VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Ergonimizace uživatelského rozhraní virtuální laboratoře počítačových sítí

Ergonimization of virtual network laboratory user interface

2009

Jan Rudovský

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2009 .....

Rád bych poděkoval všem, kteří mi pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Diplomová práce se zabývá rozšířením stávajícího webového rozhraní virtuální laboratoře počítačových sítí. Součástí práce je upravená aplikace Dia sloužící k vytváření logických topologií a zejména popis jakým způsobem aplikaci Dia rozšiřovat.

Klíčová slova: Virtlab, Dia, virtuální laboratoř

Abstract

This thesis contains information about extension of virtual laboratory. This document include modified Dia application. Dia appliaction is used for drawing logical topologies.

Keywords: Virtlab, Dia, virtual laboratory

Seznam použitých zkratek a symbolů

WYSIWYG	–	What you see is what you get
PHP	–	Hypertext Preprocessor
XML	–	eXtensible Markup Language

Seznam tabulek

1 Tabulka podporovaných prvků z VirtlabDia 16

Seznam obrázků

1 Základní architektura Virtlabu (převzato z Wikipedie Virtlabu [7]) 7
2 Fungování právě aktivní úlohy (převzato z Wikipedie Virtlabu [7]) 8
3 Současné webové rozhraní aktivní úlohy 10
4 Současný obrázek topologie úlohy 12
5 Tvorba úlohy a rezervace 13
6 Nakreslení úlohy ve VirtlabDia 15
7 Vygenerovaný obrázek z Dia XML při rezervaci úlohy 18
8 Požadované webové rozhraní aktivní úlohy 19
9 Třídní diagram aplikace Dia převzatý ze stránek aplikace Dia [2] 21
10 Zjednodušený digram správy objektů v Dia 22
11 Dialogové okno s Virtlab vlastnostmi pro objekt 31
12 Dialogové okno s Virtlab vlastnostmi pro spoje 32
13 Zjištění názvu prvku ve VirtlabDia 37
14 Zjištění názvu prvku ve VirtlabDia 38
15 Vygenerovaný graf zátěže 44

Seznam výpisů zdrojového kódu

1 Virtlab XML pro obrázek 6 . . . 17
2 Ukázka Dia Shape . . . 23
3 Makefile.am pro adresář Virtlab . . . 24
4 Ukázka Dia Sheet . . . 25
5 Virtlab Sheet . . . 26
6 Základní kostra knihovny . . . 27
7 Základní struktura prgramového Shape . . . 28
8 dopsat . . . 28
9 Rozšiřující funkce objektu . . . 28
10 Základní kostra objektu . . . 29
11 Jednoduchá tvorba parametrů objektu . . . 29
12 Ukázka asociativních polí propojujících Virtlab a VirtlabDia prvky . . . 34
13 Ukázka asociativních polí propojujících Virtlab a VirtlabDia prvky pro generování clicable mapy . . . 35
14 Nový záznam v poli s atributy prvku . . . 36
15 Provázání atributu Prvku a proměnné objektu . . . 36
16 Přidání nové stránky do Virtlabu . . . 47
17 Vytvoření grafu . . . 49
18 Vytvoření emailu za pomoci PHPMailer . . . 49
19 Vložení FCKeditoru do stránky . . . 50
20 Vytvoření archívu za pomoci Archive/Tar.php . . . 50
21 Kompilace a instalace VirtlabDia . . . 54
22 Šablona DTD Dia XML . . . 55

Obsah

1 Úvod **6**

2 Virtuální laboratoř počítačových sítí (Virtlab) **7**

3 Zjednodušení tvorby úlohy a zpřehlednění práce s aktivní úlohou **10**
3.1 Současný stav tvorby úlohy a rezervace 10
3.2 Požadovaný průběh tvorby úlohy 11
3.3 Zjednodušení tvorby úlohy a zpřehlednění práce s aktivní úlohou 12

4 Úprava aplikace Dia **20**
4.1 Jazykové mutace Dia 20
4.2 Struktura aplikace 21
4.3 Správa prvků 22
4.4 Přidání nového Shape do aplikace Dia 23
4.5 Vytvoření nového Sheet 25
4.6 Vytvoření programovaného Shape 27
4.7 Kompilace VirtlabDia pro OS Windows 32
4.8 Kompilace aplikace VirtlabDia pro OS Linux 32

5 Rozšíření webové aplikace Virtlabu pro práci s Dia XML **33**
5.1 Dia XML a jeho struktura 33
5.2 Virtlab XML a jeho struktura 33
5.3 Realizace transformace Dia XML na Virtlab XML 33
5.4 Generování obrázku z Dia XML a nahrazení konstant dle namapovaných fyzických prvků 34
5.5 Generování clickable mapy pro obrázek topologie 35

6 Analogické postupy pro práci s VirtlabDia **36**
6.1 Přidání nového atributu k prvku 36
6.2 Přidání podpory nového VirtlabDia prvku do Virtlabu 37
6.3 Vytvoření nové logické topologie ve VirtlabDia 37
6.4 Vytvoření logické topologie ve VirtlabDia pro již existující úlohu 39

7 Další úpravy webového rozhraní Virtlabu **40**
7.1 Graf zátěže 40
7.2 Export úloh do archívu 41
7.3 Odesílání zpráv 41
7.4 Editor denních zpráv a úvodní stránky 41

8 Analýza dalších úprav webového rozhraní Virtlabu **43**
8.1 Graf zátěže 43
8.2 Export úloh do archívu 44
8.3 Systém pro odesílání zpráv 45

8.4 Editor denních zpráv a úvodní stránky 45

9 Realizace úprav webového rozhraní Virtlabu **47**
9.1 Přidání nové stránky do Virtlabu 47
9.2 Graf zátěže 48
9.3 Systém pro odesílání zpráv 49
9.4 Editor denních zpráv a úvodní stránky 50
9.5 Export úloh do archívu 50

10 Závěr **51**

11 Reference **52**

Přílohy **53**

A Kompilace aplikace Dia **54**
A.1 Potřebné balíčky pro kompilaci a samotná kompilace VirtlabDia 54

B Šablony a schémata XML **55**
B.1 Šablona DTD Dia XML 55

1 Úvod

Pro údržbu počítačových sítí je potřeba odborníků, kteří musí mít dostatečnou odbornou kvalifikaci. Proto je nutné, aby měli k dispozici nástroj, který jím umožní testování, učení se a zkoušení konfigurací počítačových sítí. Tento nástroj musí být ale dostatečně reálný, aby rozdíl mezi reálnou konfigurací a testovací byl co nejmenší. Jedním z takových nástrojů je "Virtuální laboratoř počítačových sítí" (Virtlab).

Virtlab je založen na úlohách, které se v něm řeší. Úlohy obsahují popis logické topologie v XML, které určuje propojení jednotlivých fyzických prvků a jejich vlastnosti. K úlohám se váží další informace, obrázek topologie, zadání úlohy, počáteční konfigurace a ukázková cílová konfigurace.

Obrázek topologie je v současné době grafické znázornění logické topologie, ale neobsahuje informace o vlastnostech prvků a linek.

Cílem této práce je zpříjemnění používání Virtlabu a to tak, aby uživatelé Virtlabu strávili co nejmenší čas nad přípravou a mohli se soustředit zejména na řešení problémů počítačových sítí. Jde hlavně o změny v přípravě úlohy, kdy se pokusíme přenést popis topologie se všemi potřebnými informaci do obrázku topologie.

Dalším zpříjemněním je zpřehlednění uživatelského rozhraní webové aplikace Virtlabu, kdy se pokusíme realizovat úpravy, na základě podnětů uživatelů Virtlaby.

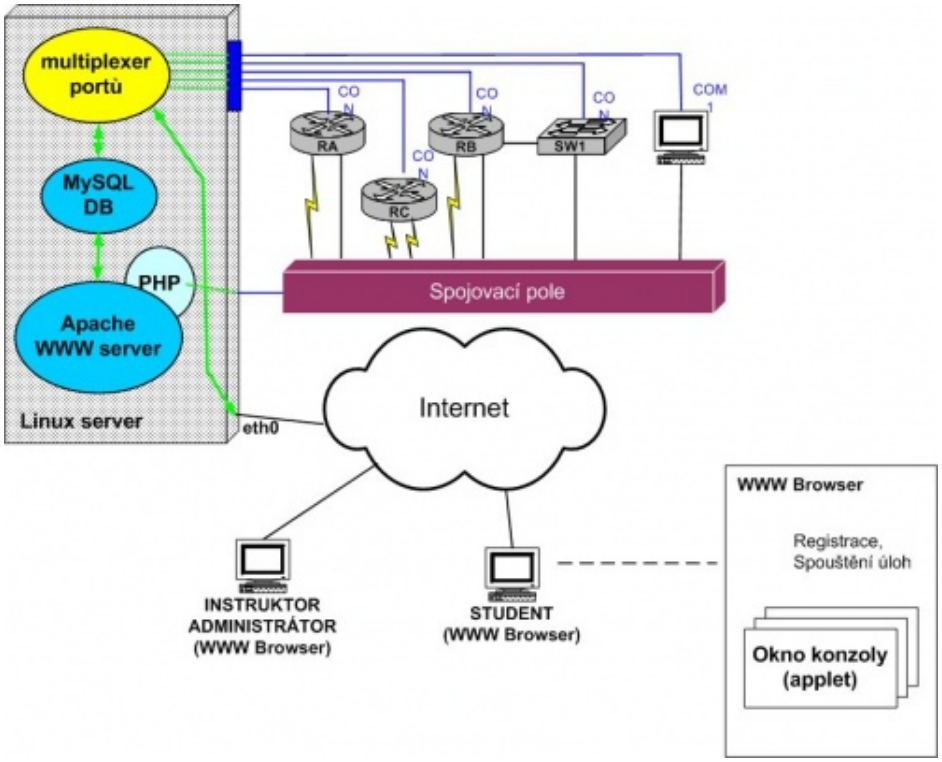
Nedílnou součastí práce bude popis řešení a technologických postupů zejména v oblasti tvorby úlohy.

2 Virtuální laboratoř počítačových sítí (Virtlab)

Smyslem projektu Virtlab je zpřístupnit laboratorní prvky pro praktickou výuku počítačových sítí vzdáleně prostřednictvím Internetu. Studenti si mohou pomocí WWW rozhraní rezervovat laboratorní prvky na určitý časový interval a následně k nim přistupovat pomocí běžného WWW prohlížeče s podporou Java appletů. Propojení laboratorních prvků se uskuteční automaticky podle výběru konkrétní úlohy ze souboru nabízených laboratorních úloh, nebo si student může zadat svou vlastní topologii.

Systém nyní dovoluje spolupráci více lokalit vzájemně sdílejících síťové prvky a realizaci virtuálních síťových topologií přes Internet. Fyzické síťové prvky nutné pro vytvoření studentem vybrané topologie jsou v době rezervace vyhledávány dynamicky ve všech lokalitách.

Citováno z Wikipedie Virtlabu [7] , kde naleznete také více informací o architektuře Virtlabu.

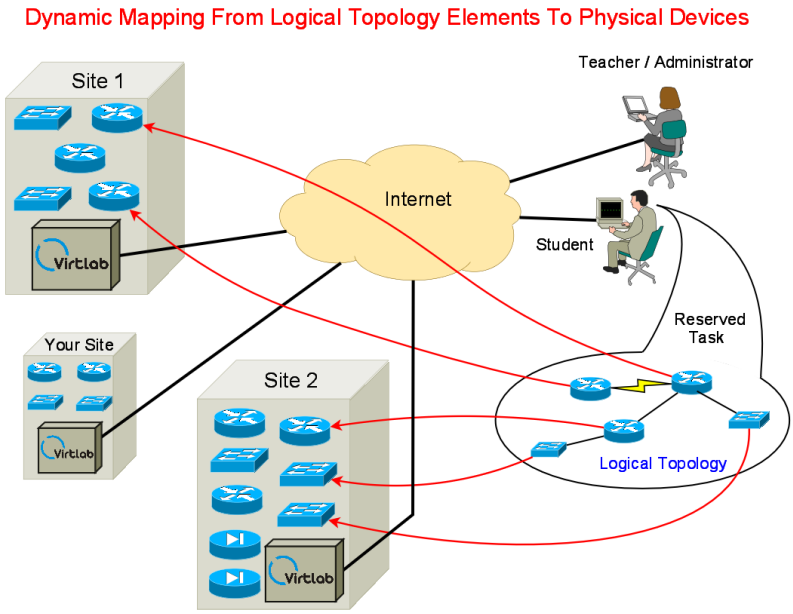


Obrázek 1: Základní architektura Virtlabu (převzato z Wikipedie Virtlabu [7])

Jednou z hlavních přístupových bran do systému Virtlab je internetový prohlížeč a webové rozhraní Virtlabu. Aplikace slouží jako informační systém pro uživatele Virtlabu a správci Virtlabu zde naleznou řadu podpůrných nástrojů, které jim usnadňují tvorbu úloh, správu samotného webového rozhraní Virtlabu nebo správu uživatelů. Jak je patrno, každý uživatel má jiná práva, existují tři hlavní skupiny administrátor (správce), tutor a běžný uživatel.

Hlavním úkolem webového rozhraní je práce s úlohami, které zde vytváříme, spravujeme, rezervujeme a hlavně realizujeme. Rezervací se myslí zarezervování fyzických prvků Virtlabu na konkrétní časový interval. V tomto čase pak lze konfigurovat samotné fyzické prvky a realizovat tak zadání úlohy. Při realizaci lze přistupovat přímo na konzole fyzických zařízení, která jsou v topologii použita, přes Java applet a zde zadávat příkazy a nastavovat tak například router.

Na obrázku 2 můžeme vidět jak funguje přiřazení fyzických prvků k logické topologii úlohy, při její rezervaci.



