživatele, i tuto skutečnost aplikace vypíše. Uživatel má tak kontrolu nad provedenými operacemi.

4.4 Typy grafů

Grafy v aplikaci jsou v podobě obrázků. Toto zpracování umožňuje graf uložit pro pozdější použití. Grafy v sobě integrují všechny informace pro potřebnou identifikaci uživatele i zobrazeného časového období. Při výběru typu grafu jsem vybíral z několika možností. Sloupcový graf jsem nemohl použít pro velké množství informací obsažených v grafu. Jako nejvhodnější varianta se jevil graf spojnicový. Graf umožňuje přehledně porovnat data v jednotlivých dnech, případně hodinách. Z grafu jsou též dobře patrná maxima a minima dosažených hodnot. V grafu se vyskytují dvě vertikální osy. Je to z toho důvodu, aby bylo možné porovnat množství přenesených dat a počet přenesených paketů. Celkově tedy graf zobrazuje informace o stažených datech, odeslaných datech a přenesených paketech. Každá z těchto hodnot je pro lepší čitelnost vyznačena jinou barvou.

Aby bylo možné podrobně sledovat průběh přenesených dat, zvolil jsem dva typy grafů. Jeden graf zobrazuje data v měsíčním cyklu. Časový horizont je zvolen záměrně na rozsah

jednoho měsíce, umožňuje tak porovnávat jednotlivá období v roce. Každý den má vyznačeno přesné datum, aby bylo jednoduché se v grafu orientovat. V grafu je naznačen počet dní v měsíci a hodnoty jsou postupně doplňovány tak, jak jdou jednotlivé dny po sobě. Druhým typem je graf s denní statistikou začínající vždy od 0:00. Zde se nabízela ještě možnost zobrazovat 24 hodinový graf, kde by na ose X odpovídala maximální časová hodnota aktuálnímu času a data by se zobrazovala 24 hodin zpětně. Toto řešení jsem však zavrhl jako méně potřebné.

U všech grafů je také obsažena informace o celkovém množství přenesených dat za zobrazené období. Jednotky použité v grafech jsou zvoleny s ohledem na množství přenesených dat tak, aby graf zůstával dobře čitelný.

4.5 Rozvržení kódu

Při návrhu aplikace jsem vycházel z potřeby rozdělit pro větší přehlednost a údržbu kódu zdrojové soubory do více souborů. Nejdříve jsem rozdělil soubory na dvě skupiny: přístupné uživateli a administrátorovy. Toto rozdělení umožňuje už od začátku zamezit neoprávněným osobám v přístupu k některým souborům. Dále jsem oddělil všechny funkce aplikace do souboru function.php. Pomocí tohoto rozdělení se lze snadněji orientovat v kódu a provádět potřebné aktualizace. Pro jednu funkci jsem však vyčlenil samostatný soubor. Funkce print_graph() je pro svoji rozsáhlost kódu umístěna v samostatném souboru. Jiným samostatným souborem je soubor address.php. Tento kód generuje novou stránku s navštívenými servery, proto je umístěn samostatně. JavaScriptový kód je rovněž umístěn v samostatném souboru. Jedná se o jiný skriptovací jazyk, proto jsem ho umístil do samostatného souboru.

4.6 Ochrana před neoprávněným přístupem

Pro přístup do aplikace slouží přihlašovací dialog na úvodní stránce aplikace. Pro přihlášení je potřeba znát přihlašovací jméno a heslo. Bez nich se nelze do aplikace přihlásit. Pro zvýšení bezpečnosti se hesla do databáze ukládají šifrovaně, pomocí algoritmu MD5. V této podobě nejsou téměř čitelná a nemělo by dojít k zneužití hesla. Vstupní přihlašovací pole jsou také omezena svojí maximální délkou znaků a ošetřena proti úmyslnému poškození aplikace. Celá aplikace je taky navržena tak, aby nebylo možné zneužít SQL injection. Vstupní textová pole po přihlášení již toto ošetření nemají, při návrhu jsem nepředpokládal, že by se někdo z administrátorů pokoušel aplikaci narušit.

Na ochranu jednotlivých souborů slouží kontrolní procedura na začátku všech skriptů. Tato ochrana má zabránit neoprávněnému spuštění skriptu. Není-li uživatel přihlášen nebo jeho úroveň práv neodpovídá požadované autorizaci, dojde k ukončení skriptu ještě před vykonáním jakýchkoliv jiných operací. Pomocí tohoto ošetření lze zabránit i samostatnému spouštění skriptů.

