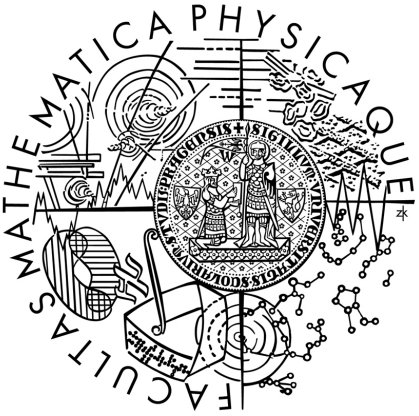
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Horký

Monitor síťové aktivity pro větší datové toky

Středisko informatické sítě a laboratoří

Vedoucí bakalářské práce: RNDr. Libor Forst
Studijní program: Informatika, správa počítačových systémů

2008

Na tomto místě bych chtěl poděkovat vedoucímu práce RNDr. Liborovi Forstovi za jeho cenné připomínky k textu práce. Mé poděkování také patří Mgr. Jiřímu Caldovi a Danu Lukešovi za umožnění testování aplikace v budově MFF na Malé Straně a na kolejích UK v Troji. Slova díky si zaslouží také má přítelkyně Bc. Magdaléna Čípová za podporu a pomoc se stylizací a úpravou textu.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 29. května 2008 Jiří Horký

Obsah

Název práce: Monitor síťové aktivity
Autor: Jiří Horký
Katedra (ústav): Katedra softwarového inženýství
Vedoucí bakalářské práce: RNDr. Libor Forst
e-mail vedoucího: libor.forst@mff.cuni.cz

Abstrakt: Cílem práce je návrh a implementace aplikace pro sledování síťového provozu v reálném čase zaměřená na větší toky dat. Aplikace bude určena pro správce sítě, kteří nemají možnost či nechtějí nasadit řešení monitoringu přímo v aktivních prvcích sítě. Snahou této práce je minimalizovat množství systémových prostředků potřebných k běhu aplikace - aplikace by měla být nasaditelná na běžný serverový hardware. Aplikace bude umožňovat konfiguraci množství zachytávaných dat (rozlišování portů, MAC adres, nastavení agregace) a tak snižovat objem statistických dat. Nedílnou součástí práce bude i grafické rozhraní programu umožňující efektivní zobrazení zachycených statistik ve formě uživatelem definovanových grafů a tabulek. Aplikace bude naprogramována v jazyce C a bude určena pro UNIXové systémy

Klíčová slova: monitoring, síť, paket, zachytávání

Title: Network activity monitor
Author: Jiří Horký
Department: Department of Software Engineering
Supervisor: RNDr. Libor Forst
Supervisor's e-mail address: libor.forst@mff.cuni.cz

Abstract: Aim of this work is to design and implement real-time network monitoring application for higher traffic. Application will be intended for network administrators, who can't or don't want to monitor network directly, using active network equipment such as routers or switches. One of main effort would be to minimalize system resource requirements - application should be usable on common server hardware. Application will allow to configure amount of captured data (ports, MAC addresses) and thus lower the statistic data size. Integral part of this work will be graphical frontend enabling to effectively observe captured data and statistics in the form of user-defined charts and tables. Application will be programmed in C language and it will be aimed to UNIX systems.
Keywords: monitoring, network, packet, capture

Cíl práce

Cílem práce je návrh a implementace aplikace pro sledování síťového provozu se zaměřením na větší toky dat. Základním požadavkem je navrhnout aplikaci, která bude nasaditelná na běžném hardware, tedy bez nutnosti speciálních síťových karet s hardwarovou podporou či super výkonných procesorů. Je zřejmé, že tento požadavek klade i určité omezení v množství datových toků, které by měla aplikace zvládat. Rozumným odhadem se zdají být datové toky okolo 1 Gbit. Z výše uvedeného důvodu také není úkolem práce detailní zkoumání paketů, tj. inspekce na úrovni aplikační vrstvy.

Zároveň aplikace musí umožňovat vizualizovat zachycená data ve formě uživatelem definovaných grafů a tabulek. Uživatel si tedy bude moci zvolit různá omezení a časové období tak, aby mohl sledovat právě jen ty údaje, které ho zajímají. To vše bude možné již z jednou zachycených dat. Nebude tedy nutné nejprve definovat statistiky, které konkrétního uživatele zajímají, a až poté začít monitorovat. I to však bude možné, čímž lze efektivně minimalizovat množství zachytávaných dat.

Vzhledem k předpokládanému obrovskému množství zachycených dat je úkolem práce také navrhnout kompromis mezi objemem statistických dat, stupňem požadadových informací a rychlostí vlastního vyhodnocování.

Součástí práce je i ověření nasaditelnosti této aplikace v reálném prostředí a úvod do problematiky monitorování sítě.

Kapitola 1

Monitorování sítí

Klíčem pro úspěšné fungování každé organizace je spolehlivá a dobře zabezpečená počítačová síť. Každý výpadek, snížená dostupnost síťových aplikací či zvýšení odezvy má za následek nespokojenost klientů, ztrátu potenciálních zákazníků a v neposlední řadě také ztrátu finanční.

S pomocí pravidelného sledování sítě lze výše uvedeným následkům a situacím úspěšně předcházet. V závislosti na použitém monitorovacím mechanismu lze sledovat zátěž jednotlivých částí sítí i jednotlivých klientů a mít tak reálnou představu o využití sítě, zahlcení jednotlivých linek, existenci kritických míst v síti[1] apod. Monitoring tedy pomáhá správcům sítě včas odhalit příčinu problému, zjišťovat původce přetížení linek či zpětně analyzovat případné problémy.

Následující seznam shrnuje výše popsané důvody a přidává další, neméně podstatné:

- Prevence problémů v síti (a hlavně případných následků)
- Efektivní dohlížení na síť a zjednodušení správy sítě pro administrátory
- Zvýšení bezpečnosti sítě odhalením vnějších i vnitřních útoků
- Určení kritických míst v síti za účelem optimalizace její infrastruktury
- Sledování dostupnosti a využití služeb
- Sledování aktivity jednotlivých uživatelů
- Účtování služeb na základě přenesených dat
- Dodržování vyhlášky č. 485/2005[1] [2] o elektronické komunikaci[1]

[1] spíše známých jako bottlenecků

[2] Dne 23. 4. 2008 byl poslaneckou sněmovnou schválen pozměňovací návrh zákona č. 127/2005 Sb. o elektronických komunikacích[2]. Pokud tento návrh schválí senát a

Je zřejmé, že k naplnění předchozích bodů je třeba různá hloubka zkoumání datových toků. Například ke zjištění celkového využití sítě stačí zkoumat celkový počet přenesených dat a paketů v čase. K účtování již bude nutné umět rozlišit jednotlivé uživatele (IP adresy), k sledování využití služeb pak i jednotlivé porty. Ne každý program a způsob monitorování sítě dokáže splnit všechny tyto body.

Monitoring sítě lze rozdělit dle způsobu, jakým je prováděn, na tři stěžejní kategorie.

1.1 Aktivní monitoring

Aktivní monitorování sítě spočívá v generování paketů na jednom místě v síti a jejich zachytávání na místě druhém. Můžeme takto měřit například zpoždění při průchodu sítí, ztrátovost nebo celkovou dosaženou propustnost. Tento provoz je však umělý, od skutečného provozu uživatelů se zpravidla výrazně liší. Další nevýhodou je možné ovlivnění samotného provozu uživatelů.

Mezi typické aplikace v této kategorii patří především: *ping*, *traceroute*, které jsou součástí snad každého operačního systému. Na těchto programech jsou pak postaveny celé projekty jako jsou *pingplotter*[3], *NeoTrace*[4], *traceping*, a mnoho dalších [3].

1.2 Pasivní monitoring

Při pasivním monitorování se do sítě neposílají žádné pakety, pakety jsou pouze zachytávány. Tento přístup má výhodu v naprosté transparentnosti pro uživatele sítě a v reálnosti monitorovaných dat. Pomocí této metody většinou probíhá měření objemových a časových parametrů provozu, které můžou být případně vztažené k jednotlivým uživatelům. Monitování zpravidla zahrnuje:

1. Zachycení celého paketu
2. Načtení hlavičky paketu
3. Uložení dat z hlavičky do databáze pro pozdější zpracování

prezident ČR, bude mj. možnost postihnout ISP sankcí až 10 mil. Kč při nedodržování vyhlášky [1].

[3]Rozsáhlý seznam aplikací zabývající se aktivním monitoringem je k nalezení na http://www.caida.org/tools/taxonomy/measurement/

Existuje nespočetné množství programů a utilit pro pasivní monitorování. Vzhledem k zaměření této práce právě na pasivní monitoring, je několik významných programů blíže popsáno v sekci 1.4.

1.3 SNMP-like monitorování

Zvláštní postavení mezi monitory sítě má protokol SNMP[4] . Jde o určitý kompromis mezi pasivním a aktivním monitorováním. Jedná se o aplikační protokol založený na modelu klient/server. Statistická data se zpracovávají na serveru, v SNMP terminologii označovanému jako *SNMP agent*. Ten monitoruje stav zařízení, na kterém běží, a ukládá aktuální sledované hodnoty do MIB[5] (pasivní část). Tato data jsou do MIB zapisována periodicky, čímž dojde k přepsání starých hodnot. Server je pak na tato data dotazován klientem, neboli tzv. *network managerem* (aktivní část).

Sledované hodnoty mají většinou aktuální (vytížení CPU, použitá paměť, uptime), či kumulativní charakter (počet příchozích, odchozích či chybových paketů). Z logiky ukládání dat do MIB je zřejmé, že k vytvoření časových statistik je nutné se periodicky dotazovat SNMP agentů na zkoumaná data a tato data zanášet do interní databáze či rovnou graficky zpracovávat. Tento proces se nazývá *SNMP polling*.

Jednou z hlavních výhod použití protokolu SNMP je jeho značná rozšířenost, v dnešní době je implementován téměř na každém lepším síťovém zařízení. To však nese i nevýhodu v podobě různorodosti rozšíření MIB různými výrobci. Mezi další často diskutované nevýhody patří bez-pečnost (viz [6]), přidaná zátěž monitorovaným zařízením a z hlediska monitorování sítě pak malá škálovatelnost sledování, tj. nemožnost monitování jednotlivých uživatelů apod.

Existuje celá řada programů využívající protokol SNMP. Pominu-li programy jednotlivých výrobců síťových zařízení, které jsou zpravidla kompatibilní jen s danými prvky, a zaměřím se na volně dostupné programy používané právě k monitorování sítě, je nutné mj. zmínit programy *Cacti*[8], *MRTG*[8] či projekt *net-snmp*[9].

[4]Simple Network Management Protocol, více informací k nalezení na [5]

[5]Management Information Base. Jedná se o databázi, která definuje, jaká data daný agent udržuje a jaké operace jsou nad těmito daty povoleny. Základní podoba MIB je definována standardem, jednotliví výrobci zařízení ji však zpravidla rozšiřují. Viz [5]

1.4 Existující řešení pro pasivní monitorování

1.4.1 NetFlow

Netflow je otevřený protokol pro sledování sítě vytvořený firmou Cisco Systems, Inc.[10]. Zařízení, na kterých je povolen (obvykle switche a routery), generují statistiky pro každý *síťový proud*. Ten je definován sedmi unikátními klíči:

- Zdrojovou IP adresou
- Cílovou IP adresou
- Zdrojovým portem
- Cílovým portem
- Použitým protokolem na síťové vrstvě (IP)
- TOS[6]
- Vstupním rozhraním

O každém proudu pak existují typicky[7] následující informace:

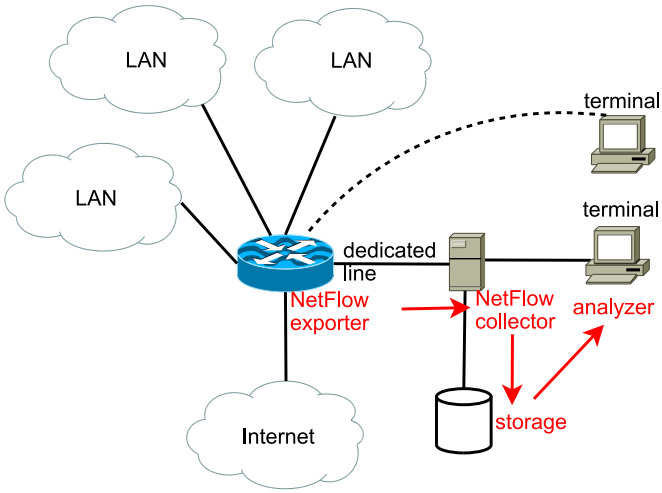
- Zdrojová a cílová IP adresa
- Zdrojový a cílový TCP/UDP port
- Počet přenesených dat a paketů
- TOS
- Čas začátku a konec toku
- Čísla vstupních a výstupních síťových rozhraních
- Flagy protokolu TCP
- Routovací informace (next-hop adresa, číslo autonomního systému zdroje a cíle, síťové masky zdroje a cíle)

Právě tato data si žádá i vyhláška o elektronických komunikacích[1].

Statistiky o jednotlivých tocích jsou poté exportovány do *netflow kolektorů*, z nichž jsou pak používány tzv. *analyzátory*. Netflow analyzátor i kolektor může být umístěn na stejném serveru. Jak lze vidět ze záznamu pro každý tok, NetFlow nám dovolí použít monitorování pro všechny body uvedené v kapitole 1.

[6] Type Of Service, nepříliš často využívaný parametr IP hlavičky

[7] Kompletní seznam pro jednotlivé verze je k nalezení v dokumentaci NetFlow, viz [10][11][12][13]



Obrázek 1.1: **Architektura Netflow.** Obrázek byl získán z [11].

Vzhledem k zatížení na monitorovacím zařízení se místo zpracovávání všech paketů často využívá tzv. vzorkované zachytávání, tj. zkoumání každého n-tého paketu.

Protokol NetFlow nemusí být nutně provozován pouze na aktivních síťových prvcích. Existuje i možnost nasadit tzv. neflow sondy (servery pasivně zachytávající veškerý provoz), které generují data místo routerů či switchů (viz [14]).

1.4.2 nTop

Program nTop[22] patří asi k nejznámějším open-source pasivním monitorům síťě. Jedná se o multiplatformní program, který pomocí různých pluginů nabízí rozšíření funkcí aplikace. Program sám o sobě obsahuje webový server a využívá databázovou knihovnu `gdbm`. Ke svému chodu tedy nepotřebuje aktivně spuštěnou žádnou jinou aplikaci.

Webové rozhraní programu umožňuje prohlížení provozu dle několika kritérií. V závislosti na nastavení program umožňuje sledovat aktuální, průměrný a celkový provoz na zvolených ip adresách, portech a protokolech spolu s poměrně detailními informacemi (operační systém strojů, hop-count, průměrná velikost paketů, atd). Dále mj. umožňuje využití NetFlow, či SNMP protokolu. Naopak bez jiných rozšíření neumožňuje generování grafů celkové zátěže, jednotlivých strojů či aplikací v čase (v základním nastavení podporuje jen celkové grafy). Ty je možné za-

pnout pomocí podpory RRD[8] grafů, je však nutné grafy, které se budou generovat, definovat předem. Nelze tedy z již jednou zachycených dat generovat nové grafy.

Během testování programu v provozu na Malé Straně (viz kapitola 4.4), bylo zatížení systému při použití nTop se zapnutou podporou RRD grafů cca 8-10x větší než s aplikací, která je předmětem této práce. Kvůli této zátěži již docházelo k zahazování paketů. Na druhou stranu je však nutno poznamenat, že vzhledem ke způsobu generování grafů, bylo jejich zobrazovaní okamžité. Navíc program nTop poskytoval mnohem detailnější informace provozu.

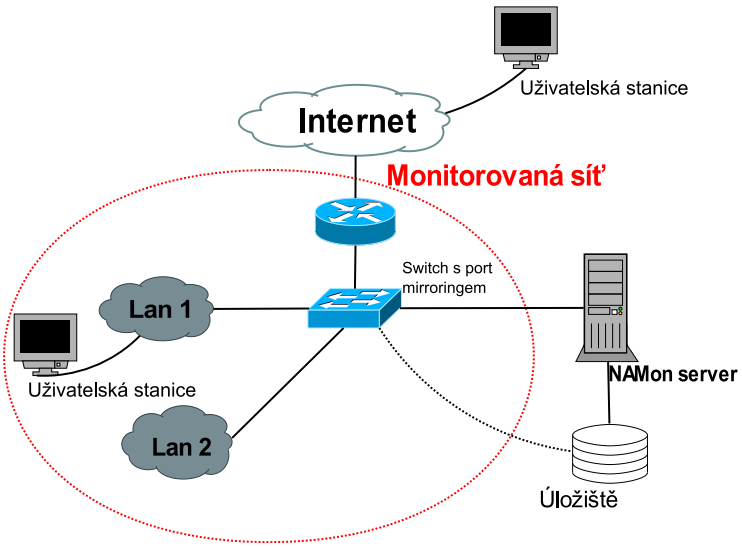
[8]Round Robin Database, populární způsob vykreslování grafů zátěže apod. Viz [25]

Kapitola 2

NAMon - Network Activity Monitor

Hlavní částí této práce je návrh a implementace vlastního programu, nazvaného NAMon. Detailnější informace o samotné implementaci jsou k nalezení v programátorské dokumentaci(viz příloha B), informace o používání aplikace pak v dokumentaci uživatelské (viz příloha A).

2.1 Model aplikace



Obrázek 2.1: **Síťový model programu NAMon.**

Síťový model aplikace zobrazuje obrázek 2.1. Monitorovací server je připojen k switchi umístěnému v dané, monitorované síti. Na switchi (lze použít také hub) je poté nastaven port mirroring[1]. V závislosti na nastavení switche lze monitorovat jen určité segmenty sítě. Tyto pakety jsou zpracovány monitorovacím serverem, který své záznamy ukládá na server, jež slouží jako úložiště. Klientské stanice, které mohou být umístěny kdekoliv, se pak dotazují serveru s daty na jednotlivé statistiky provozu. Storage server může být provozován i na stejném stroji, kde probíhá monitorování, tím se však zvýší jeho zatížení.

2.2 Návrh aplikace

Během návrhu aplikace bylo třeba provést několik zásadních rozhodnutí ovlivňujících chod programu. Jednalo se především o:

- Způsob zachytání paketů
- Způsob uchovávání zachycených dat
- Způsob agregace a cachování paketů v aplikaci
- Volba prostředí klientské aplikace

2.2.1 Zachytávání

Způsob a tedy i rychlost zachytávání paketů je pro běh aplikace zcela zásadní. Existují v podstatě dvě cesty, kterými se dá ubírat. První je napsaní vlastního low-level rozhraní k zachytávání - tedy mít plnou kontrolu nad pakety, což dává prostor k různým optimalizacím ušitým na míru programu apod. Druhou možností je použití již existujícího API, které k zachytávání slouží. Výhodou je ulehčení práce, ověřená funkčnost a v neposlední řadě také přenositelnost kódu mezi různými Unixovými systémy.