Obrázek 2: Fungování právě aktivní úlohy (převzato z Wikipedie Virtlabu [7])

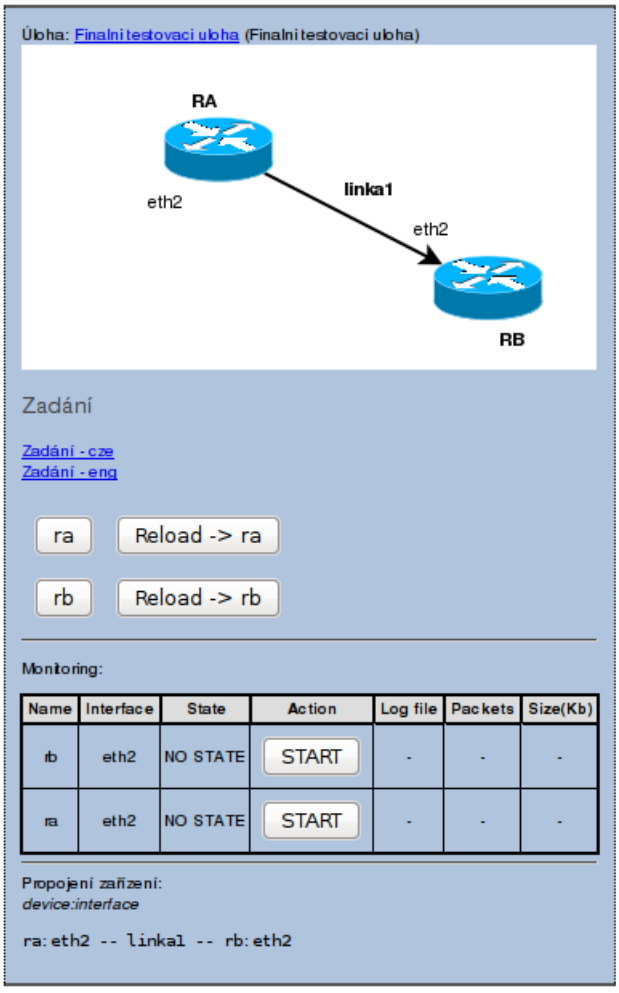
Na základě podnětů uživatelů z používání Virtlabu a zejména jeho webového rozhraní, se vybralo několik úprav a rozšíření, které práci s ním usnadní nebo zlepší. Tyto úpravy se dají rozdělit na dvě části.

Na skupinu úprav, která zjednoduší vytváření úloh a na úpravy samotného webového rozhraní (graf zátěže, export úloh do archívu, systém pro odesílání zpráv, editor denních zpráv a úvodní stránky).

3 Zjednodušení tvorby úlohy a zpřehlednění práce s aktivní úlohou

Jak jsme již zmínili hlavním úkolem webového rozhraní je práce s úlohou. Proto musí její vytváření, rezervace a následná aktivace být co nejsnazší a nejintuitivnější. Z toho plyne, že i část webového rozhraní s aktivní rezervací musí být pro řešitele přehledná, aby se zbytečně nezatěžoval hledáním potřebných informací k pochopení problému úlohy.

3.1 Současný stav tvorby úlohy a rezervace



Obrázek 3: Současné webové rozhraní aktivní úlohy

Před vytvořením úlohy si její tvůrce musí připravit několik souborů, které dohromady tvoří úlohu s plnohodnotnými informacemi. Jediným povinným souborem úlohy, je logická topologie ve formátu XML (dále Virtlab XML). Podle něj pak mapovací algoritmus hledá vhodná zařízení a přiřazuje je k rezervaci úlohy. Virtlab XML musí být validní podle Relax schématu. Více informací o požadavcích na tvorbu úlohy nalezneme na stránkách Wikipedie Virtlabu [11].

Další soubory patřící k úloze jsou přidávány kvůli přehlednosti pro řešitele, kterým dodané soubory pomohou pochopit její problematiku. Jsou to soubory obrázku topologie a zadání úlohy v různých jazykových mutacích. Zadání je textový popis obsahující informace o tom, čeho přesně uživatel má docílit.

Poslední soubory úlohy jsou konfigurační, jde o soubory s předkonfigurací a ukázková cílová konfigurace.

Pokud úloha obsahuje všechny nebo většinu těchto souborů, pak by měla podat uživateli, který si úlohu zarezervuje, velmi podrobné informace.

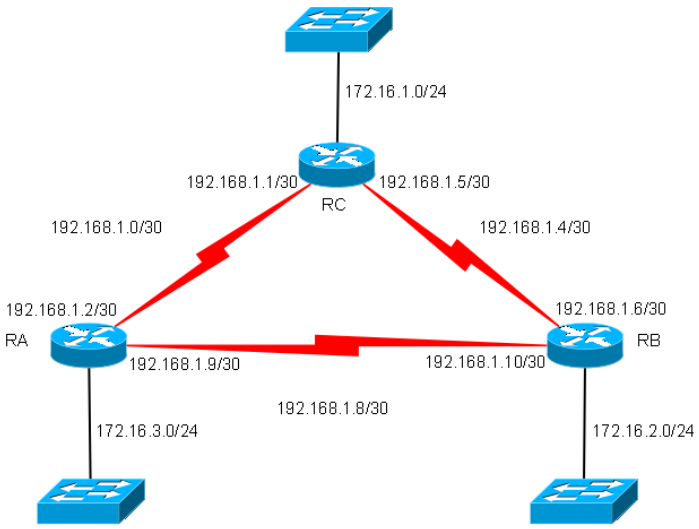
Podstránka s aktivní úlohou je nyní velice rozsáhlá. Obsahuje obrázek topologie, pod ním odkazy na zadání v jednotlivých jazykových mutacích a pak tabulku tlačítek, pomocí kterých se dostaneme k jednotlivým konzolím zařízení použitých v úloze.

Pod touto tabulkou se nachází další tabulka, se seznamem rozhraní zařízení a tlačítka, jenž umožňují jejich sledování. Nakonec následuje výpis, ve kterém je řečeno, pomocí kterých linek jsou jednotlivé zařízení topologie propojeny. Jak vypadá současný stav podstránky, můžeme vidět na obrázku 3.

3.2 Požadovaný průběh tvorby úlohy

V současné době tedy Virtlab XML a obrázek topologie (viz obrázek 4) reprezentují stejné údaje o topologii úlohy. Virtlab XML reprezentuje topologii jako XML a naopak obrázek topologie znázorňuje graficky.

Proto je vhodné sjednotit Virtlab XML s obrázkem, tak aby obrázek mohl být použit jako šablona pro vytvoření Virtlab XML i jako obrázek samotné topologie. Takto vytvořená šablona úlohy musí nést všechny potřebné informace o její topologii, které jsou potřebné pro Virtlab.



Obrázek 4: Současný obrázek topologie úlohy

Samotný obrázek topologie bude významným ovládacím rozhraním rezervované úlohy. Tedy, aby uživatel z obrázku topologie zjistil co nejvíce informací, i těch, které jsou dynamicky generovány na základě dané rezervace. Například názvy přidělených rozhraních zařízení k dané rezervaci.

Uživatel by se měl také jednoduše přes interakci s obrázkem dostat k ovládacím prvkům zařízení. Například reload zařízení, vstup na konzoli zařízení a monitoring rozhraní zařízení. K těmto volbám se musí uživatel dostat přes kontextové menu pravého tlačítka myši nad obrázkem konkrétního zařízení.

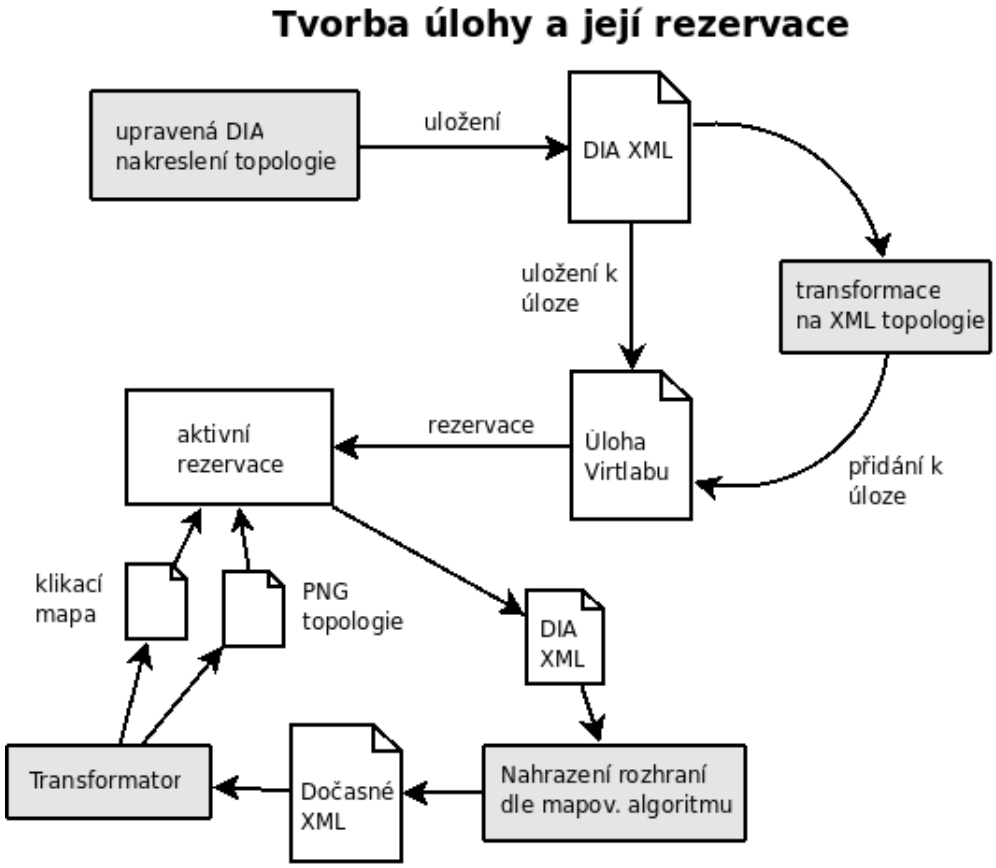
Po realizaci budou jednotlivé zařízení (router, switch, hub) na obrázku fungovat jako samotné přístupy ke skutečným konzolím a možnostem, které se nacházely doposud pod obrázkem a zadáním.

3.3 Zjednodušení tvorby úlohy a zpřehlednění práce s aktivní úlohou

Ve zjednodušení tvorby úlohy, jde zejména o přenesení tvorby do externí aplikace, ve které topologii jednoduše nakreslíme. Pak také o zpřehlednění aktivní úlohy a využití možností právě nakreslené topologie.

Realizaci tohoto komplexu úprav si rozdělíme na tři části. A to na tvorbu úlohy, převod mezi XML a úpravu XML pro rezervaci.

Na obrázku 5 vidíme jak celý proces bude fungovat. Rozšíření, která bude potřeba realizovat jsou šedě zbarvené obdélníky.



Obrázek 5: Tvorba úlohy a rezervace

První úpravou je rozšíření stávající aplikace Dia. O potřebných úpravách aplikace se dočteme více v kapitole 3.3.1. Hlavním rozšířením je možnost přidání nových vlastností k již existujícím prvkům a jejich uchování v uloženém souboru XML společně s daty o obrázku. Data musí být uložena do souboru, jenž je standartním souborem pro ukládání Obrázků v aplikaci Dia (dále již Dia XML).

Následně je potřeba vytvořit nástroj, který Dia XML překonvertuje do Virtlab XML. Této problematice se věnuje kapitola 3.3.2. Hlavní komplikací bude rozpoznání objektů v Dia XML a jejich atributů a následné vygenerování Virtlab XML podle Relax schematu.

Následně vygenerované Virlab XML a samotné Dia XML přidáme k již vytvořené úloze ve Virtlabu.

Při samotné rezervaci této úlohy zkopírujeme Dia XML do dočasného souboru a v něm vyhledáme všechny texty ve tvaru `#nazev_zarizeni:nazev_linky#`, které nahradíme skutečnými názvy rozhraní, dle mapovacího algoritmu. Transformátor použije modifikovaný dočasný soubor s Dia XML a vygeneruje z něj obrázek úlohy a kontextové menu pro jednotlivé prvky. Více v kapitole 3.3.3.

3.3.1 Tvorba úlohy v aplikaci Dia

Opensource aplikace Dia byla při jejím vzniku hlavně inspirovaná komerční aplikací Microsoft Visio. Aplikace slouží ke kreslení diagramů, UML, síťových topologií a ve směs všeho co se skládá z vrcholů a hran.

Samotná Dia je šířena pod licencí GPL, která umožňuje svobodné upravování a rozšiřování aplikace, pod podmínkou, že rozšíření na ní provedena budou také pod licencí GPL. Více o aplikaci Dia se dozvíme na jejich webových stránkách [2].

Rozšířením existující aplikace Dia, která byla doposud využívána ke kreslení obrázků topologií, získáme plnohodnotný nástroj pro tvorbu logických topologií úloh v grafickém prostředí.

Upravíme aplikaci Dia dle svých požadavků tak, aby bylo možno k prvkům zadat všechny potřebné parametry. Tyto parametry se budou k prvkům přidávat v dialogovém okně s vlastnostmi. Je tedy nutné upravit a rozšířit stávající prvky v aplikaci Dia o nové atributy.

Další výhodou je, že můžeme provedené změny v aplikaci Dia použít na další verze, které vývojáři zveřejní. Máme tedy zajištěnou aktuálnost aplikace, se všemi jejími budoucími rozšířeními a funkcemi, které přinesou.

