Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Sledování průtoků dat na síti linuxových routerů

Karel Borkovec

Bakalářská práce

2008

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Karel BORKOVEC**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Sledování průtoků dat na síti linuxových routerů**

Zásady pro vypracování:

Teoretická část: Vytvořte přehled nástrojů pro sledování průtoku dat na linuxových routerech. Porovnejte jejich vlastnosti, nároky na HW a SW, snadnost instalace a správy. Zaměřte se na platformu embedded-Linux a navrhněte vhodné nástroje pro sledování průtoku dat. Praktická část: Pro vybrané řešení sledování na embedded Linuxu navrhněte způsob předávání dat do centrální databáze. Navrhněte strukturu databáze pro ukládání naměřených hodnot tak, aby bylo možné z ní vyčíst pro vybraný router rychlosti za stanovené časové období a vygenerovat graf průtoku (s rozlišením např. 5 minut).Navrženou databázi vytvořte (např. na MySQL) a nad ní vytvořte nástroj pro získávání dat do ní uložených včetně generování grafů zátěže (např. pomocí apache + php + rdd). Pozn.: Počítejte s tím, že na jednom routeru je možné měřit více různých průtoků dat, jednotlivé průtoky je třeba odlišit podle směru toku dat (odkud kam).

Rozsah grafických prací:
Rozsah pracovní zprávy:
Forma zpracování bakalářské práce: **tištěná/elektronická**
Seznam odborné literatury:

Žádná doporučená literatura

Vedoucí bakalářské práce: **Mgr. Tomáš Hudec**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2009**
Termín odevzdání bakalářské práce: **15. května 2009**

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 17. 1. 2009

Karel Borkovec

Poděkování

Chtěl bych poděkovat panu Mgr. Tomáši Hudcovi za vedení mé práce, užitečné rady a informace a také za umožnění práce na tak zajímavém tématu.

Anotace

Práce je věnována sledování průtoků dat na linuxových směrovačích. Naměřená data v podobě počtu paketů a bytů jsou periodicky čtena a odesílána na centrální server. Jsou ukládána do databáze, ze které je možno generovat grafy průtoku a zobrazovat je v přehledné podobě ve webová vyhodnocovací aplikaci.

Klíčová slova

Linux, směrovač, síťový provoz, sledování, rrd

Title

Network traffic monitoring on a network with Linux routers

Annotation

This work is focused on network traffic monitoring on Linux routers. Measured values of packets and bytes are periodically sent to the central server. They are saved in a database. From that database it is possible to generate graphs of traffic and view it in a web based application.

Keywords

Linux, router, network traffic, monitoring, rrd

Obsah

Úvod

Router je základním prvkem každé počítačové sítě. Stará se o spojení více sítí do jednoho celku a o přeposílání paketů. Při diagnostice sítě je důležité mít přehled o toku dat na routerech. Je potřebovat vidět jak momentální zatížení, tak i průměrné zatížení v podobě množství dat a paketů. Na grafickém znázornění je pak možné vidět, kdy je router nejvíce zatížen a následně navrhnout vylepšení průchodnosti sítě.

Cílem této bakalářské práce je vytvořit přehled nástrojů pro sledování průtoku dat na linuxových routerech. Dále se zaměřit na platformu embedded-Linux, navrhnout a implementovat vhodné řešení.

V této práci se budu zabývat problematikou výše zmíněných nástrojů pro sledování průtoku dat. Nároky těchto nástrojů na hardware či software routeru, ale i náročností instalace a správy. Vhodnými nástroji naměřená data se odesílají na centrální server, kde se ukládají do databáze. Nad touto databází je postavena webová aplikace, která umožňuje zobrazit průtoky dat na jednotlivých routerech v libovolných směrech, včetně generování grafů zátěže s rozlišením 5 minut.

1 Přehled nástrojů pro sledování průtoku dat

1.1 Iptraf

Slouží především k monitorování IP sítě v reálném čase. Jedná se o čistě softwarové sledování, kdy iptraf využívá vestavěnou podporu síťových zařízení v jádře [1].

1.1.1 Nároky na hardware

Podle oficiální dokumentace vyžaduje minimálně 16 MB paměti RAM, avšak 64 MB RAM je doporučováno. Pro routery s vysokou propustností se doporučuje ještě více paměti. Zkompilovaný program zabere zhruba 2 MB prostoru na disku, pokud by bylo potřebovat logovat průtok dat na disk, je zapotřebí více prostoru v závislosti na množství logovaných dat. Jako minimální nárok na CPU je uváděno Pentium třídy II 200 MHz nebo vyšší. A nakonec samozřejmě jedno nebo více podporovaných síťových zařízení.

1.1.2 Nároky na software

Celý program je postaven na knihovně ncurses, která zajišťuje grafické uživatelské rozhraní na emulovaném terminálu [2].

Je vyžadováno jádro verze 2.2.0 nebo vyšší a minimální verze ncurses 4.2.

1.1.3 Nároky pro kompilaci

Pokud je zapotřebí zkompilovat iptraf ze zdrojových kódů, tak je nutné mít v systému tyto nástroje:

- gcc 2.7.2.3 nebo vyšší,
- GNU C (glibc) vývojová knihovna 2.1 nebo vyšší,
- ncurses vývojová knihovna 4.2 nebo vyšší.

1.1.4 Instalace

Iptraf je dostupný v každém balíčkovacím systému všech běžných distribucí (např. Debian, Ubuntu, Gentoo, apod).

Kompilace ze zdrojových kódů je usnadněna díky dodávanému skriptu Setup.sh. Stačí tedy stáhnout patřičné zdrojové kódy, rozbalit a spustit skript Setup.sh, který program zkompiluje a zároveň nainstaluje.

1.1.5 Použití iptrafu

Správa tohoto programu je poměrně jednoduchá, existuje řada přepínačů, které je možné použít při spuštění. Další možnost je spustit iptraf bez jakéhokoli parametru a použít vestavěné menu, kde je možné si vybrat, co přesně chceme sledovat. Dosáhne se stejného cíle, jen tento postup může být poněkud zdlouhavější.

1.2 Iptables

Všechny zavedená pravidla v iptables mají vestavěné čítače odchycených paketů a dat. Toho je možné využít při sledování průtoku dat routerem.

1.2.1 Nároky na hardware

Vše závisí na tom, co přesně a v jakém rozsahu se s iptables provádí. Jednoduché akce jako např. základní stavový firewall lze podle mých vlastních zkušeností provozovat na CPU Pentium I s 32 MB RAM.

Pokud by přes router mělo „téct“ velké množství dat v řádu desítek megabitů, určitě by byl zapotřebí výkonnější hardware.

1.2.2 Nároky na software

Iptables se používají od jader verze 2.4, kde nahradily staré ipchains, které jsou dnes již označeny jako „zastaralé“, stejně tak jako jádra řady 2.2 a tudíž se nedoporučuje jejich používání.

Je zapotřebí mít jádro se zakompilovanou podporou netfilteru a iptables, ať už jako moduly nebo přímo v jádře. Minimálně je potřeba toto nastavení.

```
CONFIG_NETFILTER=y
CONFIG_IP_NF_IPTABLES=y
CONFIG_IP_NF_FILTER=y
```

Dále se musí nainstalovat nástroj pro uživatelský prostor. Jmenuje se přímo iptables a je dostupný ve všech balíčkovacích systémech běžných distribucí. Nebo ho můžeme zkompilovat ze zdrojových kódů, které jsou dostupné na domovské stránce projektu [3].