Tato cesta byla nakonec zvolena. Zachytávání dat probíhá pomocí knihovny `libpcap`. Jedná se výsledek práce autorů programu tcpdump[15]. Tato knihovna představuje osvědčené a přenositelné[2] API sloužící k zachytávání paketů. Tuto knihovnu mj. využívají programy jako *ntop*[22], *tcpdump*, *ethereal* apod. Knihovna rovněž umožňuje poměrně komplexní a efektivní filtrování vstupních paketů.

[1] Technologie používaná na switchích. Switch posílá kopii veškerých paketů z daných fyzických portů na jeden další port, kde je zpravidla připojen monitor sítě.

[2] Existuje i verze pro Windows nazvaná Winpcap. Autor se však přenositelností na tuto platformu nezabýval.

2.2.2 Způsob uchovávání zachycených dat

Při rozhodování o způsobu ukládání dat bylo nutné vzít v potaz následující body:

Rychlost ukládání - ukládání musí být dostatečně rychlé, aby zvládalo ukládání v řádu deseti tisíců paketů za vteřinu.

Rychlost dotazování - je třeba mít na paměti, že z těchto dat se dále vytvářejí statistiky. Je proto nutné vybrat takovou reprezentaci, nad kterou je přirozené a rychlé dotazování.

Vzdálený přístup k datům - je nutné, aby se k datům dalo přistupovat vzdáleně, tedy ne jen ze stanice, kde jsou fyzicky uložené (která typicky nebude na zcela přístupném místě - např. v serverovně).

Vícenásobný přístup - k datům může v jeden okamžik přistupovat více uživatelů vzhledem k požadavku statistik v reálném čase. Kupříkladu bude monitorovací server ukládat data, na která se ve stejné době bude klient chtít dotazovat. Je tedy nutné zajistit konzistenci i v tomto případě.

Dostupnost - nejedná se o nejdůležitější bod, ale příjemnou vlastností ukládání by mohla být dostupnost k dalšímu zpracovávání dat externími programy. V praxi by to znamenalo nevytváření vlastního formátu datových souborů, ale používání již známého a osvědčeného řešení.

Vzhledem k výše uvedeným bodům bylo rohodnuto použít databázový server. Ten umožní skladování velkého množství zachycených dat a efektivní a rychlé dotazování. Má také prostředky k řízení vícenásobného přístupu. Lze jej provozovat na monitorovacím i na vlastním stroji. Jako doplňkovou službu dostáváme také možnost replikace dat či automatického zálohování.

Hlavní otázkou však zůstává rychlost. Zřejmě nebude možné ukládat informace o každém paketu do vlastního záznamu (průměrný provoz 17 Mbit generoval během testovacího provozu cca 50 milionů paketů za hodinu), ale bude nutné tyto informace vhodným způsobem agregovat a cachovat přímo v programu, čímž se několikanásobně sníží objem dat a tedy i počet vkládání do databáze. Testovací provoz ukázal (viz kapitola 4), že takto zvolený způsob ukládání je dostatečně rychlý.

V současnosti je na trhu spousta různých databázových serverů. Svůj produkt nabízí velké společnosti jako Oracle, IBM nebo Microsoft, ale k dispozici je i celá řada dalších produktů, které jsou určené pro volné použití – MySQL, PostgreSQL, Firebird a další. Vzhledem k cílové platformě Unix a snaze vyhnout se závislosti na komerčních produktech bylo

zvoleno MySQL[3].

Samotná databáze je strukturovaná do tabulek dle jednotlivých dnů. To má za výhodu zmenšení počtu záznamů v tabulce, čili zrychlení jednotlivých dotazů do databáze. V případě dotazu na delší časové období než jeden den není problém příslušné tabulky spojit.

Každá tabulka obsahuje následující sloupce:

- Zdrojová a cílová IP adresa
- Zdrojová a cílová MAC adres (volitelně)
- Zdrojový a cílový port (volitelně)
- Transportní protokol
- Počet paketů
- Počet přenesených dat
- Začátek časového intervalu s přesností na sekundy
- Hodina při začátku časového intervalu[4],

přičemž primárním klíčem jsou položky zdrojová a cílová IP a MAC (volitelně) adresy, porty (volitelně), protokol a časový interval.

Tento formát dovoluje i aplikacím třetích stran poměrně jednoduchý přístup k zachyceným informacím.

2.2.3 Agregace a cachování paketů v aplikaci

Jak již bylo zmíněno výše, obrovské množství paketů si vyžaduje určité shlukování a následné cachování podobných paketů před jejich uložením do databáze. *Podobné* pakety jsou, v závislosti na nastavení (viz část 2.4), takové pakety, kde se shodují následující položky(klíče):

- Zdrojová a cílová MAC adresa (volitelně)
- Zdrojová a cílová IP adresa
- Zdrojový a cílový port (volitelně)
- Použitý transportní prokol (podporovány jsou TCP, UDP a ICMP)
- Časový interval

[3]Neměl by však být problém použít libovolný jiný produkt.

[4]Tato hodnota se dá získat i z předchozího sloupce, v praxi se ale ukázalo, že je poté dotazování na konkrétní hodinu až 5x pomalejší.

Časový interval popisuje nejmenší rozlišitelnou jednotku času během zachytávání. Během jednoho intervalu jsou všechny *podobné* pakety shlukovány k sobě, čímž dochází k redukci objemu dat, ale i k určité ztrátě přesnosti měření. Je-li například tento interval nastaven na 5 minut, nelze se dotazovat na toky během první či poslední minuty, ale jen na celý tento interval. Výsledná hodnota je tedy zprůměrovaná hodnota během časového intervalu. Tento interval je plně konfigurovatelný, při krátkých intervalech je však nutné počítat s velkým nárustem objemu dat. Vliv této hodnoty na přesnost měření a velikost výsledné databáze byl předmětem zkoumání reálného provozu v kapitole 4.

Provádění této agregace je možno nechat přímo na databázovém serveru (pomocí konstrukce ON DUPLICATE KEY UPDATE), což znamená dotaz do databáze pro každý paket. Druhou možností je před samotným vložením do databáze pakety v programu cachovat. Vzhledem k zaměření na vysoké toky a tedy k potenciálnímu obrovskému množství dotazů byla zvolena druhá možnost[5].

Cachování znamená nutnost ukládat informace o paketech ve vhodné datové struktuře tak, aby bylo umožněno rychlé vyhledávání potenciálního *podobného* paketu a změna počtu přenesených paketů a velikosti dat k takovému záznamu. Z této struktury by mělo jít také efektivně mazat, k čemuž dochází v případě vkládání do samotné databáze. Převládat však jistě bude vyhledávání a následná změna záznamu nad vkládáním nového či uvolňováním starého záznamu[6].

Je dobré si uvědomit, že každé vložení nového záznamu do této datové struktury znamená vlastně samostatný dotaz do databáze. Úspěšností této cache, tzv. cache-hitu se zabývá sekce 4.3 o reálném provozu.

Z těchto hledisek se jako nejvýhodnější jeví použití hashovací tabulky. Ta umožní (při vhodné velikosti) vyhledávání, vkládání i mazání v konstantním čase (amortizovaně). Jako jediný závažný problém se jeví nalezení optimální velikosti hashovací tabulky. Ta ale může být na základě pozorování provozu v dané síti poměrně efektivně odhadnuta. Druhou možností je velikost hashovací tabulky dynamicky měnit za cenu jisté ztráty výkonu při její změně, ale s minimálním zaručeným výkonem při vkládání.

[5]Ve skutečnosti jsou data do databáze i v případě cachování vkládány pomocí konstrukce ON DUPLICATE KEY. Těchto dotazů je však s použitím cachování podstatně méně.

[6]Obvyklé posílání dat mezi dvěmi stanicemi probíhá na stejných portech, generuje tudíž *podobné* pakety.

V aplikaci byla použita kombinace obou přístupů. Na základě pozorování během testovacího provozu je nastavena minimální velikost tabulky s poměrně velkou rezervou. Ta však může být za chodu dle konkrétní situace zvětšována (detaily viz příloha B).

K tomuto byla použita částečně pozměněná implementace hashovací tabulky z knihovny *GLiB*[16].

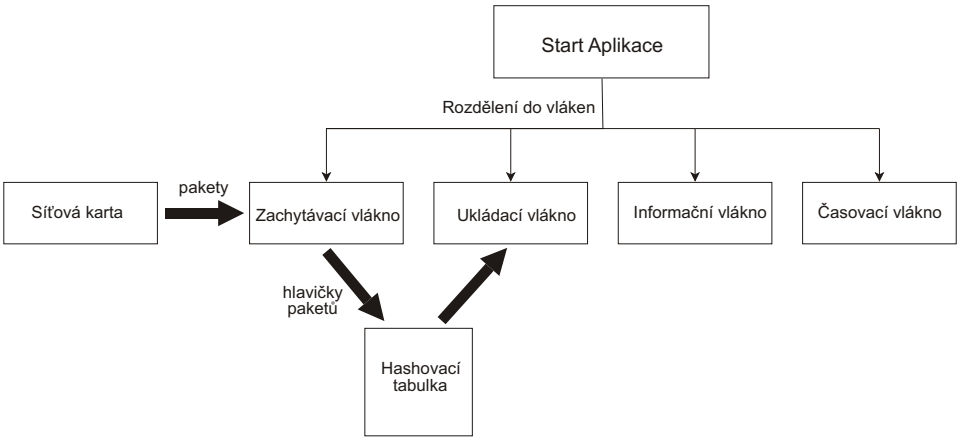
2.2.4 Prostředí klientské aplikace

Dalším zásadním rozhodnutím, které bylo potřeba učinit, je volba prostředí, ve kterém bude fungovat klientská část aplikace.

Rozhodování probíhalo mezi desktopovou aplikací a webovým rozhraním. V případě volby desktopové aplikace by bylo nutné zvolit mezi konkrétní platformou, tedy Unix nebo Windows. Naproti tomu je webové rozhraní nezávislé na platformě a odpadá i nutnost instalace klientského programu.

Vzledem k uvedeným výhodám bylo rozhodnuto pro webové rozhraní. Detailněji o této části bakalářské práce pojednává kapitola 3.

2.3 Architektura programu



Obrázek 2.2: **Architektura NAMon**

Architekturu programu zobrazuje obrázek 2.2. Program se po spuštění rozdělí na 4 vlákna.

Zachytávací vlákno provádí vlastní příjem paketů a jejich filtrování. Následně je pak ukládá do hashovací tabulky, respektive aktualizuje její záznamy v případě *podobných* paketů. Toto vlákno je spuštěno s maximální prioritou.

Ukládací vlákno je spuštěno s minimální prioritou. Pokud není hashovací tabulka prázdná, začne vlákno data ukládat z tabulky do samotné databáze. Pokud je hashovací tabulka prázdná, uspí se na nastavenou dobu. Tato doba významně ovlivňuje výsledný cache-hit, ale zároveň způsobuje zpoždění v zobrazování - data se do databáze v nejhorším případě dostanou až po této době.

V reálném provozu (viz kapitola 4) se ukázalo, že konkrétně tento přístup není dostatečně výkonný. Nakonec existují v programu dvě instance hashovací tabulky, mezi kterými se vždy přepne v případě ukládání. Nedochází tedy k současnému přístupu zachytávacího a ukládacího vlákna do hashovací tabulky.

Informační vlákno slouží k zobrazování výstupu programu (převážně ladící informace), k zachytávání uživatelských vstupů a následnému vypisování aktuálního provozního stavu.

Časovací vlákno. Jeho úkolem je nastavování aktuálního časového intervalu, slouží také k přepínání databázových tabulek při přechodu do následujícího dne.

2.4 Možnosti nastavení

Chod programu, velikost databáze, stupeň získaných informací i vzhled webové aplikace značně ovlivňuje nastavení programu. Program umožňuje dva způsoby nastavování. Pro volbu výkon zásadně ovlivňujících atributů slouží hlavičkový soubor. Tento způsob byl zvolen hlavně kvůli zrychlení samotné aplikace - případný nepotřebný kód se do výsledné aplikace vůbec nedostane.

Ostatní, pro výkon méně významné atributy, je možné zadávat přímo z příkazové řádky během spuštění či pomocí konfiguračního souboru.

2.4.1 Nastavení pomocí hlavičkového souboru

Toto nastavování probíhá pomocí souboru **namon_config.h**. Při každé změně je potřeba aplikaci znovu zkompilovat, některé změny si pak dokonce žádají úpravu pomocných struktur v databázi. Tyto úpravy automaticky provede instalační skript. Po změně v tomto souboru je tedy nutné spustit instalační skript, který se postará o všechny nutné kroky.

Nastavit lze tyto položky:

MAC_ENABLED - Zapíná podporu rozlišování MAC adres. Znamená to tedy další dva sloupce v databázi, zvětšení klíče a jistou režii navíc. Vzhledem k podstatnému nárustu velikosti databáze (cca dvakrát), bude často tato volba v reálném nasazení vypnutá.

PORT_ENABLED - Ovlivňuje podporu rozlišování portů. Pro každý záznam se tedy uchovává i cílový a zdrojový port. To potom znamená zvětšení databáze o tyto položky, ale také nárust počtu záznamů. Dojde totiž ke snížení agregačního poměru přidáním další klíčové položky. V praxi se ukázalo, že přibude cca 30% záznamů a samotná velikost databáze se zvětší o asi 80%. V reálném provozu je tedy nutné zvážit, jestli je nutné monitorovat i použité síťové služby (na základě portů), nebo stačí informace o cílové a zdrojové stanici.

STATS_ENABLED - Zapíná podporu provozních statistik samotného programu. Jedná se o počty přijatých a zahozených paketů, úspěšnost cache, počtu SQL dotazů. To vše během časového intervalu.

AGGREGATE_INTERVAL - Jedná se o časový interval popsaný v sekci 2.2.2. Zásadně ovlivňuje přesnost měření, velikost celkové databáze a také cache-hit. Například přechod z jednominutového intervalu na pětiminutový znamenal při stejném provozu zmenšení databáze cca dvakrát. Lze nastavit od jedné sekundy (má smysl jen v pomalých sítích) až po jednu hodinu.

SLEEP_INTERVAL - Udává hodnotu v sekundách, na kterou se uspí vlákno ukládající záznamy z hashovací tabulky do samotné databáze. Ovlivňuje výsledný cache-hit, tedy i počet nutných dotazů do databáze. Zároveň však způsobuje zpoždění v zobrazovaní statistik.

HASH_TABLE_MIN_SIZE - Udává počáteční a zároveň minimální velikost hashtabulky. Tato hodnota je určena při instalaci. Vzhledem k velké rezervě a k vcelku malému poklesu výkonu při nesprávném nastavení se nedoporučuje měnit.

2.4.2 Ostatní nastavení

Tato nastavení jsou zadávána pomocí příkazového řádku a konfiguračního souboru. Jedná se hlavně o údaje týkající se databáze (adresa databázového serveru, jméno databáze a tabulky, přihlašovací jméno a heslo). Dále je pak možné nastavování filtru paketů a zapínání ladícího výstupu.

Kapitola 3

Webové rozhraní

Webové rohraní slouží k zobrazování statistik ve formě grafů a tabulek. Je naprogramované pomocí PHP 5.0, zobrazování grafů je pak řešeno pomocí javascriptu - použita byla upravená verze knihovny WebFx CHart [17]. Pro nastavování vzhledu webového rozhraní slouží technologie CSS. Veškeré grafy a tabulky se generují z reálných dat (nejsou tedy předpřipraveny) a jsou definované uživatelem. Výhodou použití javascriptu je přenesení výpočetního výkonu k uživateli.

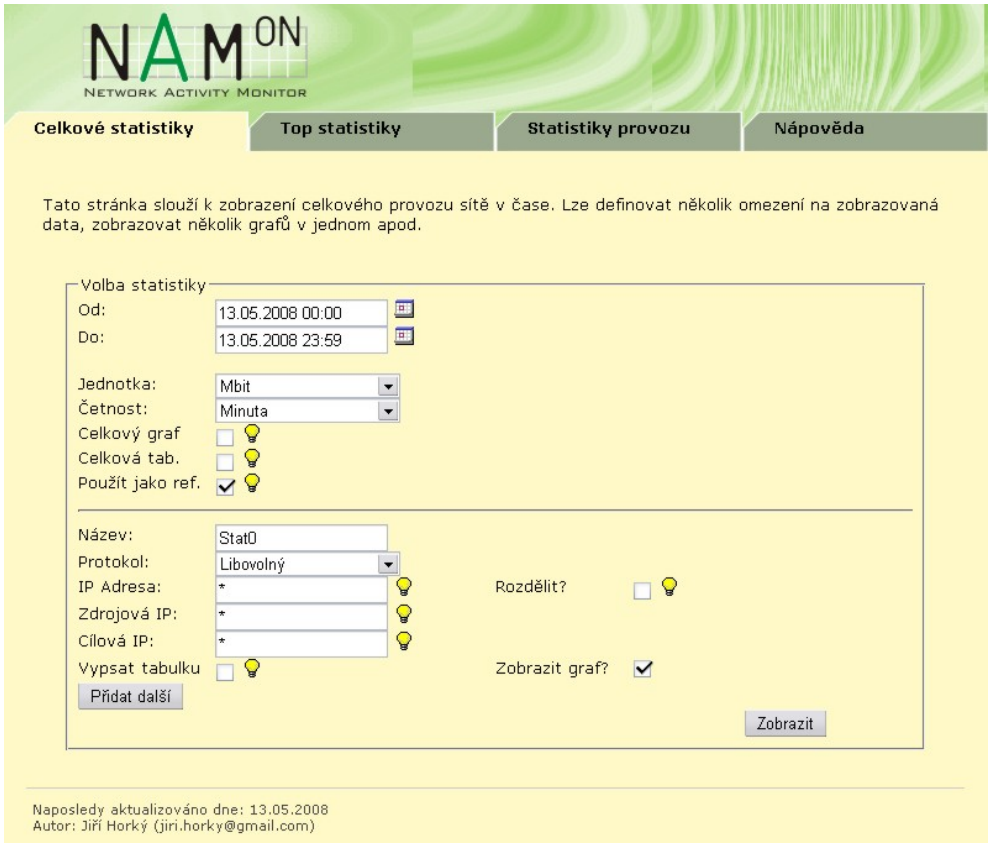
K zobrazení kalendáře slouží javacsriptové funkce nazvané JS Calendar [18].

3.1 Úvodní obrazovka

Na úvodní obrazovce probíhá přihlášování ke konkrétní databázi. Při úspěšném přihlášení dojde k vytvoření spojení a inicializují se datové struktury používané v celé aplikaci. Mimo jiné se z databáze načte konkrétní nastavení dané tabulky, tj. jestli je podporováno rozlišování portů, MAC adres a zda je zapnut modul kontrolující provozní stav aplikace. Poté dojde k načtení další stránky umožňující samotné dotazování.

