VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

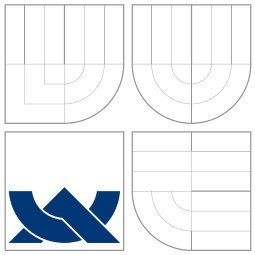
NETWORK MONITOR PRO GNOME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

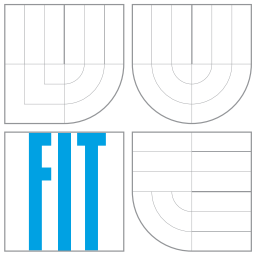
AUTOR PRÁCE
AUTHOR

JAKUB HROZEK

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NETWORK MONITOR PRO GNOME

GNOME NETWORK MONITOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB HROZEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ VOJNAR, Ph.D.

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Hrozek Jakub**
Obor: Informační technologie
Téma: **Network monitor pro Gnome**
Kategorie: Operační systémy

Pokyny:

1. Prostudujte problematiku monitorování síťového provozu v OS Linux.
2. Seznamte se s existujícími grafickými aplikacemi pro sledování síťového provozu.
3. Na základě získaných zkušeností a na základě požadavků společnosti Red Hat, ve spolupráci s níž je projekt řešen, navrhněte vlastní aplikaci pro sledování síťového provozu. Tato aplikace by měla zobrazovat aktuální síťová připojení, statistiky z firewallu, statistiky programu netstat a informace o síťových rozhraních.
4. V prostředí OS Linux naimplementujte navrženou aplikaci pomocí jazyka Python a grafického toolkitu PyGTK tak, aby vzhledově odpovídala prostředí Gnome.
5. Zhodnoťte vytvořenou aplikaci a diskutujte možnosti jejího dalšího rozvoje.

Literatura:

- Dle doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- První tři body zadání.

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

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

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

Vedoucí: **Vojnar Tomáš, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

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

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jakub Hrozek**
Id studenta: 84357
Bytem: Vrchní 164, 683 54 Bošovice
Narozen: 13. 09. 1984, Vyškov
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   bakalářská práce

Název VŠKP: Network monitor pro Gnome
Vedoucí/školitel VŠKP: Vojnar Tomáš, Ing., Ph.D.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Práce představuje novou grafickou aplikaci pro monitorování síťového provozu pro operační systém Linux. Jejím hlavním přínosem je, že je první linuxovou aplikací svého druhu, monitoruje široké spektrum aspektů síťové komunikace a zároveň poskytuje pohodlné grafické rozhraní. Je současně integrována do desktopového prostředí GNOME s možností začlenění do distribuce Fedora. Byly prostudovány a porovnány podobné existující programy, klíčovou částí práce je pak samotný návrh, implementace a testování nástroje.

Klíčová slova

GNOME, Gtk, Python, sledování sítě

Abstract

The thesis presents a new network monitoring application for the Linux operating system. It is the first graphical application with such functionality which monitors wide range of network traffic aspects and operates in graphical environment. It is integrated into the GNOME desktop environment and is also integrated into Fedora Linux distribution. The thesis discusses similar existing tools, the key part of the thesis is design, implementation and testing of the application.

Keywords

GNOME, Gtk, Python, network monitoring

Citace

Jakub Hrozek: Network Monitor pro GNOME, bakalářská práce, Brno, FIT VUT v Brně, 2007

Network Monitor pro GNOME

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Vojnara, Ph.D. a Ing. Radka Vokála.

.....
Jakub Hrozek
14. května 2007

Poděkování

Na tomto místě bych rád poděkoval vedoucímu své práce Ing. Tomáši Vojnarovi, Ph.D. a technickému vedoucímu práce Ing. Radku Vokálovi za jejich ochotu, trpělivost a velmi cenné rady.

Obsah

Kapitola 1

Úvod

Počítačové síťe jsou dnes nezbytnou součástí téměř každé organizace, a proto je vyvíjen tlak na jejich spolehlivost a bezpečnost. Z tohoto důvodu je třeba klást důraz na kvalitu nástrojů pro monitorování komunikace na síti.

Monitorování může být zaměřeno na různé aspekty síťového provozu. Jedním z nich je udržování přehledu o celkovém vytížení síťových zařízení a o tom, na jakých portech které aplikace komunikují. Důležitým prvkem je také sledování logů a správnosti nastavení firewallu. V některých případech (v případech, kdy byla zjištěna chyba v nastavaní sítě a je třeba odhalit její přesnou příčinu) probíhá specializovanější monitorování, například na úrovni obsahu paketů.

Tato bakalářská práce se zabývá návrhem a implementací linuxové grafické aplikace s názvem GNOME Network Monitor pro sledování síťového provozu na základě požadavků společnosti Red Hat, ve spolupráci s níž je řešena.

V současné době již existuje několik podobých nástrojů, avšak jejich zaměření je jiného charakteru. Jedná se buď o aplikace výhradně s textovým rozhraním (netstat, iptraf) nebo programy zaměřené na jedno konkrétní hledisko provozu na sítí (Wireshark). Podobné aplikace sice existují pro operační systém Microsoft Windows, žádná z nich ale není přenositelná na Linux, většina z nich je navíc uvolněna pod proprietární licencí.

Práce je rozdělena do sedmi kapitol. Kapitola 2 podává stručný popis existujících podobných aplikací a jejich porovnání s vyvíjeným programem. V kapitole 3 je popsáno rozdělení požadované funkcionality na jednotlivé komponenty a návrh programu. Kapitoly 4 a 5 uvádějí soubory, ze kterých aplikace čerpá zobrazované informace a technologie, použité při vývoji programu. Samotná implementace a distribuce programu je zhodnocena v kapitole 6.

Kapitola 2

Podobné existující programy

Protože operační systém Linux má časté využití na serverech, existuje pro něj mnoho různých nástrojů na sledování síťě a provozu na ní. V této kapitole jsou stručně uvedeny charakteristiky několika z nich a jejich srovnání s požadavky na vytvářený program GNOME Network Monitor. Zmiňuje se také o některých podobných aplikací pro operační systém Microsoft Windows.

2.1 Nástroje pro Linux

V Linuxových systémech jsou oproti systému Windows běžné utility, které mají pouze textové rozhraní. Ty mají některé výhody - není potřeba mít spuštěno (nebo, jak bývá z bezpečnostních a výkonnostních důvodů běžné na serverech, ani nainstalováno) grafické prostředí, je možné je skriptovat nebo pomocí rour propojovat s jinými programy. Pro uživatele, který nemá s tímto typem programů zkušenosti, ale mohou být obtížné na ovládání. Popisované programy se oproti zadané aplikaci také často specializují na jeden aspekt síťového provozu.

2.1.1 Netstat

Z nástrojů pro systém Linux (existuje ale také verze standardně dodávaná s operačním systémem Microsoft Windows) je *netstat* funkcionalitou asi nejpodobnější vytvářenému programu, nespecializuje se totiž na jednu oblast síťového provozu, ale poskytuje souhrnné informace. Mezi typická použití programu netstat patří kontrola, na kterém portu jaká aplikace komunikuje. Ke své činnosti navíc nepotřebuje práva superuživatele. Typická použití programu netstat uvádí tabulka 2.1.1.

Volání programu	Výstup
`netstat -r`	Routovací tabulka jádra
`netstat -i`	Základní statistiky síťových rozhraní
`netstat -s`	Souhrnné statistiky pro každý síťový protokol
`netstat -ta`	Aktivní i naslouchající TCP spojení

Tabulka 2.1: Příklady použití programu netstat

Volání programu	Výstup
`tcpdump icmp`	Veškerý provoz na protokolu ICMP
`tcpdump src 10.5.2.3\ and dst port 3389`	Provoz od počítače 10.5.2.3 směrovaný na port 3389
`tcpdump dst 192.168.0.2\ and src net 172.16.0.0/16`	Provoz směřující ze sítě 172.16 na počítač 192.168.0.2

Tabulka 2.2: Příklady použití programu tcpdump

2.1.2 Tcpdump

Tcpdump je textová aplikace, která umožňuje odchytávat a zobrazovat pakety na síti (tzv. *packet sniffing*). Pro správnou funkci je třeba, aby byl tcpdump spuštěn s právy superuživatele (nebo mít nastaven setuid bit). Síťové rozhraní, na kterém chceme odchytávat pakety musí být v tzv. *promiskuitním módu*, kdy uzel na síti přijímá i pakety, které mu nejsou přímo adresované.