Pro zvýšení bezpečnosti je v aplikaci nastaveno automatické odhlášení uživatele. Nedojde-li po dobu delší než 40 minut k žádné aktivitě, aplikace uživatele sama odhlásí.

4.7 Jaká data jsou získávána a jak

Data pro výpočet přenesených dat se získávají z 32bitových čítačů. Každý uživatel má vyhrazeny čtyři čítače, které průběžně zaznamenávají přenos dat. Aby nedocházelo k velkému

zatížení serveru, ale data v aplikaci byla co nejvíce aktuální, zvolil jsem 5-ti minutové intervaly pro získání nových hodnot z čítačů. V čítačích jsou data uvedena v bajtech za sekundu (B/s). Pakety jsou uloženy v paketech/s. Pro lepší čitelnost jsou ve výsledném zobrazení upraveny jednotky ve 24 hodinovém grafu na Kb/s a jejich násobky. Graf tak zobrazuje průtok dat. Pro měsíční graf jsou jednotky upraveny na KB/den a jejich násobky. Z grafu je tak možné odečíst množství přenesených dat za den.

4.8 Ošetření proti přetečení

Aby byla prezentovaná data co nejvíce přesná, bylo zapotřebí vyřešit nulování čítačů po dosažení jejich maximálních hodnot. Pro dopočítání skutečné hodnoty po vynulování vycházím z předpokladu, že čítače jsou 32bitové. Maximální hodnota čítače je tedy 4 294 967 296. Pro výpočet nové hodnoty stažených dat vždy vycházím z předchozí hodnoty čítače a aktuální hodnoty čítače. Díky tomu lze dopočítat skutečnou hodnotu přenesených dat a to následujícím způsobem: množství přenesených dat = 4 294 967 296 - předchozí hodnota čítače + nová hodnota čítače. Tento postup lze však uplatnit pouze tehdy, pokud skutečně dojde k vynulování čítače.

V případě, že dojde k jakékoliv jiné nepředvídatelné chybě nebo číselným hodnotám mimo reálný rozsah, jsou nové hodnoty nahrazovány nulami.

Kapitola 5

Implementace aplikace

Po návrhu aplikace jsem přešel k samotné implementaci. Nejprve bylo nutné zajistit potřebné hardwarové a softwarové vybavení. Toto vybavení zprovoznit a nainstalovat na něj potřebný software. Poté jsem přešel k samotné implementaci aplikace. Jako první jsem vytvořil skript pro založení databáze a databázových tabulek. Tento skript je umístěn v souboru statistika2.sql. Skript umožňuje v případě havárie nebo přesunu serveru vytvořit celou databázi bez dat. Pro první přihlášení je vytvořen administrátorský účet s těmito údaji:

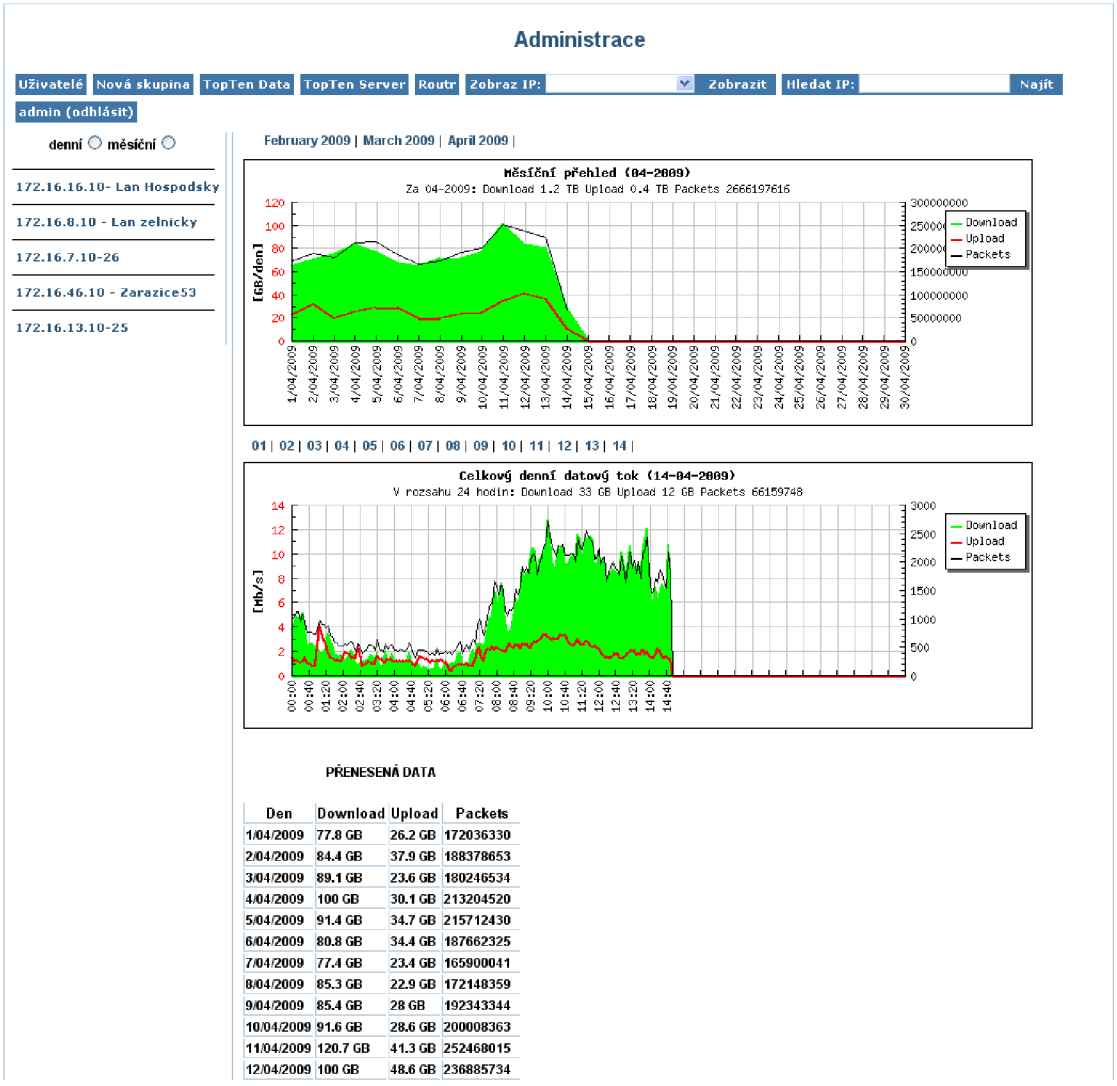
- Uživatel: admin
- Heslo: admin

Pro aplikaci jsem zvolil výchozí adresář var/www/html/. V tomto adresáři jsem vytvořil dvě složky: traffic a statistika. Pro obě složky bylo zapotřebí nastavit práva na 777. Do složky traffic jsem umístil skript, který bude spouštěn automaticky pomocí daemona Cron. Dalším krokem bylo tedy vytvořit skript, který by stahoval data z čítačů a prováděl aktualizaci dat v databázi. Tento skript jsem pojmenoval traffic.php a umístil ho do složky traffic. Jelikož se bude skript spouštět pomocí daemona, bylo zapotřebí také upravit nastavení práv. Poté jsem v daemonu nastavil automatické spouštění skriptu na každých 5 minut. Ze skriptu traffic.php je na konci jeho provádění navíc volán skript visited.php, který provádí v databázi aktualizaci navštívených serverů. Funkci provádějící aktualizaci navštívených serverů jsem umístil do zvláštního souboru úmyslně. Tato funkce prošla při návrhu aplikace několika etapami aktualizací a to zejména kvůli zvýšení efektivity provádění aktualizace dat.

Funkce visited() postupuje při aktualizaci tímto způsobem: ze souboru provoz.log umístěného v adresáři var/www/html/statistika/bakalarka/log zjišťuje navštívené servery. Jelikož je množství těchto informací příliš velké (řádově se velikost souboru po 24 hodinách provozu pohybuje ve 100MB), dochází každý den k archivaci souboru provoz.log a k vymazání souboru provoz.log. Aby tedy čtení informací ze souboru bylo co nejméně náročné na paměť a procesor, pamatuje si aplikace poslední polohu aktualizovaných dat v souboru. V praxi to tedy vypadá tak, že se ze souboru provoz.log vždy přečtou jen data přidaná v posledních 5-ti minutách. Pokud však dojde k výpadku spuštění skriptu, přečte funkce všechny zatím nevložené záznamy. Dalším testováním této funkce jsem přišel na zajímavou skutečnost. Tou bylo opakování stejných požadavků pro aktualizaci stejných dat v databázi. Abych omezil počet opakujících se dotazů, upravil jsem funkci na víceprůchodovou. Díky této úpravě došlo k omezení počtu zápisů do databáze a ke zvýšení výkonu prováděné funkce.