1.2.3 Použití

Když přijde paket na síťové zařízení a to ho přijme, dostane se ke zpracování jádrem. Při použití iptables prochází paket filtrovacími tabulkami a v nich řetězy INPUT, OUTPUT, případně FORWARD. Pokud je paket určen pro lokální použití projde řetězem INPUT, pokud je vygenerován lokálně a odeslán do sítě, projde řetězem OUTPUT, a pokud je vygenerován někde v síti a routerem prochází dál do jiné sítě, projde řetězem FORWARD.

Co se týče routeru, tak stačí zavést pravidla do řetězu FORWARD, protože se předpokládá, že lokálně generovaný provoz na routeru bude minimální, tudíž zanedbatelný.

Pravidlo pro sledování průtoku dat, které vstoupí na rozhraní eth0 a vystoupí z rozhraní eth1 by mohlo vypadat takto.

```
iptables -I FORWARD 1 -i eth0 -o eth1
```

Když se vypíše nastavení iptables, při použití pouze toho pravidla v řetězu FORWARD, výpis vypadá následovně.

```
linux # iptables -L FORWARD -x -n -v
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source
destination
       0        0            all  --  eth0   eth1    0.0.0.0/0
0.0.0.0/0
```

1.3 Ntop

Jedná se o komplexní nástroj pro monitorování sítě. Součástí je webové aplikace, kde můžeme sledovat zatížení konkrétního rozhraní nebo jen jedné IP adresy [8].

1.3.1 Nároky na hardware

Závisí pouze na rozsáhlosti a zatížení sítě, kterou je potřebovat monitorovat. Ntop lze nakonfigurovat, aby měřil pouze základní údaje, ale jeho sílu vidím především v množství informací, které nám může poskytnout.

1.3.2 Instalace

Je dostupný v běžných balíčkovacích systémech. Při instalaci ze zdrojových kódů se postupuje běžnou „trojkombinací“ ./configure, make, make install.

1.3.3 Nastavení ntopu

Jelikož jsem s ntopem nikdy předtím nepracoval, přišla mi konfigurace dost složitá. Ale není se čemu divit, protože se jedná o velmi komplexní nástroj s velkým potenciálem v nastavení. Dokonce je možné, aby ntop přijímal data od NetFlow, což je velmi silný nástroj pro monitoring sítě od společnosti Cisco.

1.3.4 Použití

Nijak zvlášť jsem se nastavením a použití ntopu nevěnoval, protože hned na první pohled je jasné, že tento program není ten pravý pro monitoring na embedded zařízeních.

Avšak webové rozhraní ntopu je pěkně zpracované, je zde vidět rozdělení podle protokolů, hostů a další užitečné údaje o využití sítě.

1.4 Vnstat

Využívá vestavěné funkce jádra pro sbírání informací o datech. Z tohoto důvodu má minimální nároky na hardware i software. Je vyžadováno pouze jádro verze vyšší než 2.2. Při prvním spuštění je potřeba vygenerovat databázi pro síťové rozhraní, které má být sledováno. Lze mu přiřadit i nějaké jméno pro lepší identifikaci. Vnstat je poté spouštěn cronem každých 5 minut a čte data z pseudofilesystému jádra. Data lze zobrazovat v seskupení po hodinách, dnech nebo měsících.

Pro základní sledování provozu se tento nástroj jeví jako velmi vhodný. Jeho nevýhoda je, že neumí dlouhodobě sledovat počet paketů. To umí zobrazovat pouze při sledování v reálném čase.

Existuje i příjemné webové rozhraní napsané v PHP, které generuje i základní grafy a je možné rychle zkontrolovat množství odeslaných, přijatých a celkových dat [6].

1.5 Iftop

Nástroj podobný programu top, který slouží pro monitoring procesů v Unixu. Poslouchá na daném síťovém zařízení a zobrazuje aktuální průtok dat ve stylu „odkud kam“. Podobný program jako iptraf, hodí se pro analýzu síťového provozu v reálném čase, v situaci kdy je potřeba okamžitě vidět kam směřuje nejvyšší provoz na síti [7].

1.5.1 Nároky na hardware

Iftop je velmi nenáročný program, který má minimální nároky na hardware.

1.5.2 Nároky na software

Vyžaduje knihovny libcurses, která se používá pro textové uživatelské rozhraní. Dále je vyžadována knihovna libcap.

1.5.3 Použití

V podstatě stačí pouze spustit iftop bez parametru a on si vybere nějaké dostupné síťové zařízení a na něm bude poslouchat. Je zde několik parametrů, které lze zadat při spouštění.

Parametry:

- -i, za nímž následuje název síťového rozhraní,
- -p – bude se naslouchat v promiskuitním módu (tzn., že se bude odposlouchávat i provoz, který není přímo určen pro síťové rozhraní na němž se naslouchá),
- -F – za ním následuje definice sítě, pro kterou chceme naslouchat,
- nebo např. parametr -P, který zajistí, že se budou vypisovat i čísla použitých portů.

1.6 SNMP/MRTG

SNMP je jednoduchý síťový protokol, který je určen ke správě síťových zařízení [9]. Pomocí SNMP agenta je možné po zařízení požadovat nějaké údaje nebo ho přimět ke změně konfigurace. Nyní existují tři verze tohoto protokolu. Verze 1 a 2 obsahuje základní ověření uživatele na základě názvu komunity. Avšak vše je posíláno v čistém textu, takže pro útočníka není problém si odposlechnout komunikaci a tím pádem by mohl komunikovat se zařízeními. Verze 3 obsahuje dokonalejší způsob ověření na úrovni uživatelského jména a hesla a navíc podporuje šifrovaný přenos dat.

Dá se říct, že MRTG je nadstavba nad SNMP protokolem. MRTG získává data pomocí SNMP protokolu a převádí je do přehledných a užitečných grafů a zároveň

i generuje webové stránky. Je tudíž možné na první pohled z grafu vidět aktuální zatížení sítě a srovnat ho s dřívějším stavem. Typicky se generují denní, měsíční a roční statistiky [10].

1.6.1 Nároky na software

MRTG je celé napsané v Perlu, takže vyžaduje jeho přítomnost v systému. Kvůli tvorbě grafů je potřeba knihovna GD, která generuje obrázky ve formátu PNG. Dříve podporovala i formát GIF, ale kvůli licenci musela být tato podpora odebrána. Knihovna GD ke své práci potřebuje ještě dvě jiné knihovny, a to libpng a zlib.

Generování grafů se dá zařídit i pomocí RRDtool [15].

1.6.2 Použití

Součástí MRTG jsou nástroje, s jejichž pomocí lze snadno vygenerovat konfigurační soubory:

- cfgmaker – program, který vygeneruje konfigurační soubor,
- indexmaker – vygeneruje indexový HTML soubor

Pak již jen stačí pravidelně spouštět MRTG nebo vytvořit init .skript a nechat MRTG běžet jako démon.

2 Výběr vhodného řešení pro embedded-Linux

Z důvodů náročnosti na software či hardware se pro sledování provozu na embedded zařízení nehodí iptraf, ntop ani iftop.