3.2 Hlavní stránka

Na hlavní stránce (obrázek 3.1) probíhá veškeré dotazování. Dle typu dotazu je stránka rozčleněna na tři hlavní části zobrazující grafy, čtvrtou částí je pak nápověda. Vzhled jednotlivých stránek závisí na nastavení zachytávání. Pokud není povoleno rozlišování portů či MAC adres, příslušné možnosti se nezobrazují.

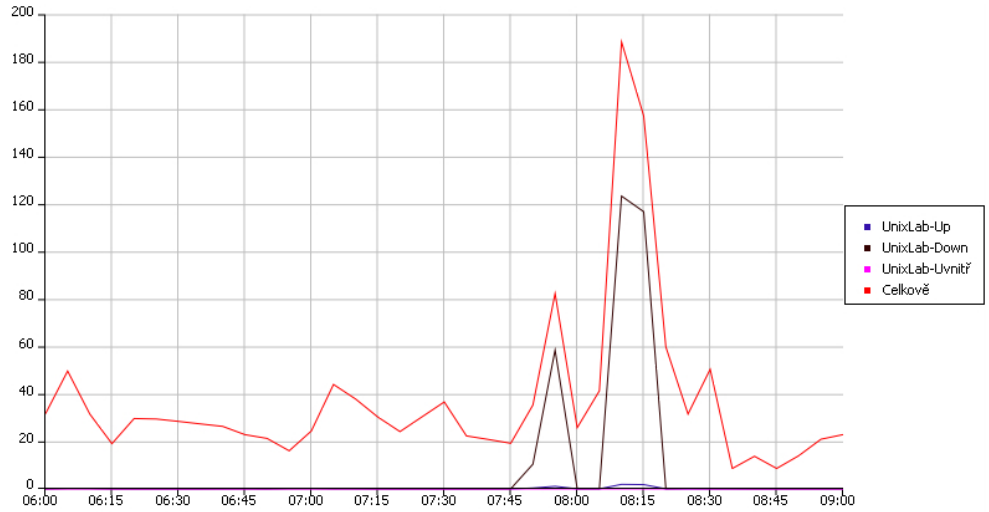


Obrázek 3.1: **Vzhled hlavního okna webového rozhraní**

3.2.1 Celkové statistiky

V této záložce probíhají dotazy na celkový provoz sítě v čase. Lze definovat několik omezení na zobrazovaná data (konkrétní IP adresa, port, rozsahy IP adres a portů apod.), zvolit časové období, jednotku zobrazování a požadovanou časovou přesnost. Zobrazovat lze několik grafů v jednom, rozhraní umožňuje počítat procentuální hodnoty vybraných statistik. Zobrazit tak lze například podíl webového provozu vzhledem k celkovému, či jen provoz zvolených počítačů během dne.

Dále lze vypsat konkrétní hodnoty v jednotlivých časových intervalech. Ukázka grafu a výsledné tabulky je na obrázcích 3.2 a 3.3. Veškeré možnosti nastavení jsou detailně popsány v uživatelské dokumentaci (viz příloha A).



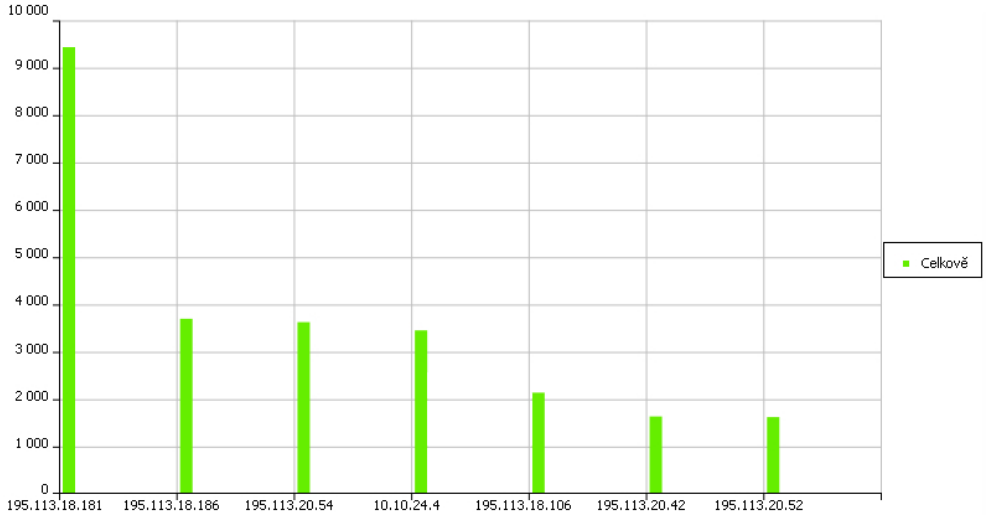
Obrázek 3.2: **Celkový graf.** Průměrný pětiminutový provoz na Malé Straně v MB dne 14.5.2008 (6:00-9:00).

Typ	Lab-Up(MB)	Lab-Down(MB)	Lab-Uvnitř(MB)	Celkově(MB)
Min	0.43[0.13%]	2.03[0.63%]	0.00[0.00%]	322.62
Max	72.08[1.02%]	4 627.35[65.51%]	0.00[0.00%]	7 064.05
Avg	7.49[0.53%]	316.26[22.54%]	0.00[0.00%]	1 403.30
Celkem	277.02[0.53%]	11 701.46[22.54%]	0.00[0.00%]	51 921.97

Obrázek 3.3: **Celková tabulka.** Průměrný pětiminutový provoz na Malé Straně v MB dne 14.5.2008(6:00-9:00).

3.2.2 Top statistiky

V této záložce se lze dotazovat na nejaktivnější IP adresy (uživatele), porty (aplikace) či poměr použitých protokolů během zvoleného období. Rovněž lze definovat několik omezení na zobrazovaná data, podobně jako u celkových statistik. Zobrazovat lze několik grafů najednou (za sebou). Konkrétní hodnoty se pak vypisují do tabulky, kde dochází k případnému překladu IP adres na doménová jména. Ukázku grafu a výsledné tabulky lze vidět na obrázcích 3.4 a 3.5.



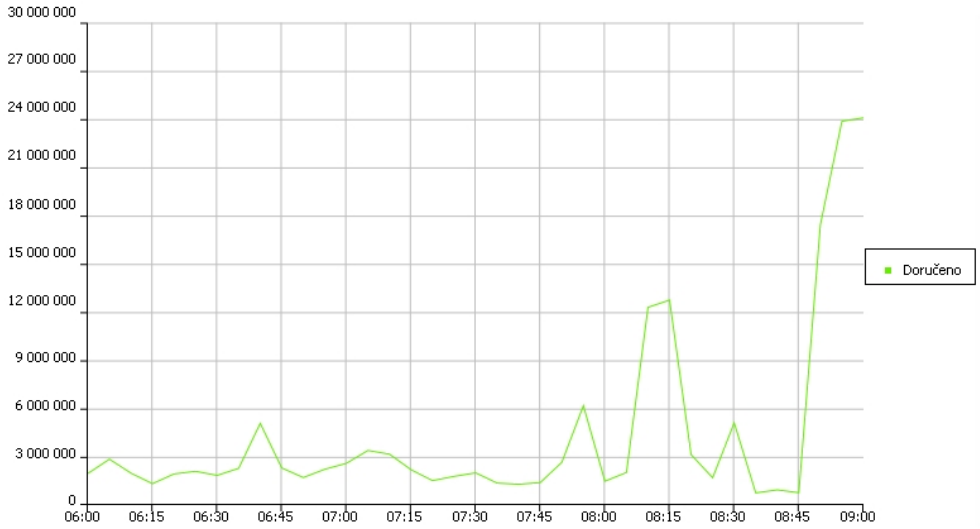
Obrázek 3.4: **Top graf.** Top 7 zdrojových IP adres během provozu na Malé Straně v MB dne 14.5.2008 (6:00-9:00).

src_ip	Celkově(MB)
195.113.18.181(ufallab.ms.mff.cuni.cz)	9 435.97
195.113.18.186(sup.ms.mff.cuni.cz)	3 699.15
195.113.20.54(lectures.ms.mff.cuni.cz)	3 627.58
10.10.24.4(deimos.ufal.hide.ms.mff.cuni.cz)	3 454.29
195.113.18.106(nikam.ms.mff.cuni.cz)	2 138.41
195.113.20.42(afs1.ms.mff.cuni.cz)	1 635.67
195.113.20.52(ufal.ms.mff.cuni.cz)	1 622.32

Obrázek 3.5: **Top tabulka.** Top 7 zdrojových IP adres během provozu na Malé Straně v MB dne 14.5.2008 (6:00-9:00).

3.2.3 Statistiky provozu

Tato záložka slouží, jak již název napovídá, ke sledování provozních statistik. Jedná se především o sledování počtu přijatých a zahozených paketů, cache-hit hashtabulky a počtu dotazů do databáze ve zvoleném časovém období. Jedná se vždy o průměrné hodnoty během nastaveného časového intervalu. Součástí výstupu je také tabulka hodnot ve všech časových intervalech ve zvoleném období. Ukázky grafů zobrazují obrázky 3.6, 3.7 a 3.8.



Obrázek 3.6: **Graf provozu.** Počet přijatých paketů během pětiminutových intervalů dne 14.5.2008 (6:00-9:00).

Typ	Doručeno	Zahozeno	Zahozeno v %	Cache hit	Cache miss	Cache hit v %
max	24 082 345.00	3 951.00	0.032	23 657 175.00	16 836.00	99.934
avg	4 419 118.19	106.78	0.001	3 395 472.35	13 629.86	98.926
min	755 958.00	0.00	0.000	437 532.00	10 801.00	96.781
sum	163 507 373.00	3 951.00	0.002	125 632 477.00	504 305.00	99.600

Obrázek 3.7: **Souhrný výpis provozu.** Data získána během provozu na Malé Straně dne 14.5.2008 (6:00-9:00).

Čas	Doručeno	Zahozeno	Zahozeno v %	Cache hit	Cache miss	Cache hit v %
06:00:00	1 962 599.00	0.00	0.00	1 234 703.00	14 379.00	98.85
06:05:00	2 852 078.00	0.00	0.00	1 819 379.00	12 116.00	99.34
06:10:00	1 985 938.00	0.00	0.00	1 244 809.00	12 617.00	99.00
06:15:00	1 328 604.00	0.00	0.00	740 927.00	12 044.00	98.40
06:20:00	1 936 447.00	0.00	0.00	1 232 411.00	10 801.00	99.13
06:25:00	2 097 860.00	0.00	0.00	1 296 151.00	13 097.00	99.00
06:30:00	1 844 617.00	0.00	0.00	1 136 430.00	11 763.00	98.98
06:35:00	2 283 395.00	0.00	0.00	1 312 079.00	13 127.00	99.01
06:40:00	5 079 646.00	0.00	0.00	1 232 522.00	13 463.00	98.92

Obrázek 3.8: **Detailní výpis provozu.** Data získána během provozu na Malé Straně dne 14.5.2008 (6:00-9:00).

3.2.4 Nápověda

Ve všech předchozích částech webového rozhraní obsahují důležité ovládací prvky popis svých funkcí, formátu vstupu apod. Nápověda se vyvolá automaticky po najetí kurzoru na obrázek žárovky u příslušného prvku. Kromě výše zmíněné pomoci je součástí webového rozhraní i samotná sekce obsahující nápovědu. Ta popisuje a na konkrétních příkladech ukazuje použití webového rozhraní.

Kapitola 4

Reálné nasazení

Hlavním úkolem reálného nasazení programu bylo ověření provozuschopnosti aplikace, tedy dokázat, že aplikace je dostatečně rychlá pro velké datové toky. Jako kritérium rychlosti zachytávání byla zvolena průměrná procentuální ztráta paketů. Ta nesměla za den překročit hodnotu **0,05%**, za jeden časový interval (obvykle 5 minut) pak **0,5%**.

Během testování bylo, kromě různých logických chyb v programu, objeveno několik výkon negativně ovlivňujících míst v aplikaci. Ta byla následně přepracována, o čemž více pojednává část 4.1.

Dalším důležitým důvodem pro nasazení v praxi bylo zjištění vlivu různých nastavení na počet záznamů a fyzickou velikost databáze. Pomineme-li nutnost tato data skladovat[1], je její velikost důležitá hlavně z hlediska rychlosti uživatelských dotazů při vykreslování dat.

4.1 Zvyšování výkonu, provedené optimalizace

Hlavním úkolem testovacího provozu bylo nalézt a opravit úzká hrdla v programu. To znamenalo změřit časovou náročnost různých míst v kódu a nejpomalejší z nich dle možností optimalizovat.

4.1.1 Úvod

Pro hledání nejpomalejších míst v programu bylo nutno změřit dobu provádění používaných knihovních funkcí. Mnohem důležitější než absolutní doba provádění[2] je poměr rychlostí jednotlivých příkazů. Tato doba

[1] což při dnešní ceně a kapacitě pevných disků není takový problém

[2] která je závislá na konkétním hardwaru, optimalizacích překladače atd.

včetně poměrů rychlostí je uvedena v tabulce 4.1.

Tabulka 4.1: Doba provádění jednotlivých funkcí použitých v programu

Funkce	Čas v ns	Poměr k nejrychlejší funkci
`strcpy`	10,4	1,0
`strncpy`	339,9	32,6
`strcat`	32,3	3,1
`strncat`	46,9	4,5
`sprintf`	277,2	26,6
`snprintf`	311,5	29,9
`sprintf2`	578,7	55,5
`time`	1455,2	139,5
`localtime`	741,6	71,1
`strftime`	728,7	69,9
`memcpy`	32,6	3,1
`lock & unlock mutex`	311,0	29,8
Bitové operace	Čas v ns	Poměr k nejrychlejší funkci
<<	0,01	0,005
&	0,01	0,005
Funkce hashtabulky	Čas v ns	Poměr k nejrychlejší funkci
lookup	70,3	6,7
insert	154,3	14,8
Funkce v programu	Čas v ns	Poměr k nejrychlejší funkci
vytvoření klíče(1)	1353,1	129,7
vytvořeni klíce(2)	64,3	6,2
existující záznam celkově	211,6	20,3
nový záznam celkově	3387,0	324,7

Volání `strcpy` a `strncpy` kopírovalo 10 znaků, `sprintf` funkce obsahovaly jeden argument typu `int`, `sprintf2` pak 2 tyto argumenty. Funkce `memcpy` kopírovala 100 bytů dat. Z této tabulky vyplývá několik závěrů a to zejména:

- Je výhodnější použít funkce nekontrolující délku vstupu.
- Volání funkcí `sprintf`, `localtime`, `strftime` a `time` je velmi drahé, je nutno je omezit.
- Bitové operace jsou opravdu velmi rychlé, je vhodné je použít všude tam, kde je to možné.

K co nejrychlejšímu zachytávání je nutno se řídit následujícími pravidly:

1. Zachytávat co nejméně paketů
2. Zachytávat jen nutnou část paketu
3. S každým paketem dělat minimum práce

První zmíněný bod v praxi znamená definování `libpcap` filtrů. Ty jsou mnohem rychlejší než filtrování v samotné aplikaci - dané pakety se do aplikace vůbec nedostanou. Pro tuto práci to znamenalo omezení paketů jen na IP protokol verze 4 a na protokoly ICMP, UDP a TCP. TCP paket navíc nesměl mít nulovou délku - v takovém případě pro naše měření nemá význam[3]. Vzhledem k variabilní délce TCP a IP paketu byly nulové pakety filtrovány až v aplikaci. Implicitní filtr aplikace byl tedy: *ip proto \tcp or ip proto \udp or ip proto \icmp*.

V praxi se ukázalo, že pro tento monitoring nezajímavé pakety tvořily 16-35 % všech paketů (údaj získaný z provozu na Malé Straně ve dnech 12.5 - 21.5 2008).

Zachytávat jen nutnou část paketů znamená zachytávat takovou část hlavičky, ze které již lze spočítat velikost přenášených dat. V praxi to je ve všech případech nastavení 51 bytů - 16 bytů pro ethernetový rámec, až[4] 32 bytů pro IP hlavičku a prvních 13 bytů z TCP hlavičky. Tuto část je nutné zachytávat i v případě nerozlišování portů, délka TCP hlavičky je proměnná a je třeba zjistit offset vlastních dat. Nestačí tedy velikost dat počítat z velikosti datagramu, která je uvedena v IP hlavičce. Zároveň těchto 13 bytů stačí na zjištění velikosti UDP či ICMP paketů.

Třetí bod je z hlediska optimalizací nejzajímavější. Je třeba si uvědomit, jaké opravdu nezbytné procesy je třeba při zachycení jednoho paketu provést v zachytávacím vláknu. Jsou to:

1. Načíst informace o paketu z hlaviček (typ protokolu, IP adresy, porty) do vlastní struktury. Důležitou částí je zjištění časového intervalu, ve kterém paket přišel.
2. Zjistit, jestli *podobný* paket již není uložen v hashtabulce. K tomu je nutné vytvořit klíč na základě informací z hlavičky paketu a provést hledání v databázi. Mohou nastat dvě možnosti:

[3]Tato aplikace je určena především pro měření výkonových charakteristik sítě. Pro bezpečnostní aplikace jsou tyto pakety podstatné. Jedná se hlavně o navazování a ukončování spojení, potvrzování dat apod.

[4]Standardní velikost IP hlavičky je 20 bytů, v případě použití tzv. optionů však až 32 bytů

(a) *podobný* paket již v hashtabulce je, stačí zvětšit informace o velikosti přenášených dat a změnit celkový počet takovýchto paketů.
(b) *podobný* paket ještě v hashtabulce není. V tomto případě je nutné naalokovat potřebnou velikost paměti pro datovou strukturu použitou v hashovací tabulce, data do ní zkopírovat a záznam do tabulky vložit.

Těmto procesům se věnují následující sekce.

4.1.2 Načítání informací o paketu z hlaviček

Na samotném analyzování paketu není příliš prostoru pro optimalizace. Důležité je neuchovávat IP adresy pomocí řetězců, jak by se nabízelo, ale použít k tomu datový typ `int`. Nejen, že tak zaberou obě IP adresy v databázi 3x méně místa, ale vyhneme se tak použití velmi drahého volání funkce `sprintf`.

Zajímavějším problémem je získávání časového intervalu zachyceného paketu. Ten je nezbytně nutné zjistit pro každý paket. Je zřejmé, že přímočaré řešení spočívající ve volání funkcí `time`, `localtime` a `strftime` pro každý paket je nejen velmi drahé, ale i zbytečné. V aplikaci proto běží vlastní vlákno, které provádí globální změnu časového intervalu a aktuálního dne (k zjištění jména tabulky, do které se paket bude ukládat), pro každý paket se pak jen tento globální parametr načte. Je však nutné zajistit atomicitu této operace, není možné, aby se nastavil jiný časový interval právě v moment, kdy ho druhé vlákno čte. Vzhledem ke snaze vyhnout se drahému uzamykání vláken, které by bylo nutno provést pro každý příchozí paket, bylo zvoleno použití jediné atomické proměnné typu `int`.