Dalším krokem bylo vytvoření uživatelského rozhraní. Soubory tvořící uživatelské rozhraní a funkce použité v aplikaci jsou umístěny v adresáři var/www/html/statistika/bakalarka/.

Náhled úvodní stránky pro uživatele s právy administrátor je na obrázku 5.1 Více náhledů



Obrázek 5.1: Úvodní stránka uživatele s právy administrátor

na aplikaci je přiloženo v příloze 1.

Po vytvoření uživatelského rozhraní bylo zapotřebí provést testovací spuštění. Toto období trvalo jeden měsíc. Během této doby došlo na základě testování k drobným změnám aplikace. Tyto změny pomohly k lepší funkci aplikace a odstranily některé nedostatky vzniklé při implementaci.

Po skončení testovacího období byla aplikace spuštěna do ostrého provozu. Na základě toho bylo vytvořeno více uživatelských účtů.

Aplikace je tedy uložena na adrese: `http://88.146.210.13/statistika/bakalarka/`. Přístupové jméno a heslo pro přístup do aplikace pro uživatele s právy administrátor:

- Uživatel: adminbp
- Heslo: 321546

Přístupové jméno a heslo pro přístup do aplikace pro uživatele s právy uživatel:

- Uživatel: userbp
- Heslo: 321547

Pro ověření korektnosti zobrazení v internetových prohlížečích posloužila vizuální kontrola a internetové stránky validátoru W3C, kterými aplikace prošla. To by mělo zaručit korektní zobrazení aplikace i v připravovaných verzích prohlížečů. Bylo zapotřebí také ověřit přesnost aplikace v množství přenesených dat. I těmito testy aplikace prošla.

Kapitola 6

Závěr

Seznámil jsem se s technologiemi pro vývoj webových aplikací a informačních systémů. Zjistil jsem požadavky na systém statistik stažených a odeslaných dat v lokální síti pro společnost VnorovyNET. Dále jsem navrhnul aplikaci umožňující registraci uživatelů sítě zaznamenávající a prezentující množství přenesených dat v obou směrech v různých časových intervalech včetně grafického zpracování, vyhledávání a řazení těchto dat. Dále pak ukládání a zobrazení seznamu navštívených stránek pro každého uživatele včetně souhrnných statistik, dále rozlišení typů přenesených dat. Na základě informací o přenesených datech jsem navrhnul systém statistik přenesených dat v různém časovém horizontu. Navržený systém jsem realizoval a otestoval. V případě statistických údajů jsem vytvořil výstupy pomocí tabulek i grafů. Nakonec jsem zhodnotil dosažené výsledky a nastínil další možné pokračování tohoto projektu.

Všechny body zadání byly splněny a jejich řešení je popsáno v této práci. V průběhu testování aplikace vyšlo najevo, že zvolená hardwarová konfigurace obsluhující běh databázového systému je málo výkonná. Aplikace by měla pracovat na serverovém systému s rychlým diskovým polem. Stejně tak i zvolená databáze MySQL by měla být nahrazena výkonější databází z produkce firmy Oracle. Tyto změny by pomohly podstatným způsobem zrychlit běh aplikace.

Oproti původním požadavkům byla aplikace rozšířena o rozdělení přenesených dat ze sítí www a p2p. Díky tomuto rozšíření lze lépe nastavovat omezení tak, aby byl zachován plynulý provoz sítě. Dalším rozšířením je informace použitého portu u navštíveného serveru.

Jako další možné pokračování projektu by mohlo být rozšíření aplikace o účetní systém, který by fakturoval částky uživatelům sítě podle množství přenesených dat. Případně by systém mohl vést evidenci o provedených platbách a umožnit uživateli nahlédnout do jednotlivých faktur.

Literatura

[1] Burget, R.: *Učební texty předmětu ITW*. FIT-VUT v Brně.