Co se týče SNMP, tak existuje verze, která se dá zkompilovat pomocí knihovny µClibC. Výsledný SNMP server je i po této „mikrokompilaci“ vcelku velký a zbytečně by jeho běh zatěžoval směrovač.

O vnstat by se dalo uvažovat v případě, že by bylo dostačující měřit pouze celkový provoz na jednotlivých rozhraních, a to jen množství dat bez počtu paketů.

Iptables se k základnímu sledování provozu, které je cílem této práce hodí nejvíce. Jsou závislé pouze na jádře a na nástroji pro uživatelský prostor. Pomocí definice vhodných pravidel lze sledovat nejen celkový provoz na síťovém rozhraní, ale např. provoz ve více směrech nebo provoz určený pro konkrétní sítě.

3 Implementace

3.1 Pravidla iptables

Pokud se vytvoří pravidla, které nemají definovaný přepínač -j, tak jimi pakety pouze prochází a pokračují dál. To znamená, že se počítá množství dat a počet prošlých paketů. Tím se dá velmi snadno vytvořit sledovací systém provozu. Tabulky IP byly primárně vytvořeny pro sestavování firewallu a řízení toku dat v síti IP. Existuje mnoho přepínačů a parametrů, které se dají využít při tvorbě pravidel.

Pro lepší přehlednost je vhodné si vytvořit vlastní řetěz pravidel s názvem např. „monitoring“ a do něj umisťovat pravidla související se sledováním provozu.

```
iptables -N monitoring
```

Jelikož se jedná o sledování provozu na směrovači, předpokládám, že lokálně generovaný provoz bude minimální a bude se sledovat pouze řetěz FORWARD, kterým prochází všechny pakety, které směrovač přeposílá dál pro stanice v síti. Je třeba začlenit nový řetěz monitoring do řetězu FORWARD. V tomto okamžiku je třeba si uvědomit, že již na směrovači mohou být ve FORWARDu zavedena nějaká pravidla, a princip tabulek IP je takový, že pokud nějaký paket vyhoví pravidlu, dál již nepostupuje! Proto je potřeba řetěz s monitorovacími pravidla zařadit vždy jako první řetěz ve FORWARDu, čehož lze dosáhnout parametrem -I, za kterým následuje název řetězu a pořadí kam se má vložit:

```
iptables -I FORWARD 1 -j monitoring
```

Např. pro sledování provozu mezi síťovými zařízeními eth0 a eth1 v obou směrech postačí tato dvě pravidla.

```
iptables -A monitoring -i eth0 -o eth1 -j RETURN
iptables -A monitoring -i eth1 -o eth0 -j RETURN
```

Přepínač -A říká, že toto pravidlo přidáváme do řetězu uvedeného za ním, -i a -o je vstupní, resp. výstupní síťové rozhraní a -j nám vrací paket zpět do řetězu, odkud byl zavolán řetěz monitoring, k dalšímu zpracování.

Sledování provozu z resp. do konkrétní sítě 192.168.0.0/24 se zajistí těmito pravidly.

```
iptables -A monitoring -s 192.168.0.0/24 -j RETURN
iptables -A monitoring -d 192.168.0.0/24 -j RETURN
```

Přibyl parametr -d a -s, za který se uvádí cílová, resp. zdrojová adresa paketu, která lze zadat i v podobě rozsahu sítě.

Pokud jsou všechna monitorovací pravidla zavedena, lze si je vypsat a tím získat potřebné údaje v podobě množství přenesených dat a počtu paketů.

```
router:~# iptables -L monitoring -x -n -v -Z
Chain monitoring (1 references)
    pkts      bytes target     prot opt in     out     source
destination
       9     3480 RETURN     all  --  eth0   eth1    0.0.0.0/0
0.0.0.0/0
       8     2362 RETURN     all  --  eth1   eth0    0.0.0.0/0
0.0.0.0/0
       0        0 RETURN     all  --  *   *    0.0.0.0/0
192.168.0.0/24
       0        0 RETURN     all  --  *   *    192.168.0.0/24
0.0.0.0/0
```

Za parametrem -L následuje název řetězu, který chceme vypsat (když se vynechá, vypisují se všechny), -x udává, že ve výpisu budou použity základní jednotky, v tomto případě bajty, a nebude se převádět na kilobajty atd., -n značí, že veškerý výstup adres IP a portů bude v numerické podobě a -v je tzv. „více upovídaný výstup“, který poskytne všechny důležité informace. Ještě je potřeba zmínit parametr -Z, který vynuluje všechny čítače v řetězu. To je velmi vhodné při sběru dat. Po každém přečtení dat se vynulují čítače a naměřené hodnoty se pouze sčítají, nejsou tedy nutné žádné operace navíc.

Ve skriptu, který je součástí přílohy, se zavádějí pravidla pro sledování provozu mezi všemi síťovými zařízeními ve všech směrech. Dále se měří průtok dat pro jednotlivé síťové rozsahy.

3.2 Předávání naměřených dat na centrální server

Získaná data ze zavedených pravidel iptables je potřeba nějak dostat ze směrovačů na centrální server. Zde nastává několik úskalí, které je potřeba vyřešit.

3.2.1 Zvolení vhodného softwaru pro embedded-Linux

Při hledání vhodného způsobu předávání dat na centrální server jsem dospěl k pěti možným způsobům.

- *Protokol SNMP* – jak jsem se již zmiňoval, toto řešení není úplně nejvhodnější, pouze z důvodu použití na embedded-Linuxu. Jinak se jedná o velmi vhodný způsob. Protokol SNMP byl vymyšlen speciálně pro podobné účely a ve verzi 3 je již dostatečně zabezpečený.

- *Naprogramovat vlastního jednoduchého démona* – tato možnost vyžaduje dostatečné zkušenosti v programování síťových aplikací, aby se dal napsat démon, který by se staral o předávání dat. S dostatkem zkušeností by pravděpodobně toto řešení zvítězilo nad ostatními.

- *Použití vzdáleného syslogu* – logovací démon je možné nastavit tak, aby zasílal zprávy na jiný démon syslog, který je umístěn na nějakém vzdáleném stro-

ji. Je použit UDP protokol a port 514. Pokud je v cestě nějaký firewall, je nutné zajistit, aby pakety prošly. Za předpokladu, že syslog démon je v každém směrovači, který je vhodné monitorovat, se jedná také o vhodný způsob, jak dostat data do úložiště. Nastavení se na první pohled nezdá nijak složité. Lze najít i mnoho tutoriálů, kde je vysvětlen základní princip [11].

- *Protokol SSH* – pokud je zaručeno, že na všech routerech bude přítomen klient SSH, důrazně bych doporučoval předávání dat pomocí tohoto protokolu. Komunikace je sama o sobě zabezpečená pomocí šifrování, což vidím jako velkou výhodu. Na centrálním serveru je možné vytvořit speciálního uživatele, kterému by se jako jeho shell nastavil pouze skript, který by přijímal data z routerů. Tím se dá dosáhnout poměrně bezpečného řešení.