Otázkou však zůstává formát čísla, který se do příslušné hodnoty bude ukládat. Při použití formátu, který vrací volání funkce `time`[5], je stále nutné volat funkce `localtime` a případně `strftime`, které umožní zjistit aktuální čas. Tyto operace jsou však relativně pomalé, a proto byl nasazen vlastní formát tohoto atomického čísla využívající bitové posuny (viz příloha B). Použitím tohoto formátu se zjišťování času zrychlilo po řadě 5609, 2711 a 1298 krát (v porovnání s původním řešením, s použitím časovače a `strftime`, s použitím časovače bez `strftime`).

[5]počet sekund od 1.1. 1970 00:00

4.1.3 Vytvoření klíče pro dotazování do hashtabulky

Operace vytváření klíče je nutná pro každý paket zachycený aplikací, proto je důležité, aby byla co nejrychlejší. Klíč se vytváří z cílové IP adresy, protokolu, časového intervalu a volitelně také z portů a MAC adres. Vzhledem k počtu klíčových informací není možné použít jako klíč číslo, ale řetězec. Původní řešení spočívalo ve vypsání klíčových parametrů do řetězce voláním funkcí `sprintf`, případně `strcpy` u protokolu.

Ukázalo se však, že takovéto vytváření klíče je příliš pomalé. Proto byl vymyšlen vlastní formát klíče využívající bitové operace. Není totiž důležité, v jakém formátu klíč bude (nemusí tedy obsahovat vypsanou IP adresu v textovém formátu), důležité je, aby byl pro unikátní klíčové hodnoty také unikátní. Nesmí se tedy stát, že pro dva různé pakety dostaneme stejný klíč. Idea spočívá v tištění jednotlivých bytů čtyř (IP adresy) či dvou (porty) bytových čísel do řetězce. Je třeba si dávat pozor na případné nulové hodnoty (znamenalo by to ukončení řetězce), ty jsou nahrazeny jedničkami a případné pozice nul jsou bitově zaznamenány do pomocných čísel, které jsou pak vypsány na konci řetězce. Dostaneme tedy také jednoznačné přiřazení, navíc však daleko rychlejší.

Nové vytvoření klíče je dle měření cca 25 krát rychlejší. Ač se může zdát, že vytváření klíče není příliš důležité, opak je pravdou. Při zatížení 150 tisíc paketů za sekundu, představovalo samotné vytváření klíče původním způsobem 28% zátěže CPU (procesor pentium P4 3.0GHz).

4.1.4 Vyhledání záznamu dle klíče, existující záznam

Tato část je z hlediska optimalizace nezajímavá. Probíhá zde volání příslušné funkce hashtabulky, na kterou má vliv pouze velikost a naplnění tabulky. Velikost je však zvolena s dostatečnou rezervou již během instalace, nemělo by tak docházet ke kolizím či změně velikosti tabulky. Navíc je dle měření hledání v tabulce rychlé, více než 3x rychlejší než volaní funkce `sprintf`, viz 4.1. V případě již existujícího záznamu v tabulce je nutné ještě upravit příslušnou velikost přenesených dat a počet paketů. Celková doba potřebná pro existující záznam se dle měření pohybovala okolo 200ns, čímž byla více než 15x rychlejší než vkládání nového záznamu do hashovací tabulky.

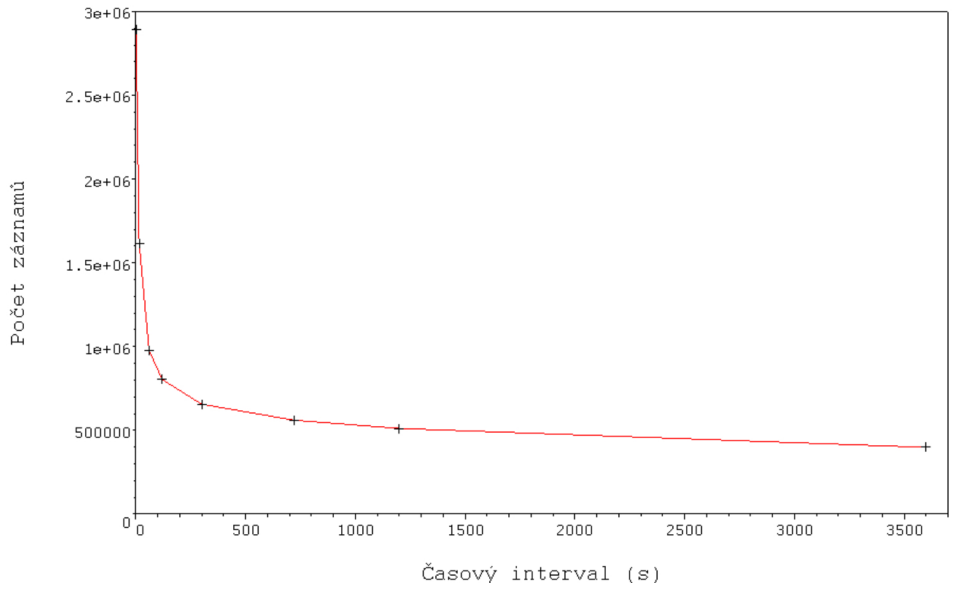
4.1.5 Vložení nového záznamu

Vkládání nového záznamu do tabulky je náročnější operací. Je třeba vytvořit klíč, prohledat tabulku, naalokovat paměť pro příslušné da-

tové struktury, zkopírovat do nich obsah a poté provést samotné vložení záznamu. Vzhledem k relativní pomalosti této operace jsou žádoucí vysoké procentuální hodnoty cache-hitu. Vliv nastavení na tuto hodnotu se více zabývá sekce 4.3.

4.2 Vliv nastavení na velikost databáze

Cílem bylo zjistit vliv nastavení časového intervalu na celkovou velikost databáze, respektive na počet záznamů v ní. Měření probíhalo v čase 01:00 - 02:00 na kolejích UK v Troji, nebylo zapnuto rozlišování portů, ani MAC adres. Během této hodiny bylo přeneseno cca 20GB dat. Spuštěno bylo několik zachytávání najednou, s různými nastaveními časových intervalů, a to: 1s, 15s, 1min, 2min, 5min, 12min, 20min, 60min.



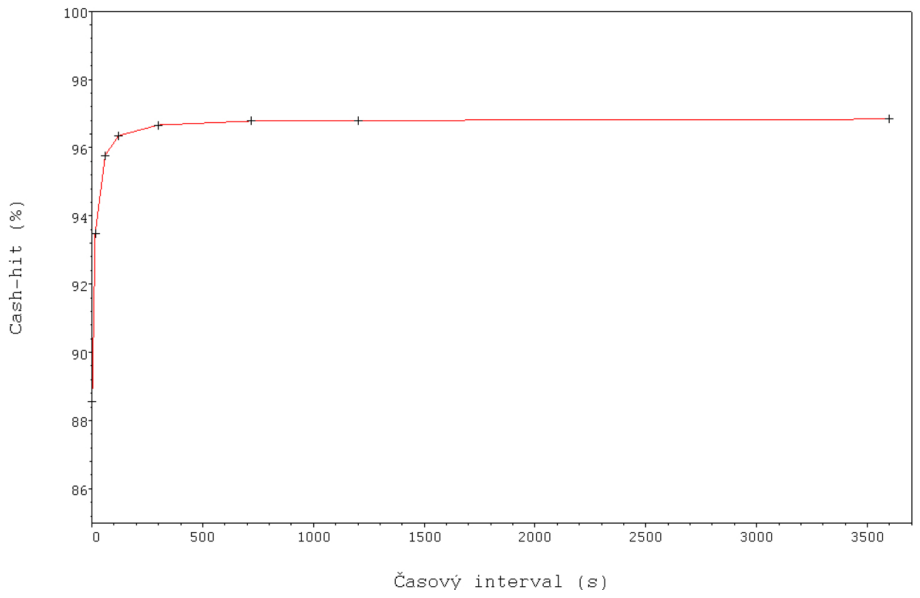
Obrázek 4.1: **Vliv nastavení časového intervalu na velikost databáze.**

Jak lze vidět z grafu 4.1, velikost databáze prudce klesá cca do pětiminutového časového intervalu, poté je již vliv nastavení daleko menší. Je zřejmé, že bude vždy nutno zvolit kompromis mezi velikostí databáze a přesností zachycených dat, rozumnou volbou se však jeví právě pětiminutový interval.

4.3 Vliv nastavení na hodnotu cache hitu

Cílem tohoto měření bylo zjistit vliv nastavení časového intervalu na hodnotu cache-hitu, tedy i počtu nutných dotazů do databáze. Měření probíhalo za stejných podmínek a nastavení jako v 4.2, sleep interval byl nastaven na 240 sekund.

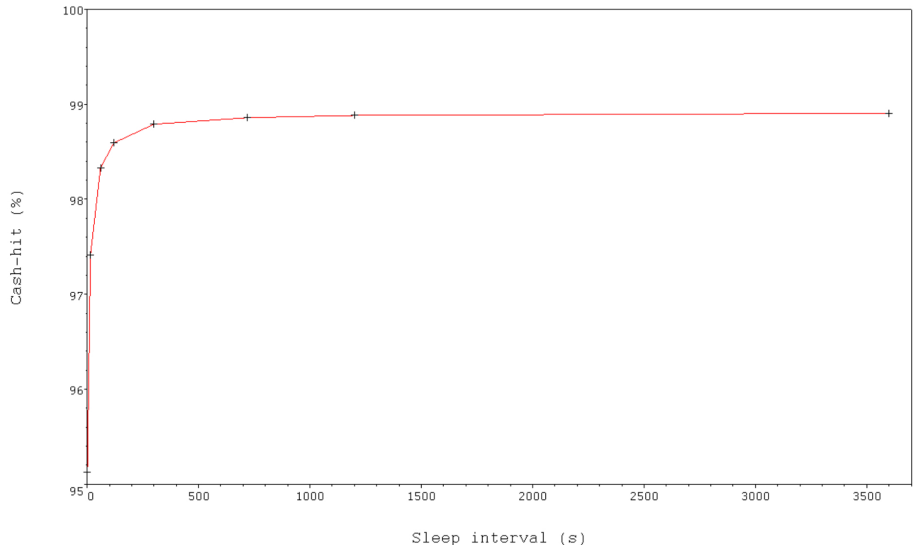
Test ukázal (viz graf 4.2), že větší časový interval než 5 minut na hodnotu cache-hitu již prakticky nemá význam. Na druhou stranu při menším než minutovém intervalu hodnota cache-hitu značně klesá.



Obrázek 4.2: **Vliv nastavení časového intervalu na cache-hit.**

Dalším provedeným měřením bylo zjišťování vlivu sleep intervalu na cache-hit při pevném časovém intervalu. Ten byl nastaven na pět minut. Spuštěno bylo několik zachytávání najednou, s různými nastaveními sleep intervalů, a to: 1s, 15s, 1min, 2min, 5min, 12min, 20min, 60min. Měření probíhalo během hodiny provozu na kolejích UK v Troji.

Tento test ukázal, že i sleep interval má důležitý vliv na úspěšnost cache. Jak lze vidět na grafu 4.3, delší sleep interval než pět minut má na hodnotu cache-hitu již pouze minimální význam. Naopak kratší sleep interval než jedna, respektive dvě minuty, znamená prudké zvýšení počtu dotazů do databáze. Pro lepší představu: s 15s sleep intervalem bylo potřeba 1,9 milionů dotazů za hodinu, s pětiminutovým intervalem 921 tisíc, s dvacetiminutovým pak 851 tisíc. Pokud tedy nevadí jisté zpoždění



Obrázek 4.3: **Vliv nastavení sleep intervalu na cache-hit.**

v zobrazování dat, je vhodné tento interval nastavit alespoň na hodnotu okolo pěti minut.

4.4 Malá Strana

Tento testovací provoz aplikace probíhal v budově Matematicko-fyzikální fakulty UK na Malé Straně. Monitorován byl provoz všech počítačových laboratoří a učeben, PC v kancelářích a serverů. Celkově se jednolo o 1800 strojů, je třeba si však uvědomit, že z tohoto počtu bylo cca 670 serverů a řada z registrovaných počítačů nebyla během provozu používána.

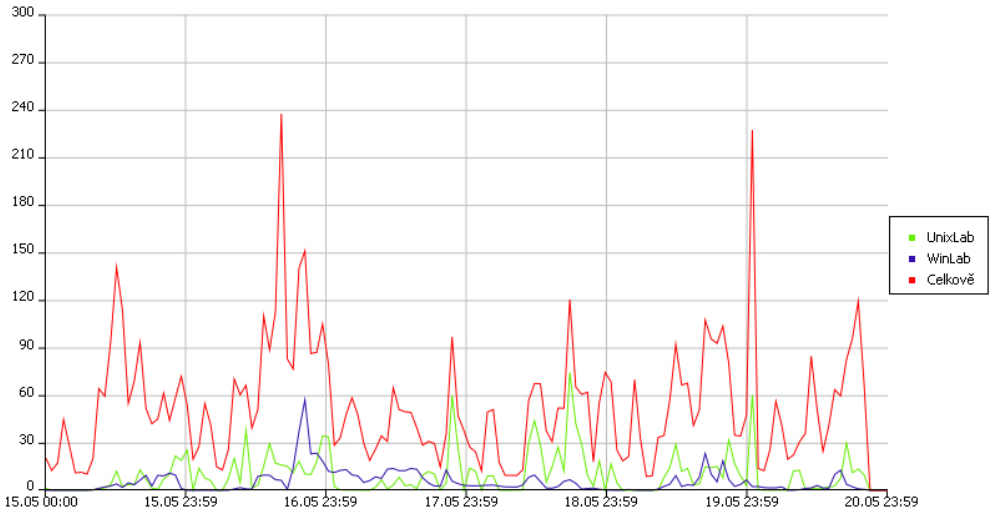
Testování probíhalo na stroji s procesorem Intel Pentium4 3.0GHz, 1GB RAM se síťovou kartou Intel PRO 1000 napojenou na 33MHz 32 bitovou sběrnici PCI. Rychlost této sběrnice je pro gigabitové sítě limitující, celková teoretická propustnost je 800 Mbit, ta je však sdílená se všemi PCI zařízeními. Praktická propustnost se proto pohybuje okolo 400-500 Mbit. Na serveru bylo nainstalován operační systém Gentoo Linux s jádrem 2.6.23.

Karta využívala NAPI rozhraní a zároveň bylo provedeno několik nastavení[6] jádra systému ovlivňující bufferování paketů:

[6] více kapitola 5

- net.core.rmem_max = 16777216
- net.core.rmem_default = 16777216
- net.core.netdev_max_backlog = 10000

Během tohoto testovacího provozu bylo provedeno mnoho optimalizací samotného programu, které umožnily výrazně snížit počet zahazovaných paketů. Dokonalého provozu však přesto nebylo dosaženo - stále docházelo k zahazování paketů. Procento zahozených paketů se již však téměř dostalo do stanovených hranic. Průměrné procento zahozených paketů během provozu ve dnech 15.5. az 20.5. 2008 činilo 0,006 procent, přičemž k překročení hranice zahazování (0,5%) pro časový interval (nastaveno 5 minut) došlo jen dvakrát a to 1,50% a 0,87% paketů během intervalu. Nejpravděpodobnějším důvodem k zahazování paketů byla již zmíněná pomalá sběrnice, protože k němu docházelo při průměrné zátěži nad 200 Mbit během pětiminutového intervalu.



Obrázek 4.4: **Průměrný hodinový provoz na Malé Straně ve dnech 15.5. - 20.5.2008 v Mbit**

Naměřená data ve dnech 15.5. - 20.5. 2008

Měření probíhalo s nastaveným pětiminutovým časovým intervalem, sleep interval byl nastaven na 240s. Nebyly rozlišovány porty, ani MAC adresy. Všechny údaje v tabulce 4.2 jsou vztaženy k časovému intervalu.

Tabulka 4.2: Naměřená data během provozu na Malé Straně 15.5. - 20.5. 2008

Provoz	
Průměrný tok	52.35 Mbit
Maximální tok	288,15 Mbit
Celkem dat	3 292.92 GB
Průměrný počet paketů	9 556 387
Průměrný cache hit	98.6%
Datábaze	
Celkový počet záznamů	16 179 179
Průměrný počet dotazů	10 949.5
Celková velikost databáze	650.5 MB

4.5 Troja

Provoz aplikace probíhal v budově kolejí Univerzity Karlovy v Troji. Monitorován byl provoz všech počítačů obou budov kolejí (cca 1400 strojů). Ačkoliv je tento počet menší než počet PC při testování na Malé Straně, bylo zatížení především ve večerních hodinách větší.

Testování probíhalo na stroji se čtyřjádrovým Intel Core2 Quad Q6600 2.40GHz, 2GB RAM a síťovou kartou Intel PRO 1000 napojenou na sběrnici PCI-Express s teoretickou propustnostní 2,5 Gbit. Rychlost sběrnice již tedy nebyla omezujícím faktorem. Testování probíhalo na operačním systému FreeBSD v8.3. Dále bylo v systému provedeno několik nastavení:

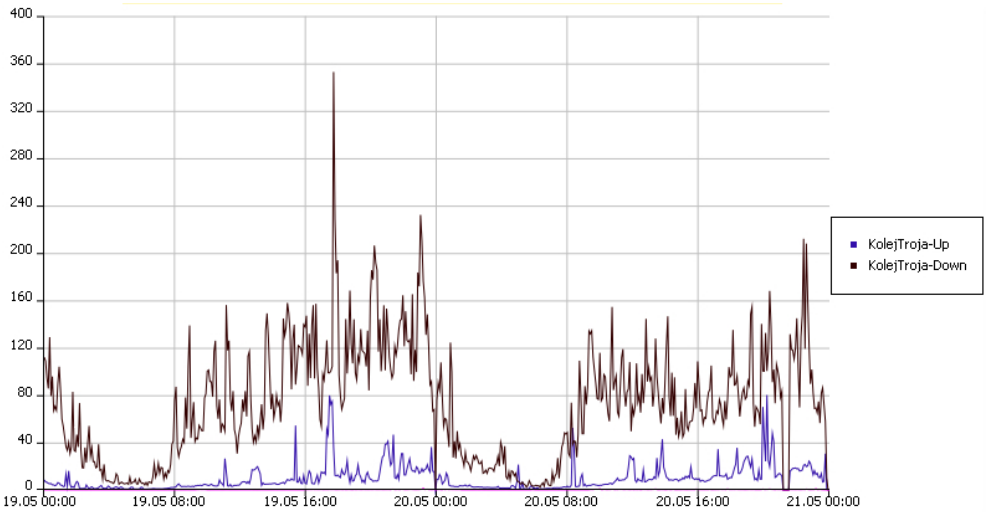
- kern.polling.burst_max=1000
- kern.polling.each_burst=50
- net.bpf.bufsize=10485760
- net.bpf.maxbufsize=10485760

Testovací provoz probíhal ve dnech 19.5. - 25.5, vzhledem k problémům se stabilitou stroje však jediným delším souvislým obdobím, kdy probíhalo zachytávání, bylo 19.5.-20.5.2008. Toto období je proto uvedeno v grafu 4.5 a tabulce 4.3. Během celého testovacího provozu nedošlo (dle aplikace) k zahazování paketů, celková zátěž stroje aplikací byla do 2% CPU.