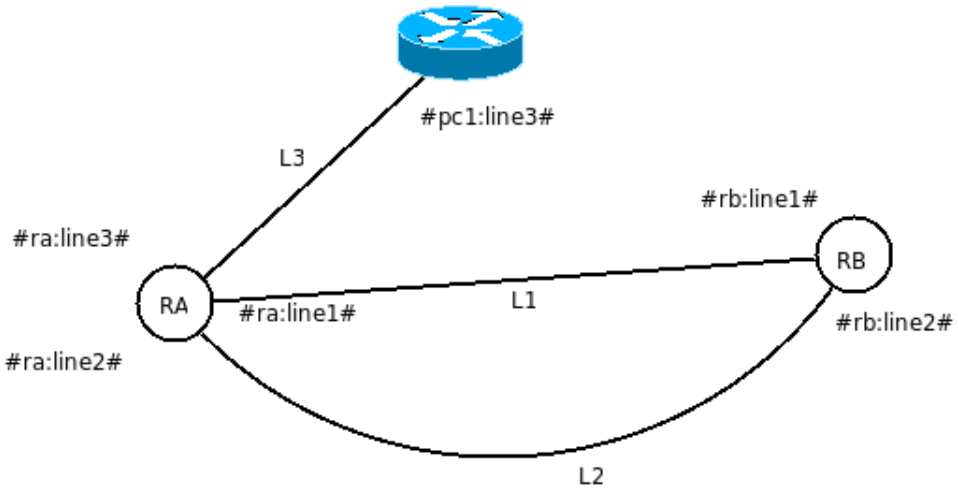
Uživatel vytvoří obrázek topologie jak byl doposud zvyklý, pouze rozšíří používané prvky o nové vlastnosti. Nebude třeba tedy nutit uživatele učit se používat nový editor nebo nástroj pro tvorbu úlohy. Navíc pokud stávající obrázky topologií byly kreslené v Dia, pak nebude problém je rozšířit, tak aby byly připraveny pro zpracování k ulehčení práce s aktivní úlohou a zároveň reprezentovali logickou topologii úloh.

Výsledná práce s aplikací bude vypadat následovně. Uživatel v námi upravené aplikaci Dia (dále jen již VirtlabDia) nakreslí topologii za pomocí prvků a spojů, doplní k ním potřebné informace.

Dále má možnost vložit specifický text do obrázku topologie, který určuje rozhraní jimž je zařízení spojeno s linkou `#nazev_zarizeni:nazev_linky#`. Tento text bude později při rezervaci této topologie nahrazen skutečným rozhraním, který mapovací server přiřadil k tomuto zařízení a lince, viz 3.3.3.

Nakonec je potřeba kolem celé topologie vytvořit obdélník s atributem "Virtlab - Ohraničení". Takto vytvořený obdélník je potom využit k rozpoznání ohraničení obrázku a jeho souřadnice využity pro generování mapy obrázků. O jeho využití se dočteme v kapitole 3.3.3.

Z VirtlabDia budou podporovány prvky:



Obrázek 6: Nakreslení úlohy ve VirtlabDia

Následně takto upravený a připravený soubor jednoduše uloží jako nekomprimovaný. Uložený dokument bude obsahovat všechny potřebné údaje pro tvorbu logické topologie ve Virtlabu a zároveň bude obsahovat i abstraktní informace o v budoucnu požitých rozhraní.

Název prvku ve VirtlabDia	Název přiřazeného prvku ve Virtlabu
Cisco - Router	router
Virtlab - Router	router
Cisco - PC	pc
Virtlab - PC	pc
Network - A Bigtower PC	pc
Cisco - Small hub	hub
Virtlab - Small hub	hub
Cisco - Workgroup switch	switch
Virtlab - Workgroup switch	switch
Cisco - PIX Firewall	firewall
Virtlab - PIX Firewall	firewall
Cisco - PIX Firewall Left	firewall
Virtlab - PIX Firewall Left	firewall
Cisco - PIX Firewall Right	firewall
Virtlab - PIX Firewall Right	firewall
Cisco - BBFW	firewall
Network - WAN Link	serial
Standard - Line	ethernet
Standard - ZigZagLine	ethernet
Standard - PolyLine	ethernet
Standard - Bezierline	ethernet
Standard - Arc	ethernet

Tabulka 1: Tabulka podporovaných prvků z VirtlabDia

Nakreslená topologie může vypadat jako obrázek 6. Kde vidíme zařízení, linky a budoucí rozhraní. Tento obrázek graficky reprezentuje logickou topologii zapsanou ve Virtlab XML 1.

```
<?xml version="1.0" encoding="UTF−8"?> <!DOCTYPE virtual_topology SYSTEM "topology.dtd">
<virtual_topology>
    <edge technology="ethernet" ether_type="legacy" name="line1">
        <vertex name="ra"/>
        <vertex name="rb"/>
    </edge>
    <edge technology="ethernet" ether_type="legacy" name="line2">
        <vertex name="ra"/>
        <vertex name="rb"/>
    </edge>
    <edge technology="ethernet" ether_type="fast" name="line3">
        <vertex name="ra"/>
        <vertex name="pc1"/>
    </edge>
    <vertex_detail type="switch" name="ra"> </vertex_detail>
    <vertex_detail type="router" name="rb"> </vertex_detail>
    <vertex_detail type="pc" name="pc1"> </vertex_detail>
</virtual_topology>
```

Výpis 1: Virtlab XML pro obrázek 6

3.3.2 Převod Dia XML na Virtlab XML

Po uložení souboru Dia XML, ve kterém je uložená topologie se všemi potřebnými vlastnostmi a povinnými atributy, pak můžeme provést převod na Virtlab XML. Tedy z Dia XML vytvořit Virtlab XML, které reprezentuje identickou topologii. Převod bude možné provést v části webového rozhraní, kde bude možné vložit Dia XML.

Vložené Dia XML převedeme pomocí skriptu na Virtlab XML úlohy a to tak, že vyhledáme v Dia XML všechny objekty, které specifikujeme. Tzn. Dia prvky budou mít svůj ekvivalentní protějšek ve Virtlab XML. Například Dia objekt `Cisco - Router` je objekt `router` ve Virtlab XML. Takto nalezené prvky zapíšeme do Virtlab XML. Dále je nutné rozpoznat vlastnosti, které jsme přidaly k Dia objektům a zapsat je korektně do Virtlab XML ke správným prvkům.

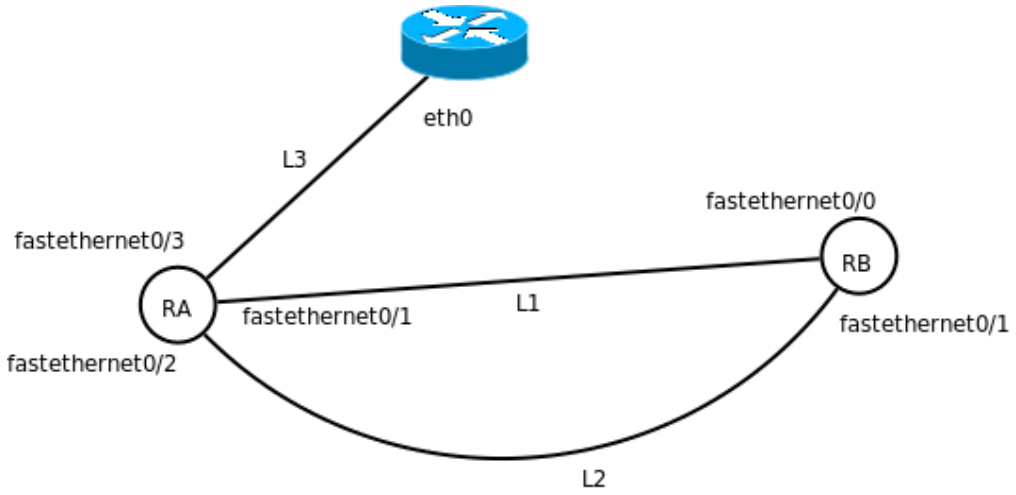
Vytvořené Virtlab XML ověříme Relax schématem, které specifikuje strukturu Virtlab XML. Při platné kontrole validity rovnou uložíme Virtlab XML do databáze. Jinak ozná-

míme chyby, které validátor nahlásí. Například chybějící povinné atributy nebo špatné pořadí atributů prvků.

3.3.3 Úprava Dia XML pro aktivní rezervaci úlohy

Při rezervaci zúročíme předchozí přípravy úlohy a nabídneme uživateli co možná nejpříjemnější práci s úlohou.

Vygenerujeme obrázek topologie z Dia XML, ve kterém budou již nahrazené prozatímní texty konkrétními rozhraními. V obrázku rozpoznáme dle Dia XML souřadnice prvků (router, switch, hub) a vytvoříme k nim jejich kontextová menu přes pravé tlačítko myši.



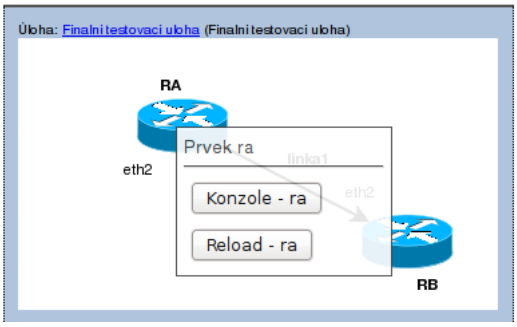
Obrázek 7: Vygenerovaný obrázek z Dia XML při rezervaci úlohy

Pro nahrazení textů je potřeba převzít rozhraní prvků, které jim přiřadil mapovací algoritmus. Najít pro ně v Dia XML text a nahradit jej přiřazeným rozhraním.

Následně vygenerujeme z Dia XML PNG obrázek, který zobrazíme na podstránce s aktivní rezervací. Vygenerovaný obrázek z topologie (obrázek 6), můžeme vidět na obrázku 7. Je patrno, že došlo k nahrazení textů, například `#pc1:line3#` za rozhraní `eth0`.

Vygenerovaný obrázek použijeme znova a to pro tvorbu kontextových menu nad prvky. Musíme nalézt souřadnice prvků v Dia XML, které jsou určeny v cm a musíme

je převést na souřadnice v px. Pro správný výpočet budeme vycházet z faktu, že díky obdélníku, který je kolem celé topologie, známe šířku a výšku obrázku v cm. Poté také šířku a výšku v px, které zjistíme ze samotné velikosti vygenerovaného obrázku.



Obrázek 8: Požadované webové rozhraní aktivní úlohy

Z takto převedených souřadnic vygenerujeme nad každým prvkem pomocí HTML elementu `div` čtverec, kterému pomocí JavaScriptu přiřadíme jednotlivá kontextová menu. Ukázku kontextového menu nad prvkem můžeme vidět na obrázku 8. Tyto kontextová menu musí být jednoduše rozšířitelná a modifikovatelná.

4 Úprava aplikace Dia

Všechny níže uvedené změny jsou aplikovány na verzi Dia 0.96.1., což je poslední stabilní verze aplikace. Změny budeme provádět v adresáři se zdrojovými kódy. Všechny zmínky na adresáře v této kapitole jsou jeho podadresáři.

Aplikace je vyvíjena zejména pod operačním systémem Linux a poté dodatečně kompilována pro Windows. Proto se soustřeďme na stejný postup.

Kompilace pro Windows není bohužel, tak jednoduchá, více o této problematice nalezneme v kapitole 4.7. Postup kompilace VirtlabDia pro Linux nalezneme v kapitole 4.8.

Zdrojové kódy Dia však nejsou nejlépe zdokumentovány. Proto se v následujících kapitolách jemně zaměříme na strukturu aplikace jako celku. Konkrétně se budeme věnovat správě prvků a jejich rozšíření, přidání a editaci.

4.1 Jazykové mutace Dia

Dia podporuje mnoho jazykových mutací a je potřeba na to brát zřetel při editaci jejího zdrojového kódu.

Je potřeba pamatovat, že všechny zdrojové soubory Dia musíme editovat pouze v kódování 7-bitovém ASCII. To znamená možnost používání pouze základních 127 znaků. Hlavním omezením je tedy nepoužívat diakritiku. Pokud chceme použít výraz s diakritikou je potřeba vytvořit pro něj konstantu, pro kterou vytvoříme překlad v přislušném souboru jazykové mutace.

Pokud vytvoříme konstantu a nevytvoříme k ní překlady, bude použit název konstanty. Znamená to tedy, chceme-li ve zdrojovém kódu vytvořit text, který se poté zobrazí v nějakém dialogovém okně a má být přeložen dle lokalizace systému, musíme doplnit překlad konstant do všech jazykových mutací.

Všechny překlady konstant nalezneme v adresáři `po/`, kde je pro každou jazykovou mutaci vytvořen jeden soubor s příponou `po`. Například pro českou jazykovou mutaci jde o soubor `cs.po`.