Častěji než k získávání přehledu o celkovém dění na síti je program tcpdump používán k ladění komunikace mezi dvěma aplikacemi. Umožňuje také snadno zobrazit obsah nešifrované komunikace - např. využívající protokolu HTTP. Několik typických příkladů použití je uvedeno v tabulce 2.1.2.

2.1.3 Wireshark

Program *Wireshark* (dříve vyvíjený pod názvem Ethereal) je funkcionalitou podobný programu tcpdump, slouží tedy primárně k analýze komunikace na síťovém rozhraní. Oproti nástroji tcpdump má ale grafické rozhraní a velké množství filtrů pro omezení zobrazení různých síťových protokolů.

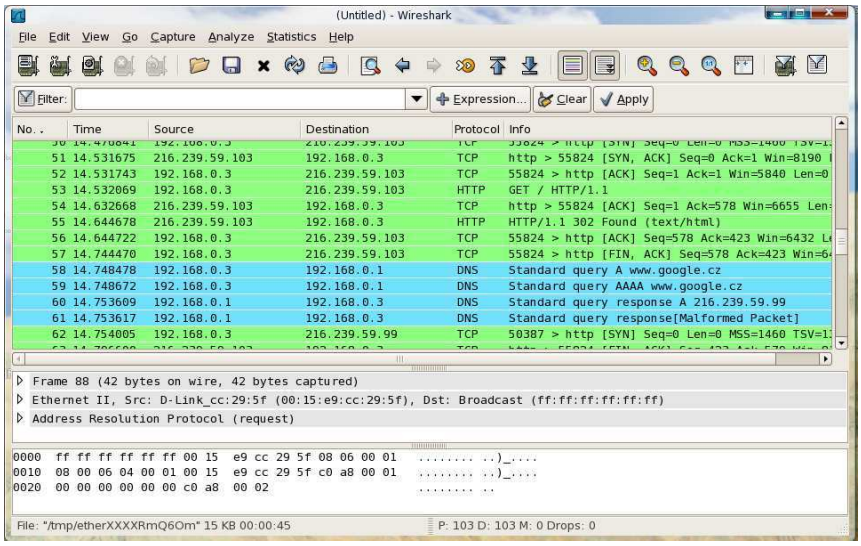
K tomu, aby byly dostupné všechny funkce (konkrétně odchytávání paketů), je třeba, aby byl Wireshark spuštěn s právy superuživatele. Protože se ale jedná o grafický program, běží pak i jeho GUI s EUID 0, což může být z bezpečnostního hlediska nepřijatelné. Proto je při startu programu zobrazen dialog, který uživateli nabízí spuštění Wiresharku v privilegovaném nebo neprivilegováném módu - tento postup byl použit i při vytváření programu GNOME Network Monitor (viz kapitola 3.4.3. Na obrázku 2.1 je zachycen Wireshark při odchytávání všech paketů ze síťového rozhraní.

2.2 Nástroje pro Microsoft Windows

Protože převážná většina aplikací pro operační systém Microsoft Windows je grafická, existují pro tuto plaformu nástroje, které se zaměřením a vlastnostmi více blíží vytvářenému programu.

2.2.1 X-Netstat Professional

Aplikace X-Netstat Professional je ze zde uvedených nástrojů svými funkcemi nejvíce podobná vytvářenému programu. Slouží primárně ke sledování aktivních síťových spojení, podobně jako textový netstat. Kromě toho zobrazuje podrobné informace o síťových zařízeních a souhrnné statistiky síťového provozu. Kromě pasivního monitorování umožňuje také ukončit vybraný proces nebo všechna spojení s vybranými parametry zaznamenávat do

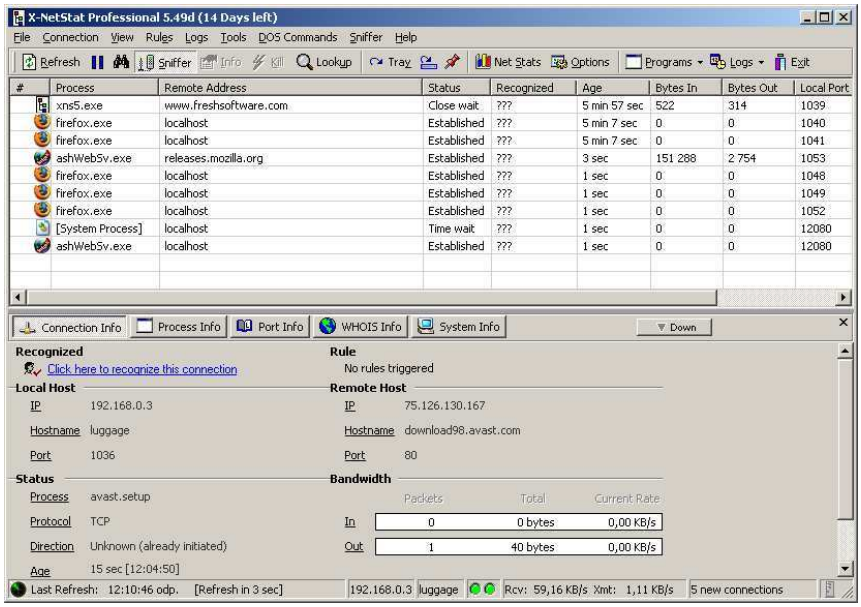


Obrázek 2.1: Ukázka z programu Wireshark

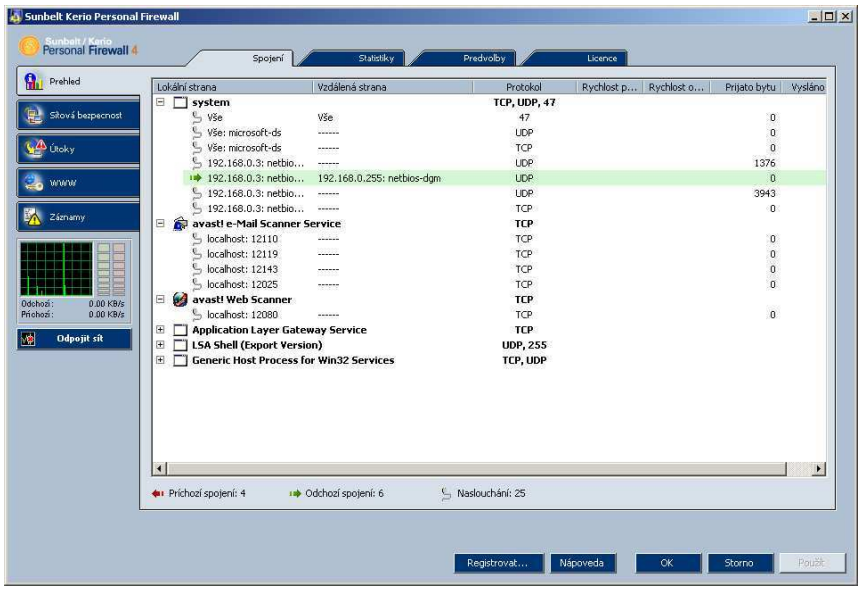
logu pro jejich další analýzu. Na obrázku 2.2 je zachycen pohled na procesy komunikující po síti v programu X-Netstat Professional.