Měření probíhalo s nastaveným pětiminutovým časovým intervalem, sleep interval byl nastaven na 240s. Nebyly rozlišovány porty, ani MAC adresy. Všechny údaje v tabulce 4.3 jsou vztaženy k časovému intervalu.

Tabulka 4.3: Naměřená data během provozu v Troji 19.5. - 20.5.2008

Provoz	
Průměrný tok	81.13 Mbit
Maximální tok	388.99 Mbit
Celkem dat	1 720.35 GB
Průměrný počet paketů	4 944 070.31
Průměrný cache hit	97.2%
Datábaze	
Celkový počet záznamů	30 307 642
Průměrný počet dotazů	60 951.46
Celková velikost databáze	1218.54 MB



Obrázek 4.5: **Průměrný pětiminutový provoz na kolejích UK v Troji ve dnech 19.5. - 20.5.2008 v Mbit**

Jak lze vidět z tabulky 4.3 tento provoz generoval daleko větší počet záznamů než provoz na Malé Straně. Jednak zde byl větší průměrný provoz, ale hlavně to plyne z rozdílného charakteru používání obou sítí. V Troji tvořili provoz ubytovaní studenti, kdežto na Malé Straně byla většina provozu vytvářena tamními servery. Tomu odpovídá i počet unikátních IP adres během jednoho dne provozu. Na Malé Straně byl tento počet okolo 300 tisíc, kdežto v Troji to bylo přes 2,5 milionu.

Kapitola 5

Možnosti rozšíření aplikace, ladění systému

Tato kapitola shrnuje základní směry, kterými by se v budoucnu mohl vývoj aplikace NAMon ubírat. Cílem dalšího vývoje by mělo být ještě efektivnější zachytávání, tudíž možnost zvládat větší datové toky. Toho lze dosáhnout několika způsoby.

5.1 Nastavení kernelu

K úspěšnému zachytávání datových toků okolo 1 Gbit již nestačí jen aplikace, která je dostatečně rychlá. Kernel Unixových operačních systémů totiž není standardně nastaven pro tento typ práce. Je nutné dostatečně zvýšit velikost zachytávacích bufferů a zapnout tzv. device polling.

5.1.1 Vstupní buffery

Konkrétní nastavení parametrů se liší v závislosti na použitém operačním systému.

Operační systém FreeBSD používá k zachytávání paketů zařízení BPF[1]. Úkolem tohoto zařízení je spuštění filtru pro každý paket a jeho uložení v bufferu. Velikost této vyrovnávací paměti lze nastavit pomocí utility `sysctl` a parametrů:

- net.bpf.bufsize
- net.bpf.maxbufsize

[1]Berkeley Packet Filter, více [20]

Dle [19] je rozumnou hodnotou pro velikost bufferu 20 MB.

Vzhledem k tomu, že linuxové systémy nemají k zachytávání vlastní buffer, ale frontu ukazatelů na jednotlivé pakety, je nutné zvýšit velikost vyrovnávací paměti pro všechny příchozí pakety a zároveň zvětšit délku fronty. To se provede pomocí souborového systému `/proc`, či změnou `sysctl.conf`. Jedná se o parametry:

- net.core.rmem_default
- net.core.rmem_max
- net.core.netdev,_max_backlog

kde `rmem\_*` je velikost bufferů pro pakety a netdev_max_backlog je maximální délka výše zmíněné fronty. Stejně jako u systému FreeBSD je doporučovaná hodnota pro buffery 20 MB, pro délku fronty pak 10000.

5.1.2 Device polling

V normálním režimu vyvolává síťová karta přerušení, kdykoliv si vyžaduje pozornost. To v závislosti na velikosti bufferů na síťové kartě a nastavení ovladačů může nastat pro každý příchozí paket zvlášť nebo pro několik paketů zároveň. Při každém přerušení pak dojde ke změně kontextu procesoru a k obsluze tohoto přerušení. Z toho vyplývá, že tato operace je procesorově velmi náročná. Při vysokých datových tocích pak karta generuje takové množství přerušení, že procesor obsluhuje jen je, přičemž nezbývá výkon pro samotnou aplikaci.

Tzv. device polling je metoda, která k problematice přistupuje opačně. Systém se sám periodicky dotazuje daného zařízení, jestli nechce obsloužit. Tím dochází k redukci přepínaní kontextu procesoru, čímž se celá operace výrazně zrychlí.

Na systému FreeBSD se polling zapíná pomocí utility `sysctl` s parametrem `kern.polling.enable=1`. Dále lze nastavit celá řada parametrů, které ovlivňují polling, jako je frekvence, maximální množství přidělených zdrojů na dotazování apod. Tato nastavení jsou podrobně popsána ve zdrojovém kódu kernelu, tj. v `src/sys/kern/kern_poll.c`.

Na linuxových systémech existuje podobný systém nazvaný NAPI[2], známý také pod jménem Rx polling. Nedochází k úplnému vypnutí pře-

[2]New API. Dle autorů tak bylo pojmenováno z důvodu nedostatku lepších nápadů. Pro více informaci viz [21]

rušení, nýbrž k jejímu vypínání a zapínání dle vytížení systému. Toto rozhraní musí být podporované síťovou kartou, podporu je pak třeba zapnout při kompilaci kernelu, respektive ovladače síťové karty. Dle [23] tak dojde k výraznému odlehčení zátěže procesoru.

5.2 Ovladače síťové karty

Vzhledem k tomu, že síťová karta je navržena pro klasický síťový provoz, nejsou ovladače karet (ani samotné karty) optimalizovány pro pasivní monitorování. Bylo by tedy možné upravit ovladače karty tak, aby se minimalizovalo množství práce nutné k zpracování paketu. Vzhledem k tomu, že na dané pakety se nebude odpovídat, není nutné některé úkony s pakety provádět (přerovnávání paketů apod.).

Dále je v závislosti na daném ovladači karty možno nastavovat velikost bufferů pro příchozí, respektive odchozí pakety, maximální počet generovaných přerušení apod.

5.3 PF_RING socket

PF_RING je nový druh socketu speciálně vytvořený pro pasivní zachytávání paketů. Jeho autorem je Luca Deri, člen sdružení ntop.org[23]. Princip spočívá v použití cyklického bufferu pro příchozí pakety v kernelu. Buffer je při vytvoření socketu naalokován, nedochází tedy k alokaci/dealokaci pro každý paket, pakety jsou zapisovány přímo do tohoto bufferu, čímž se přepisují již zpracované pakety. Na tento buffer je pak pomocí sdílené paměti a volání mmap() napojena vlastní měřící aplikace v uživatelském prostoru, nedochází tedy k žádným drahým systémovým voláním, jak je tomu u klasických socketů. Nedochází ani ke kopírování paketů do vlastních struktur kernelu a poté k následnému kopírování do uživatelského prostoru. O přijímání paketů se však stále stará jádro systému pomocí přerušení či device pollingu.

Výhodou použití je několikanásobné zrychlení zachytávání. Vzhledem k tomu, že existuje upravená verze knihovny `libpcap` podporující tyto sockety, daly by se použít i v této práci.

Značnou nevýhodou je nutnost modifikace kernelu.

5.4 nCap

nCap je stejně jako PF_RING dílem Luca Deriho. Jedná se o modifikaci ovladačů síťové karty, jádra systému a příslušnou knihovnu v uživatelském prostoru. Podobně jako PF_RING je také založen na cyklickém bufferu, s optimalizací jde však ještě dále. Pomocí několika technik, úprav jádra a samotných ovladačů síťové karty (viz [24]) nedochází po inicializaci k žádnému volání kernelu. Pakety jsou tedy skrze cyklické buffery předávány přímo monitorovací aplikaci, čímž dochází k maximálnímu urychlení zachytávání. Dle [24] je tak možné s procesorem Intel PIII 500Mhz zachytávat beze ztráty přes 500 milionů paketů za vteřinu.

Nevýhodou je velký zásah do systému. Je nutná úprava kernelu, ovladačů síťové karty (podporovány jsou navíc jen 1 a 10 Gbit karty Intel) a knihovny `libpcap`. I tak však lze nCap úspěšně použít s programem NAMon.

5.5 Další bufferování v aplikaci

Toto rozšíření se týká samotné aplikace. Na vícejádrových procesorech, či systémech s více procesory je při nynější architektuře programu vytíženo v podstatě jen jedno jádro / procesor. Aplikace je sice vícevláknová, avšak nejvíce práce má stále zachytávací vlákno[3]. Hlavní myšlenkou rozšíření je využití cyklického bufferu (podobně jako nCap či PF_RING) přímo v aplikaci, který by sloužil jako vyrovnávací paměť před vkládáním do hashtabulky. Zachytávací vlákno, které by mělo být co nejrychlejší, aby stíhalo odebírat pakety ze síťové karty, tak bude mít daleko méně práce. Při přijetí paketu, který není *podobný*, by tento paket jen vložilo do bufferu. Případné pomalé vkládání nového záznamu do hashtabulky se tak přenechá jinému vláknu (vláknům), které budou pakety z cyklického bufferu odebírat.

5.6 Vylepšení webového rozhraní

I přes to, že webové rozhraní umožňuje generování a zobrazování veškerých zachycených informací, je jistě možné jeho funkčnost vylepšit. Je nutno přiznat, že zobrazení některých dat je poněkud těžkopádné. Jistým vzorem by mohlo být webové rozhraní programu nTop[22].

[3]Ukládací vlákno je v momentě vkládání do databáze také zatížené, většinu času je však uspáno.

Kapitola 6

Závěr

Monitoring sítě je v dnešní době stále aktuálnější téma, ať už z důvodu dodržování zákona, prevence problémů v sítí či za účelem měření výkonu.

Cílem této práce bylo navrhnout a implementovat program pro monitoring sítě schopný fungovat ve vysokých datových tocích. Jedním z hlavních kritérií byla nasaditelnost této aplikace na běžném hardware a její praktická využitelnost. Aplikace měla umožnit volbu míry detailu zachycených informací, od celkových statistik, přes toky jednotlivých počítačů, až po sledování jednotlivých aplikací na bázi portů. Tato data pak mělo být umožněno vizualizovat ve formě grafů a tabulek, s možností nastavení zobrazovaného časového období a definování vlastních filtrů na zobrazovaná data.

Všechny tyto body se podařilo naplnit. Vhodnost návrhu byla prověřena v praxi při zkušebním provozu v budově MFF UK na Malé Straně a na kolejích UK v Troji. Potvrdilo se, že aplikace je dostatečně rychlá pro monitorování gigabitových sítí s více než tisíci stanicemi. Jediným praktickým problémem, který však vyplývá z obrovského počtu zachycených dat, se ukázala být velikost databáze, která ovlivňuje rychlost uživatelských dotazů. Vzhledem k tomu, že množství těchto dat při daném nastavení nelze ovlivnit, jako jediné řešení se jeví nutnost použít dostatečně výkonný dedikovaný databázový server.

Seznam obrázků

Literatura

[1] Vyhláška č. 485/2005 o elektronické komunikaci
http://www.mvcr.cz/sbirka/2005/sb169-05.pdf

[2] Vládní novela zákona 127/2005 Sb., o elektronických komunikacích
http://www.psp.cz/sqw/text/tiskt.sqw?o=5&ct=398&ct1=0&v=PZ&pn=4&pt=1

[3] Program pingplotter
http://www.pingplotter.com

[4] Program NeoTrace
http://www.neotrace.com

[5] Informace o protokolu SNMP
http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

[6] SNMP Vulnerabilities FAQ
http://www.cert.org/tech_tips/snmp_faq.html

[7] Cacti
http://www.cacti.net

[8] The Multi Router Traffic Grapher
http://oss.oetiker.ch/mrtg/

[9] Net-SNMP
http://net-snmp.sourceforge.net/

[10] Cisco Systems, Inc.
http://www.cisco.com

[11] Wikipedia, the free encyclopedia : NetFlow
http://en.wikipedia.org/wiki/Netflow

[12] NetFlow Portal
http://netflow.caligare.com/

[13] RFC 3954 - Cisco Systems NetFlow Services Export Version 9
http://www.ietf.org/rfc/rfc3954.txt

[14] INVEA-TECH, a.s. : FlowMon sondy
http://www.invea.cz/cs/products/flowmon-probes

[15] Tcpdump
http://www.tcpdump.org/

[16] GNOME Documentation Library: GLiB
http://library.gnome.org/devel/glib/stable/

[17] WebFx Chart. Distribuováno pod MIT licencí.
http://webfx.eae.net/dhtml/chart/chart.html

[18] JS Calendar. Distribuováno licencí GNU.
http://www.dhtmlgoodies.com/scripts/js_calendar/js_calendar.html

[19] Fabian Schneider, Jörg Wallerich, Anja Feldmann. Packet Capture in 10-Gigabit Ethernet Environments Using Contemporary Commodity Hardware.
http://www.net.t-labs.tu-berlin.de/papers/SWF-PCCH10GEE-07.pdf

[20] Berkeley Packet Filter
http://en.wikipedia.org/wiki/Berkeley_Packet_Filter

[21] NAPI
http://www.cyberus.ca/~hadi/usenix-paper.tgz

[22] nTop
http://www.ntop.org

[23] Luca Deri, Improving Passive Packet Capture: Beyond Device Polling
http://luca.ntop.org/Ring.pdf

[24] Luca Deri, nCap: Wire-speed Packet Capture and Transmission
http://luca.ntop.org/nCap.pdf

[25] Tobi Oetiker, RRDtool http://oss.oetiker.ch/rrdtool/

Přílohy

Příloha A

Uživatelská dokumentace

Aplikace NAMon je určena pro pasivní sledování sítě a byla navržena pro vysoké datové toky. Je primárně určena pro správce středně velkých sítí, lze s ní však monitorovat i samotný stroj, na kterém běží. Aplikace je navržena s ohledem na výkon, proto je vhodná i do gigabitových sítí.

Prostřednictvím webového rozhraní umožňuje sledovat datové proudy v celé síti, jednotlivých stanic či aplikací na bázi TCP a UDP portů.

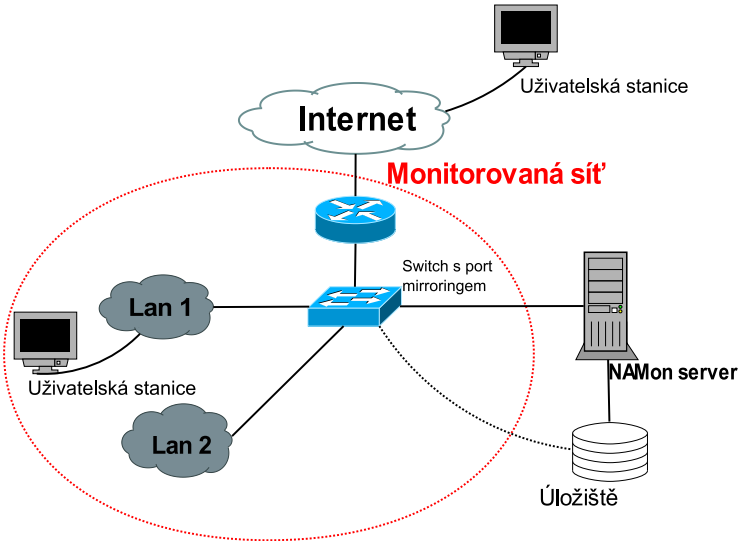
Aplikace je rozdělena na dvě části, které mohou (ale nemusí) být provozované na stejné stanici.

První částí je samotné zachytávání, tato část je určena pro platformu UNIX. Druhou částí je pak webové rozhraní, které je platformě nezávislé a může být (a typicky taky bude) provozováno na jiném než monitorovacím serveru.

A.1 Začlenění aplikace do sítě

Síťový model aplikace zobrazuje obrázek A.1. Monitorovací server je připojen k switchi umístěnému v dané monitorované síti. Na switchi (lze použít také hub) je poté nastaven port mirroring. V závislosti na nastavení switche lze monitorovat jen určité segmenty sítě. Tyto pakety jsou zpracovány monitorovacím serverem, který své záznamy ukládá na server, jež slouží jako úložiště. Klientské stanice se pak dotazují serveru s daty na jednotlivé statistiky provozu. Storage server může být provozován i na stejném stroji, kde probíhá monitorování. Tím se však zvýší jeho zatížení.

V případě použití switche s port mirroringem je třeba mít v monitorovacím serveru dvě síťové karty, jednu pro zachytávání, druhou pro



Obrázek A.1: **Začlenění aplikace NAMon do sítě**

samotné kontrolní spojení.

A.2 Požadavky

A.2.1 Softwarové požadavky

K úspěšnému provozovaní aplikace je potřeba:

1. Přístup k MySQL serveru s právy zakládat tabulky v dané databázi
2. Přístup k webovému serveru podporujícímu technologii PHP 5.0
3. Dostatečná oprávnění v systému k přepnutí karty do promiskuitního režimu a zachytávání paketů

Třetí bod se liší systém v závislosti na systému. Na systému Linux je bez jiných zásahů nutné mít práva superuživatele(root), na FreeBSD pak stačí práva ke čtení zařízení `/dev/bpfX`.

K úspěšnému zkompilování aplikace je pak třeba mít v systému následující knihovny:

1. `glib v2.16.1` či vyšší
2. `libpcap`
3. `libmysqlclient`
4. `libpthread`

A.2.2 Hardwarové požadavky

Hardwarové požadavky na běh aplikace se značně liší v závislosti na velikosti a rychlosti dané sítě. Pro 1Gbit sítě jsou požadavky pro zachytávací část programu následující:

- **Procesor** - Intel P4 2.4GHz nebo kompatibilní
- **Paměť RAM** - 512MB
- **Síťová karta** - Gigabitová karta, doporučovány jsou karty Intel, naopak se nedoporučuje používat integrované síťové karty

Pro databázový server požadavky vychází z publikovaných HW požadavků MySQL serveru. Pro větší gigabitové sítě se ovšem doporučuje co nejvýkonnější stroj, jeho rychlost zásadně ovlivňuje rychlost uživatelských dotazů.

Hardwarové požadavky webové rozhraní se odvíjejí od použitého webového serveru.

A.3 Instalace

Tato část popisuje instalaci zachytávací části aplikace a webového rozhraní. Celý programový balík je šířen v komprimované podobě v souboru `namon.tar.gz`. Tento soubor je třeba rozbalit pomocí příkazu:

```
tar xzvf namon.tar.gz [cílový adresář]
```

Pokud není uveden cílový adresář, je soubor rozbalen do současného adresáře. Po rozbalení vzniknou v cílové složce dva podadresáře:
`namon-cap` obsahující zachytávací část projektu a
`namon-web` obsahující webové rozhraní.