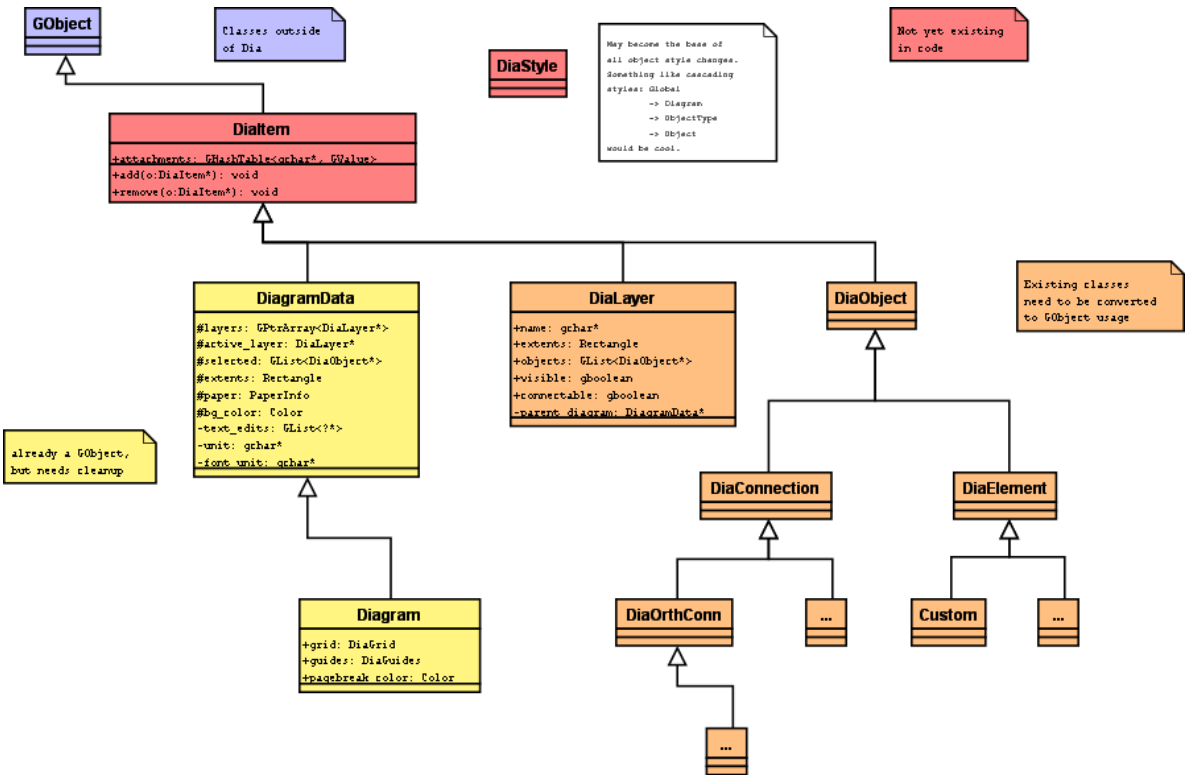
Například chceme použít text `Žlutý kůň`, tak si vytvoříme konstantu `_("Yellow horse")`. Do souboru `cs.po` doplníme řádek `msgid "Yellow horse"` a pod něj `msgstr "Žlutý kůň"`.

Podobným způsobem se překládají i XML soubory, které reprezentují například prvky nebo palety v aplikaci Dia. Nepoužívá se ovšem zápis `_("nazev konstanty")`, ale elementy, jejíž název začíná podtržítkem `<_element>nazev konstanty</_element>`.

Před samotnou kompilací zdrojových kódů se projdou všechny soubory uvedené v souboru `po/POTFILES.in` a vyhledají se v nich konstanty zapsané ve formátu `_("nazev konstanty")` nebo `<_element>nazev konstanty</_element>`. Nalezené konstanty se pak přepíší dle pravidel v souborech s jazykovou mutací, kterou určí lokalizace systému.

4.2 Struktura aplikace

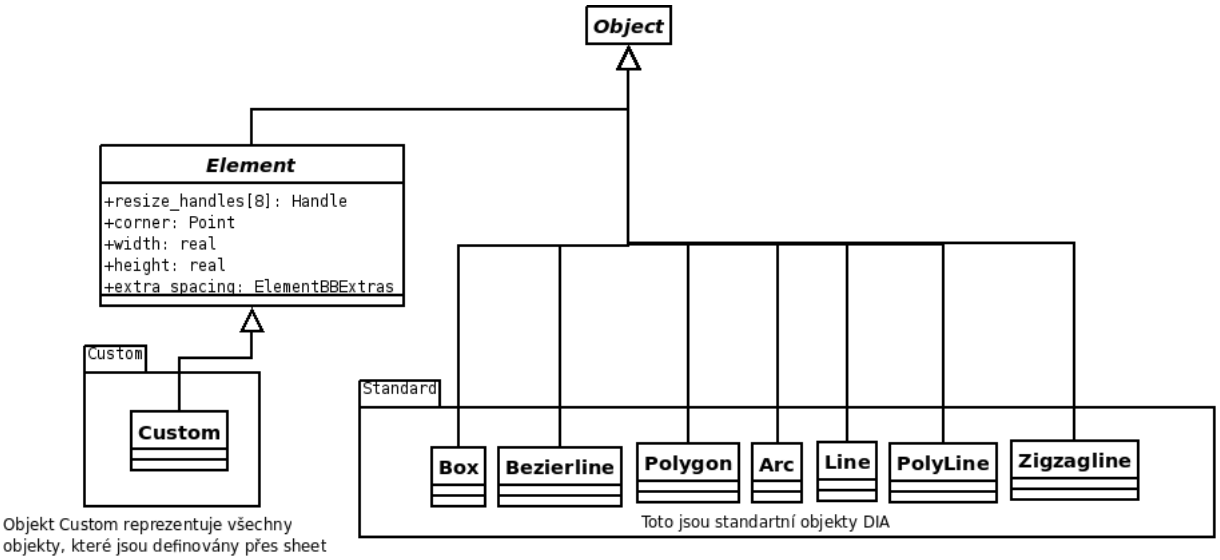
Vycházejme z třídního diagramu, který nám nabízí přímo vývojáři Dia, můžeme jej vidět na obrázku [9]. Bohužel sami vývojáři připouštějí, že není plně aktuální.



Obrázek 9: Třídní diagram aplikace Dia převzatý ze stránek aplikace Dia [2]

V třídním diagramu můžeme vidět, že hlavním objektem je GObject. Jelikož aplikace Dia je postavena nad GTK+ 2.0 a využívá jej k vytváření GUI vše je pod GObjectem.

Objekt DiagramData sdružuje obecné informace o samotném obrázku. Například okraje, formát papíru a nastavení mřížky. Z DigramData dědí vlastnosti objekty vrstev DiaLayer. Objekty DiaLayer obsahují informace o vrstvách použitých v obrázku, hlavně viditelnost a název. A obsahují všechny objekty prvků obrázku, DiaObject. Tento objekt je pro naší práci nejdůležitějším objektem. Budeme se mu věnovat samostatně v následující kapitole.



Obrázek 10: Zjednodušený digram správy objektů v Dia

4.3 Správa prvků

Správu prvků jsme zachytili třídním diagramem 10. Prvky mohou být tvořeny dvojím způsobem.

- Jako prvek s vlastnostmi objektu Custom, takto vytvořenému prvku říkáme Shape. Tato možnost je nejjednodušší, ovšem objekty vytvořené jako Shape mají omezené vlastnosti dle standardního objektu Custom.

- Další možností je vytvoření objektu dědícího z obecného objektu Object, programovaný Shape. Tato možnost je složitější avšak nabízí více možností, jak ovlivnit prvek, který takto vytváříme. Podobně jsou vytvořeny standardní objekty pro kreslení (Line, Zigzagline, Box ..), které reprezentují stejnojmenné prvky.

4.4 Přidání nového Shape do aplikace Dia

Shape umožňuje vytvoření vlastního prvku do aplikace Dia bez potřeby psaní kódu v C. Namísto toho vytvoříme soubor, do kterého napíšeme jednoduché XML popisující prvek. Prvek je popsán pomocí SVG specifikace a dalších dodatečných atributů.

Ukázka jednoduchého XML reprezentující objekt `Circuit - NPN Transistor`, viz kód 2.

```
<?xml version="1.0"?>
<shape xmlns="http://www.daa.com.au/~james/dia-shape-ns"
       xmlns:svg="http://www.w3.org/2000/svg">
  <name>Circuit - NPN Transistor</name>
  <icon>npn.xpm</icon>
  <connections>
    <point x="0" y="0"/>
    <point x="6" y="-4"/>
    <point x="6" y="4"/>
  </connections>
  <aspectratio type="fixed"/>
  <svg:svg width="3.0" height="3.0">
    <svg:line x1="0" y1="0" x2="3" y2="0" />
    <svg:line x1="3" y1="-3" x2="3" y2="3" />
    <svg:line x1="3" y1="-2" x2="6" y2="-4" />
    <svg:line x1="3" y1="2" x2="6" y2="4" />
    <svg:polyline points="5,4 6,4 5.6154,3.0769" />
  </svg:svg>
</shape>
```

Výpis 2: Ukázka Dia Shape

Pouze elementy `name` a `svg` jsou povinné. Element `name` je jednoznačným a tedy nutně unikátním identifikátorem prvku. Pomocí něj budeme s prvkem dále pracovat.

Element `icon` obsahuje cestu k XPM nebo PNG souboru, který má být použit pro grafické znázornění prvku v dialogovém okně s prvky v aplikaci Dia. Cesta k souboru je relativní k umístění Shape souboru. Pokud není element zadán je použita základní ikona prvku. Více o souboru XPM nalezneme ve Wikipedii [17].

V elementu `svg` se popisuje prvek pomocí standardu SVG, více o specifikaci SVG [12]. Podporovány jsou line, polyline, polygon, rect, circle, ellipse, path a g elements. Path element podporuje pouze = M,m,L,l,H,h,V,v,C,c,S,s,Z a z přikazy. Transformace a CSS jednotky nejsou podporovány (pouze "uživatelské") a je podporovaný pouze omezený počet CSS atributů.

Přidání Shape lze vykonat pomocí poslední verze aplikace Dia, která umožňuje tvorbu Shape bez znalosti výše zmíněných pravidel. Nakreslíme v aplikaci Dia obrázek, který posléze vyexportujeme jako Shape a jako ikonu. Poté můžeme naimportovat do již existujících Sheet, podrobnější manuál nalezneme v [13].

4.4.1 Realizace vytvoření Shape pro Virtlab

Pro potřeby Virtlabu jsme vybrali několik již existujících Shape, které chceme používat pro kreslení logických topologií.

Vytvoříme adresář `shapes/Virtlab/`, kam budeme umisťovat vybrané Shape. Pro kreslení topologií jsme používali již existující prvky, proto stačí, když překopírujeme jejich Shape a ikony do námi nově vytvořeného adresáře. Prvky v souborech následně zeditujeme tak, že změníme jejich unikátní název. Další úpravy nejsou vzhledem k potřebám Virtlabu nutné.

V adresáři `shapes/Virtlab/` vytvoříme soubor `shapes/Virtlab/makefile.am`, do kterého zapíšeme názvy nových souborů se Shape, které se nachází v adresáři, viz kód 3.

Dále rozšíříme proměnou `SUBDIRS` v souboru `shapes/makefile.am` o název námi vytvořeného adresáře.

Doplníme cestu k makefile do souboru `/configure.in`, který se nachází v hlavním adresáři se zdrojovými kódy, do proměnné `AC_OUTPUT`, ve které se nachází cesty ke všem makefile souborům.

```
shapedir = $(pkgdatadir)/shapes

 virtlabdir   = $(shapedir)/ Virtlab
SHAPES = \
  router.shape\
  router.png\
  switch.shape\
```

```
    switch.png\
    hub.shape\
    hub.png\
    pc.shape\
    pc.png\
     pix_firewall .png\
     pix_firewall .shape\
      pix_firewall_left  .png\
      pix_firewall_left  .shape

EXTRA_DIST = $(SHAPES)

virtlab_DATA = $(SHAPES)
```

Výpis 3: Makefile.am pro adresář Virtlab

4.5 Vytvoření nového Sheet

Sheet slouží ke sdružování prvků vytvořených jako Shape nebo jako programovaný Shape. Jednoduše řečeno, je to pouze schránka, ve které jsou uchovány prvky. Jde vlastně o paletu s prvky. Sheet nemá žádné vlastnosti a žádně vlastnosti nepředává prvkům k němu přiřazených.

Sheet je reprezentován souborem XML, jenž obsahuje elementy `name`, `description` a `contents`. Element `name` s atributem `xml:lang` určuje název Sheet a musí být jedinečný, tedy takový název nesmí být použit u žádného jiného Sheet. Element `description` s atributem `xml:lang` obsahuje popis tohoto Sheet.

Element `contents` obsahuje další elementy `object`, které mají atribut `name` jenž musí být shodný s názvem použitým v Shape objektu. Element `object` obsahuje elementy `description` s atributem `xml:lang`, který obsahuje popis objektu. Ukázka XML pro Sheet viz. kód 4.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<sheet xmlns="http://www.lysator.liu.se/~alla/dia/dia-sheet-ns">
    <name>Plug-in Sheet</name>
    <name xml:lang="cs">Ukazkovy Sheet</name>
    <description xml:lang="cs">Popis ukazkoveho Sheetu pro cesky jazyk</description>
    <contents>
        <object name="Plug-in_Objekt">
            <description>Obecny popis/nazev pro Shape</description>
            <description xml:lang="en_GB">Shape</description>
        </object>
    </contents>
```

```
</sheet>
```

Výpis 4: Ukázka Dia Sheet

Pravidla pro použití správného názvu Sheet a správných jazykových mutací jsou následující. Aplikace Dia rozpozná jazykové prostředí, ve kterém je spuštěna. Dle něj vyhledává příslušné názvy a popisy k Sheet a Shape. Pokud nenalezne příslušný překlad, použije základní, tzn. obsah elementu, ve kterém není uveden atribut `xml:lang`.

4.5.1 Realizace vytvoření Sheet pro Virtlab

Aby tvůrci úloh nemuseli hledat ve všech paletách prvky podporované Virtlabem, umístíme všechny podporované do společné palety s názvem Virtlab.

Pro vytvoření nového Sheet vytvoříme soubor `Virtlab.sheet.in` v adresáři `sheets/` a doplníme jeho název do proměnné `sheet_in_files` v souboru `sheets/makefile.am`. Soubor `Virtlab.sheet.in` slouží jako šablona, která je později před kompilací dotvořena dle pravidel pro jazykovou mutaci a přejmenována na `Virtlab.sheet`.