2.2.2 Kerio Personal Firewall

Hlavní funkcí této aplikace je ochrana počítače formou filtrování příchozích a odchozích spojení. Protože ale kontrola správnosti nastavení firewallu patří mezi hlavní úlohy při sledování sítě, obsahuje také poměrně sofistikovanou monitorovací komponentu. Ta umožňuje udržovat přehled o zatížení sítě a procesy, které po síti komunikují. Obsahuje také grafický histogram zatížení síťových rozhraní. Pohled na monitorovací komponentu je na obrázku 2.3.



Obrázek 2.2: X-Netstat Professional



Obrázek 2.3: Kerio Personal Firewall

Kapitola 3

Funkcionalita a návrh programu

Tato kapitola se podrobněji zabývá popisem funkcí a vlastností programu, tak, jak byly formulovány zadáním práce a požadavky společnosti Red Hat. Obsahuje také popis architektury programu a diagram tříd.

3.1 Monitorovací komponenty

V této podkapitole práce se zabývám detailním popisem funkcí jednotlivých částí programu. Celkem se jedná o čtyři níže uvedené části, každá z nich představuje jednu monitorovací komponentu a v grafickém rozhraní také jednu záložku okna.

3.1.1 Souhrnné statistiky síťového provozu

Komponenta souhrnné síťové statistiky zobrazuje informace o síťovém provozu na úrovni protokolů TCP, UDP, IP a ICMP. Statistiky jsou tříděny podle protokolu, ke kterému se vztahují. Uživatel si také může nastavit časový interval pro automatické obnovování zobrazovaných informací.

3.1.2 Pohled na procesy komunikující po síti

Program zobrazuje všechny procesy, které komunikují po síti, a to jak ty aktivně komunikující, tak procesy, které pouze naslouchají a čekají na příchozí spojení. Údaj o spojení obsahuje síťový protokol, pomocí kterého proces komunikuje (TCP nebo UDP), adresu a port místní i vzdálené strany, stav spojení a pokud lze zjistit, tak i jméno nebo PID komunikujícího procesu. Pro větší přehlednost je možné přeložit IP adresy komunikujících stran na doménová jména nebo čísla portů na názvy služeb. Komponenta grafického rozhraní, která zobrazuje síťová spojení umožňuje údaje pro snadnější orientaci uživatele třídit a nastavit časový interval, ve kterém budou informace automaticky obnovovány.

3.1.3 Statistiky síťových rozhraní

Část aplikace, monitorující statistiky síťových zařízení, zobrazuje seznam dostupných rozhraní v systému. Pro každé rozhraní je kromě tradičního Unixového jména (např. `eth0`, `tun1`, ...), i jeho podrobnější popis – například „Wireless Interface“ pro `wlan`. Pro zvolené zařízení program zobrazuje následující informace:

- IP adresa

- MAC adresa
- síťová maska
- aktuální objem přenášených dat
- počet paketů přenesených oběma směry
- objem dat celkem přenesených přes síťové rozhraní

Objem přenášených dat za poslední minutu je také vykreslen jako grafický histogram, který obsahuje křivku pro příchozí a odchozí data. Všechny zobrazované údaje jsou aktualizovány jednou za vteřinu.

3.1.4 Informace z logů firewallu

Komponenta, která zobrazuje informace, logované standardním firewallem netfilter, je dostupná pouze, pokud je program spuštěn s právy superuživatele. Zprávy, nalezené v logovacích souborech jsou jednak přímo zobrazeny, ale také z nich program zpracovává několik statistik – seznam portů, na něž firewall nejčastěji nepovolí spojení a IP adresy (případně doménová jména) počítačů, které netfilter nejvíce odmítá. Program je schopen kromě typicky používaného souboru `/var/log/messages` vyhledat v konfiguraci logovacího démona syslog další soubory, do kterých mohou být ukládány zprávy z netfilteru.

3.2 Integrace s GNOME

Distribuce firmy Red Hat (komunitní větev Fedora i komerční Red Hat Enterprise Linux) používají jako své výchozí desktopové prostředí GNOME, postavené na grafickém toolkitu GTK. Jedním z požadavků zadání firmy Red Hat byla integrace aplikace s tímto grafickým prostředím. Vzhled, filozofie ovládání a například klávesové zkratky jsou podobné jako u ostatních GNOME aplikací.

3.2.1 Human Interface Guidelines

Dokument nazvaný Human Interface Guidelines [7] popisuje doporučení pro návrháře a vývojáře aplikací pro GNOME. Obsahem jsou jak obecné principy prostředí GNOME (například známé KISS - Keep It Simple, Stupid), tak konkrétní rady pro návrh grafických rozhraní, rozložení jednotlivých komponent nebo standardní klávesové zkratky.

Cílem takových doporučení je snadnější používání GNOME aplikací - principy, které si uživatel osvojí v jedné aplikaci jsou platné ve všech podobně navržených.

3.2.2 GConf

Desktopové prostředí GNOME používá pro ukládání nastavení svých aplikací systém s architekturou klient - server, nazvaný GConf. Nad databází, ve které jsou uložena uživatelská nastavení [1], běží uživatelský démon *GConfd*. Ten zaznamenává veškeré změny v databázi a upozorňuje na ně aplikace, které tato nastavení používají. Důsledkem je, že změna hodnoty se projeví okamžitě ve všech aplikacích které jej používají (tzv. *Auto-apply*).

[1] Zatím je implementován pouze XML backend

Samotná nastavení jsou uložena ve formátu klíč–hodnota ve struktuře, která je podobná např. registrům systému Windows. Nastavení aplikací jsou typicky uložena v cestě `/apps/aplikace`, pro můj program jsem proto zvolil `/apps/gnome-network-monitor`. Každý klíč s sebou nese také metadata, nazvaná *schéma* která popisují (tento popis je vhodné zahrnout do překladu) danou volbu nastavení a uvádějí její výchozí hodnotu.

3.3 Diagram tříd

Ze zadání práce je možné vysledovat logické rozdělení na jednotlivé komponenty programu tj. souhrnné informace o síťovém provozu, informace o síťových zařízeních, pohled na síťové procesy a statistiky z firewallu.

Každá komponenta se skládá z části, která tyto informace zobrazuje do grafického rozhraní a z části, která ze systému sbírá požadované informace (viz. 4). Komponenty, které zobrazují informace, jsou dále integrovány do hlavního okna aplikace. Cílem takového rozdělení sběru dat od jejich prezentace je usnadnění případných pozdějších změn jak v modulech pro získávání informací, tak naopak v grafickém rozhraní aplikace (nebo např. i její kompletní přepis do jiného grafického prostředí - Qt/KDE, WxWidgets,. . .). Kromě toho je třeba vytvořit třídu pro ukládání konfigurace (podrobněji se o uložení konfigurace zmiňuje kapitola 3.2.2).

Každá část programu je implementována jako samostatná třída v jazyce Python. Podrobný pohled na hierarchii tříd je na obrázku 3.1. Pro přehlednost neobsahuje diagram atributy tříd, pouze třídy samotné a vztahy mezi nimi. Atributy tříd i s jejich podrobnějším popisem ve formě dokumentačních komentářů je možné nalézt např. v generované dokumentaci ke zdrojovým kódům - A.

Hlavní okno aplikace reprezentuje třída `MainWindow`. Část okna se záložkami, které obsahují jednotlivé komponenty jsou implementovány ve třídě `Notebook`. Každá záložka je reprezentována vlastní třídou (`TabNetstat`, `TabSnmp`, `TabIptables` a `TabInterfaceStats`), ty se starají o aktualizaci a grafickou reprezentaci získaných informací. Každá z těchto specializovaných tříd dědí z třídy `Tab`, která obsahuje kód společný všem záložkám. Samotný sběr zobrazovaných informací je implementován ve specializovaných třídách, případně dalších podtřídách. Každá komponenta, která umožňuje nastavovat vlastnosti zobrazení má svoji vlastní třídu, která obsahuje její konfiguraci. Tyto konfigurační třídy mají jako svého předka třídu `ConfigBase`, která implementuje jim společné atributy.