A.3.1 Instalace zachytávácí části

Monitorovací část programu NAMon se po rozbalení nachází v adresáři `namon-cap`.
Pozor! Některá nastavení programu je nutné provést před samotnou instalací, respektive kompilací programu! O nastavení více pojednává sekce A.4.3.
Instalace probíhá pro systém Unix standardní cestou a to serií příkazů (je nutné se nacházet v adresáři `namon-cap`):

1. `./configure` [1]
2. `make`
3. `make install`

Poté je nutné ještě provést inicializaci pomocných struktur v databázi. K tomu slouží právě nainstalovaný program `namon-db-init`. Je ho tedy třeba spustit:

```
namon-db-init
```

Během této inicializace budete vyzváni k zadání přihlašovacích údajů k databázi. Program vytvoří pomocné struktury v databázi a založí konfigurační soubor `.namonrc` (formát tohoto souboru je popsán v sekci A.4.3) v domovském adresáři. Po dokončení inicializace je program úspěšně nainstalován a připraven ke spuštění.

A.3.2 Instalace webového rozhraní

Webové rozhraní programu NAMon se po rozbalení nachází ve složce `namon-web`. Tyto soubory stačí jen zkopírovat do adresáře vašeho webového serveru.

A.4 Hlavní aplikace

A.4.1 Spuštění aplikace

Hlavní aplikace se spouští příkazem

```
namon <nazev_zarizeni> [parametry]
```

, kde `nazev_zarizeni` je název příslušné síťové karty (např. eth0, lnc0). Aplikaci lze spustit i s více parametry, které jsou detailně popsané v sekci A.4.3.

Při spuštění aplikace dojde k načtení konfiguračního souboru `.namonrc` v domovském adresáři, případně dalšího souboru, pokud je zvolen z příkazové řádky, až poté jsou čteny parametry na řádce. Stejné parametry jsou přepisovány, parametry na příkazové řádce tedy mají nejvyšší prioritu. Po spuštění aplikace lze sledovat aktuální provozní data stisknutím klávesy Enter (viz obrázek A.2). Tyto statistiky jsou vždy vztaženy k časovému

```
Statistiky behem poslednich 0.58s:
Doruceno paketu: 1988 (3442/s)
Pocet podobnych / novych paketu: 1186 / 40. Cachehit: [96.737%]
Velikost hashtabulky: 2618
-------------------------------------------------
Celkove doruceno: 100082
Celkove zahozeno: 0 (0.00%)
```

Obrázek A.2: Ukázka výstupu programu po zmáčknutí klávesy Enter.

období mezi dvěma výpisy, lze tedy tak sledovat aktuální zatížení. Program dále na obrazovku tiskne chybové informace, při zapnutém ladícím režimu i různé informace týkající se provozu aplikace.

Aplikace nepodporuje spuštění na pozadí (jako démon), proto je nutné ji spouštět na jiném terminálu či pomocí aplikace `screen`. V opačném případě dojde při odhlášení uživatele k ukončení aplikace.

A.4.2 Ukončení aplikace

Aplikace se ukončuje stisknutím CTRL+C, neboli doručením signálu SIGINT procesu. Aplikace se poté ještě před svým ukončením pokusí zapsat informace ze své cache do databáze. Toto vyprazdňování cache je však omezeno pětisekundovým intervalem, aplikace se po této době ukončí i v případě, že cache není zcela uvolněna.

A.4.3 Nastavení aplikace

Program umožňuje dva způsoby nastavování. Pro volbu výkon zásadně ovlivňujících atributů slouží hlavičkový soubor, který je třeba nastavit před samotnou instalací programu. Tato nastavení také ovlivňují vzhled webové aplikace. Pokud je příslušná volba vypnutá, nebude přístupná ani z webové aplikace. Ostatní, výkon méně ovlivňující atributy, je možné zadávat přímo z příkazové řádky během spuštění či pomocí konfiguračního souboru.

Nastavení pomocí hlavičkového souboru

Nastavení probíhá pomocí hlavičkového souboru `namon_config.h`, který se nachází v adresáři `namon-cap`. Daná volba se zapíná číslem 1, vypíná

[1]Pro vypsání všech možností programu configure slouží příkaz `./configure --help`.

pak 0. Příkladem zapnuté volby je např. `#define MAC_ENABLED 1`. Nastavit lze tyto parametry:

MAC_ENABLED - Zapíná podporu rozlišování MAC adres. Do databáze se bude ke každému záznamu zaznamenávat MAC adresa stroje / posledního routeru. Znamená to tedy další dva sloupce v databázi, zvětšení klíče a jistou režii navíc. Vzhledem k podstatnému nárustu velikosti databáze (cca dvakrát) je tato volba standardně vypnuta.

PORT_ENABLED - Zapíná podporu rozlišování portů. Pro každý datový tok se tak rozpoznává i cílový a zdrojový port. To potom znamená zvětšení databáze o tyto dva sloupce, ale také nárust počtu záznamů. Dojde totiž ke snížení agregačního poměru přidáním další klíčové položky. Pokud chcete monitorovat i použité síťové služby (na základě portů), zapněte tuto volbu. Pokud vám stačí informace o cílové a zdrojové stanici, nechte tuto volbu vypnutou.

STATS_ENABLED - Zapíná podporu provozních statistik samotného programu. Jedná se o počty přijatých a zahozených paketů, úspěšnost cache, počtu SQL dotazů během jednoho časového (AGGREGATE) intervalu. Doporučuje se nechat zapnuté, nastavení má minimální dopad na výkon.

AGGREGATE_INTERVAL - Jedná se o agregační interval použitý pro ukládaní dat, jde tedy o nejmenší rozlišitelnou časovou jednotku. Veškerá zobrazovaná data jsou pak průměrná data během tohoto intervalu. Tento parametr zásadně ovlivňuje přesnost měření, velikost celkové databáze a také cache-hit. Lze nastavit od jedné sekundy až po jednu hodinu. Rozumnou hodnotou z pohledu velikosti databáze je 300s. Hodnota se udává v sekundách a výsledné číslo musí být dělitelem čísla 3600.

SLEEP_INTERVAL - Udává dobu, na kterou se uspí vlákno ukládající záznamy z interní cache programu do samotné databáze. Ovlivňuje výsledný cache-hit, tedy i počet nutných dotazů do databáze. Zároveň však způsobuje zpoždění v zobrazovaní statistik, maximální zpoždění je dáno právě touto hodnotou. Pokud je důležitá aktuálnost zobrazovaných dat, zvolte tento interval malý. Pro minimalizaci dotazů do databáze pak hodnotu okolo 300s. Delší interval nemá smysl. Hodnota se udává v sekundách.

HASH_TABLE_MIN_SIZE - Jen pro zkušené uživatele. Udává počáteční a zároveň minimální velikost hashtabulky. Pokud nevíte, co je to hashtabulka, nebo si nejste opravdu jisti, neměňte toto nastavení. Tato hodnota musí být prvočíslo.

Po jakékoliv změně v tomto souboru je nutné program překompilovat a inicializovat pomocné struktury v databázi. Stačí znovu provést proces instalace popsaný v sekci A.3.1. Přiložený program `namon-db-init` pak automaticky provede potřebné změny v databázi a upozorní na případné problémy.

Nastavení pomocí voleb na příkazovém řádku či konfiguračním souboru

Program umožňuje pomocí přepínačů na příkazové řádce nastavovat tyto parametry:

- `-c <filename> --config <filename>` název konfiguračního souboru, který má být načten
- `-d <db> --database <db>` jméno datábaze v databázovém serveru
- `-F <filter> --pcap_filter <filter>` nastaví "`libpcap`-kompatibilní" filtr zachytávání paketu. Formát zadávání filtru je stejný jako u programu tcpdump. Kompletní popis je možno nalézt na `http://www.tcpdump.org/tcpdump_man.html`. Nutno uvést v úvozovkách.
- `-h --help` vypíše nápovědu
- `-p <pass> --password <pass>` přihlašovací heslo do k databázi
- `-P <port> --sql_port <port>` port, na kterém je spuštěn databázový server
- `-s <server> --server <server>` hostname/IP adresa MySQL serveru
- `-t <table> --table <table>` jméno databázové tabulky, do které se budou ukládat data
- `-u <sql-user> --sql-user <jmeno>` přihlašovací jméno k databázi
- `-v --verbose` zapne vypisování ladících informací
- `-V --version` vypíše číslo verze a ukončí se

Seznam těchto argumentů lze také vypsat pomocí příkazu `namon -h` nebo `namon --help`.

Konfigurační soubor

Všechny výše uvedené parametry s výjimkou parametru `-c` (`--config`) lze nastavit pomocí konfiguračního souboru. Každý řádek souboru odpovídá parametru příkazové řádky ve své dlouhé formě, který je následován znakem `=` a příslušnou hodnotou. Řádky začínající znakem # jsou

```
#Defaultni konfiguracni soubor programu Namon.
sql_server=localhost
sql_user=root
sql_password=topsecret
sql_database=namon
sql_table=troja
sql_port=3306
#pcap_filter=change_me
verbose=1
```

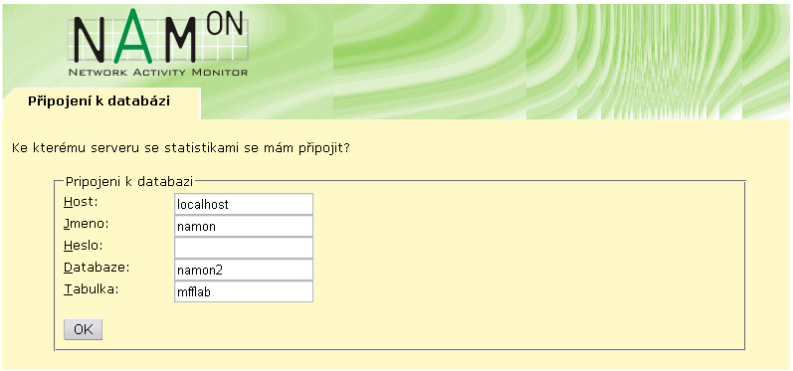
Obrázek A.3: **Ukázka konfiguračního souboru**

brány jako komentáře. Řádky, které neodpovídají žádnému z parametrů příkazové řádky jsou ignorovány, je však vypsáno varování. Řádky, které sice obsahují spravné klíčové slovo, ale mají nesmyslný obsah za rovnítkem, způsobí vypsání chyby a ukončení programu. Příklad validního konfiguračního souboru lze vidět na obrázku A.3.

Konfigurační soubor je automaticky vytvořen při instalaci a pokud neexistuje, tak i při spuštění programu. Vzhledem k tomu, že heslo do databáze je v souboru uloženo v otevřené podobě, je soubor vždy vytvářen jen s právy pro čtení a zápis daného uživatele.

A.5 Webové rozhraní

K prohlížení webového rozhraní slouží libovolný webový prohlížeč podporující javascript. Po zadání příslušné webové adresy dojde k načtení úvodní stránky (obrázek A.4).

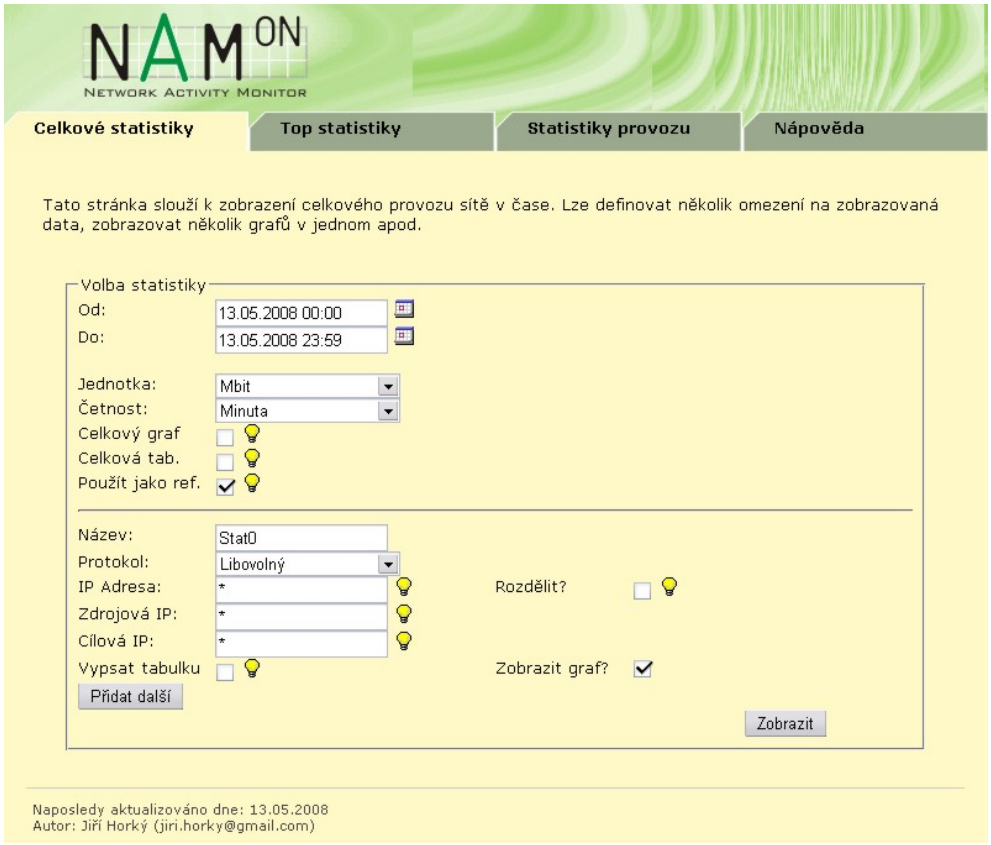


Obrázek A.4: **Úvodní obrazovka webového rozhraní**

Na této obrazovce dochází k přihlašování k příslušnému databázovému serveru. Při otevření této stránky dojde taky k ukončení případných spojení k jiným databázovým serverům. Předvyplněný text přihlašovacího formuláře lze ovlivnit nastavením příslušných položek v souboru `namon.php`. Jsou to položky:

```
var $db_host
var $db_user
var $db_pass
var $db_name
var $db_table
```

Toto nastavení má vliv pouze na předvyplněný text, zadáním jiných parametrů se lze příhlásit k jiným serverům. Po úspěšném přihlášení dojde k otevření hlavního okna webové aplikace (obrázek A.5).

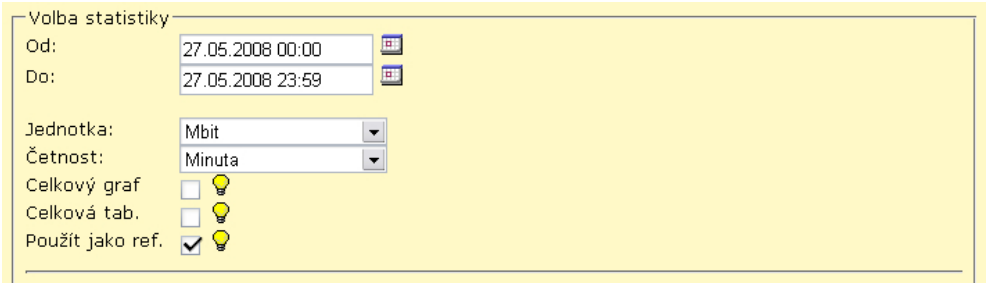


Obrázek A.5: **Vzhled hlavního okna webového rozhraní**

Hlavní okno je dle typu dotazů rozčleněno na tři hlavní části zobrazující grafy, čtvrtou částí je pak nápověda.

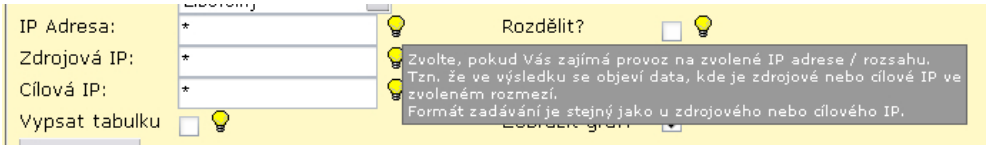
A.6 Ovládání

Všechny tři části webového rozhraní zobrazující grafy mají obsahují společné ovládací prvky, které jsou popsány zde. Prvky specifické pro konkrétní část webového rozhraní jsou popsány níže.



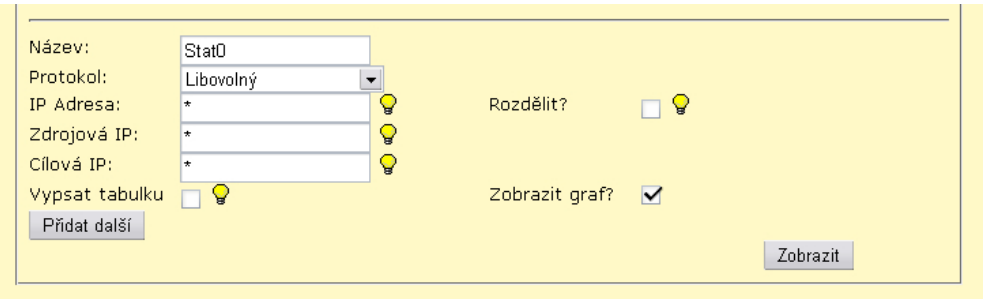
Obrázek A.6: **První (globální) část formuláře.**

Formulář sloužící k výběru zobrazovaných dat je rozdělen na dvě části. První část (viz obrázek A.6) slouží k nastavení časového období, jednotky grafu, případně jiných položek pro všechny grafy. Druhá část (obrázek A.8) slouží k vybrání příslušných omezení jednotlivých grafů, pomocí tlačítek `Přidat další` a `Odebrat` lze nastavovat jejich počet. Konkrétní ovládácí prvky jsou popsány níže. U každého významného



Obrázek A.7: **Ukázka nápovědy ve formě tipu.**

prvku je umístěn obrázek žárovky. Při najetí kurzoru na tento obrázek je zobrazena příslušná nápověda (obrázek A.7).



Obrázek A.8: **Druhá část formuláře.**

Nastavení pro všechny grafy

Volba časového období - K nastavování časového období pro všechny grafy slouží položky `Od:` a `Do:`. Čas se zadává ve formátu `DD.MM.YYYY hh:mm`, tedy například `27.05.2008 14:28`. Datum lze zapsat ručně nebo s pomocí interaktivního kalendáře, který se vyvolává kliknutím na ikonku kalendáře u příslušného pole.

Volba jednotky - Volba jednotky probíhá pomocí jednoduchého výběru ze seznamu. Data je možné zobrazit s jednotkou: Kbit, Mbit, kB, MB, GB. Tato jednotka je vázaná na všechny grafy.

Nastavení konkrétního grafu

Název - Tato položka slouží k pojmenování konkrétního grafu, respektive křivky grafu. Tento název bude vypsán v legendě grafu a ve vypsané tabulce.

Protokol - Slouží k výběru konkrétního transportního protokolu. Do výstupu budou zahrnuta jen ta data, která byla přenášena daným protokolem. Volba probíhá vybráním ze seznamu, na výběr jsou protokoly TCP, UDP a ICMP.