```
<?xml version="1.0" encoding="UTF-8"?>
<sheet xmlns="http://www.lysator.liu.se/~alla/dia/dia-sheet-ns">
  <_name>Virtlab</_name>
  <_description>Router and switch shapes by Cisco</_description>
  <contents>
    <object name="Virtlab - Router">
        <_description>Router</_description>
    </object>
    <object name="Virtlab - Workgroup switch">
        <_description>Switch</_description>
    </object>Virtlab - PC
    <object name="Virtlab - Small hub">
   <_description>Hub</_description>
    </object>
    <object name="Virtlab - PC">
   <_description>PC</_description>
    </object>
    <object name="Virtlab - PIX Firewall">
        <_description>PIX Firewall</_description>
    </object>
    <object name="Virtlab - PIX Firewall Left">
        <_description>PIX Firewall Left</_description>
    </object>
  </contents>
</sheet>
```

Výpis 5: Virtlab Sheet

4.6 Vytvoření programovaného Shape

Programové Shape reprezentované objekty je vhodné sdružovat pod jednou centrální knihovnou. Není to podmínkou, ale pro přehlednost kódu doporučené. Takovou knihovnu vytvoříme jako třídu v programovacím jazyku C, ve které zaregistrujeme objekty do ní patřící. Realizaci budeme demonstrovat na tvorbě imaginární knihovny `ImgLibrary` a objektu `my_object`

Knihovnu `ImgLibraty.c` vytvoříme tedy do adresáře `objects/ImgLibrary/`, do podadresáře `./pixmaps/` umístíme ikony objektů ve formátu XPM. Struktura knihovny je pevně daná a musí obsahovat minimálně přesně tyto funkce, viz kód 6.

```
#include <string.h>
#include <stdio.h>
#include <assert.h>

/* nutne DIA knihovny */
#include "intl.h"
#include "object.h"
#include "plug-ins.h"

/* Promena reprezentujici nas novy objekt */
extern DiaObjectType myobject_type;

DIA_PLUGIN_CHECK_INIT

PluginInitResult dia_plugin_init(PluginInfo *info)
{
  /* check for errors */
  if (!dia_plugin_info_init(info, "Library Info",
        "Popisek knihovny",
        NULL, NULL))
    return DIA_PLUGIN_INIT_ERROR;

  /* zde registrace objektu */
  object_register_type(&myobject_type);

  return DIA_PLUGIN_INIT_OK;
}
```

Výpis 6: Základní kostra knihovny

Tímto jsme vytvořili jednoduchou knihovnu, která obsahuje jeden objekt. Je potřeba vytvořit pro tento objekt vlastní třídu. Proto v adresáři s knihovnou vytvoříme nový soubor `my_object.c` reprezentující náš objekt. Objekt je definován pomocí struktury,

ve které nejdříve definujeme rodiče našeho objektu, základní strukturu našeho nového objektu vidíme v kódu 7.

```
typedef struct _MyObject MyObject;
struct _MyObject
{
   Element element;
   /* zde definujeme typ objektu, pozici, spojeni, vysku, sirku */
   /* a ostatni atributy */
};
```

Výpis 7: Základní struktura prgramového Shape

To umožňuje jakémukoliv MyObject být přetypován na Element a jelikož Element dědí z obecného objektu Object, tak náš MyObject můžeme přetypovat na Object.

Dále musíme nadefinovat vtable `ObjectTypeOps` objektu našeho nového prvku. Vtable slouží k dynamickému přiřazování metod k metodám třídy, pro kterou vtable vytváříme.

Definujeme tak některé operace, které jsou volány ostatními objekty Virtlabu v určitých situacích, jako je tvorba, nahrání, uložení objektu. Registraci základních funkcí vidíme v kódu 8.

```
ObjectTypeOps shape_type_ops =
{
/*
(CreateFunc) Vytvoreni noveho Shape
(LoadFunc) Nacteni Shape z Dia XML
(SaveFunc) Ulozeni Shape do Dia XML
*/
(CreateFunc)      shape_create,
(LoadFunc)    shape_load,
(SaveFunc)    shape_save,
};
```

Výpis 8: dopsat

Dále nadefinujeme strukturu samotného obecného objektu reprezentující náš objekt, která obsahuje název, číslo verze, pixmap a vtable našeho objektu. Název musí být jedinečným, slouží jako hlavní identifikátor našeho objektu a na něj odkazujeme ze Sheet. Objekt registrujeme ve třídě knihovny viz. kód 6.

```
DiaObjectType myobject_type =
{
  "MySheet - MyObject", /* name */
  0,                    /* version */
```

```
  (char **) my_xpm, /* pixmap */
  &shape_type_ops     /* ops */
};
```

Výpis 9: Rozšiřující funkce objektu

Také musíme nadefinovat strukturu vtable pro objekt `ObjectOps`. Vtable pro `ObjectOps` může definovat mnoho operací. Jejich seznam a popis nalezneme v kódu 10, tento seznam není úplný, obsahuje pouze ty, které by nás mohly zajímat. Více informací nalezneme na stránkách [14].

Pro plnou funkčnost objektu je potřeba naimplementovat většinu těchto metod, protože jinak se vlastně při událostech nebude dělat nic. Například pokud nenaimplementujeme metodu pro kopírování objektů, objekt nepůjde kopírovat.

```
ObjectOps shape_ops =
{
/*
(DestroyFunc) Metoda volana, kdyz je objekt odstranovan
(DrawFunc) Vykresleni objektu
(DistanceFunc) Musi vracet vzdalenost od objektu k jakemukoliv bodu
(SelectFunc) Metoda volana pri oznaceni objektu
(CopyFunc) Volana kdyz dochazi ke kopirovani objektu
(MoveFunc) Presun objektu
*/
/* Funkce shape_copy bude obstaravat kopirovani prvku */
(CopyFunc) shape_copy
};
```

Výpis 10: Základní kostra objektu

Dvě poslední potřebné pole jsou typu `PropDescriptions` a `PropOffsets`, dohromady definují typy atributů objektu. Pomocí těchto dvou polí dokážeme vytvořit například dialogové okno, které obsahuje záložky, výběry, checkboxy a textové pole.

Pole `PropDescriptions` je jednou z nejdůležitějších částí objektu. Definuje typy všech dat našeho objektu a charakterizuje layout dialogového okna. Pole `PropOffset` říká do jakých proměnných se mají data převzatá z dialogu uložit. Ukázku jednoduchého definování struktury objektu, jeho atributů můžeme vidět v kódu 11. Popis jednotlivých konstant a prvků nalezneme na [15].

Dialogové okna prvku jsou generovány analogicky na základě parametrů objektu.

```
static PropDescription entity_props[] = {
  ELEMENT_COMMON_PROPERTIES,
```

```
  { "name", PROP_TYPE_STRING, PROP_FLAG_VISIBLE,
    N_("Name:"), NULL, NULL },
  { "weak", PROP_TYPE_BOOL, PROP_FLAG_VISIBLE,
    N_("Weak:"), NULL, NULL },
  { "associative", PROP_TYPE_BOOL, PROP_FLAG_VISIBLE,
    N_("Associative:"), NULL, NULL },
  PROP_STD_LINE_WIDTH,
  PROP_STD_LINE_COLOUR,
  PROP_STD_FILL_COLOUR,
  PROP_STD_TEXT_FONT,
  PROP_STD_TEXT_HEIGHT,
  PROP_DESC_END
};

static PropOffset entity_offsets [] = {
  ELEMENT_COMMON_PROPERTIES_OFFSETS,
  { "name", PROP_TYPE_STRING, offsetof(MyObject, name) },
  { "weak", PROP_TYPE_BOOL, offsetof(MyObject, weak) },
  { "associative", PROP_TYPE_BOOL, offsetof(MyObject, associative) },
  { "line_width", PROP_TYPE_REAL, offsetof(MyObject, border_width) },
  { "line_colour", PROP_TYPE_COLOUR, offsetof(MyObject, border_color) },
  { " fill_colour ", PROP_TYPE_COLOUR, offsetof(MyObject, inner_color) },
  { "text_font", PROP_TYPE_FONT, offsetof (MyObject, font) },
  { "text_height", PROP_TYPE_REAL, offsetof(MyObject, font_height) },
  { NULL, 0, 0}
};
```

Výpis 11: Jednoduchá tvorba parametrů objektu

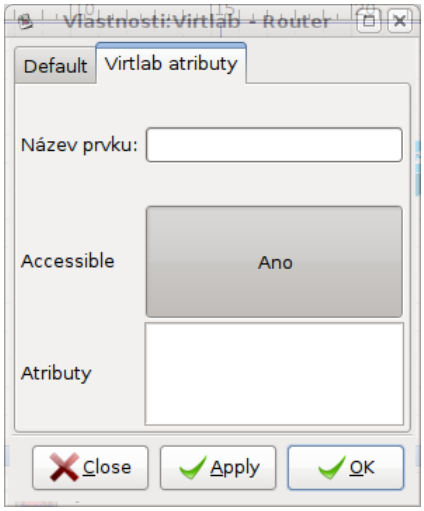
Nesmíme zapomenout vytvořit a rozšířit příslušné `makefile.am` a `configure.in` o námi vytvořené soubory aby došlo k jejich kompilaci.

4.6.1 Realizace úprav Dia objektů pro potřeby Virtlab

Pro potřeby Virtlabu nebylo potřeba vytvářet nové objekty s rozšířenou funkčností. Naopak bylo žádoucí, aby potřebné vlastnosti mohl mít jakýkoliv obecný Shape, ne programovaný Shape. Obecný Shape tvořený XML souborem dědí vlastnosti objektu Custom, který nalezneme v `objects/custom/custom_object.c`.

Je tedy potřeba rozšířit jeho funkčnost o nové vlastnosti. Musíme přidat do dialogového okna novou záložku, ve které se budou vyplňovat potřebné vlastnosti, tedy název prvku, atribut accessible a popřípadě další parametry objektu.

Výsledný dialog můžeme vidět na obrázku 11.



Obrázek 11: Dialogové okno s Virtlab vlastnostmi pro objekt

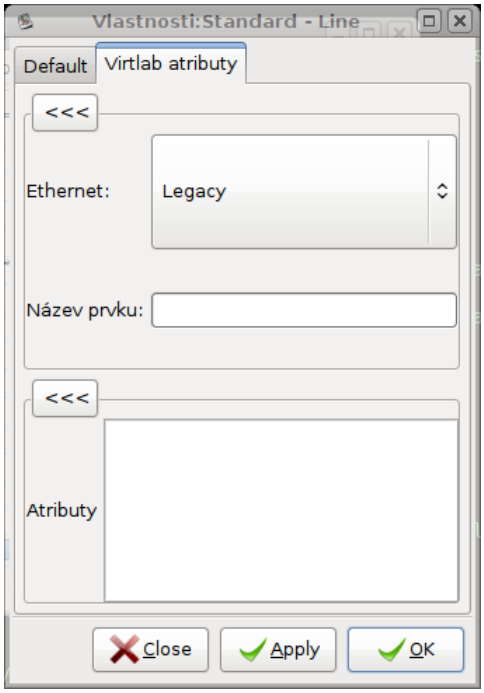
Dále musíme rozšířit objekty, které slouží k propojení prvků a reprezentují tak propojení v logické topologii. Konkrétně se jedná o objekty `zigzagline.c`, `polyline.c`, `arc.c`, `bezier.c` a `line.c`, tyto prvky nalezneme v adresáři `objects/standard/`.

Mezi nastavitelné povinné vlastnosti linky pro topologii Virtlabu jsou ethernet, název linky a další atributy (link features). Dialogové okno pro linku můžeme vidět na obrázku 12.

Dalším objektem spojující prvky, který se využívá pro potřeby Virtlabu je objekt `objects/network/wanlink.c`, jenž musí mít stejné atributy jako obecný objekt bez vlastnosti accessible. Tento objekt reprezentuje WAN linku.

Posledním prvek, který musíme rozšířit je objekt `box.c`, se nachází v adresáři se standardními objekty. Přidáme mu pouze vlastnost “Virtlab - ohraničení“, která určuje, že právě tento obdélník je ohraničením celé topologie.

Ve všech upravených objektech musíme rozšířit metody pro načítání, ukládání a kopírování prvku. Rozšířit pole `PropDescriptions`, `PropOffsets` a strukturu samotného objektu o nové proměnné reprezentující vlastnosti.



Obrázek 12: Dialogové okno s Virtlab vlastnostmi pro spoje

4.7 Kompilace VirtlabDia pro OS Windows

Pro správnou kompilaci VirtbalDia v operačním sytému Windows je potřeba mít k dispozici vývojové prostředí Visual Studio C++ 6.0 se servisním balíčkem verze 6, dále je potřeba mít nainstalován balíček all-in-one GTK+ 2.0 pro Windows.

Více o problematice kompilace VirtlabDia nalezneme na internetových stránkách, které byly 22.4.2009 aktualizovány, viz [16].

Bohužel kompilace sebou často nese nečakané a nepředvídatelné obtíže, které nelze obsáhnout v této práci.

4.8 Kompilace aplikace VirtlabDia pro OS Linux

Kompilace pro operační systém linux je podstatně bezproblémovější. V systému musí být nainstalovány potřebné balíčky, jejich výčet nalezneme v příloze A.1.

5 Rozšíření webové aplikace Virtlabu pro práci s Dia XML

Jakmile máme uložené Dia XML s logickou topologií pro Virtlab, je potřeba ho přetransformovat na Virtlab XML, které bude reprezentovat vše co znázorňuje nakreslená topologie.

5.1 Dia XML a jeho struktura

Dia XML se skládá z dvou hlavních elementů a to `digramdata` a `layer`. V elementu `diargramdata` se nachází informace o samotném diagramu. Důležitým elementem je `layer`, kde se nachází prvky a skupiny prvků. Prkem je vše, co je v diagramu zaneseno, tedy jak prvky, tak i spoje. Prvek se skládá z atributů a popřípadě z elementu `connections`, který nese informaci o tom, které prvky tento spoj propojuje.

Celou DTD šablonu můžeme vidět v příloze B.1.

5.2 Virtlab XML a jeho struktura

Virtlab XML má dva hlavní elementy `edge` a `vertex_detail`. Elementy `edge` reprezentují vnitřními elementy propojení dvou prvků a v atributech nesou informace o linkách. Informace o názvu linky, o jaký typ linky jde, přenosovou rychlost linky a další.

Elementy `vertex_detail` obsahují informace o prvcích použitých v topologiích. O jaký typ prvku jde, zda-li je dostupný, jeho operační systém, platformu a další vlastnosti.

Celé relax schéma nalezneme v [18].

5.3 Realizace transformace Dia XML na Virtlab XML

Pro transformaci vytvoříme novou podstránku `transform-xml.php`, která využije naší třídu `XMLTransform`.

Ve třídě `XMLTransform` jsou nejdůležitější dvě asociativní pole `$virtlab_objects` a `$virtlab_objects_con`.

První pole obsahuje všechny prvky z VirtlabDia, kterým chceme přiřadit prvek ve Virtlabu, jejíchž název je klíčem v poli obsahující název ekvivalentního prvku v systému

Virtlab. Pole můžeme vidět v kódu 12. Vidíme zde například, že prvek `Cisco - Router` je ve Virtlabu reprezentován jako `router`.