- *Netcat* – je tradiční nástroj, který je součástí snad každé běžně dostupné distribuce. Velikost zkompilované binárky se pohybuje pouze kolem 26 kB, avšak jeho schopnosti jsou obrovské. Dokáže číst nebo posílat data skrze síť pomocí TCP nebo UDP protokolu. Může být spuštěn ve třech režimech:

  - režim připojení – nejvíce používaný, je možno se s ním připojit na jakoukoli službu která naslouchá na nějakém soketu,

  - režim naslouchání – netcat se sám bude chovat jako služba, která bude naslouchat na socketu,

  - režim v podobě tunelu – nejnovější režim, slouží k vytváření tunelů mezi vzdálenými stanicemi.

  Rozhodl jsem se pro použití právě tohoto nástroje. Jeho síla spočívá hlavně v jeho jednoduchosti a malé velikosti. Pokud by nebyl součástí embedded-Linuxu na sledovaném směrovači, není problém ho tam dodat.

Jako jedinou nevýhodu při použití netcatu se mi jeví nezabezpečený přenos. Vše, co je netcatem posíláno skrze síť, je v čistém textu, neprobíhá žádné šifrování přenosu. Případný útočník by tedy mohl jednoduchým způsobem odchytit data. Důrazně se nedoporučuje používat přenos netcatem rizikovou sítí a už vůbec ne internetem.

3.2.2 Použití netcatu pro odeslání naměřených dat

Na centrální server je potřeba odeslat text v tomto formátu:

```
2009-04-21 20:25:59 asus_laptop 1287 5786 RETURN all -- eth0 eth1
0.0.0.0/0 0.0.0.0/0
```

První dva sloupce skýtají informaci o datu, která je velice důležitá. Třetí sloupec je hostname směrovače, odkud pochází naměřená data, čtvrtý a pátý sloupec je počet paketů a bytů. Poslední čtyři sloupce nesou informaci o vstupních/výstupních síťových zařízeních a zdrojové/cílové adrese.

Vše od čtvrtého sloupce včetně je výstupem z iptables. První tři sloupce se však musí k tomuto výstupu přidat. Název směrovače je návratová hodnota příkazu hostname, ale problém nastává s časem.

Přichází v úvahu několik možností, jak získat čas do logu.

1) Brát čas ze směrovače.

2) Na směrovači se časem nezabývat a přidávat ho až na centrálním serveru.

3) Získat čas z centrálního serveru a ten posílat v logu zpět.

V případě první možnosti by nebylo zapotřebí řešit nic navíc, ale je potřeba si uvědomit, že z naměřených dat se budou generovat grafy, ze kterých je potřeba „číst“ zatížení sítě a navrhovat řešení pro vylepšení propustnosti. Z toho vyplývá, že čas v logu získaných dat musí být za každou cenu synchronizován. Tohle řešení synchronizaci času nezaručí.

Druhá možnost již zajistí potřebnou synchronizaci, ale mohl by nastat problém, pokud by bylo efektivnější předávat naměřená data na centrální server v dávkách. V některých případech by mohlo být vhodnější naměřit data za delší časové období a ty odeslat najednou. V tomto případě by se narazilo na problém.

Třetí možnost zajišťuje synchronizaci času a zároveň i řeší problém druhé možnosti. Navíc díky tomuto přístupu je možné rovnou nastavit čas na směrovači. Existuje ná-

stroj rdate, který byl vytvořen přímo pro tyto účely [13]. Umí získat čas ze vzdáleného stroje a vypsat na standardní výstup nebo dokonce i nastavit získaný čas na zdrojovém stroji.

Získání času ze vzdáleného stroje a vypsání:

```
router ~ # rdate -p 10.10.1.1
rdate: [10.10.1.1]    Wed Apr 22 10:07:38 2009
```

Pokud se rdate zadá s parametrem -s, tak pouze získá čas a nastaví zdrojový stroj tímto časem.

Jak jsem již zmínil, bylo by možné odesílat data na centrální server v určitých dávkách. Pokud by bylo vhodné sbírat naměřená data třeba každou minutu, určitě by bylo lepší nashromáždit více dat a ty odeslat najednou. Na směrovači je možné uložit tuto dávku dočasně do RAM-disku.

Vzniká zde ještě jedna otázka. Co se stane v případě, že směrovač nedokáže navázat spojení s centrálním serverem. Příkaz, který odesílá data, by šlo vložit do cyklu a pokusit se spojení navázat znovu po určitém časovém intervalu, samozřejmě rozložení intervalu záleží na čase, ve kterém je zapotřebí logovat další naměřená data. Avšak prozatím se ve skriptu předpokládá, že směrovače a centrální router musí být neustále ve spojení a pokud nastane nějaký výpadek, data budou zahozena.

3.2.3 Pravidelné spouštění skriptu

Nejjednodušší řešení je spouštět tento skript na směrovačích každých pět minut pomocí démona Cron.

Příkazem

```
crontab -e
```

se vstoupí do editačního prostředí úloh. Stačí přidat následující řádek.

```
*/5 * * * * /cesta/ke/skriptu/skript.sh
```

Všechny úlohy lze zobrazit následujícím příkazem.

```
crontab -l
```

Je však možné, že na embedded zařízení bude tento démon chybět. Celý skript by se v tomto případě musel uzavřít do nekonečné smyčky a na konci zajistit zastavení provádění skriptu na 300 sekund, než se skočí zase na začátek smyčky. Tím se ovšem nedodrží naprosto přesně pětiminutový interval, který je nutný k přesnému znázornění dat na grafu. Vůbec není nutné se o tento jev starat, protože databáze RRD s tímto problémem počítá a vyrovná se s ním po svém.

3.3 Konfigurace centrálního serveru

Na straně centrálního serveru je zapotřebí zajistit, aby data odesílaná směrovači byla přijata a zpracována. Opět se nabízí několik řešení:

- naprogramovat vlastní démon, který se postará o přijetí dat,
- netcat,
- super-server démon.

Programování vlastního démona jsem se vyhnul kvůli nedostatku zkušeností v této oblasti. Rozhodl jsem se využít již nějaké hotové řešení sloužící právě k těmto účelům.

Netcat je již použit k odesílání dat na centrální server, avšak šlo by jej využít i pro přijímání dat. Tuto možnost jsem zkoušel, ale nezdálo se mi to jako zrovna ideální způsob.

3.3.1 Super-server démon

Použil jsem tzv. super-server démon, který je velice vhodný pro podobné situace.

Jedná se o aplikaci, která naslouchá na vybraném portu (portech) a vždy když na tento port přijde TCP nebo UDP paket, super-server spustí daný program, aby se postaral o to konkrétní připojení.

Pokud aplikace není využívána nějak zásadně s vysokým využitím procesoru, je tento přístup velmi efektivní z hlediska úspory paměti i procesorového času. Vybraná aplikace je spouštěna pouze tehdy, když je skutečně potřeba. Další výhoda je ta, že v obslužné aplikaci není potřeba programovat žádné věci týkající se sítě, to vše obstará super-server.

Původním super-serverem byl program inetd [14]. Dodnes je součástí základního systému distribuce Debian Lenny GNU/Linux. Avšak doporučuje se používat nějakou novější implementaci toho programu, jako je například xinetd. Jelikož je o mnoho více zabezpečen oproti původnímu inetd super-serveru.

Pozn.: operační systém Mac OS X používal standardně xinetd super-server démon, než byl nahrazen démonem launchd.