3.4 Další oblasti návrhu

Kromě návrhu částí programu, které implementují monitorování síťového provozu, je také třeba vyřešit několik dalších problémů souvisejících s vývojem aplikace a také jejím snadným používáním. Patří sem například možnost pohodlné instalace programu, intuitivní umístění do aplikační nabídky nebo možnost čtení systémových logů při současném zachování možnosti spustit aplikaci jako neprivilegovaný uživatel.

3.4.1 Instalační balíčky

Pro distribuci programu směrem k uživateli je důležité, aby jej bylo možné snadno nainstalovat, a odistalovat. Tato podkapitola diskutuje některé varianty, které jsem při návrhu aplikace zvažoval.

Tradiční metodou je použití programu GNU make [10]. Tento způsob není příliš uživatelsky přívětivý, obtížná je také distribuce nových verzí programu nebo instalace knihoven, které aplikace vyžaduje. Výhodou ale je, že instalace pomocí make není závislá na konkrétní linuxové distribuci a mohou ji tedy použít uživatelé, pro jejichž distribuce neexistuje binární balíček.

Většina linuxových distribucí také nabízí možnost použití předkompilovaných binárních balíčků pomocí takzvaného správce softwaru. Ten kromě samotného kopírování souborů na disk zajistí i doinstalací případných dalších programů či knihoven, které instalovaná aplikace potřebuje ke svému chodu. Formát instalačního souboru se liší s konkrétní linuxovou distribucí. Protože distribuce Fedora a RHEL používají formát RPM [8], je součástí implementace také vytvoření takového instalačního balíčku.

3.4.2 Umístění do aplikačního menu

Aby uživatel mohl aplikaci po nainstalování snadno používat, musí mít program intuitivní zařazení do aplikačního menu. Standardy skupiny freedesktop.org definují (například v [2]) za tímto účelem soubor, který se umísťuje do `/usr/share/applications` a má název `aplikace.desktop`, pro náš program tedy `gnome-network-monitor.desktop`. Tento soubor obsahuje například příkaz, který se má spustit po zvolení programu z nabídky, stručný popis funkce programu, kategorii, v níž je zařazen nebo ikonu aplikace.

3.4.3 Čtení logů neprivilegovaným uživatelem

Většina linuxových distribucí umožňuje čtení standardních logů pouze superuživateli [2]. Vytvářená aplikace je ale zamýšlena pro spouštění neprivilegovaným uživatelem. Je proto třeba najít způsob, jak zpřístupnit systémové logy pro čtení jinému uživateli než root.

Při návrhu programu jsem zvažoval tyto alternativy:

- Nastavení setuid bitu pro vytvářený program. Spustitelný soubor by vlastnil uživatel root a s jeho právy by také celý program běžel. Výhodou takového přístupu je, že je nejpohodlnější pro uživatele, ten nemusí zadávat při spouštění programu žádné heslo. To je zároveň největší nevýhoda - v systému pak existuje aplikace, která bez nutnosti zadat heslo běží s nejvyššími právy a případná chyba v ní může kompromitovat celý systém. Většina administrátorů se snaží omezit počet takových programů ve svém systému na minimum a kód setuid programů často prochází bezpečnostním auditem.

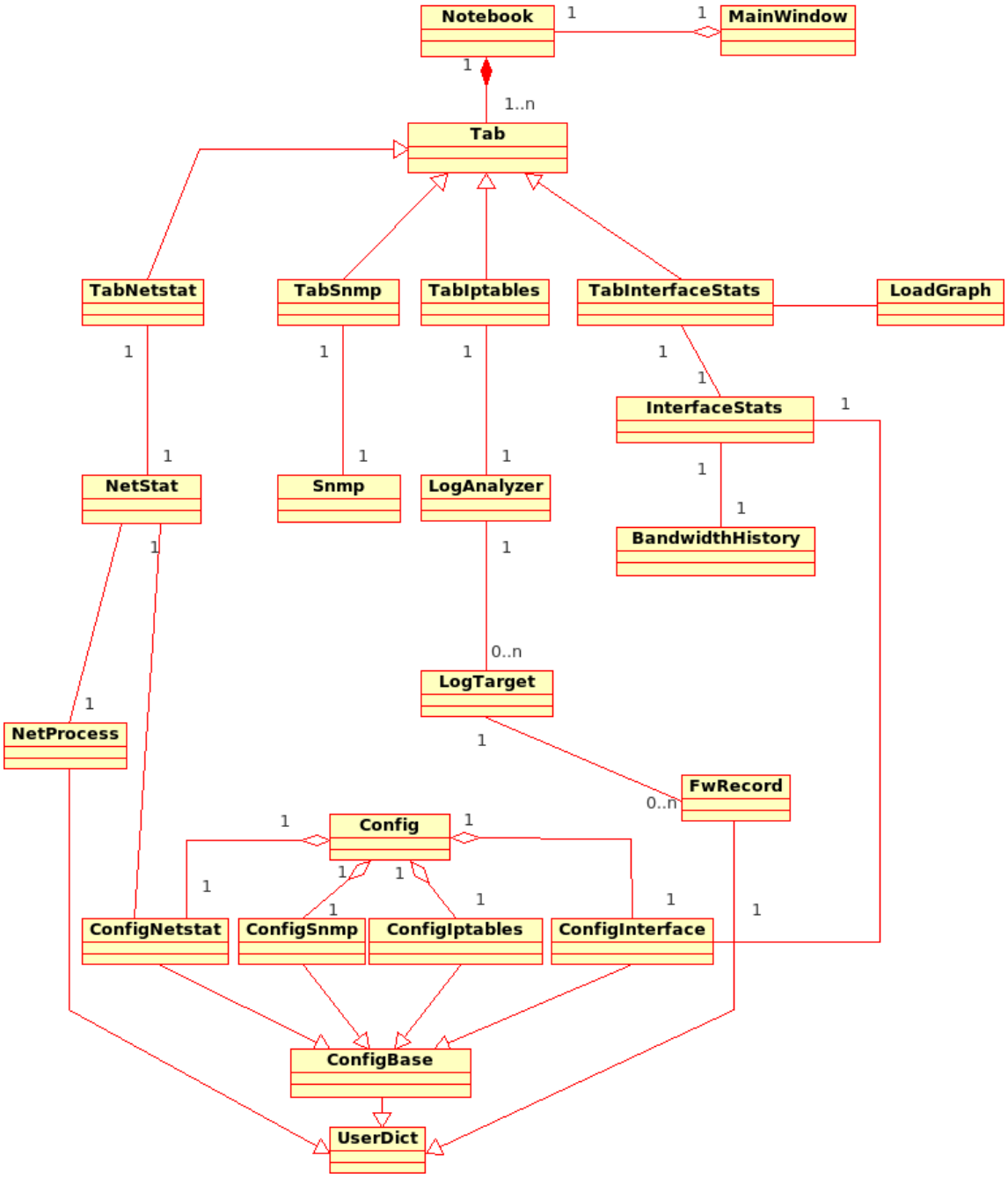
- Spouštění přes pomocný program, který zajistí případné zvýšení práv. Protože ostatní komponenty aplikace prává uživatele root nevyžadují, je možné jej spustit s EID 0 jen pokud je třeba číst logy z firewallu. Způsob nastavení aplikace pro spouštění s administrátorskými právy se liší v každé linuxové distribuci. Systémy Fedora a RHEL používají nástroj *usermode*, kdy je při spuštění třeba zadat heslo uživatele root, v distribucích Ubuntu je zvykem spouštění přes utilitu *sudo*, při kterém uživatel

[2] Jedná se typicky o soubory v adresáři `/var/log`, konfigurace logovacího démona `syslog(8)` umožňuje přesměrovat zprávy do jakéhokoliv souboru

musí mít záznam v konfiguračním souboru `/etc/sudoers` a zadává své uživatelské heslo. Plusem takového řešení je, že aplikace běží s UID 0 jen, pokud je třeba číst logy. Nevýhodou je, že s nejvyššímy právy běží celá aplikace včetně grafického rozhraní, takže bezpečnostní chyba v grafickém toolkitu může ohrozit celý systém.