[2] Petr, B.: *Tvorba WWW stránek pro úplné začátečníky*. Computer Press, 2001. ISBN 80-7226-423-0

[3] Wikipedie, otevřená encyklopedie. Dostupná na URL `http://cs.wikipedia.org/wiki/HyperText_Markup_Language` (březen 2009)

[4] Domovská stránka W3C. Dostupná na URL `http://www.w3c.org/` (březen 2009)

[5] PHP Documentation: PHP Manual. Dostupné na URL `http://www.php.net/docs.php/` (březen 2009)

[6] JavaServer Pages: Vývoj aplikací. Dostupné na URL `http://interval.cz/clanky/javaserver-pages-pro-vsechny/` (březen 2009)

[7] Slavoj, P.: *JavaScript* efektivní nástroj oživení WWW stránek. Grada Publishing, 2001. ISBN 80-247-0014-X

[8] Ryan, A. - Nathaniel, T.S.: *AJAX*. Vytváříme vysoce interaktivní webové aplikace. Computer Press, 2006. ISBN 80-251-1285-3

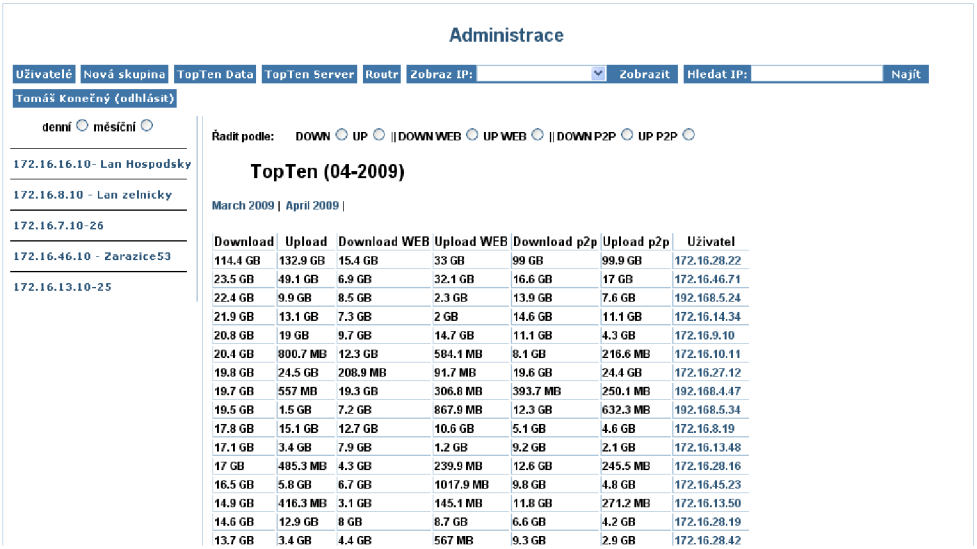
[9] Luboslav, L.: *SQL hotová řešení*. Computer Press, 2003. ISBN 80-7226-975-5

Příloha A

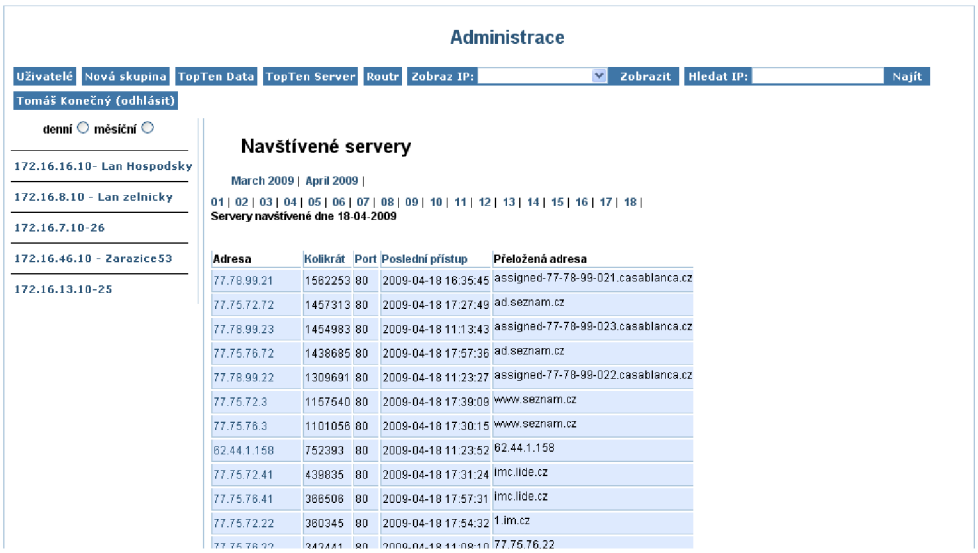
Seznam příloh

Příloha 1. Ukázky uživatelského rozhraní
Příloha 2. Seznam skriptů použitých v aplikaci
Příloha 3. CD s kompletními zdrojovými soubory