Nastavení démona inetd v distribuci Debian GNU/Linux se nachází v souboru /etc/inetd.conf a má následující formát.

```
monitoring      stream  tcp     nowait  monitoring
/home/monitoring/process.sh
```

První položka je název služby, který je obsažený v souboru /etc/services.

```
monitoring      16001/tcp
```

To znamená, že pod názvem „monitoring“ se skrývá připojení přes TCP na port 16001.

Druhá položka v konfiguračním souboru inetd je typ socketu, která podle manuálové stránky může nabývat hodnot – stream, dgram, raw, rdm nebo seqpacket. Třetí položka je použitý protokol. Čtvrtá položka určuje, zda má inetd čekat na návratovou hodnotu z programu nebo může dále zpracovávat přijímaná data na socketu. Pokud je použit typ socketu stream doporučuje se použít nowait, pouze v případě pokud by je-

den program zpracovával zároveň vícenásobné připojení, mělo by být nastaveno wait. Pátá položka je uživatel, pod kterým se bude spouštět program/skript, který je uveden jako šestá položka. Jako poslední sedmou položku je možné ještě přidat parametry pro spouštěný program/skript.

Z hlediska bezpečnosti doporučuji použít novější xinetd. Má implementováno mnoho důležitých funkcí navíc. Jako je například TCP Wrapper ACL, rozšířená možnost logování nebo zpřístupnění služeb pouze v určitý čas. Nevýhoda je v tom, že syntaxe konfiguračního souboru není kompatibilní, proto je nutný zásah administrátora.

Stejné konfigurace jako je uvedena výše pro inetd, vypadá v xinetd následovně.

```
service monitoring
{
  socket_type  = stream
  port         = 16001
  protocol     = tcp
  wait         = no
  user         = monitoring
  server       = /home/monitoring/process.sh
}
```

Na první pohled se nastavení moc neliší, ale xinetd má plno dalších možností nastavení, především co se týče bezpečnosti nebo logování.

3.3.2 Skript zpracovávající přijatá data

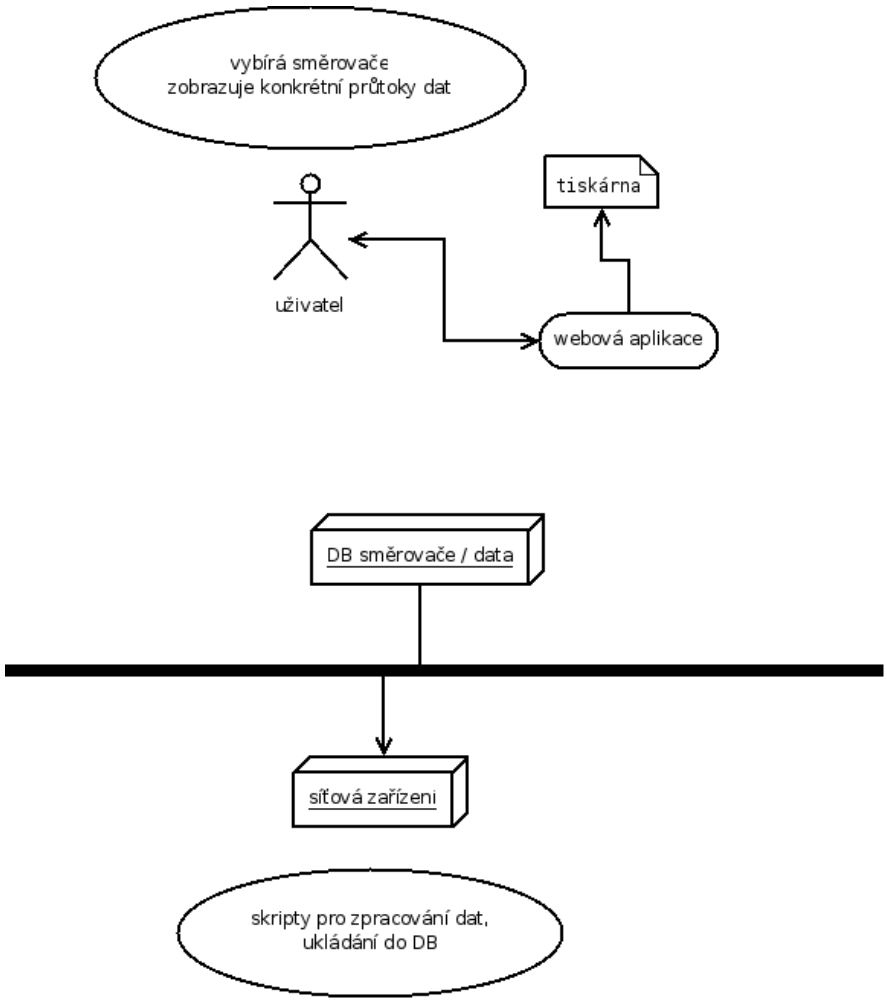
Super-server démon spustí skript a předává mu přijatá data ze směrovače na standardní vstup. Potom není problém zpracovat data v cyklu řádek po řádku.

V první fázi jsou data ukládána do souboru pro případné další zpracování, dále jsou ukládána do databází MySQL a RRD pro zobrazení ve webové vyhodnocovací aplikaci.

Databáze je vytvořena podle normálních forem, tudíž je rozdělena do více tabulek. V jedné tabulce jsou evidovány směrovače a v dalších tabulkách naměřená data. Pokud by se objevil nový směrovač, který ještě není v evidenci, skončil by skript chybou. Když nastane tato situace, tak jsem vytvořil podmínku, která se postará o vložení nového směrovače do evidence v databázi. Pro přidání nového směrovače do monitorovacího systému tedy stačí pouze pravidelně posílat data na centrální server. Směrovač se nemusí se přidávat do evidence ručně, ale vloží se automaticky.

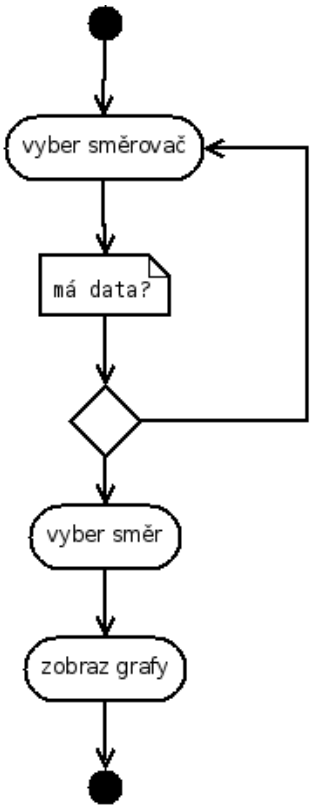
4 Vyhodnocovací webová aplikace

Pro prvotní znázornění webové aplikace jsem vytvořil tzv. rich picture.



Ilustrace 1: Rich picture

UML activity diagram zachycuje „chování“ webová aplikace.



Ilustrace 2: UML activity diagram

4.1 MySQL databáze – uložení naměřených dat

Nejjednodušší způsob by byl vytvořit si dvě tabulky. V první by byly „názvy“ všech sledovaných směrovačů a ve druhé jejich naměřená data. Je však nutné si uvědomit, že pro jeden směrovač se může měřit více směrů a interval sbírání měřených dat bude 5 minut. Tím pádem by velikost tabulky s daty stále rostla do enormních rozměrů.