- Samostatný démon, který čte informace z logů a předává je grafické aplikaci. Výhodou je, že s právy uživatele root běží pouze démon oddělený od zbytku aplikace, která pouze zobrazuje dodané informace. Kód démona by také byl výrazně jednodušši a proto by bylo jednodušší provést audit jeho kódu. Pro komunikaci mezi těmito komponentami přichází v úvahu například protokoly CORBA nebo D-Bus. Nevýhodou je, že démon by musel běžet stále, což zatěžuje systém a pro jednoduchý grafický program je samostatná služba neúměrně silné řešení.

Při implementaci programu jsem zvolil přístup se spouštěním přes pomocný program. Jedná se totiž o řešení, které pro získání administrátorských práv vyžaduje znalost root hesla v případě použití usermode nebo zásah administrátora pro spouštění pomocí sudo. Zároveň neúměrně nezatěžuje systém běžícím službou, jako v případě řešení se samostatným démonem.



Obrázek 3.1: Diagram tříd

Kapitola 4

Zdroje informací pro program

Úlohou tvořeného programu je zobrazování informací o různých aspektech síťového provozu. Jednou z hlavních otázek, kterou bylo třeba vyřešit, je tudíž problematika zdrojů informací pro aplikaci, tj. místa v systému, ze kterých je možné číst údaje, které zobrazuje grafické rozhraní programu.

4.1 Souborový systém procfs

Většina informací, které program zobrazuje, existuje v nějaké podobě ve virtuálním souborovém systému `procfs`. Ten zobrazuje interní datové struktury jádra Linuxu a umožňuje tak uživatelským aplikacím sbírat informace o systému. Některé virtuální soubory jsou také zapisovatelné a změnou jejich obsahu je možné nastavit parametry jádra. Typicky bývá připojen do adresáře `/proc` a má hierarchickou strukturu organizovanou do virtuálních adresářů a souborů. Údaje o síťovém provozu jsou umístěny v adresáři `/proc/net/` [3]. Tabulka 4.1 ukazuje přehled všech souborů ze systému procfs, které program využívá, v dalším textu je pak detailněji popsán formát těchto souborů.

4.2 Zdroje informací pro jednotlivé komponenty

V této části práce jsou popsány zdroje informací pro jednotlivé monitorovací komponenty aplikace. S vyjímkou čtení informací z logů firewallu je možné všechny požadované informace číst ze souborového systému procfs.

Cesta k souboru	Získaná informace
`/proc/net/snmp`	Přehled statistik o síťovém provozu
`/proc/net/dev`	Data přenesená po síťových rozhraních
`/proc/net/tcp`	Otevřené TCP sockety
`/proc/net/udp`	Otevřené UDP sockety
`/proc/<pid>/fd`	Seznam otevřených souborů včetně socketů
`/proc/<pid>/status`	Jméno běžícího procesu

Tabulka 4.1: Přehled Použitých souborů z `procfs`

4.2.1 Souhrnné statistiky síťového provozu

Souhrnné informace o síťovém provozu rozdělené podle zobrazených protokolů (TCP, IP, UDP a ICMP) se nacházejí v souboru `/proc/net/snmp`. Je z něj možné zjistit například počty přenesených paketů nebo počet otevřených spojení. Údaje pro každý protokol jsou na dvou řádcích, první obsahuje popisky jednolivých polí, druhý samotné hodnoty jako desítková čísla.

4.2.2 Pohled na procesy komunikující na síti

Pro zobrazení procesů komunikujících na síti je v souborech `/proc/net/tcp` a `/proc/net/udp` k dispozici seznam otevřených socketů pro síťové protokoly TCP, respektive UDP. Pro každý socket je možné z těchto souborů mimo jiné zjistit IP adresu a port místní i vzdálené strany komunikace, uid procesu, kterému tato schránka patří a číslo i-node schránky. Popis všech polí lze najít např. v [4] nebo v [9].

Formát IP adresy a portu komunikujícího procesu

Pro reprezentaci IP adresy a portu se v souborech v procfs často používá notace, kdy je zapsána nejdříve IP adresa jako sekvence osmi hexadecimálních číslic a poté port jako posloupnost čtyř hexadecimálních číslic. Tato pole se oddělují dvojtečkou. Takový způsob zápisu se používá např. v souborech `/proc/net/tcp` a `/proc/net/udp`.

Přiřazení procesu k otevřenému socketu

Číslo i-node je možné využít k zjištění, kterému procesu otevřená schránka patří. Každý běžící proces v Linuxu má totiž v souborovém systému procfs adresář se stejným jménem, jako je PID procesu. Tento adresář obsahuje podadresář *fd*, ve kterém jsou uloženy symbolické odkazy na všechny otevřené soubory procesu, včetně socketů a rour. Za předpokladu, že máme dostatečná oprávnění ke vstupu do tohoto adresáře, respektive ke čtení souborů v něm obsažených, je možné procházet tyto „adresáře procesů“ a hledat čísla i-node, která jsme zjistili ze souborů `/proc/net/tcp` a `/proc/net/udp`. Pokud takové číslo i-uzlu najdeme, z názvu samotného adresáře, ve kterém se nacházíme je možné zjistit pid procesu, který používá socket s tímto i-node. V souboru `/proc/<pid>/status` se navíc v poli *Name* nachází jméno procesu, které je při zobrazování informací preferováno před údajem PID.

4.2.3 Statistiky síťových rozhraní

Program využívá ke zjištění informací o síťových zařízeních několika zdrojů. Metriky samotných rozhraní jsou získány pomocí systémového volání `ioctl` (respektive jemu ekvivaletní funkce stadartní knihovny Pythonu) a informace o přenesených datech se čtou ze souboru `/proc/net/dev`.

Adresy síťových rozhraní

IP adresy, přiřazená síťovému rozhraní jsou uloženy podobně jako většina ostatních informací zobrazovaných programem v souborovém systému procfs, konkrétně v souboru `/proc/net/arp`. Nejsou v něm ale uloženy informace o „virtuálních“ rozhraních, např. VPN tunelech. Proto jsem zvolil robustnější řešení ve formě systémového volání ioctl.

Parametr volání `ioctl`	Získaná informace	C struktura, její prvek
`SIOCGIFADDR`	IP adresa	`sockaddr_in.sin->addr`
`SIOCGIFHWADDR`	MAC adresa rozhraní	`ifreq.ifr_hwaddr.sa_family`
`SIOCGIFBRDADDR`	Adresa pro broadcast	`sockaddr_in.sin->addr`
`SIOCGIFNETMASK`	Maska rozhraní	`ifreq.ifr_addr`

Tabulka 4.2: Použité parametry volání ioctl

Sytémové volání ioctl umožňuje komunikovat s ovladačem zařízení a získat tak informace o jeho nastavení nebo v některých případech jeho nastavení měnit. Pro každé zařízení má ioctl jiné argumenty nebo návratové kódy, obecně tyto charakteristiky dány parametrem request. Seznam parametrů systémového volání ioctl lze nalézt například v manuálových stránkách [1] (konkrétně stránka `ioctl_list`(2)). Pokud je voláno na otevřený socket asociovaný s určitým síťovým rozhraním, je možné zjistit parametry tohoto rozhraní - viz. 4.2.3. Funkce standardní knihovny Pythonu jsou pouze mapování na odpovídající systémová volání v jazyce C a proto vrací stejné datové struktury. Python ale pojem struktura, ve smyslu, v jakém se používá v C nezná a proto jsou data mapována na typ string. Konverzi z něj zajišťuje modul `struct` standardní knihovny jazyka Python.