```
private $virtlab_objects = array(
'Cisco - Router' => 'router',
' Virtlab - Router' => 'router',
'Cisco - PC' => 'pc',
' Virtlab - PC' => 'pc',
'Cisco - Small hub' => 'hub',
' Virtlab - Small hub' => 'hub',
'Cisco - Workgroup switch' => 'switch',
' Virtlab - Workgroup switch' => 'switch',
' Virtlab - PIX Firewall' => 'firewall' ,
'Cisco - PIX Firewall' => 'firewall' ,
' Virtlab - PIX Firewall Left' => 'firewall' ,
'Cisco - PIX Firewall Left' => 'firewall' ,
'Network - A Bigtower PC' => 'pc',
'Cisco - BBFW' => 'connector'
);
private $virtlab_objects_con = array(
'Standard - Line' => 'ethernet',
'Network - WAN Link' => 'serial',
'Standard - ZigZagLine' => 'ethernet',
'Standard - PolyLine' => 'ethernet',
'Standard - Bezierline' => 'ethernet',
'Standard - Arc' => 'ethernet'
);
```

Výpis 12: Ukázka asociativních polí propojujících Virtlab a VirtlabDia prvky

Nejdůležitější metodou třídy `XMLTransform` je `create_xml()`. V této metodě se zpracovávají data a generuje se Virtlab XML. Ošetřují se zde volitelné atributy u prvků a linek, přiřazují se názvy prvkům a generuje se popis propojení mezi nimi dle Dia XML

5.4 Generování obrázku z Dia XML a nahrazení konstant dle namapovaných fyzických prvků

Ke generování obrázku topologie dochází v okamžiku, kdy je zobrazena aktivní úloha, ať již během její realizace a v čase, kdy byla zarezervována nebo před zarezervovaným časem v detailu rezervace. Obrázek se generuje pouze jednou a při ukončení rezervace je smazán.

Generování obstarává třída `VirtlabDia` metodou `generateDIAimage`, které předáme jeden povinný a jeden nepovinný atribut. Povinným atributem je identifikační číslo rezervace a nepovinným argumentem příznak, zda-li současně generovat s obráz-

kem i clickable mapu nebo negenerovat. Více o generování clickable mapy v kapitole 5.5

V metodě `generateDIAimage` nejdříve získáme data o rezervaci, přiřazených prvcích a jejich propojení z mapovacího algoritmu. Poté získáme z databáze Dia XML. Vytvoříme řetězce dle struktury `#nazev_zarizeni:nazev_linky#`, pokusíme se je vyhledat v Dia XML a nahradit skutečnými rozhraními. Nakonec takto upravené Dia XML převedeme pomocí VirtlabDia na PNG obrázek.

5.5 Generování clickable mapy pro obrázek topologie

O generování mapy se stará třída `VirtlabDia`. Celé vygenerování mapy je rozděleno na dvě fáze.

V první fázi procházíme Dia XML a vyhledáváme v něm obdélník s atributem `Virtlab - Ohraničení`. Pokud existuje ohraničující obdélník je možné vygenerovat mapu. Dále hledáme objekty, které mají svůj ekvivalent ve Virtlabu, ty se nachází v asociativním poli `$virtlab_objects`, které můžeme vidět v kódu 13.

```
private $virtlab_objects = array(
'Cisco - Router' => 'router',
' Virtlab - Router' => 'router',
'Cisco - PC' => 'pc',
' Virtlab - PC' => 'pc',
'Cisco - Small hub' => 'hub',
' Virtlab - Small hub' => 'hub',
'Cisco - Workgroup switch' => 'switch',
' Virtlab - Workgroup switch' => 'switch',
' Virtlab - PIX Firewall' => 'firewall',
'Cisco - PIX Firewall' => 'firewall',
' Virtlab - PIX Firewall Left' => 'firewall',
'Cisco - PIX Firewall Left' => 'firewall',
'Network - A Bigtower PC' => 'pc',
'Cisco - BBFW' => 'connector'
);
```

Výpis 13: Ukázka asociativních polí propojujících Virtlab a VirtlabDia prvky pro generování clicable mapy

Druhá fáze je samotné vygenerování mapy a přiřazení ji k samotnému obrázku, vygenerování kontextových menu pro každý prvek topologie.

6 Analogické postupy pro práci s VirtlabDia

6.1 Přidání nového atributu k prvku

Ukázka přidání nového atributu pro všechny obecné prvky, tedy prvky Shape. Budeme přidávat atribut `Popis`, který budeme vkládat do textového pole.

Musíme editovat soubor `objects/custom/custom_object.c` ve zdrojových kódech virtlabdia.

První si vytvoříme ve struktuře `_Custom` proměnou `char* virtlab_popis`. Poté přidáme další záznam do pole `custom_props` viz kód 14.

```
{"virtlab_popis", PROP_TYPE_MULTISTRING,PROP_FLAG_VISIBLE|
    PROP_FLAG_DONT_MERGE, N_("Popis:"), NULL, NULL}
```

Výpis 14: Nový záznam v poli s atributy prvku

Řetězec `"virtlab_popis"` musí být shodný s názvem proměnné, ke které se tento atribut váže. `PROP_TYPE_MULTISTRING` určuje o jaký typ atributu jde, v tomto případě jde o textarea. Další část určuje, zda-li je prvek viditelný a zda má být editovatelný pro skupinu, tzn. pokud je prvek použit ve skupině, jde tento atribut editovat. Více informací o různých typech atributů a jejich vlastnostech nalezneme na [15]. `N_("Popis:")` je popisek atributu, který se objeví před atributem.

Dále rozšíříme pole `custom_offsets`, ve kterém se říká, ze které proměnné se má plnit atribut v dialogovém okně, viz kód 14. Typ atributu je stejný jako byl uveden v předchozím poli. `offsetof(Custom, virtlab_atributes)` říká, ze struktury objektu `Custom` vezmi obsah proměnné `virtlab_atributes`.