Kdyby byl součástí sledování pouze jeden směrovač, na kterém by nás zajímalo měření ve čtyřech různých směrech, ať už mezi síťovými rozhraními nebo tekoucí data pro různé sítě, tak v pěti minutovém intervalu, by bylo v tabulce za jeden den 288 řádků, za jeden měsíc přibližně 8640 řádků a za rok zhruba 105 tisíc řádků. A to pouze pro jeden směrovač. Tato databáze by byla velmi neefektivní.

Rozhodl jsem se proto pro uložení dat ve stejném formátu jako používá RRDtool. Vytvořil jsem 4 tabulky:

- denní data – v této tabulce se uchovávají data za poslední den s rozlišením 5 minut,
- týdenní data – uchovává data za poslední týden s rozlišením 30 minut,
- měsíční data – uchovává data za poslední měsíc s rozlišením 2 hodiny,
- roční data – uchovává data za poslední rok s rozlišením 1 den.

Tímto způsobem se v rozumném množství ukládají data, ze kterých lze vygenerovat denní, týdenní, měsíční a roční graf průtoku. Tabulky mají pro každý sledovaný směr pouze zhruba 1350 řádků za rok oproti 105 tisícům jako v prvním případě.

4.2 RRDtool – vytvoření databáze

Jedná se o zkratku Round Robin Database tool [15]. Round Robin je velmi vhodný způsob jak uchovávat data (tento pojem je známý z plánování procesů v operačních systémech). V tomto případě se jedná o techniku, která pracuje s fixním množstvím dat a ukazatelem na poslední vložený údaj.

Pokud by se uchovávaly všechny naměřené hodnoty každých pět minut, soubor by narůstal do stále větších rozměrů. Round Robin tento důsledek velice vhodně eliminuje. Při vytváření Round Robin databáze se definuje tak zvaný Round Robin archiv. V něm se rozhodne kolik dat a v jakém intervalu se bude uchovávat. V jedné Round Robin databázi je možné ukládat více Round Robin archivů, tak jak je to v konkrétním případě vhodné. Díky tomu, že je předem dáno kolik dat se bude uchovávat, je možné aby RRDtool předem alokoval potřebné množství paměti, které je okamžitě známé.

V konkrétním případě při sledování průtoku dat na směrovačích by byla vhodná následující Round Robin databáze.

```
rrdtool create db.rrd DS:pkts:ABSOLUTE:600:U:U
DS:bytes:ABSOLUTE:600:U:U RRA:AVERAGE:0.5:1:600
RRA:AVERAGE:0.5:6:700 RRA:AVERAGE:0.5:24:775
RRA:AVERAGE:0.5:288:797
```

DS značí „Data Source“, to znamená jaká data se budou do databáze ukládat. Opět je možné uvést více zdrojů dat. V tomto případě se jedná o počet bytů a počet paketů. Klíčové slovo „ABSOLUTE“ značí způsob čtení dat. Jsou k dispozici tyto možnosti:

- GAUGE – používá se pro data jako je např. teplota nebo počet přístupů na webovou stránku.

- COUNTER – čítače v linuxovém jádře fungují, tak že začínají od nuly a jak data přibývají, tak se stále přičítají (inkrementují). Tento typ se používá pro čtení právě těchto hodnot. Na 32bitových systémech čítač přeteče po dosažení čísla $2^{32} = 4294967296$. RRDtool obsahuje mechanismus, který odhalí přetečení čítače a zanese do databáze správné údaje.

- DERIVE – funguje úplně stejně jako COUNTER, ale nepočítá se s přetečením čítače, tzn. nekontroluje se zda data nepřesáhla hranici.

- ABSOLUTE – použije při sběru dat, které se po každém přečtení vynulují. V mém skriptu je to právě tento případ, kdy parametr -Z u výpisu iptables zajistí vynulování čítačů. Pokud by bylo nutné čítače nenulovat, použil by se typ COUNTER.

- COMPUTE – je určen pro speciální účely, kdy je potřeba data přepočítat podle daného vzorce.

Číslo 600 je tak zvaný „heartbeat“, který značí maximální počet vteřin, které mohou uběhnout mezi dvěma aktualizacemi datového zdroje než bude hodnota označena jako neznámá. Poslední dvě možnosti, které jsou v tomto případě označeny jako U, značí minimum a maximum měřených hodnot. Pokud minimální ani maximální

hodnoty nejsou známé, uvede se U. V případě sledování průtoku dat, by se jako minimální hodnota mohla uvést 0 a jako maximální hodnota propustnost daného síťového zařízení. Vždy, když jsou tyto dvě hodnoty známe, tak se doporučuje je uvést, protože se usnadní výpočet.

Dále jsou definovány čtyři Round Robin archivy, které uchovávají samotná data. Po definici klíčového slova RRA následuje tak zvaná slučovací (konsolidační) funkce. Tyto jsou k dispozici:

- AVERAGE – je ukládána průměrná hodnota naměřených dat. Pokud je interval 5 minut (jedná se o implicitní hodnotu) a ukládaná data např. 10240, ukládá se 10240/300 = 34,1.

- MIN – ukládá pouze minimální naměřenou hodnotu.

- MAX – ukládá pouze maximální naměřenou hodnotu.

- LAST – ukládá poslední naměřenou hodnotu.

Za slučovací funkcí následují 3 její parametry. Těmi jsou *xff:steps:rows*:

- xff – udává procento dat, která mohou být neznámá, aby celková uložená jednotka nebyla nulová,

- steps – určuje kolik naměřených jednotek dat se spojuje v jednu ukládanou hodnotu,

- rows – určuje kolik záznamů se v Round Robin archivu uchovává.

4.3 RRDtool – aktualizace databáze

Jakmile je databáze vytvořená, je možné do ní začít ukládat data. Slouží k tomu rrdtool nástroj update. Jako argument přijímá cestu k souboru s RRD databází, časovou značku ve formátu „unix timestamp“ a data v pořadí, v jakém byly definovány datové zdroje při vytváření databáze.

Příklad pro pakety a byty:

```
rrdtool update db.rrd N:1387:16943
```

Jako časovou značku je možné uvést „N“, což znamená že se použije aktuální čas. Také se dá v jedné aktualizaci uvést více hodnot k zapsání do databáze.

```
rrdtool update db.rrd 1240869600:1387:16943 1240869900:1983:24921
```

4.4 RRDtool – tvorba grafu

Nástroj graph slouží k vytváření přehledných grafů, ale dá se pomocí něj generovat i přehledný textový výstup.

Je k dispozici mnoho konfiguračních voleb, pomocí nichž lze přidat popisky grafu, upravit osy, měnit barvy, atd.

Pomocí tohoto příkazu se vygeneruje zcela základní graf.

```
rrdtool graph graf.png --start end-1d
DEF:Bps=db.rrd:bytes:AVERAGE DEF:packets=db.rrd:pkts:AVERAGE
AREA:Bps#00FF00:"Bps" LINE1:packets#0000FF:"packets"
```

První argument značí soubor, do kterého se uloží výsledný graf. Jsou podporovány typy PNG, SVG, EPS a PDF. Je možno zadat časový interval, pro který se má graf vygenerovat. V tomto případě se jedná aktuální čas minus jeden den (vygeneruje se graf za posledních 24 hodin). Instrukce „DEF“ získává data z databáze. Datovou sadu je možné si pojmenovat nějakou proměnnou, která se dá dále v příkazu použít (např. pro výpočty). Instrukce „LINE“ a „AREA“ jsou vykreslené křivky v grafu. Area se liší od Line tím, že vše pod křivkou je vybarvené.