IP Adresa - Slouží k omezení dat na bázi IP adres, do výstupu budou zahrnuta jen ta data, která byla přenášena na nebo z dané adresy či rozsahu adres. Pole IP adresa akceptuje následující formáty:

- Celá IP adresa, například: `192.168.10.20`
- Část IP adresy s hvězdičkou, například `192.168.*`
- Rozsah IP adres v CIDR notaci, například `192.168.20.0/24`
- Rozsahu IP adres, například: `192.168.10.1-192.168.10.20`. Je nutné uvést obě IP adresy v plném rozsahu, druhá IP adresa musí být vyšší, než první.

- Negace předchozích formátů, například: `!192.168.*` nebo `!10.0.0.0-10.1.0.0`
- Kombinace předchozích formátů. Separátor je čárka. Například: `192.168.10.20, 10.7.*, !10.7.0.12`

Zdrojová IP adresa - Slouží k omezení dat na bázi zdrojových IP adres. Do výstupu budou zahrnuta jen ta data, která pocházela z dané IP adresy / rozsahu adres. Formát vstupu je popsán výše.

Cílová IP adresa - Slouží k omezení dat na bázi cílových IP adres. Do výstupu budou zahrnuta jen ta data, která směřovala na danou IP adresu / rozsahu adres. Formát vstupu je popsán výše.

MAC adresa, Zdrojová MAC adresa, Cílová MAC adresa - Tyto položky fungují stejně jako odpovídající položky s IP adresou. Tyto možnosti se zobrazí, pouze pokud byl program nainstalován s podporou rozlišování MAC adres. Pole MAC adresa akceptuje následující formáty:

- Celá MAC adresa, například: `00:0f:f0:cd:10:20`
- Část MAC adresy s hvězdičkou, například `00:50:*`
- Negace předchozích formátů, například: `!00:0f:f0:cd:10:20` nebo `!00:50:*`
- Kombinace předchozích formátů. Separátor je čárka. Například: `00:0f:f0:cd:10:20, 00:50:*, !00:50:40:*`

Port, Zdrojový port, Cílový port - Tyto položky fungují stejně jako odpovídající položky s IP či MAC adresou. Tyto možnosti se zobrazí, pouze pokud byl program nainstalován s podporou rozlišování portů. Pole port akceptuje následující formáty:

- Jeden port, například: `80`
- Rozsah portů, například `1-1024`
- Negace předchozích formátů, například: `!80` nebo `!1-1024`
- Kombinace předchozích formátů. Separátor je čárka. Například: `1-1024, 6000, !80`

Rozdělit - Pokud je toto zaškrtávací políčko u příslušné položky zvoleno, ve výsledku se zobrazí 3 grafy. Prvním z nich bude graf provozu, kde *cílová* IP adresa, MAC adresa, či port byla zvolená hodnota. V dalším budou data, kde *zdrojová* IP adresa, MAC adresa, či port byly ve zvoleném rozsahu. Třetím budou data, kde *zdrojová* i *cílová* IP adresa, MAC adresa, či port byly ve zvoleném rozsahu. Lze tak tedy například zobrazit aktivitu části sítě rozdělenou na upload, download a aktivitu uvnitř této sítě.

Zobrazit graf - Pokud je tato položka zaškrtnuta, bude zobrazen příslušný graf. Pokud ne, graf se ve výsledku neobjeví.

Zobrazit tabulku - Pokud je tato položka zaškrtnuta, bude zobrazena i příslušná tabulka dat rozdělená. Pokud ne, tabulka se ve výsledku neobjeví. Pokud není zvolena ani jedna z možností `Zobrazit graf` nebo `Zobrazit tabulku`, nebudou z daného grafu zobrazena žádná data. To se může hodit v případě, že je definováno grafů víc a ten se tak stává nepřehledným. Nechceme však přijít o nastavení tohoto grafu, a proto místo tlačítka `Odebrat` jen odškrtneme položky zobrazit graf a zobrazit tabulku.

A.7 Celkové statistiky

V této záložce probíhají dotazy na celkový provoz sítě v čase. Lze definovat několik omezení na zobrazovaná data, zvolit jednotku a časové období, jak bylo popsáno výše. Pokud je definováno více grafů, budou zobrazeny v jednom. Kromě již uvedených možností jsou k dispozici ještě další:

Časový interval - Slouží k nastavení agregačního intervalu, ve kterém budou data zobrazována. Pokud je např. zvolen jako 1 hodina, bude zobrazen průměrný hodinový provoz v daném časovém období. Pokud je zvolen jako minuta, budou zobrazena data s nejmenší možným agregačním intervalem, tj. stejným intervalem, který byl zvolen před instalací v souboru `namon_config.h`.

Celkový graf - Ve výsledném grafu bude zobrazena celková zátěž sítě v daném období, tedy data bez žádných omezení.

Celková tabulka - Ve výsledku bude zobrazena tabulka obsahující informace o celkové zátěži sítě v daném omezení, tedy data bez žádných omezení.

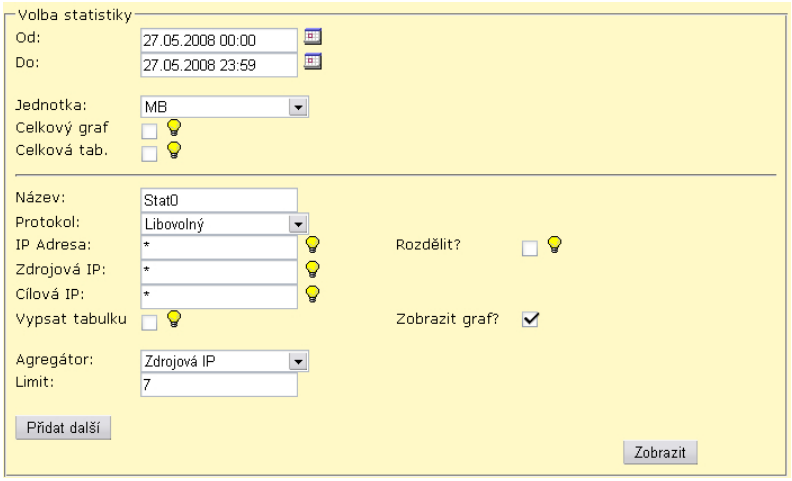
Použít jako referenční - Ve výsledné tabulce budou u ostatních statistik spočítány procentuální hodnoty k celkové statistice. Má smysl zvolit, pouze pokud je zaškrtnuta volba `Celková tabulka`.

Konkrétní příklad použití naleznete v sekci A.11.1.

A.8 TOP statistiky

V této záložce se lze dotazovat na nejaktivnější IP adresy (uživatele), porty (aplikace) či poměr použitých protokolů během zvoleného období.

Podobně jako u *Celkové statistiky* lze zvolit několik omezení na zobrazovaná data, jak toho lze dosáhnout je již popsáno výše. Konkrétní hodnoty pak lze vypsat do tabulky, kde dochází k případnému překladu IP adres na doménová jména.



Obrázek A.9: **TOP statistiky, ukázka rozhraní**

V této záložce se nachází, kromě již uvedených ovládacích prvků, další prvky (obrázek A.9):

Celkový graf - Ve výsledném grafu bude zobrazena celková zátěž sítě v daném období, tedy data bez žádných omezení. Agregátorem tohoto grafu bude první agregátor grafů uvedných níže.

Celková tabulka - Ve výsledku bude zobrazena tabulka obsahující informace o celkové zátěži sítě v daném omezení, tedy data bez žádných omezení. Agregátorem této tabulky bude první agregátor grafů uvedných níže.

Agregátor - Nastavuje, ze které položky se budou tvořit TOP statistiky. Je-li tedy nastavena např. na cílovou IP adresu, ve výsledku se zobrazí ti uživatelé (IP adresy), kteří v daném období nejvíce stahovali.

Limit - Nastavuje maximální počet položek zobrazených v grafu a tabulce.

Konkrétní příklad použití naleznete v sekci A.11.2.

A.9 Statistiky provozu

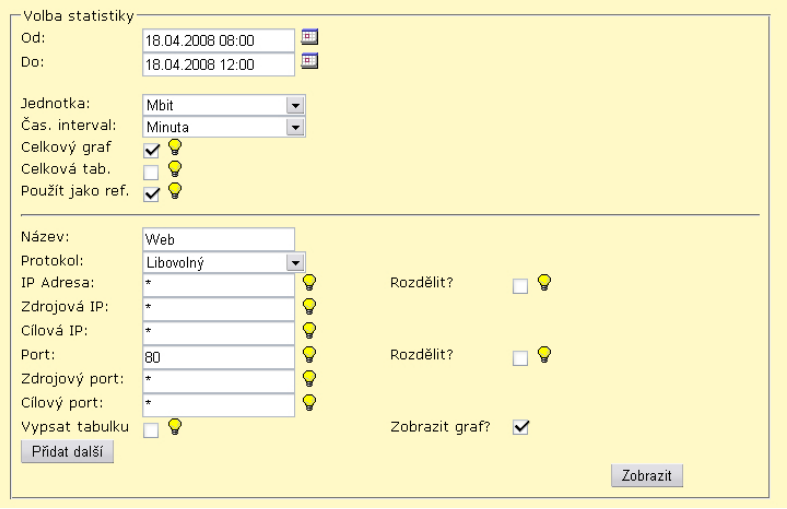
Tato záložka slouží ke sledování provozních statistik. Jedná se o sledování počtu přijatých a zahozených paketů, cache-hit interní cache programu a počtu dotazů do databáze ve zvoleném časovém období. Jedná se vždy o průměrné hodnoty během nastaveného časového intervalu. Součástí výstupu je také tabulka hodnot ve všech časových intervalech ve zvoleném období. Jediným možným nastavením je zvolení časového období.

A.10 Nápověda

Tato záložka obsahuje konkrétní příklady použití webové aplikace. Jedná se o stejné příklady jako jsou v této dokumentaci.

A.11 Příklady použití

A.11.1 Celkové statistiky

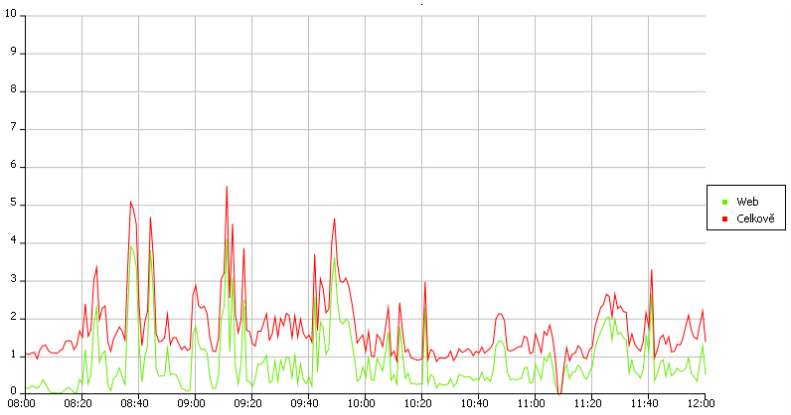


Obrázek A.10: **Příklad použití celkových statistik**

V tomto příkladu byl monitorován provoz firemní sítě. Monitorování probíhalo pomocí port-mirroringu na centrálním routeru připojeném do internetu. Správce sítě zajímá celkový provoz této sítě během večera a dále

poměr webových služeb na celkovém provozu. Provede proto nastavení ukázaná na obrázku A.10.

Po stisknutí tlačítka `Zobrazit` dojde k vykreslení grafu a vypsání tabulky, jak ukazuje obrázek A.11.



Celková statistika

Typ	Web(Mbit)	Celkově(Mbit)
Min	0.00[1.40%]	0.00
Max	4.11[74.70%]	5.50
Avg	0.84[48.93%]	1.72

Obrázek A.11: **Příklad použití celkových statistik, výsledný graf**

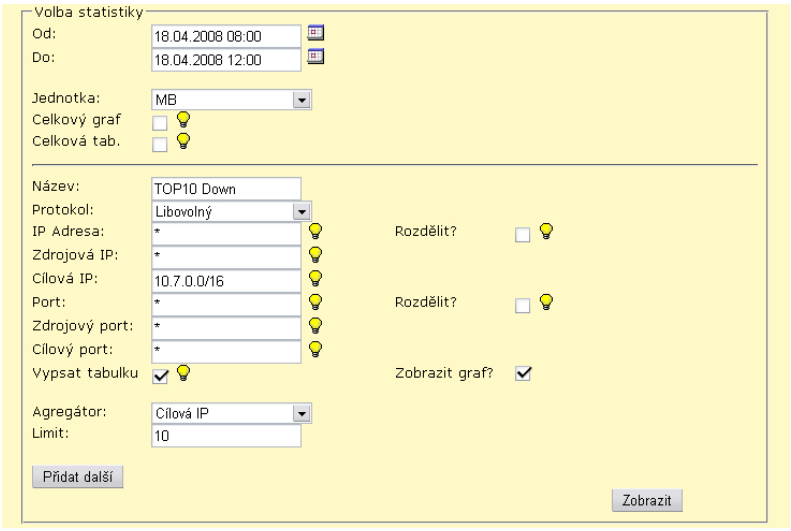
Administrátor sítě tedy získal požadované informace. Zjistil, že webový provoz tvořil cca 50% veškerého provozu, což se zdá být v pořádku.

A.11.2 TOP statistiky

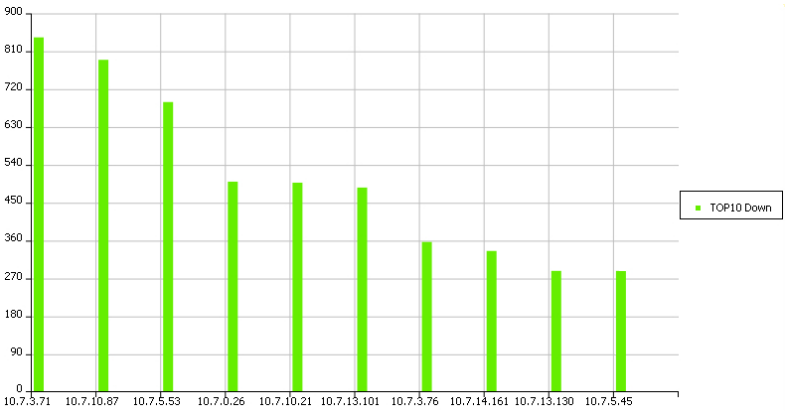
Monitorován je provoz stejné firemní sítě jako v příkladě A.11.1. Tentokrát správce zajímají uživatelé (IP adresy) ze sítě 10.7.0.0/16, kteří během daného období nejvíce stahovali. Provede tedy nastavení jako na obrázku A.12.

Po stisknutí tlačítka `Zobrazit` dojde k vykreslení grafu a vypsání tabulky, jak ukazuje obrázek A.13

Administrátor sítě tedy získal požadované informace. Vzhledem k tomu, že IP adresy v tabulce nemají vlastní DNS záznam, jsou v závorkách místo doménového jména uvedeny opět IP adresy.



Obrázek A.12: **Příklad použití TOP statistik**



Tabulková statistika (TOP10 Down)

dst_ip	TOP10 Down(MB)
10.7.3.71(10.7.3.71)	843.10
10.7.10.87(10.7.10.87)	789.77
10.7.5.53(10.7.5.53)	689.41
10.7.0.26(10.7.0.26)	500.27
10.7.10.21(10.7.10.21)	497.86
10.7.13.101(10.7.13.101)	486.25
10.7.3.76(10.7.3.76)	356.97
10.7.14.161(10.7.14.161)	335.57
10.7.13.130(10.7.13.130)	288.24
10.7.5.45(10.7.5.45)	288.16

Obrázek A.13: **Příklad použití TOP statistik, výsledný graf**

Příloha B

Programátorská dokumentace

B.1 Technické informace

Hlavní, zachytávací, část aplikace je napsaná v jazyce C, webové rozhraní pak pomocí jazyka PHP 5.0, HTML, CSS a Javascript. Jako databázový server je použit server MySQL. Zachytávací část aplikace je napsaná pro platformu Unix, webové rozhraní je pak platformě nezávislé. K jeho prohlížení slouží libovolný webový prohlížeč podporující javascript.

Softwarové požadavky

Pro zachytávací část aplikace jsou potřeba následující položky:

- *knihovna GLiB v2.16.1 nebo vyšší* - z této knihovny byla použita implementace hashtabulky, funkce zaručující atomické operace s čísly a ukazateli a funkce umožňující efektivnější alokaci paměti (slice allocator).
- *knihovna libpthread* - tato knihovna slouží pro podporu vláken v aplikaci.
- *knihovna libmysqlclient* - tato knihovna je zde kvůli podpoře MySQL serveru, slouží k připojení k databázi a podpoře dotazování přímo z aplikace.
- *knihovna libpcap* - knihovna obsahuje API pro zachytávání paketů, obsahuje možnost poměrně komplexních filtrů paketů, je multiplatformní.
- *`autotools`* - nástroje sloužící k automatickému vytváření `configure` skriptů a `Makefile` souborů dle dané šablony. Nutné pouze při úpravě kódu.

Webová část aplikace nevyžaduje, kromě samotného webového serveru, instalaci žádné další knihovny.

B.2 Instalace

K usnadnění instalace a kompilace zachytávacího programu na různých verzích Unixových systémů stejně byly použity nástroje `autotools`. Byla tedy napsána konfigurační šablona nacházející se v souboru `congigure.ac` sloužící k vygenerování konfiguračního skriptu `configure`. Spolu s příslušnými šablonovými `makefile` soubory, které jsou pojmenované `Makefile.am` a nachází se v každém adresáři obsahující zdrojový kód, pak dojde při spuštění příkazu `./configure` k vygenerování souborů `Makefile`.

Z důvodů usnadnění instalace byl také napsán program `namon-db-init` nacházející se v adresáři `namon-cap/src/sql-install`. Tento program slouží k inicializaci databáze, vytvoření pomocných tabulek a také k vytvoření konfiguračního souboru.

B.3 Struktura databáze

Data v databázi jsou ukládána do tabulek pojmenovaných dle zvoleného názvu a konkrétního dne, výsledný název je tedy `<nazev>YYYYMMDD`. Každý den má tedy vlastní tabulku. Tato tabulka má následující strukturu:

Tabulka B.1: Struktura hlavní tabulky

Sloupec	Typ	Primární klíč
`src_ip`	`int(10)`	Ano
`dst_ip`	`int(10)`	Ano
`src_port`	`smallint(5)`	Ano
`dst_port`	`smallint(5)`	Ano
`src_mac`	`varchar(20)`	Ano
`dst_mac`	`varchar(20)`	Ano
`protocol`	`enum`	Ano
`size`	`bigint(20)`	Ne
`count`	`mediumint(8)`	Ne
`hour`	`tinyint(4)`	Pouze klíč (ne primární)

Sloupce `src_port`, `dst_port`, `src_mac` a `dst_mac` se v tabulce a tedy i v primárním klíči objevují, pouze pokud byla při instalaci podpora příslušné položky zapnutá. Sloupec `time` značí začátek časového intervalu, ve kterém byly pakety zachyceny, `count` jejich počet a `size` jejich celkovou velikost.