Objem přenesených dat

Důležitou informací při sledování sítě je objem dat, přenesený přes jednotlivá síťová rozhraní. Tyto údaje jsou uloženy v souboru `/proc/net/dev`. První dva řádky souboru obsahují popisky sloupců, samotné objemy datových přenosů jsou v druhém a třetím sloupci pro počet bytů resp. paketů směrovaných do síťového rozhraní, v desátém a jedenáctém jsou tytéž informace pro objem odchozích dat. Ukázka souboru `/proc/net/dev` je v příloze C.1.

4.3 Logy firewallu

Sledování logů patří k nejdůležitějším úkolům při administraci systému. Jedna z komponent vytvářeného programu má za úkol umožnit sledování a analýzu informací, které do logů posílá firewall. Časté použití analýzy logů firewallu je sledování, jaká spojení, případně z jakých strojů firewall odmítá a případná úprava bezpečnostní politiky.
Linux obsahuje od verze 2.2 firewall netfilter, uživatelský nástroj pro jeho administraci se jmenuje iptables. Pokud některé z pravidel firewallu obsahuje direktivu `LOG`, pošle netfilter zprávu do systémového logu. Pro sběr takových informací je třeba zjistit, do kterých souborů jsou zprávy ukládány. Zprávy, směřující do logů s sebou nesou informaci o zdroji a důležitosti logované zprávy. Příjem zpráv a jejich odesílání do specializovaných logů na základě těchto parametrů je úkolem systémového démona syslog.

Protože netfilter je součástí kernelu, jsou zprávy z firewallu označeny jako zprávy z jádra, tj. parametr zdroj má hodnotu kern, priorita zprávy je ve výchozím nastavení info [6]. Je ale možné při definici pravidla nastavit prioritu zprávy parametrem `--log-level`.

Komponenta programu, která hledá soubory, do kterých se ukládají logované informace tedy prohledá konfiguraci démona syslog a najde soubory,do kterých se ukládají zprávy z jádra s libovolnou prioritou.

Kapitola 5

Použité technologie

Po vytvoření návrhu a stanovení konkrétních funkcí aplikace je dalším krokem při vývoji výběr nástrojů a knihoven, které budou použity při implementaci a testování. Všechny použité nástroje jsou uvolněny pod svobodnou licencí, takže jsou snadno dostupné v repozitářích většiny linuxových distribucí.

5.1 Vývoj aplikace

Tato podkapitola popisuje nástroje a knihovny, použité při samotném vývoji programu nebo návrhnu jeho grafického rozhraní a také nástroj GNU gettext, který může být použit pro vytváření nových jazykových mutací aplikace.

5.1.1 GTK

GTK je grafický toolkit původně vyvinutý pro implementaci grafického editoru GIMP, dnes známý hlavně jako knihovna použitá v desktopovém prostředí GNOME. Red Hat používá GNOME jako své výchozí desktopové prostředí, zároveň je integrace s tímto prostředím jedním z bodů zadání programu, proto byla tatko knihovna zvolena pro implementaci projektu. Nejčastěji používaným jazykem při tvorbě GTK programů je C, ale k dispozici jsou i rozhraní pro Python, C#, C++ a další programovací jazyky.

5.1.2 Python

Python je interpretovaný, objektově orientovaný programovací jazyk. Pro implementaci projektu byl zvolen, protože je de facto standardem pro tvorbu grafických aplikací v distribucích Red Hatu. Návrh i implementace programu v Pythonu je efektivnější než např. v C. Navíc velká část práce programu spočívá ve zpracování textu, kde Python oproti C získává díky dynamické práci s pamětí a možnosti práce s regulárními výrazy.

5.1.3 Glade

Glade je vizuální nástroj na tvorbu grafických uživatelských rozhraní v GTK. Je primárně určen pro použití s jazykem C, existují ale i knihovny pro Python. Oproti tradičnímu přístupu, kdy je GUI vytvořeno přímo ve zdrojovém kódu, je výstupem programu Glade XML soubor, který popisuje rovržení komponent v oknech.

5.1.4 GNU gettext

Tento nástroj slouží k distribuci programů v různých jazykových mutacích a k usnadnění překladů, kdy pro vznik nového překladu není potřeba žádný zásah do samotného zdrojového textu.

Standartní knihovna jazyka Python obsahuje rozhraní k utilitě `gettext`, které je v programu použito na překlad speciálně označených textových řetezců přímo ve zdrojovém textu (např. chybová hlášení). Proces označování přeložitelných řetězců se nazývá internacionalizace, často se zkracuje na i18n. Takto zpracovaný kód se dá použít jako vstup utility `intltool-extract`, výstupem je tzv. *Portable Object Template*, který obsahuje seznam řetězců a jejich umístění v kódu.

Při vytváření nové jazykové mutace je třeba pro každý jazyk vygenerovat ze souboru typu Portable Object Template, je soubor typu *Portable Object*, ten obsahuje pro každý přeložitelný řetězec jemu odpovídající v jazyce, do kterého překládáme. Samotný proces překladu se často označuje jako lokalizace nebo zkratkou l10n.

Posledním krokem při lokalizaci programu je konverze souboru typu PO, který je zamýšlen pro čtení člověkem do formátu *Machine Object*, který je binární a používá jej přímo GNU gettext.

Protože je při vytváření programu použit nástroj Glade, je třeba stejným způsobem vygenerovat Portable Object i z XML souboru, ve kterém Glade ukládá popis grafického rozhraní.

5.2 Testování a doržování programovacích konvencí

Při vývoji programu bylo použito několik nástrojů, které sloužily na automatizovanou kontrolu funkčnosti (PyUnit a Dogtail) a kontrolu dodržování programátorských konvencí. Motivací k použití těchto nástrojů je fakt, že program je zamýšlen jako open source. Zdrojový kód bude volně přistupný uživatelům, kteří by jej potenciálně mohli upravovat, a proto je nutné zachovávat vseobecně rozšířené programovací konvence. V případě začleňování patchů od dalších vývojářů nebo úprav programu pro správnou inegraci do jiných linuxových distribucí než Fedora mohou pomoci automatizované testy ověřit, že je zachována správná funkcionalita aplikace.

5.2.1 PyUnit

Knihovna pro automatické testování PyUnit (často označovaná jako unittesting) je portem knihovny JUnit z programovacího jazyka Java. Slouží k ověřování, že jednotlivé části programu (typicky se k testování vybírají třídy nebo moduly) fungují správně, například test, zda pro předem známé vstupní hodnoty je výstupem funkce očekávaná návratová hodnota, zda je vyvolána výjimka v případě nekorektního vstupu, apod.

5.2.2 Pylint

Pro statickou analýzu správnosti zdorjového kódu byl při vývoji programu použit program Pylint. Motivací pro jeho použití byla dostupnost zdrojových kódů, kdy je třeba, aby byl text programu snadno čitelný a přehledný i pro ostatní vývojáře.

Umožňuje sledování všeobecně rozšířených programovacích konvencí, jako například přítomnost doc stringů, překrývání identifikátorů apod. Výstupem nástroje Pylint je HTML stránka (nebo dokument v jiném formátu), která obsahuje seznam zjištěných nedostatků a

celkové hodnocení modulu v rozsahu od 0 do 10. Při vývoji programu bylo vyžadováno na této stupnici hodnocení alespoň 8,5.

5.2.3 Dogtail

Dogtail je framework pro vytváření automatizovaných testů pro grafické aplikace. Oproti většině podobných nástrojů nevyužívá k popisu grafického rozhraní a komunikaci s testovanou aplikací souřadnice jeho komponent, ale asistenční technologie prostředí GNOME. Ty pomocí standardu pro meziprocesovou komunikaci CORBA simulují interakci s testovaným programem. Podrobnější popis architektury systému Dogtail je možné najít na [5]

Samotné testy jsou stejně jako Dogtail implemetovány v programovacím jazyce Python.