Je nutné si všimnout, že tímhle způsobem se vykreslí průměrné hodnoty bytů za sekundu, ale bylo by vhodnější vykreslovat bity za sekundu. RRDtool umožňuje použít libovolné výpočty na data uložená v databázi. Slouží k tomu instrukce CDEF.

```
rrdtool graph graf.png --start end-1d
DEF:Bps=db.rrd:bytes:AVERAGE DEF:packets=db.rrd:pkts:AVERAGE
CDEF:bps=Bps,8,\* AREA:bps#00FF00:"bps"
LINE1:packets#0000FF:"packets"
```

Předcházejícím zápisem se zajistí, že hodnoty z databáze se budou násobit 8, a tím se na výstupu objeví bity za vteřinu.

Ještě je vhodné upravit osy.

```
rrdtool graph graf.png --start end-1d --x-grid
HOUR:2:HOUR:2:HOUR:2:86400:%H -Y -v "values per second"
DEF:Bps=$rrdFileName:bytes:AVERAGE
DEF:packets=$rrdFileName:pkts:AVERAGE CDEF:bps=Bps,8,\*
AREA:bps#00FF00:"bps" LINE1:packets#0000FF:"packets"
```

Mřížka se bude vykreslovat každé dvě hodiny a jako popisky se zobrazí hodina ve formátu 00, 02, 04, atd. To, co se zobrazí jako popisek, se řídí funkcí z jazyka C strftime. V tomto případě %H.

4.5 Použití MySQLi pro PHP

Tento adaptér pro připojení k databázi MySQL z PHP nahradil staré rozhraní MySQL. Důvodů je hned několik [16].

- Podpora nových vlastností serveru MySQL verze 4.1 a vyšší.
- Snadnější udržování kódu, jelikož kód starého rozhraní MySQL se stával molochem.
- Zpětná a lepší kompatibilita s klientskou knihovnou MySQL.

Mezi hlavní výhody, kterými nové rozhraní disponuje, patří:

- procedurální přístup, který vznikl zejména kvůli snadnějšímu přechodu ze starého rozhraní, které používá pouze tento přístup,
- objektově orientovaný přístup, který je v dnešní době modernější, přehlednější a i jednodušší z hlediska rozšiřitelnosti,
- podpora pro nový binární protokol MySQL, který byl nově představen ve verzi 4.1 (například umožňuje použití prepared statements),
- podpora samotné klientské knihovny MySQL psané v jazyce C.

Tyto důvody jsou dostatečné k tomu, aby se v nových projektech vždy používalo toto nové rozhraní. Navíc dokumentace uvádí, že v některých případech je MySQLi až čtyřicetkrát rychlejší než starší rozhraní MySQL. Co se týče bezpečnosti, tak ta byla také velice posílena.

4.5.1 SQL injection

V každé aplikaci, ve které se pracuje s SQL dotazy v kombinaci s uživatelskými vstupy, si programátor musí dát pozor na tak zvané podstrčené dotazy (SQL injection). Všechny vstupy od uživatele, které se stávají součástí dotazu, musí být řádně ošetřeny, jinak by se mohlo stát, že útočník vloží do pole tento řetězec.

```
a'; DROP TABLE tabulka; --
```

Pokud by se výsledný SQL dotaz skládal z proměnné $vstup (výše uvedený podstrčený SQL dotaz) a následující definice

```
SELECT * FROM tabulka WHERE sloupec = '$vstup';
```

vznikl by velice nepříjemný dotaz.

```
SELECT * FROM tabulka WHERE sloupec = 'a'; DROP TABLE tabulka;
--';
```

K zabránění podstrčených dotazů v PHP se používají již zmiňované prepared statements, které jsou dostupné v MySQLi. V starší verzi MySQL adaptéru pro PHP se používala funkce mysql_real_escape_string(), která ovšem pouze „escapovala“ nepovolené znaky tím způsobem, že před ně vložila zpětné lomítko.

4.5.2 Prepared statements

Jedná se o lepší způsob, jak zabránit SQL injection. V pokročilejších relačních databázích se tento způsob používá již delší dobu. Podporu pro PHP přineslo až vydání adaptéru MySQLi.

Jsou k dispozici dva druhy – první se týká parametrů a druhý výsledků. Jednoduše řečeno, první se používá pro ukládání dat do databáze a druhý pro získávání dat z databáze.

Funguje to tak, že se nejdřív vytvoří jakási šablona SQL dotazu a ta se odešle na server, ten ji rozparsuje a pokud je syntakticky v pořádku, tak ji uloží do zásobníku. Server poté vrací zpět aplikaci odkaz na tento prepared statement, aby se s ním dalo dále pracovat. Když je poté nutné spustit dotaz s konkrétními daty, odesílají se na server pouze tato data a nic navíc. Výhoda je v tom, že největší režie, kterou server musí vykonat, jako například parsování a validace dotazu, se provede pouze jednou, i když se dotaz může spustit mnohokrát.

Prepared statement vypadá takto.

```
SELECT hostname FROM routers WHERE id_router = ?;
```

Ukázka PHP kódu za použití MySQLi adaptéru.

```
$mysqli = new mysqli(MYSQL_HOST, ...

$stmt = $mysqli->prepare("SELECT hostname FROM routers WHERE
id_router = ?");

$stmt->bind_param("i", $id);

$code = 12;

$stmt->execute();
```

```
while ($stmt->fetch()) {
        echo ...;
}
$stmt->close();
$mysqli->close();
```

4.6 Načítání nově vygenerovaných grafů prohlížečem

Při testování výsledné webové vyhodnocovací aplikace jsem narazil na problém, že v některých prohlížečích, jako například Internet Explorer nebo Opera, se při odeslání formuláře, kdy si uživatel zvolil směr, pro který chce zobrazit patřičné grafy, se nenačetly nové vygenerované obrázky. Načetly se pouze v případě, když se provedlo obnovení stránky pomocí tlačítka „Obnovit“ nebo klávesové zkratky F5.

Zkoušel jsem experimentovat s hlavičkou HTML stránky, konkrétně s nastavením http-equiv a hlavičky Cache.

```
<meta http-equiv="cache-control" content="no-cache" />
```

Dále jsem zkoušel, dle rad různých uživatelů, nastavit hlavičku Expires na datum v minulosti, aby se vynutilo kompletní obnovení stránky, ale to také nepomohlo.

```
<?php header("Expires: Sat, 26 Jul 1997 05:00:00 GMT"); ?>
```

Nakonec jsem objevil jediné funkční řešení, které donutí Internet Explorer a Operu znovu načíst obrázky. Do URL adresy obrázku se přidá jako parametr nějaký náhodně vygenerovaný řetězec nebo číslo, které bude vždy jiné. Tento postup funguje jak v IE, tak i v Opeře.

```
<?php
$t = rand(10000, 1000000);
echo '<img src="graf.png?t='. $t .'" alt="graf prutoku"/>';
?>
```

Závěr

V práci se mi podařilo dosáhnout všech požadovaných kritérií v zadání. Po prvotním nastavení centrálního serveru, stačí pravidelně spouštět skript na směrovačích a dále je již vše zautomatizováno.