Kromě této tabulky existují ještě pomocné tabulky, které jsou potřebné k chodu aplikace a které při instalaci vytváří program `namon-db-init`.

Jedná se o tabulku `<nazev>_settings` (viz tabulka B.2) nesoucí nastavení programu během instalace (délka časového intervalu, povolení zachytávání portů a MAC adres). Tato tabulka obsahuje jediný řádek.

Tabulka B.2: Struktura tabulky nastavení

Sloupec	Typ	Primární klíč
`port_enabled`	`tinyint(1)`	Ne
`mac_enabled`	`tinyint(1)`	Ne
`interval`	`int(11)`	Ne

Dále tabulka `<nazev>_statsYYYYMMDD` (viz tabulka B.3) obsahující provozní informace programu v daný den.

Tabulka B.3: Struktura tabulky provozních statistik

Sloupec	Typ	Primární klíč
`time`	`(time`	Ano
`received`	`bigint(20)`	Ne
`dropped`	`bigint(20)`	Ne
`already`	`bigint(20)`	Ne
`new`	`bigint(20)`	Ne

Další pomocnou tabulkou je tabulka `dummy` (tabulka B.4) neobsahující žádná data, sloužící pouze k vracení prázdných dotazů.

Tabulka B.4: Struktura dummy tabulky

Sloupec	Typ	Primární klíč
`day`	`int(8)`	Ne
`agregator`	`int(10)`	Ne
`size`	`bigint(20)`	Ne

Poslední dvě pomocné tabulky pojmenované `<nazev>time_table` (tabulka B.5) a `<nazev>time_table_hour` (tabulka B.6) slouží k zajištění

správnosti časových intervalů při výstupu. Tabulka `time_table` obsahuje hodnoty začátků včech časových intervalů během jednoho dne (pro pětiminutový interval například 00:00:00, 00:05:00, 00:10:00 atd.), tabulka `time_table_hour` obsahuje seznam hodin během jednoho dne (čísla 0-23).

Tabulka B.5: Struktura time_table tabulky

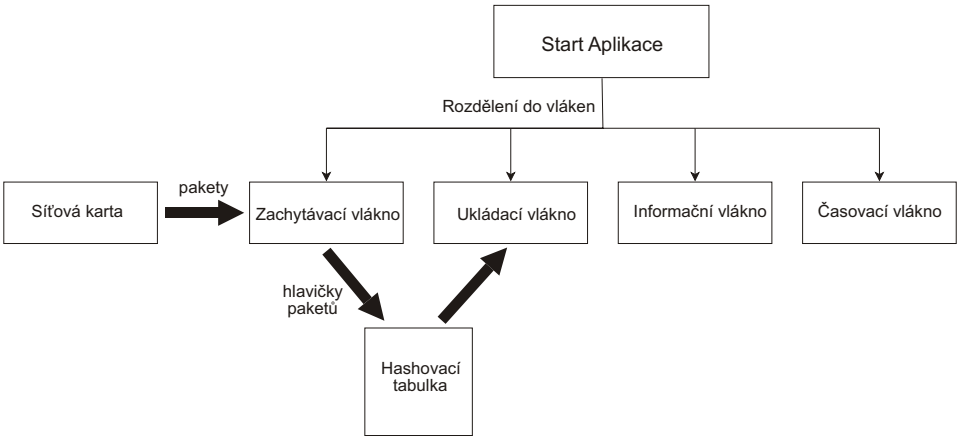
Sloupec	Typ	Primární klíč
`time`	`time`	Ano

Tabulka B.6: Struktura time_table_hour tabulky

Sloupec	Typ	Primární klíč
`hour`	`tinyint(4)`	Ano

B.4 Hlavní aplikace

B.4.1 Architektura programu



Obrázek B.1: **Architektura NAMon**

Architekturu programu zobrazuje obrázek B.1. Program se po spuštění rozdělí na 4 vlákna.

Zachytávací vlákno provádí vlastní příjem paketů a jejich filtrování. Následně je pak ukládá do hashovací tabulky, respektive aktualizuje její záznamy v případě, že se paket shoduje v klíčových parametrech. Klíčové parametry jsou: zdrojová a cílová IP adresa, použitý protokol, čas doručení paketu a případně také zdrojovou a cílovou MAC adresa a zdrojový a cílový port, je-li program zkompilován s těmito možnostmi. Toto vlákno je spuštěno s maximální prioritou.

Ukládací vlákno je spuštěno s minimální prioritou. Pokud není hashovací tabulka prázdná, začne vlákno data ukládat z tabulky do samotné databáze. Pokud je hashovací tabulka prázdná, uspí se na nastavenou dobu (sleep interval). Ve skutečnosti nedochází k současnému přístupu obou vláken do hashovací tabulky. Z tohoto důvodu existují v programu dvě instance hashtabulky. V případě, že je třeba data z hashtabulky uložit do samotné databáze, dojde jen k atomickému prohození ukazatelů na aktivní hashtabulku. Zachytávácí vlákno pak ukládá data do jiné hashtabulky, než ze které ukládací vlákno data odebírá. Vyhneme se tak nutnému a drahému uzamykání hashtabulky pomocí mutexu při každém vkládání paketu.

Informační vlákno slouží k zobrazování výstupu programu (převážně ladící informace), k zachytávání uživatelských vstupů a následnému vypisování aktuálního provozního stavu.

Časovací vlákno. Jeho úkolem je nastavování aktuálního časového intervalu, slouží také k přepínaní databázových tabulek při přechodu do následujícího dne. Pro toto nastavování byl použit vlastní formát čísla umožňující rychleji zjistit čas pro každý paket. Tento formát je popsán v sekci B.5.1.

B.4.2 Zdrojové soubory

Zdrojový kód aplikace je rozdělen do několika souborů (viz obrázek B.2). Aplikace původně měla umožňovat ještě další možnosti, z časových důvodů k jejich plné implementaci však již nedošlo. Proto jsou ve zdrojových kódech k nalezení fragmenty kódu, které jsou zakomentované a připravené pro budoucí verzi programu.

Následuje popis významu jednotlivých souborů. V tomto manuálu nejsou rozebrány jednotlivé funkce programu, jejich popis je vždy k nalezení v příslušném zdrojovém souboru.

```
.
|-- MAC.h
|-- common.h
|-- conf.c
|-- conf.h
|-- data.c
|-- data.h
|-- [glib]
    `-- ghash.c
|-- if_ether.h
|-- malloc.c
|-- malloc.h
|-- namon-cap.c
|-- namon-cap.h
|-- namon_config.h
|-- namon_mysql.c
|-- namon_mysql.h
|-- port.h
|-- [sql-install]
    `-- namon-install.c
```

Obrázek B.2: Seznam zdrojových souborů hlavního programu

Popsán je vždy obsah příslučného zdrojového souboru s příponou `.c`, hlavičkové soubory obsahující jen seznam funkcí nejsou v seznamu uvedeny.

`MAC.h` - obsahuje makra použitá při povolení, respektive vypnutí MAC adres. Jedná se o části MySQL dotazů apod.

`common.h` - obsahuje společné definice konstant (maximální délky řetězců), velikosti vnitřních struktur apod. Tento soubor je vložen pomocí `#include\` ve všech zdrojových souborech.

`conf.c` - obsahuje funkce k inicializaci, načítání a ukládání konfiguračního souboru.

`conf.h` - kromě definice funkcí ze souboru `conf.c` obsahuje také definici struktury `struct conf_t`, která slouží k uchovávání nastavení načteného z příkazové řádky či konfiguračních souborů.

`data.c` - obsahuje celý kód ukládacího vlákna. V tomto souboru jsou veškeré funkce spojené s ukládáním a odebírání dat do/z hashtabulky.

`glib/ghash.c` - obsahuje upravený kód implementace hashtabulky z knihovny `GLiB`. Upravena je především počáteční velikost hashtabulky,

která v původní verzi byla nastavena na 11. V této verzi je velikost konfigurovatelná pomocí souboru `namon_config.h`.

`if_ether.h` - obsahuje definice ethernetových protokolů, respektive jejich čísel. Jedná se upravený standardní hlavičkový soubor operačního systému Linux. Vzhledem k tomu, že na systému FreeBSD podobný soubor neobsahuje, je v případě kompilace na FreeBSD použit.

`malloc.c` - obsahuje funkce na alokaci a uvolňování paměti pro datovou strukturu použitou v hashtabulce. Původní myšlenkou bylo použít rychlé alokování pomocí tzv. memory slices knihovny GLiB, od toho však bylo upuštěno, a proto se zde volá jen standardní funkce `malloc`.

`namon-cap.c` - hlavní soubor celého programu. Obsahuje zachytávací, informační a časovací vlákno. Mimojiné obsahuje také funkci `main`, ve které dochází k analyzování vstupních parametrů, inicializaci spojení k MySQL, kontrole tabulek apod.

`namon-cap.h` - obsahuje definici struktury `struct namon_t`, která slouží jako nosná proměnná v celém programu. Jsou v ní uloženy veškeré nastavení, obsahuje také ukazatel na připojení k databázi apod. Instance této struktury je v programu globální.

`namon_config.h` - obsahuje různá nastavení, od kterých se pak odvíjí funkce programu (povolení zachytávání portů, MAC adres apod, nastavení různých časových intervalů). Toto nastavování probíhá pomocí makra `#define`, v jednotlivých zdrojových souborech je pak toto nastavení testováno pomocí `#ifdef`. Toto je jediný soubor, který by měl editovat samotný uživatel aplikace.

`namon_mysql.c` - obsahuje funkce potřebné pro práci s MySQL databází, jako je inicializace spojení, mazání a zakládání tabulek apod.

`port.h` - má podobný význam jako soubor MAC.h s tím rozdílem, že se makra uvedená v souboru týkají portů a ne MAC adres.

`sql-install/namon-install.c` - obsahuje zdrojový kód pomocné aplikace `namon-db-init`, která slouží k inicializaci databáze a založení konfiguračního souboru po instalaci.

B.5 Speciální formáty použité v aplikaci

B.5.1 Formát času použitý v aplikaci

Kvůli rychlosti získávání konkrétního času pro každý paket bylo v aplikaci použito atomické číslo, které periodicky nastavuje časovací vlákno.

Místo standardního formátu čísla typu `time_t` byl použit vlastní formát. Standardně by bylo totiž nutné pro získání konkrétního času volat funkci `localtime`, ta je však dle měření velmi pomalá. Použito je 4 bytové číslo, které s použitím správných funkcí dokáže zaručit atomické operace přiřazení. Myšlenka tkví v bitovém ukládání jednotlivých částí času (rok, měsíc, den, hodina, sekunda) do příslušných částí 32 bitového čísla. To umožní rychlejší extrahování jednotlivých časových období pomocí bitových operací. Na jednotlivé časové období byly vyhrazeny následující počty bitů:

- Rok - 6 bitů, tedy hodnoty 0-63. Podporány jsou jen roky 2000-2063.
- Měsíc - 4 bity
- Den - 5 bitů
- Hodina - 5 bitů
- Minuta - 6 bitů
- Sekunda - 6 bitů

Obrázek B.5.1 ukazuje příklad použití tohoto formátu a pořadí jednotlivých období. Zobrazeným časem je 13:48:00 28.05.2008.

```
001000|01  01|11100|0  1101|1100  01|000000
------|------|-----|-------|--------|-------
 rok  |mesic | den |hodina | minuta |sekunda
```

Obrázek B.3: Speciální formát času

B.5.2 Vytváření klíče do hashtabulky

Klíč do hashtabulky je tvořen zdrojovou a cílovou IP adresou, časovým intervalem příchodu paketu a případně zdrojovým a cílovým portem a zdrojovou, cílovou MAC adresou, pokud jsou příslušné položky povoleny.

Kvůli maximálnímu zrychlení vytváření vyhledávacího klíče byl nasazen vlastní způsob jeho vytváření namísto prostého vypsání příslušných klíčových položek za sebou do řetězce pomocí funkce `sprintf`, která je relativně pomalá.

Myšlenka spočívá v tištění jednotlivých bytů čtyřbytového (IP adresa) či dvoubytového čísla (porty) za sebou do řetězce. Pomocí bitové operace AND a bitových posunů jsou z těchto čísel po řadě získávány jednotlivé byty. Ty jsou pak zapisovány do pole bytů představující řetězec. Je však třeba si dát pozor na případné nulové byty klíčových položek. Ty by

při vypisování řetězce způsobily jeho ukončení. Případné nuly jsou tedy nahrazeny jedničkou a pozice této nuly je zapsána na příslušnou bitovou pozici pomocného čísla. Toto pomocné číslo je pak vytištěno na konci řetězce. Tímto dostaneme rychlé a jednoznačné zobrazení mezi klíčovými položkami a klíčem do hashtabulky.

B.6 Webové rozhraní

Webové rozhraní je naprogramováno v jazyce PHP 5.0, k dotazování databáze je použit jazyk SQL. K zobrazovaní grafů a interaktivních prvků je použit Javascript, celkový vzhled aplikace se pak ovlivňuje pomocí kaskádových stylů (CSS).

B.6.1 Použité komponenty

Webové rozhraní ke svému běhu nevyžaduje instalaci žádné knihovny, několik komponent přesto obsahuje. Ty jsou šířeny spolu s programem. Použity jsou následující komponenty:

- `Chart 1.0 - WebFX` - soubor javascriptových funkcí sloužící ke generování grafů. Viz `http://www.webfx.nu`
- `JS Calendar` - javascriptové funkce sloužící k zobrazení interaktivního kalendáře. Viz `http://www.dhtmlgoodies.com`
- `qTip` - jednoduchý skript v javascriptu umožňující vypsání formátovaného textu po najetí kurzorem na daný prvek. Viz `http://qrayg.com/learn/code/qtip/`

B.6.2 Architektura

Webová aplikace je psána objektově, aplikace obsahuje celkem čtyři třídy. Následuje jejich základní popis a role v aplikaci. Podrobný popis jednotlivých metod a atribitů tříd je obsažen v příslušných zdrojových kódech aplikace.

`Namon` - toto je hlavní třída webové aplikace. Obsahuje jako atributy všechna nastavení programu, tato třída také zodpovídá za přihlašování uživatele na úvodní stránce. Jako své proměnné obsahuje třídy `Stats` a `Chart`. Během běhu aplikace existují maximálně dvě instance této třídy - jedna pro celkové statistiky, druhá pro TOP statistiky. Statistiky provozu jsou pro svou jednoduchost řešeny bez této třídy. V aplikaci tato třída představuje vždy celé okno

statistik (TOP, celkové). Mezi jednotlivými stránkami je tato třída se všemi proměnnými předávána pomocí techniky sessions.

`Stats` - tato třída v aplikaci představuje tu část formuláře, kde probíhá vybírání jednotlivých omezení grafu. Počet těchto částí může být více (přidání pomocí tlačítka `Přidat`), každá tato část formuláře pak odpovídá jedné instanci této třídy. Třída obsahuje metody na kontrolu uživatelského vstupu tohoto formuláře.

`Chart` - třída zodpovědná za vykreslování grafů a výsledných tabulek. Úkolem této třídy je nastavení správného měřítka, správných popisků grafu a vyvolání správného dotazu pomocí třídy `Query`, která je součástí této třídy jako její proměnná. V záložce celkových statistik existuje jen jedna instance(probíhá vypisování více křivek do jednoho grafu), v TOP statistikách pak případně více (každá statistika má svůj vlastní graf).

`Query` - třída zodpovědná za vytváření správných dotazů do databáze v závislosti na vstupu uživatele. Výsledky dotazů se ukládají do dočasných pomocných tabulek, vzhledem k nutnosti pracovat s těmito daty vícekrát. Proto třída obsahuje metody na práci s touto tabulkou včetně její zrušení.

Kromě těchto tříd obsahuje aplikace několik obecných pomocných funkcí obsažených v souboru `functions.php`.

B.6.3 Zdrojové soubory

Aplikace je členěna do několika zdrojových souborů a adresářové struktury, jak lze vidět na obrázku B.4.

Následuje popis významu jednotlivých souborů. V tomto manuálu nejsou více rozebrány jednotlivé funkce programu, jejich popis je vždy k nalezení v příslušném zdrojovém souboru.

`calendar.ini` - obsahuje HTML kód potřebný pro zobrazení kalendáře. Je inkludován soubory `stats.php`, `statstop.php` a `health.php`.

`chart.php` - obsahuje třídu `Chart`, nejedná se o samostatnou stránku.

`editform.php` - tato stránka slouží pro vyvolání příslušných rutin při přidávání a odebírání statistik ve formuláři, slouží také k vyvolání obnovení stránky po kliknutí na tlačítka `Přidat` a `Odebrat`.

`foot.ini` - obsahuje HTML kód záhlaví stránky. Je includován ve všech souborech sloužících k výstupu.

`functions.php` - obsahuje obecné funkce použité v celé aplikaci.

```
.
|-- calendar.ini
|-- [chart]
|   `-- složka obsahující zdrojové soubory knihovny WebFX
|-- chart.php
|-- constants.php
|-- [dhtmlgoodies_calendar]
|   `-- složka obsahující zdrojové soubory komponenty JS Calendar
|-- editform.php
|-- foot.ini
|-- functions.php
|-- header.ini
|-- health.php
|-- help.php
|-- [images]
|   `-- složka obsahující obrázky použité v aplikaci
|-- index.php
|-- menu.php
|-- namon.php
|-- query.php
|-- statclass.php
|-- stats.php
|-- statstop.php
|-- [style]
|   `-- složka obsahující kaskádové styly
|-- [tooltip]
    `-- složka opsahující zdrojové soubory komponenty qTip
```

Obrázek B.4: Seznam zdrojových souborů webového rozhraní

`header.ini` - obsahuje HTML kód hlavičky stránky. Je includován ve všech souborech sloužících k výstupu.

`health.php` - jedná se o kód stránku obsahující provozní statistiky.

`help.php` - jedná se o kód stránky obsahující nápovědu.

`index.php` - úvodní obrazovka webového rozhraní. Na této stránce dochází po úspěšném přihlášení k inicializaci sessions, třídy `namon` a k přesměrování uživatele na stránku `stats.php`.

`menu.php` - představuje menu webové aplikace. Tento soubor je includován ve všech souborech sloužících k výstupu.

`namon.php` - obsahuje zdrojový kód třídy `namon`, nejedná se o samostatnou stránku.

`query.php` - obsahuje zdrojový kód třídy `query`, nejedná se o samostatnou stránku.

`statclass.php` - představuje stránku sloužící k zobrazování TOP statistik.

`stats.php` - představuje stránku sloužící k zobrazování celkových statistik, tato stránka se načítá po úspěšním přihlášení.

`statstop.php` - představuje stránku sloužící k zobrazování TOP statistik.

Příloha C

Obsah přiloženého CD

```
.
|-- [src]            - zkomprivaný zdrojový kód obou části aplikace
|   `-- namon.tar.gz
`-- [doc]            - text bakalářské práce v různých formátech
    |-- bakalarska_prace.pdf
    |-- bakalarska_prace.ps
    `-- bakalarska_prace.dvi
```

Obrázek C.1: Obsah přiloženého CD