5.2.4 Epydoc

Pro usnadnění případných úprav programu po jeho vypuštění je také třeba, aby byl zdrojový kód dobře komentovaný a byla dostupná aktuální dokumentace. K tomu byl při vývoji použit nástroj epydoc. Umožňuje z dokumentačních komentářů (v Pythonu označovaných jako doc stringy) automaticky generovat popis jednotlivých tříd a modulů spolu s náhledy na prototypy funkcí a tříd nebo diagramy dědičnosti. Aby byla zajištěna aktuálnost takové dokumentace, bylo samotné generování podobně jako testy spouštěno automaticky démonem `cron` z kódu z repozitářů na SourceForge.net. Dokumentace k aktuálním zdrojovým kódům je dostupná na adrese `http://gnetworkmonitor.sourceforge.net/docs/`.

Kapitola 6

Implementace a distribuce

Podle postupů navržených v kapitolách 3 a s využitím technologií uvedených v kapitole 5 jsem implementoval program GNOME Network Monitor. V průběhu implementace jsem dosavadní výsledky průběžně konzultoval s technickým vedoucím práce. Několik náhledů na vytvořený program je na obrázcích 6.1 a 6.2.

6.1 Distribuce a prezentace programu

Po implementování všech požadovaných vlastností programu (tak, jak byly uvedeny v kapitole 3) jsem s použitím technik diskutovaných v kapitole 3.4.1 vytvořil instalační balíčky vytvořeného programu.

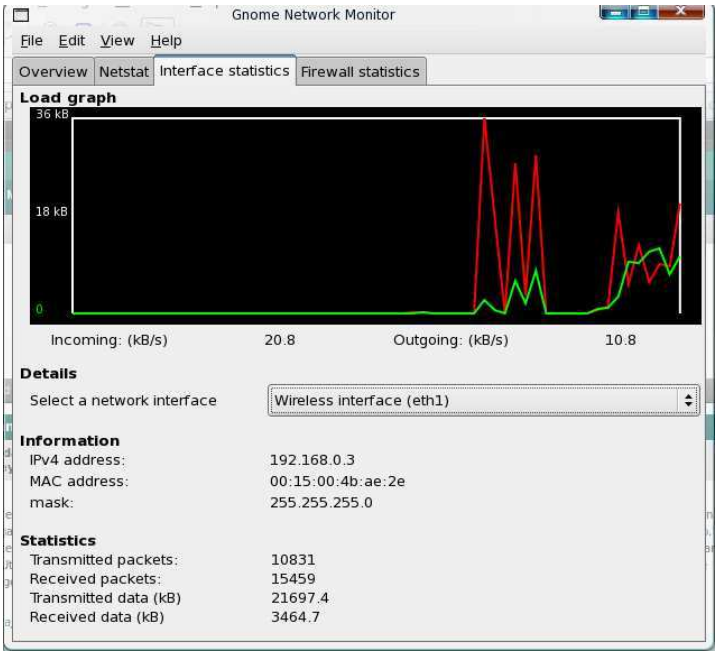
V případě distribuce ve formě zdrojových textů s možností instalace pomocí utility GNU make jsem se snažil vyhnout technikám specifickým pro konkrétní distribuci Linuxu (například pomocné programy na změnu uživatele, se kterým je aplikace spuštěna).

6.1.1 SourceForge.Net

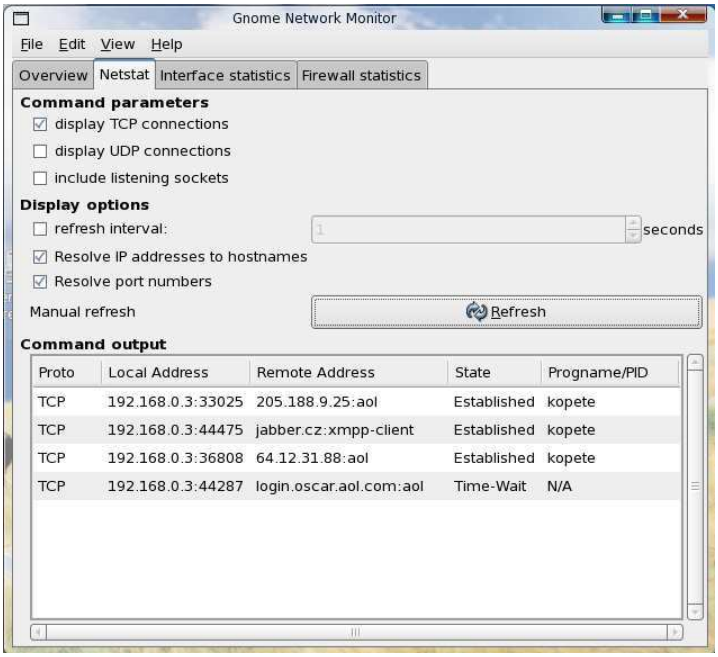
Zdrojové kódy programu byly od začátku vývoje umístěny v repozitářích služby SourceForge.Net. Ta kromě samotného systému správy zdrojových kódů nabízí také možnost vytvoření webové prezentace pro projekt, místo pro uložení ukázek programu nebo správu distribučních balíčků (`http://sourceforge.net/projects/gnetworkmonitor/download/`). Webová prezentace, umístěná na (viz `http://gnetworkmonitor.sf.net`) obsahuje stručný popis funkcí programu, odkazy, pomocí kterých je možné přímo nahlásit chybu v aplikaci a několik ukázek z hotového programu.

6.1.2 Fedora

V současné době probíhá začleňování aplikace do repozitářů distribuce Fedora. K tomu, aby byl program přijat, je nutné, aby byl vytvořen instalační balíček ve formátu RPM podle pravidel projektu Fedora, tak, jak jsou uvedeny na stránkách [11] a jeho specifikaci zkontroloval někdo z registrovaných recenzentů [12]. Samotná recenze je veřejná a probíhá v nástroji Bugzilla firmy Red Hat pod číslem 239435 (viz. A).



Obrázek 6.1: Statistiky síťových rozhraní



Obrázek 6.2: Přehled procesů komunikujících po síti

6.2 Testování

Pro testování funkctionality jednotlivých komponent programu je v samostatném modulu `unittesting` v době psaní práce asi 50 testů, které pokrývají téměř všechny části programu. Samotný běh testů byl plně automatizovaný, během vývoje aplikace byly testy spouštěny každou noc démonem *cron*. Pokud některý test neuspěl, odeslal testovací skript svůj výstup k analýze e-mailem.

Testy většinou využívají faktu, že program je vysoce konfigurovatelný, včetně možnosti zadávání cest k souborům, ze kterých čte informace. Proto je možné na vstup aplikace poslat soubor se známým obsahem a výstup testované části programu porovnat s předpokládanými daty – například předem známé informace o otevřených socketech porovnat s výstupem ve formě seznamu síťových procesů.

6.3 Další vývoj

Kromě funkcí a vlastností, implementovaných pro tuto bakalářskou práci, zahrnují požadavky firmy Red Hat několik dalších, které budou dokončeny později.

Jedná se například o těsnější spoluprácí s firewallem netfilter. Komponenta, která v současnosti pouze zobrazuje logy z netfilteru by v budoucnu měla být schopna uživateli vytvořit pravidlo pro firewall, kterým by se dané spojení povolilo nebo naopak zakázalo. Část aplikace, která zobrazuje seznam procesů, komunikujících po síti bude obsahovat možnost poslat zobrazenému procesu signál a tím jej například ukončit. Další uvažovanou funkcí související s tímto pohledem na procesy je běh celého programu v systémové liště [1] prostředí GNOME a upozorňování na některé procesy nebo nové záznamy v logu netfilteru formou grafických notifikací.