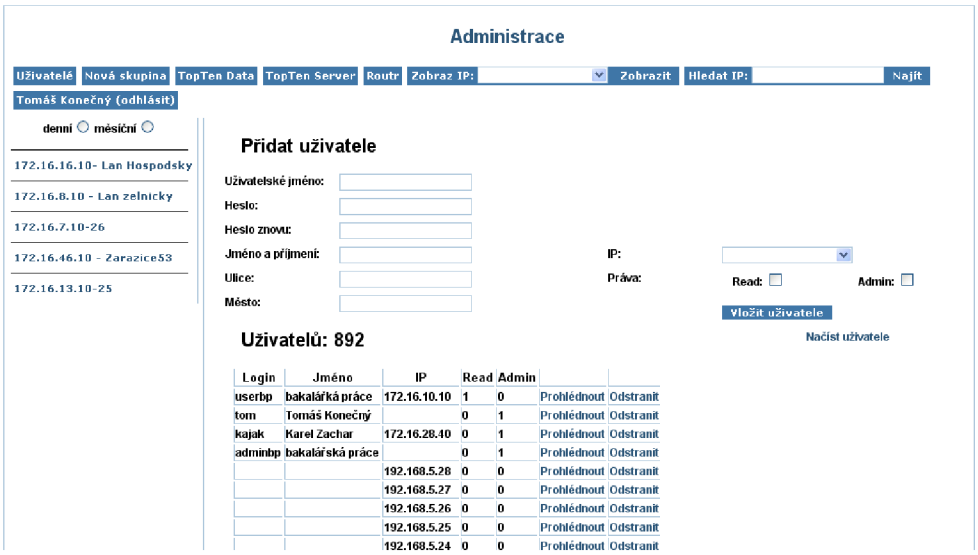
Ukázky uživatelského rozhraní



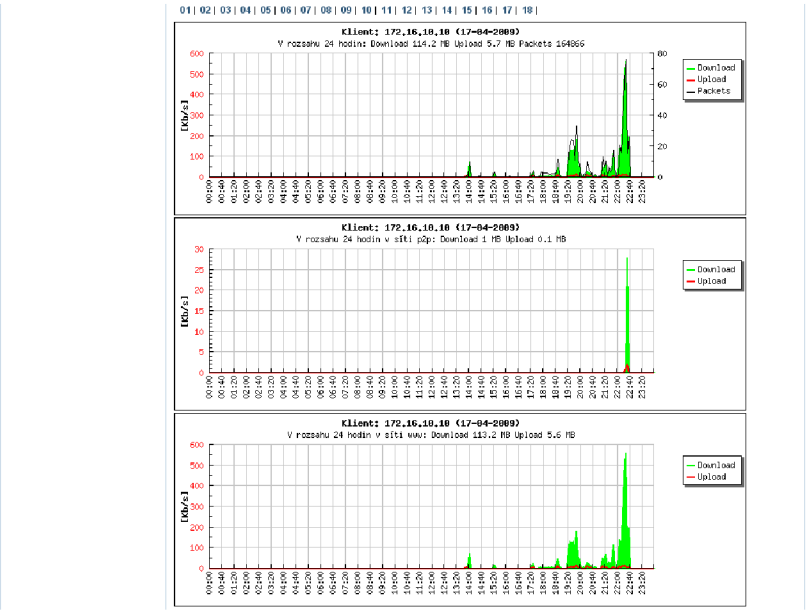
Obrázek A.1: Ukázka aplikace - sekce TopTen



Obrázek A.2: Ukázka aplikace - sekce Navštívené servery



Obrázek A.3: Ukázka aplikace - sekce Uživatelé



Obrázek A.4: Ukázka aplikace - sekce Uživatel - detail uživatele

Seznam skriptů použitých v aplikaci

index.php - vytváří přihlašovací dialog
admin.php - vytváří uživatelské rozhraní pro uživatele s právy administrátor
user.php - vytváří uživatelské rozhraní pro uživatele s právy uživatel
function.php - obsahuje funkce dostupné v aplikaci
print_graph.php - tiskne graf se zvolenými parametry
address.php - vytváří uživatelské rozhraní pro výpis navštívených serverů
compute.php - pomocný skript pro Ajax
script.js - umožňuje dynamický překlad IP adres
styl.css - soubor definující styl aplikace
styl-ie.css - soubor definující styl aplikace
styl2.css - pomocný soubor definující styl
statistika2.sql - skript pro vytvoření databáze
src - složka knihovny JpGrapf
traffic
- traffic.php - skript provádějící aktualizaci dat v databázi
- visited.php - skript provádějící aktualizaci dat v databázi
log
- provoz.log - soubor obsahující navštívené servery