```
{ "virtlab_popis", PROP_TYPE_MULTISTRING, offsetof(Custom, virtlab_atributes)}
```

Výpis 15: Provázání atributu Prvku a proměnné objektu

Tímto jsme vytvořili nový atribut pro všechny prvky tvořené Shape. Ovšem obsah atributu se nebude přenášet při kopírování prvku. Proto musíme rozšířit metodu `custom_copy`, kde provedeme přiřazení obsahu naší proměnné do proměnné nově vytvářeného prvku. Objekt `Custom` řeší načítání dat atributů z XML metodu `object_load_props`, data načítá dle nadefinovaných atributů objektů. Bohužel ne všechny objekty tuto vý-

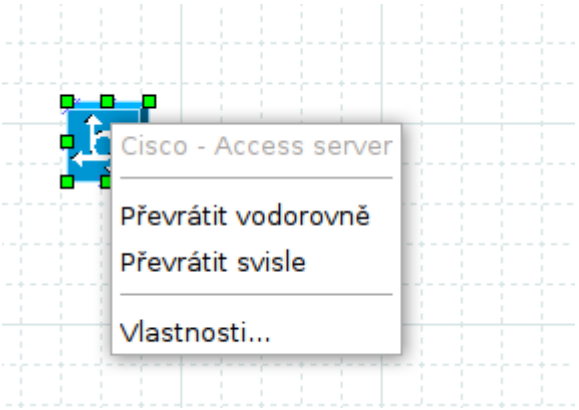
hodu vyžívají, to znamená, že pokud budeme editovat jiný než `Custom` objekt, je potřeba upravit metody, které se starají o načítání a ukládání prvku.

Poté je nutné provést kompilaci virtlabdia.

Po těchto úpravách bude nový atribut uložen v Dia XML. Musíme rozšířit webovou aplikaci Virtlab o jeho podporu. A to ve třídě pro generování Virtlab XML, viz 5.3, a její metodě `generate_xml()`.

6.2 Přidání podpory nového VirtlabDia prvku do Virtlabu

- Z VirtlabDia zjistíme název prvku. Název prvku nalezneme v jeho kontextovém menu pravého tlačítka myši (obrázek 13).



Obrázek 13: Zjištění názvu prvku ve VirtlabDia

- O tento název rozšíříme příslušná pole ve třídách `XMLTransform` a `VirtlabDia`. Podrobnější popis těchto polí nalezneme v kapitole 5.3 a 5.5.

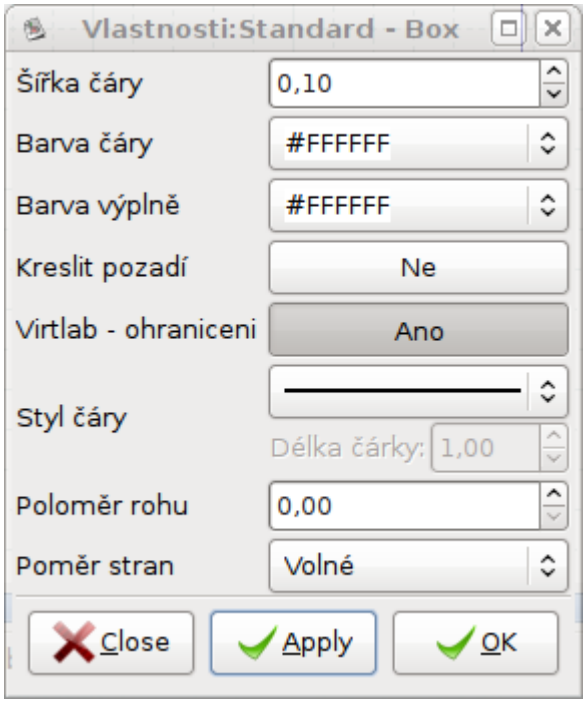
- Pokud jsme rozšířili pole v obouch třídách, získali jsme podporu nového prvku, jak při generování clickable mapy, tak i při transformaci z Dia XML na Virtlab XML.

6.3 Vytvoření nové logické topologie ve VirtlabDia

- Otevřít virtlabdia

- Umístit prvky logické topologie a pospojovat je

- K prvkům a linkám doplnit jejich názvy a vlastnosti (záložka Virtlab v dialogovém okně, které se objeví po dvojkliku na prvek nebo linku)
- Vložit prozatimní reprezentaci interface prvků ve tvaru `#nazev_zarizeni:nazev_linky#` jako textový prvek
- Kolem celé topologie vytvořit obdélník, tzn. tak, aby vše bylo uvnitř obdélníku
- Zakliknout u obdélníku vlastnost `Virtlab - ohraniceni` na hodnotu Yes (obrázek 14)



Obrázek 14: Zjištění názvu prvku ve VirtlabDia

- Uložit topologii a v dolním rohu dialogu pro ukládání odškrtnout vlastnost „Komprimovat soubory s diagramy“
- Takto uloženou topologii vložit do formuláře v sekci „Transformace Dia topologie“ (dočasně se jmenuje Transform XML)
- Uložit vygenerované XML s topologií do seznamu souborů

- Přiřadit soubor s topologií a soubor s Dia obrázkem k úloze

6.4 Vytvoření logické topologie ve VirtlabDia pro již existující úlohu

- Otevřít virtlabdia
- Vložit prvky, jenž jsou zapsány v XML topologie (router, switch, hub...)
- Podle propojení v XML logické topologie propojit prvky v obrázku pomocí linek. Pozor pro ethernet jsou používány obecné čáry pro spoj (line, zigzagline...) a pro WAN je používán objekt `WAN LINK` ze Sheet `Network`
- K linkám a prvkům doplnit názvy podle XML topologie a zároveň doplnit vlastnosti. Například volitelný atribut `os: linux`, další atributy jsou odděleny entrem
- Vložit prozatímní reprezentaci interface prvků ve tvaru `#nazev_zarizeni:nazev_linky#` jako textový prvek
- Kolem celé topologie vytvořit obdélník, tzn. tak, aby vše bylo uvnitř obdélníku
- Zakliknout u obdélníku vlastnost `Virtlab - ohraniceni` na hodnotu Yes
- Uložit topologii, v dolním rohu v dialogu pro ukládání odškrtnout vlastnost „Komprimovat soubory s diagramy“
- Uložit soubor do systému Virtlab jako dia obrázek
- Přiřadit soubor s Dia obrázkem k úloze

7 Další úpravy webového rozhraní Virtlabu

Webové rozhraní Virtlabu slouží jako hlavní prostředník mezi uživatelem a Virtlabem. V tomto prostředí pracují administrátoři, uživatelé a tutoři. Proto je potřeba provést některé úpravy, které zpříjemní nebo zlepší práci s webovou aplikací.

Mezi potřebné rozšíření patří graf zátěže, který poskytne uživatelům Virtlabu představu o obsazenosti a vytížení Virtlabu. Export úloh do archívu, který usnadní přenos úloh mezi lokalitami a také usnadní jejich archivaci. Díky systému pro odesílání zpráv budou správci Virtlabu schopni informovat uživatele o specifických odstávkách, například o odstávce konkrétního zařízení nebo o kompletní odstávce systému. Editor denních zpráv díky kterému můžou správci informovat jednotlivé uživatele Virtlabu o novinkách ve všech jazykových mutacích a podle práv. Tedy pro administrátory, tutory a běžného uživatele Virtlabu je denní zpráva odlišná. A nakonec editor úvodní stránky, který umožní pohodlnou úpravu úvodní stránky a poskytne tak lehčí udržovaní její aktuálnosti.

7.1 Graf zátěže

O využití zařízení Virtlabu a skutečné obsazenosti nemá uživatel v podstatě žádnou ucelenou představu. O tom, kolik je rezervací úloh ve Virtlabu, se dozvíme ze seznamu rezervací. Ovšem tato informace není ucelená a přehledná.

Proto pro větší přehlednost o využití a obsazení zařízení Virtlabu zobrazíme tyto informace v grafu. Graf zátěže bude zobrazovat využití jednotlivých zařízení Virlabu v časovém intervalu. Ten musí být nastavitelný uživatelem. Na základě grafu zátěže lze zjistit, zda mapovací algoritmus vhodně rozkládá zátěž mezi všechna zařízení Virtlabu a nebo jsou některá úplně nevyužita. Zároveň demonstruje a ukazuje využití samotného Virtlabu uživateli. Což pomůže k plánování odstávek a údržby Virtlabu.

Také uživatelé Virtlabu zjistí díky grafu zátěže, kdy je možné provést jejich rezervaci úlohy, která například vyžaduje specifické zařízení Virtlabu, nebo potřebuje mnoho zařízení.

7.2 Export úloh do archívu

Pokud chceme nyní přenést úlohu z Virtlabu do jiné lokality je potřeba najít všechny soubory, které s úlohou souvisí. Ty nelezeme na stránce editace úlohy. Poté musíme v seznamu souborů příslušné soubory nalézt a pak uložit, což je velmi zdlouhavé a nepohodlné.

Proto z důvodu jednoduššího přenášení úloh nejen mezi lokalitami Virtlabu je potřeba vytvořit export úloh do archívu. Export úloh musí vytvořit archív, ve kterém budou obsaženy všechny potřebné soubory úlohy (zadání, topologie, obrázek topologie, obrázek aplikace Dia, více o aplikaci Dia v kapitole 3.3.1).

Díky rozšíření exportu úloh bude správce systému Virtlab schopný na pár kliknutí vygenerovat archív obsahující všechny úlohy a jejich potřebné soubory.

7.3 Odesílání zpráv

Informace, například o odstávce Virtlabu, se uživatelům nyní podávají velmi nevhodně a to tak, že se informují všichni zaregistrovaní, bez rozdílu, zda se jich informace týká. Navíc nebylo možné poslat informaci jen uživatelům, kteří chtějí využít konkretní zařízení nebo mají rezervaci v čase, ve kterém je plánován výpadek zařízení, nebo celého Virtlabu.

Systém pro odesílání zpráv bude schopný rozesílat emaily vybraným uživatelům dle různých kriterií, kterými jsou skupina, čas rezervace, prvek v rezervaci použitý. Konkrétně tedy skupina, do které uživatel patří, časový interval od do, ve kterém má uživatel rezervaci nebo prvek v rezervacích použitý.

Rozšíření usnadní rozesílání emailů o odstávkách Virtlabu nebo jeho zařízeních. Uživatelé budou tedy včasně informováni. Navíc nebude třeba zatěžovat všechny uživatele všemi informacemi, naopak uživateli se dostanou opravdu pouze informace, které ho mohou potencionálně zajímat.

7.4 Editor denních zpráv a úvodní stránky

V současné době úprava denních zpráv a úvodní stránky je pouze textovým souborem, který se musí ručně editovat na serveru. Což znamenalo připojit se na server, edito-

vat konkrétní soubor. Soubory jsou totiž různé dle práv uživatelů dělí se tedy soubory pro administrátora, uživatele, tutora. A také podle jazyku (angličtina, čeština). Úprava úvodní stránky nebo denní zprávy je tedy velice zdlouhavá a nepohodlná.

Je tedy potřeba, aby tato data byly načítány z databáze a velice jednoduše editovatelná ve webovém rozhraní Virtlabu. Navíc u denních zpráv je potřeba rozlišit, komu je zpráva určena (administrátor, tutor, uživatel) a o jakou jazykovou mutaci jde.

Tato úprava usnadní a hlavně zpříjemní úpravu denních zprav a úvodní stránky. Tedy denní zprávy budou aktualizovány častěji a přinesou uživateli informace. Nepohodlná editace úvodní stránky nyní odrazovala k její častější editace, po tomto rozšíření bude možné častěji upravovat a udržovat tak aktuálnost úvodní stránky.

8 Analýza dalších úprav webového rozhraní Virtlabu

Pro rozšíření a úpravy současné webové aplikace Virtlabu, bude potřeba vycházet ze současného stavu. Seznámit se se stávajícím fungováním a strukturou aplikace. Postupovat tak, aby nedošlo k narušení stávající funkčnosti webové aplikace. Je tedy nutné využívat technologie, které mají potenciál a jejichž vývoj je stabilní. A to z důvodů, aby bylo jednoduché pokračovat a rozšiřovat úkoly a úpravy v této práci započaté.

Webové rozhraní běží na serveru s operačním systémem linux a distribucí Debian. Webové rozhraní je založeno na skriptovacím jazyku PHP [1] a databází je MySQL [3]. V databázi je uchovávána většina dat webové aplikace, datové soubory úlohy, informace o uživatelích, rezervacích a úlohách.

Pro většinu rozšíření použijeme PHP framework Pear [4], jenž je součástí repositářů distribuce Debian. Což nám zaručí aktuálnost frameworku. Navíc jeho zakomponování do stávající aplikace nenese žádné riziko a je bezproblémové. Stačí připojit potřebnou knihovnu s třídou do třídy, kde budeme framework Pear využívat.

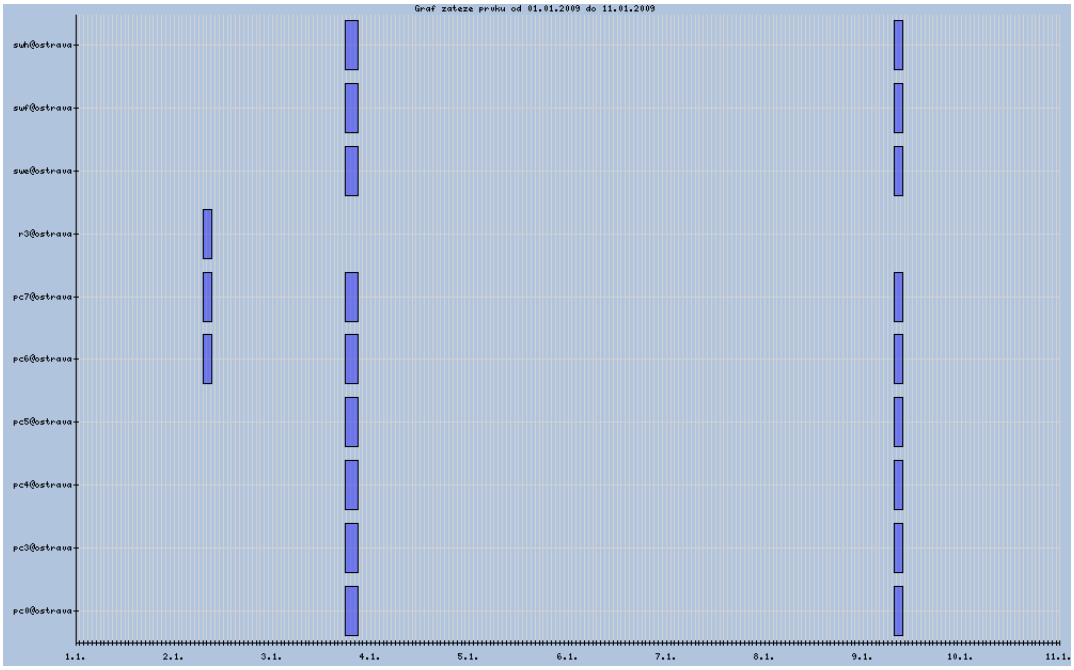
Všechny rozšíření budou mít svojí třídu, která bude obstarávat funkčnost, popřípadě budou upravovat již stávající třídy Virtlabu, aby došlo k zachování stávající struktury.

8.1 Graf zátěže

Graf zátěže musí znázorňovat využití prvků v daném časovém intervalu. Tak aby uživatel získal představu o vytíženosti samotného Vitlabu a jeho zařízení.

Pro realizaci grafu zátěže, je potřeba rozšířit stávající webové rozhraní o novou podstránku, kde se graf zobrazí. Při prvním zobrazení podstránky se vykreslí graf aktuálního měsíce. Pro graf musí být nastavitelné časové rozmezí, které má zobrazit. Toto rozmezí nesmí být větší než půl roku a musí být schopné rozpoznat prohozená data (počáteční datum je pozdější než koncový datum).

Pro generování grafu použijeme knihovnu Graph z frameworku Pear [4], která usnadňuje tvorbu různorodých grafů založených na číselných datech. Umožňuje vytvářet koláčové, sloupcové, křivkové a mnohé další grafy viz. [5]. Celý proces vytváření grafu je



Obrázek 15: Vygenerovaný graf zátěže

velice variabilní a lze tak dosáhnout téměř jakéhokoliv vzhledu grafu. Grafický výstup grafu lze uložit do mnoho grafických souborů, například do PNG, SVG a JPG.

Pro zadávání času bude použit JavaScriptový kalendář [9], který zpřehlední uživateli zadávání data.

Graf se bude generovat do adresáře `graph` pod názvem `login-lokalita.png`. Maximálně tak může vzniknout tolik souborů s grafem, kolik je uživatelů Virtlabu.

Na obrázku 15 můžeme vidět jak bude vygenerovaný graf vypadat pro časový interval od 1.1.2009 do 11.1.2009. Na ose X jsou jednotlivé dny a na ose Y zařízení v tomto termínu použitá. Světle modré obdélníky značí použití tohoto zařízení.

8.2 Export úloh do archívu

Export úloh bude prováděn na podstránce Editace úloh, kde v seznamu editovaných úloh přibude tlačítko pro export do archívu. Pro tvorbu archívu využijeme knihovnu `Archive/Tar.php` z frameworku Pear. Knihovna podporuje vytváření, procházení,

doplňování a rozbalování tar archívů. Pokud PHP obsahuje rozšíření zlib nebo bz2, pak lze pracovat i s těmito typy archívů.

Jelikož v podstránce, lze filtrovat úlohy dle skupiny do které úloha patří je vhodné, aby bylo možné vyexportovat celý takto vyfiltrovaný seznam úloh. Struktura archívu musí být přehledná, tedy v archívu musí být tolik složek kolik je exportovaných úloh a jejich název musí odpovídat krátkému názvu úlohy. Samotný název archívu bude podle názvu úlohy, nebo pokud půjde o export seznamu úloh, bude tvořen aktuálním datem exportu.

Exportovaný archív se poté vytvoří a uloží na server do adresáře `tasks_export`, bude tak k dispozici uživateli ke stažení.

8.3 Systém pro odesílání zpráv

Systém pro odesílání zpráv musí umožňovat co nejsnazší vytvoření emailu, který se následně odešle všem specifikovaným uživatelům dle zvolených filtrů.

Systém pro odesílání zpráv bude tedy nová podstránka, kde bude možno zadat konkrétní seznam emailových adres, vybrat skupinu, zvolit časový interval rezervace nebo konkrétní zařízení. Výběry jsou slučovací, ne vylučovací. Vybráním skupiny a zadáním konkrétní emailové zprávy, bude tedy email zaslán jak skupině, tak i na konkrétní email. Poté tvůrce zadá předmět zprávy a napíše samotnou zprávu.

Pro odesílání emailů použijeme knihovnu `PHPMailer` [10]. Je potřeba, aby na serveru byl nainstalován server pro odesílání emailů `sendmail`. Editor pro psaní zprávy bude FCKEditor [8], který umožňuje psát jednoduše formátované zprávy, umožňuje tvořit HTML texty jako WYSIWYG editor a nebo jako prostý text.

8.4 Editor denních zpráv a úvodní stránky

Editor denních zpráv bude také nová podstránka. Na které se bude nacházet možnost zvolit jazykovou mutaci, pro kterou chceme editovat denní zprávu. Poté se zobrazí FCKEditor pro jednotlivé práva uživatelů, tedy administrátora, tutora a běžného uživatele. Každý uživatel Virtlabu má přiřazené práva, které mu vymezují i to, jaká denní zpráva se mu zobrazí.

Editor úvodní strany bude fungovat na stejném principu jen bez rozdělení podle práv uživatelů a bude umístěn na svojí vlastní podstránce.

Všechny informace, jak denní zprávy, tak i úvodní zpráva budou uloženy v databázi v tabulce `motd`, kterou je potřeba vytvořit.

9 Realizace úprav webového rozhraní Virtlabu

Nejčastější úpravou webového rozhraní je vložení nové podstránky, kterou popíšeme podrobněji v kapitole 9.1. Z tohoto technického základu vychází většina úprav. Mezi úpravy, které potřebují novou podstránku patří graf zátěže, odesílání zpráv, editor denních zpráv. Úpravy je potřeba realizovat s ohledem na již používaný systém správy podstránek a zároveň realizovat úpravy s ohledem na budoucí inovaci systému.

Mezi základní pravidla pro implementaci úprav patří oddělování funkčnosti od prezentační vrstvy. Pro každé komplexnější řešení je nutné vytvořit vlastní třídu.

9.1 Přidání nové stránky do Virtlabu

Pro všechny nové podstránky je potřeba upravit soubory `virtlabWeb.php.inc`, `virtlabLanguage.php.inc`, `index.php`, `menu.php` a vytvořit soubor, jenž bude reprezentovat novou podstánku `nova_stranka.php`. Nově vytvořený soubor umístíme do adresáře `virtlab`, kde se nachází všechny soubory stránek.

Nejprve v souboru `virtlabLabguage.php.inc` nadefinujeme konstantu názvu stránky a její význam pro všechny jazykové mutace v metodách jazykových mutací. Konkrétně v metodách `CZE_text()` a `ENG_text()`.

Do `virtlabWeb.php.inc` do metody `Define_page_constants()` vložíme novou definici konstanty, která bude obsahovat identifikační číslo naší nové stránky.

V metodě `Authorization()` přidáme konstantu do správné části switch, podle toho pro jaký typ práv se stránka má zobrazovat, přidáme další konstantu, která reprezentuje práva naší nové stránky.

V souboru `index.php` doplníme, z čeho se naše stránka skládá a které další soubory potřebuje (JavaScripty, kaskádové styly a samotný název souboru se stránkou).

Nakonec v souboru `menu.php` přidáme řádek do části menu, ve které chceme zobrazit odkaz na naší novou stránku.

Konkrétní příklad přidání podstránky viz zdrojový kód 16

soubor: virtlabLanguage.php.inc