Kromě výše uvedených nových funkcí bude jistě třeba opravovat ve stávajících verzích programu nalezené chyby. K jejich hlášení mohou sloužit webová rozhraní služby SourceForge.Net [2]nebo, po začlenění projektu do distribuce Fedora, specializované nástroje jako Bugzilla.

[1] Nazývané také systray

[2] V době psaní práce nalezli a nahlásili uživatelé 4 chyby - viz. A

Kapitola 7

Závěr

V této práci jsem představil grafickou aplikaci pro monitorování síťového provozu pro operační systém Linux. Program byl navržen, implementován a testován ve spolupráci s firmou Red Hat.

Výsledná aplikace dodržuje konvence desktopového prostředí GNOME [7] a v budoucnu bude začleněna do Linuxové distribuce Fedora. Nástroj umožňuje sledovat procesy komunikující na síti, zobrazuje informace o jednotlivých síťových zařízeních včetně grafického histogramu zatížení a poskytuje přehledné statistiky o provozu. Pokud má uživatel dostatečná oprávnění, poskytuje aplikace také statistiky z firewallu netfilter, který je standardní součástí Linuxového jádra. Program je šířen pod licencí GNU GPL verze 2, tak jak je uvedena na stránkách FSF [13]. Zdrojové kódy, testy i další materiály jsou hostovány službou SourceForge.net.

Vývoj aplikace bude dále pokračovat, ať již v podobě přidávání nových funkcí nebo opravování nalezených chyb. Do budoucna plánuji také začleňování do dalších linuxových distribucí.

Literatura

[1] Manuálové stránky systému Fedora Core 6. ftp://ftp.fi.muni.cz/pub/linux/fedora-core/6/source/SRPMS/man-pages, poslední návštěva - květen 2007.

[2] Bastian Waldo, Gouget Francois, Graveley Alex a kol. Desktop Menu Specification. http://standards.freedesktop.org/menu-spec/latest/index.html, poslední návštěva - květen 2007.

[3] Zdrojové kódy jádra Linux verze 2.6.20-1. Documentation/filesystems/proc.txt. ftp://ftp.fi.muni.cz/pub/linux/fedora-core/6/source/SRPMS/, poslední návštěva - duben 2007.

[4] Zdrojové kódy jádra Linux verze 2.6.20-1. Documentation/networking/proc_net_tcp.txt. ftp://ftp.fi.muni.cz/pub/linux/fedora-core/6/source/SRPMS/, poslední návštěva - duben 2007.

[5] Di Maggio Len. Automated GUI testing with Dogtail. http://www.redhat.com/magazine/020jun06/features/dogtail , poslední návštěva - březen 2007.

[6] Andreasson Oskar. Iptables tutorial 1.2.2. http://iptables-tutorial.frozentux.net/iptables-tutorial.html, poslední návštěva - březen 2007.

[7] Benson Calum, Elman Adam, Nickell Seth. GNOME Human Interface Guidelines. http://developer.gnome.org/projects/gup/hig/, poslední návštěva - březen 2007.

[8] Domovská stránka balíčkovacího systému RPM. Dokumentace k formátu rpm. http://www.rpm.org/max-rpm/ , poslední návštěva - květen 2007.

[9] WWW stránky. Red Hat Enterprise Linux 4 Reference Guide. http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/ref-guide/, poslední návštěva - březen 2007.

[10] WWW stránky. Domovská stránka programu GNU Make. http://www.gnu.org/software/make/ , poslední návštěva - květen 2007.

[11] WWW stránky. Doporučení pro tvorbu RPM balíčků pro distribuci Fedora. http://fedoraproject.org/wiki/Packaging/Guidelines, poslední návštěva - květen 2007.

[12] WWW stránky. Pravidla pro přijetí balíčků pro distribuci fedora. `http://fedoraproject.org/wiki/Packaging/ReviewGuidelines`, poslední návštěva - květen 2007.

[13] Licenční text programu. GNU General Public License. `http://www.gnu.org/licenses/gpl.txt`, poslední návštěva - březen 2007.

Dodatek A

Odkazy na webové stránky, související s projektem

- `http://gnetworkmonitor.sf.net`
  Domovská stránka programu
- `http://gnetworkmonitor.sf.net/docs`
  Automaticky generovaná dokumentace ke zdrojovým kódům
- `http://sourceforge.net/tracker/?group_id=188966`
  Webové rozhraní, ve kterém mohou uživatelé hlásit chyby nalezené v aplikaci
- `http://sourceforge.net/projects/gnetworkmonitor/`
  Stránka projektu na službě SourceForge.net
- `http://sourceforge.net/projects/gnetworkmonitor/download/`
  Přímý odkaz na stažení zdrojových kódu a instalačních balíčků
- `https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=239435`
  Žádost o zařazení balíčku do repozitářů distribuce Fedora

Dodatek B

Seznam použitých zkratek a odborných výrazů

- *RHEL* – Red Hat Enterprise Linux
  Komerční distribuce Linuxu vyvíjená firmou Red Hat
  `http://www.redhat.com/rhel/`
- *Fedora*
  Komunitní distribuce Linuxu vyvíjená pod záštitou firmy Red Hat
  `http://www.fedoraproject.org`
- *GNU GPL* GNU General Public License
  Open Source licence, pod níž je šířeno mj. jádro Linuxu
  `http://www.fedoraproject.org`
- *GNOME* – GNU Object oriented Model Environment
  Grafické prostředí pro Unixové operační systémy.
  `http://www.gnome.org`
- *GTK* – GIMP ToolKit
  Knihovna, použitá pro implementaci grafického prostředí GNOME
  `http://www.gtk.org`
- *RPM* – RPM Package Manager (dříve Red Hat Package Manager)
  Formát instalačních balíčků, používaný v distribucích firmy Red Hat
  `http://www.rpm.org`
- *i-node*
  Datová struktura, sloužící k uložení informací o souboru
  `http://en.wikipedia.org/wiki/Inode`
- *PID* – Process identifier
  Číslo, jednoznačně identifikující proces
  `http://en.wikipedia.org/wiki/Pid`
- *UID* – User identifier
  Číslo, jednoznačně identifikující uživatele
  `http://en.wikipedia.org/wiki/Pid`

Dodatek C

Ukázky souborů ze souborového systému procfs

C.1 /proc/net/dev

```
Inter-|   Receive
 face |bytes    packets errs drop fifo frame compressed multicast
    lo: 2773120    2398    0    0    0     0          0         0
  eth0:       0       0    0    0    0     0          0         0
  eth1:29645656   10725    0    0    0     0          0         0
  tun0:   28158     197    0    0    0     0          0         0

|  Transmit
|bytes    packets errs drop fifo colls carrier compressed
  2773120    2398    0    0    0     0       0          0
        0       0    0    0    0     0       0          0
  3400508   11660    0    0    0     0       1          0
    14224     194    0    0    0     0       0          0
```

C.2 /proc/net/tcp

```
Inter-|   Receive
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when
   0: 0100007F:1F40 00000000:0000 0A 00000000:00000000 00:00000000
   1: 00000000:0FA1 00000000:0000 0A 00000000:00000000 00:00000000
   2: 00000000:008F 00000000:0000 0A 00000000:00000000 00:00000000

      |  Transmit
      retrnsmt   uid  timeout inode
      00000000     0        0 9290 1 d03dfa40 3000 0 0 2 -1
      00000000     0        0 8750 1 d3f565c0 3000 0 0 2 -1
      00000000     0        0 10345 1 d03df040 3000 0 0 2 -1
```

C.3 /proc/net/netstat

```
TcpExt: SyncookiesSent SyncookiesRecv SyncookiesFailed EmbryonicRsts
TcpExt: 0 0 0 0 178 0 0 0 0 0 716 6 0 0 0 79 61576 5 10087 0 0 12671
```

C.4 Formát souboru popisujícího socket v /proc/<pid>/fd

```
lrwx------ 1 jakub jakub 64 May  9 16:58 /proc/7627/fd/5 -> socket:[890126]
```