Původní myšlenka v zadání práce byla taková, že se data budou ukládat pouze do databáze MySQL. Avšak v průběhu implementace jednotlivých částí jsem si uvědomil, že by bylo výhodné mít možnost volby, kam naměřená data ukládat. Vyřešil jsem to takovým způsobem, že se data ukládají do logovacího souboru, databáze MySQL i databáze RRD.

Další věc, která by ještě stála za zamyšlení, je ta, že pokaždé když se ve webová vyhodnocovací aplikaci žádá o nějaké grafy, tak se generují nové. Ne vždy je to však potřeba. V PHP skriptu by se dalo doprogramovat opatření, které by mohlo kontrolovat, jak starý je vygenerovaný graf a jestli má smysl ho generovat znovu. Tedy jestli se nějaká data v uplynulém časovém intervalu od poslední tvorby grafu změnila. Ještě je možné jedno řešení, a to využít modul webového serveru Apache mod_expires.

U takových optimalizací je vždy vhodné posoudit, kolik klientů bude data prohlížet. Vzhledem k tomu, že tato aplikace je primárně určena pro dohled nad sítí pro poskytovatele internetu, jsou použitá řešení plně dostačující a vyhovující.

Tento systém by mohl být přínosem pro každou síť, kde je zapojen více než jeden směrovač a je vhodné sledovat průtok dat.

Seznam zkratek a pojmů

IP – Internet Protocol, používá se pro přenos dat v paketových sítích

MB – Megabyte, jednotka množství dat

RAM – Random-access Memmory, paměť která bez energie ztrácí svůj obsah

CPU – Central Processing Unit, procesor počítače

Mhz – Megahertz, jednotka frekvence

GNU – GNU's Not Unix, projekt zaměřený na svobodný software

Mbit – Megabit, jednotka informace

PHP – Hypertext Preprocessor, původně Personal Home Page, skriptovací jazyk používaný především pro programování dynamicky generovaných webových stránek

SNMP – Simple Network Management Protokol, slouží pro sběr dat a k účelům správy sítí

MRTG – Multi Router Traffic Grapher, slouží pro monitoring zařízení pomocí SNMP protokolu

PNG – Portable Network Graphics, grafický formát s bezeztrátovou kompresí rastrové grafiky

GIF – Graphics Interchange Format, grafický formát určený pro rastrovou grafiku

PERL – Practical Extraction and Report Language, programovací jazyk

HTML – HyperText Markup Language, značkovací jazyk používaný především pro tvorbu webových stránek

UDP – User Datagram Protocol, protokol transportní vrstvy modelu OSI/ISO

TCP – Transmission Control Protocol, protokol transportní vrstvy modelu OSI/ISO

SSH – Secure Shell, program a zároveň i zabezpečený komunikační protokol

SQL – Structured Query Language, dotazovací jazyk používaný pro práci s daty

MySQL – typ databáze podporující SQL dotazy

ACL – Access Control List, používá se pro definici seznamu oprávnění k nějakým objektům

IE nebo MSIE – Microsoft Internet Explorer, proprietární webový prohlížeč společnosti Microsoft

RRD – Round Robin Database, používá k uchovávání dat v programu RRDtool

RRA – Round Robin Archive, způsob uložení dat v archivu

Unix Timestamp – časová epocha, počet sekund které uplynuly od 1. 1. 1970

UML – Unified Modeling Language, grafický jazyk, který se používá pro vizualizaci programových systémů

URL – Uniform Resource Locator, udává umístění zdroje informací

Použitá literatura

[1] GERARD PAUL, Java. *IPTraf – An IP Network Monitor: IPTraf User's Manual* [online]. 1997-2001 [cit. 2009-04-08]. Dostupný z WWW: <http://iptraf.seul.org/2.7/manual.html>.

[2] Ncurses. *Wikipedia, the free encyclopedia* [online]. 2009 [cit. 2009-04-08]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Ncurses>.

[3] AYUSO, Pablo Neira. *Netfilter/iptables project homepage* [online]. 1999-2008 [cit. 2009-04-08]. Dostupný z WWW: <http://iptables.org/>.

[4] KRETCHMAR, James M., DOSTÁLEK, Libor. *Administrace a diagnostika sítí: pomocí OpenSource utilit a nástrojů*. Brno : Computer Press, 2004. 216 s. ISBN 80-251-0345-5.

[5] TOIVOLA, Teemu. *VnStat – network traffic monitor for Linux and *BSD* [online]. 2002-2009 [cit. 2009-04-22]. Dostupný z WWW: <http://humdi.net/vnstat/>.

[6] DIJKSTRA, Bjorge. *Vnstat PHP frontend* [online]. 2005-2007 [cit. 2009-04-22]. Dostupný z WWW: <http://www.sqweek.com/sqweek/index.php?p=1>.

[7] LIGHTFOOT, Chris, WARREN, Paul. *Iftop: display bandwidth usage on an interface* [online]. 2002-2006 [cit. 2009-04-22]. Dostupný z WWW: <http://www.ex-parrot.com/~pdw/iftop/>.

[8] *Ntop – network top* [online]. 1998-2009 [cit. 2009-04-22]. Dostupný z WWW: <http://www.ntop.org/>.

[9] Carnegie-Mellon University. *Net-SNMP* [online]. 1992-2007 [cit. 2009-04-22]. Dostupný z WWW: <http://www.net-snmp.org/>.

[10] OETIKER, *Tobi. Tobi Oetiker's MRTG – The Multi Router Traffic Grapher* [online]. 1995-2008 [cit. 2009-04-22]. Dostupný z WWW: <http://oss.oetiker.ch/mrtg/>.

[11] CHANTRA, (pseudonym). How-To: Remote syslog logging on Debian and Ubuntu. *Debian/Ubuntu Tips & Tricks* [online]. 2008 [cit. 2009-04-22]. Dostupný z WWW: <http://www.debuntu.org/how-to-remote-syslog-logging-debian-and-ubuntu>.

[12] Hess, Joey: *Nc(1)* [manuálová stránka]. [cit. 2009-04-22]. Součást instalace Debian Lenny 5.0 GNU/Linux.

[13] Elliot, Lee: *Rdate(1)* [manuálová stránka]. 2001-05-01. [cit. 2009-04-22]. Součást instalace Debian Lenny 5.0 GNU/Linux.

[14] *Inetd(8)* [manuálová stránka]. [cit. 2009-04-22]. Součást instalace Debian Lenny 5.0 GNU/Linux.

[15] OETIKER, Tobias. *RRDtool – About RRDtool* [online]. 1999-2009 [cit. 2009-04-28]. Dostupný z WWW: <http://oss.oetiker.ch/rrdtool/>.

[16] GREANT, Zak, RICHTER, Georg. Ext/mysqli: Part I - Overview and Prepared Statements. *Zend developer zone* [online]. 2004 [cit. 2009-07-06]. Dostupný z WWW: <http://devzone.zend.com/article/686-extmysqli-Part-I---Overview-and-Prepared-Statements>.

Seznam ilustrací