```
...
```

```
public function CZE_text($identifier) {
    case 'pagename-nova_stranka':   return 'Nova stranka';
}

public function ENG_text($identifier) {
    case 'pagename-nova_stranka':   return 'Nova stranka';
}

soubor: virtlabWeb.php.inc
 ...
public function Define_page_constants()
{
    define('VC_NOVA_STRANKA',          100);
}

public function Authorization($page_id, $rights=NULL) {
     ...
    case 'page-nova_stranka':
     ...
}

public function Define_pagename_constants() {
    define('VC_NOVA_STRANKA', $_lang->Text('pagename-nova_stranka'));
}

soubor: index.php
 ...
    case VC_NOVA_STRANKA:
        $script['script'][]  = $dir_java . 'javasctipt.js'
        $css['css'][]  = $dir_css . 'nova_stranka.css';
        $body['body'] = $dir_virt . 'nova_stranka.php';
            break;

soubor: menu.php
 ...
    if ($all->Authorization('page-nova_stranka'))
                print('<li><a href="index.php?page='.VC_NOVA_STRANKA.'&amp;' .SID. '">' .$_lang
                    ->Text('pagename-nova_stranka'). "</a></li>\n");
```

Výpis 16: Přidání nové stránky do Virtlabu

9.2 Graf zátěže

Realizace grafu zátěže je postavena na frameworku Pear a jeho knihovny `Graph`.

Do systému přidáme podstránku `graf-zateze.php` a třídu `Graph.php.inc`, která bude obsluhovat všechnu práci s daty a vracet obrázek grafu. Vstupními daty pro správné vygenerování grafu je název lokality, přihlašovací jméno přihlášeného uživatele, časový interval (od, do), počet dat, a data.

Data musí být pole, obsahující asociativní pole s klíči device, od, do. Kde každé takové asociativní pole znamená obsazení zařízení v čase od do.

Poté pomocí funkcí knihovny `Graph` z Pear sestrojíme graf, který uložíme do souboru s názvem `lokalita-login.png` do adresáře `graphs`. Příklad vytvoření grafu za použití naší knihovny viz kód 17.

```
$data[] = array("device" => "pc1@ostrava", "od" => 13242020 , "do" => 1323123); // od, do je
      timestamp urcujici zacatek a konec rezervace zarizeni
$graf = new Graph("nazev_lokality", "login_uzivatele");
$graf->setFrom(20, 8, 2008);
$graf->setTo(27, 8, 2008);
$graf->setData($data);
$graf->setCount(1);
$graf->done();
// zobrazime vytvoreny graf
$graf->showIt();
```

Výpis 17: Vytvoření grafu

9.3 Systém pro odesílání zpráv

Systém pro odesílání zpráv se nachází na podstránce `mail-sender.php` a využívá přímo knihovnu `PHPMailer`.

Knihovna nabízí velmi široké možnosti sestrojení zprávy a hlavně velmi zjednoduší jeho tvorbu. Umožňuje k emailu přidávat soubory, podporuje tvorbu těla zprávy jako HTML nebo prostý text a mnoho dalšího, více na stránkách projektu [10].

Ukázka tvorby zprávy pro potřeby Virtlabu a jeho odeslání pomocí knihovny viz kód 18.

```
$mail = new PHPMailer();
$mail->From      = "od_koho@email.cz";
$mail->FromName  = "Od koho je email";
$mail->Subject   = "Predmet emailu";
$mail->MsgHTML("Text emailu.");
$mail->IsHTML(true); // nastaveni, ze email obsahuje HTML znacky
$mail->AddAddress("Jan Rudovsky", "email@email.cz");
if (!$mail->Send()) {
  $msg = "Mailer Error: " . $mail->ErrorInfo;
} else {
  $msg = "Message sent!";
}
```

Výpis 18: Vytvoření emailu za pomoci PHPMailer

9.4 Editor denních zpráv a úvodní stránky

Editor denních zpráv a úvodní stránky se nachází v podstránce `motd-editor.php` a nabízí uživateli pro vkládání textu editor FCKeditor. FCKEditor je velmi flexibilní nástroj, který lze jednoduše modifikovat dle svých požadavků. Jednoduchá ukázka vložení FCKeditoru do stránky s šířkou 800px a výškou 400px viz kód 19.

```
$sBasePath = 'class/fckeditor/' ;
$oFCKeditor = new FCKeditor('text') ;
$oFCKeditor->BasePath = $sBasePath ; // cesta ke zdrojovym souborum FCKEditoru
$oFCKeditor->Value = '' ;  // predefinovany text v okne pro editaci
$oFCKeditor->Width  = '800px';
$oFCKeditor->Height  = '400px';
$oFCKeditor->Create();
```

Výpis 19: Vložení FCKeditoru do stránky

9.5 Export úloh do archívu

Pro export úlohy rozšíříme funkčnost v souboru `tasks-edit.php`, který reprezentuje podstránku pro editaci úloh. Také přidáme novou třídu `virtlabTaskExport.php.inc`, která využívá funkčnost Archive/Tar.php z frameworku Pear.

Jedna z možností jak přidat soubor do archívu viz kód 20. Více informací o práci s knihovnou je v [6].

```
$obj = new Archive_Tar('nazev_archivu.tar',  'gz');
$obj->addString('nazev_souboru_v_archivu.txt', 'datovy stream souboru');
```

Výpis 20: Vytvoření archívu za pomoci Archive/Tar.php

10 Závěr

Cílem práce bylo zjednodušení tvorby úlohy, práce s aktivní úlohou a rozšířit stávající webovou aplikaci Virtlabu o nové možnosti.

Modifikovali jsme opensource aplikaci Dia pro požadavky Virtlabu, a to tak, že nyní v ní lze nakreslit topologii. Prvky v ní použité mají nastavitelné parametry, které dále slouží pro účely rezervace úlohy.

Podařilo se nám přenést tvorbu úlohy do aplikace VirtlabDia, ve které lze vytvořit úlohy se všemi potřebnými informacemi. Následně jsme na obrázku z VirtlabDia postavili novou funkčnost obrázku topologie. Nyní obrázek slouží při rezervaci úlohy jako rozhraní pro práci s prvky. To znamená, že každý prvek na obrázku má sve kontextové menu, které obsahuje volby pro práci s ním (přístup na jeho konzoli a jeho realod). Kontextové menu je modulární a flexibilní. Dále dochází k nahrazování symbolických rozhraní za opravdu přiřazené rozhraní.

Další část diplomové práce pojednávala o změnách provedených na webovém rozhraní Virtlabu. Vytvořili jsme rozšíření, která usnadní jeho používání. Mezi ně patří graf zátěže, odesílání zpráv, editor denních zpráv a úvodní stránky a export úloh do archívu.

Diplomová práce splnila všechny požadavky, které na ni byly kladeny v úvodu. Zejména upravená aplikace Dia se bude využívat k tvorbě úloh. Většina realizovaných úprav je dnes již využívána ve webové aplikaci Virtlabu.

Konkrétně jsou již nasazeny systém pro odesílání zpráv, editor denních zpráv, graf zátěže, export úloh a nahrazování textu v obrázku topologie. Rozšíření generující clickable mapy a kontextová menu je v současně době testován na virtuálních lokalitách a měly by být přidány do webového rozhraní Virtlabu během roku 2009.

11 Reference

[1] PHP [online]. Dostupný na URL: http://www.php.net/

[2] Dia [online]. Dostupný na URL: http://projects.gnome.org/dia/

[3] MySQL [online]. Dostupný na URL: http://www.mysql.com/

[4] Framework Pear [online]. Dostupný na URL: http://pear.php.net/

[5] Knihovna Graph frameworku Pear [online]. Dostupný na URL: http://pear.veggerby.dk/samples/

[6] Knihovna Archive Tar frameworku Pear [online]. Dostupný na URL: http://pear.php.net/package/Archive_Tar

[7] Virtlab Wikipedie [online]. Dostupný na URL: http://infra2.cs.vsb.cz/vl-wiki

[8] FCKEditor [online]. Dostupný na URL: http://www.fckeditor.net/

[9] Mootools calendar [online]. Dostupný na URL: http://www.electricprism.com/aeron/calendar/

[10] PHPMailer [online]. Dostupný na URL: http://phpmailer.codeworxtech.com/

[11] [online]. Dostupný na URL: http://infra2.cs.vsb.cz/vl-wiki/index.php

[12] SVG dokumentace [online]. Dostupný na URL: http://www.w3.org/Graphics/SVG/

[13] Tvorba Shape [online]. Dostupný na URL: http://dia-installer.de/howto/create_shape/

[14] Dokumentace ObjectOps [online]. Dostupný na URL: http://faemalia.org/wiki/view/Technical/ObjectOps

[15] Dokumentace StdProps [online]. Dostupný na URL: http://faemalia.org/wiki/view/Technical/StdProps

[16] Kompilace Dia pro Windows [online]. Dostupný na URL: http://dia-installer.de/howto/compile_msvc/index.html

[17] Dokumentace XPM [online]. Dostupný na URL: http://en.wikipedia.org/wiki/XPM_(image_format)

[18] Relax schema pro Virtlab ulohu [online]. Dostupný na URL: https://virtlab.cs.vsb.cz/relax/topology.rng

A Kompilace aplikace Dia

A.1 Potřebné balíčky pro kompilaci a samotná kompilace VirtlabDia

Důležité balíky

- pkg-config version 0.14.0 or higher.
- GTK version 2.0.0+
- libxml version 2.3.9 nebo vyšší
- intltool version 0.21 nebo vyšší
- freetype version 2.0.9 nebo vyšší
- libart version 2.1.0 nebo vyšší
- libpng (pro podporu exportu do PNG)
- gnome-libs (pro podporu Gnome)
- libxslt pro XSLT plug-in.
- Python a PyGtk2 pro podporu Python

Příkazy pro kompilaci VirtlabDia spouštěné v adresáři se zdrojovými kody VirtlabDia:

```
automake
./configure --program-prefix=virtlab
make
sudo make install
```

Výpis 21: Kompilace a instalace VirtlabDia

Po úspěšné instalaci se aplikace spouští pomocí příkazu `virtlab`. Více informací a podrobnější popis instalace, kompilace nalezneme na CD v souboru `cti me.txt`.

B Šablony a schémata XML

B.1 Šablona DTD Dia XML

```
<!ELEMENT dia:diagram (dia:diagramdata, (dia:layer)*) >
<!ATTLIST dia:diagram
   xmlns:dia CDATA #FIXED "http://www.lysator.liu.se/~alla/dia/">

<!ELEMENT dia:diagramdata (dia:attribute)* >

<!ELEMENT dia:layer (dia:object | dia:group)*>
<!ATTLIST dia:layer
   name CDATA #REQUIRED
   visible (true|false) #REQUIRED >

<!ELEMENT dia:object ((dia:attribute)*, dia:connections?)>
<!ATTLIST dia:object
   type CDATA #REQUIRED
   version NMTOKEN #REQUIRED
   id ID #REQUIRED >

<!ELEMENT dia:connections (dia:connection)*>

<!ELEMENT dia:connection EMPTY>
<!ATTLIST dia:connection
   handle NMTOKEN #REQUIRED
   to IDREF #REQUIRED
   connection NMTOKEN #REQUIRED>

<!ELEMENT dia:group (dia:object | dia:group)*>

<!ELEMENT dia:attribute (dia:composite | dia:int | dia:enum | dia:real |
          dia:boolean | dia:color | dia:point | dia:rectangle |
          dia:string | dia:font )*>
<!ATTLIST dia:attribute name CDATA #REQUIRED >

<!ELEMENT dia:composite (dia:attribute|dia:composite)*>
<!ATTLIST dia:composite type CDATA #IMPLIED>

<!ELEMENT dia:int EMPTY>
<!ATTLIST dia:int val NMTOKEN #REQUIRED>

<!ELEMENT dia:enum EMPTY>
<!ATTLIST dia:enum val NMTOKEN #REQUIRED>

<!ELEMENT dia:real EMPTY>
<!ATTLIST dia:real val CDATA #REQUIRED>

<!ELEMENT dia:boolean EMPTY>
<!ATTLIST dia:boolean val (true|false) #REQUIRED>

<!ELEMENT dia:color EMPTY>
<!ATTLIST dia:color val CDATA #REQUIRED>
```

```
<!ELEMENT dia:point EMPTY>
<!ATTLIST dia:point  val  CDATA #REQUIRED>

<!ELEMENT dia:rectangle EMPTY>
<!ATTLIST dia:rectangle val  CDATA #REQUIRED>

<!ELEMENT dia:string (#PCDATA)>

<!ELEMENT dia:font EMPTY>
<!ATTLIST dia:font
  name CDATA #REQUIRED
  style  CDATA #IMPLIED
  family  CDATA #IMPLIED>
```

Výpis 22: Šablona DTD Dia XML