

**VŠB - Technická univerzita Ostrava**  
**Fakulta elektrotechniky a informatiky**

**DIPLOMOVÁ PRÁCE**

**2010**

**Bc. Pavel Burda**

**VŠB - Technická univerzita Ostrava**  
**Fakulta elektrotechniky a informatiky**  
**Katedra informatiky**

**Implementace konektorů pro dynamické propojování  
distribuovaných virtuálních topologií v systému Virlab**

**The implementation of connectors for dynamic  
connections of distributed virtual topologies  
in the Virlab system**

**2010**

**Bc. Pavel Burda**

## Zadání diplomové práce

Student: **Bc. Pavel Burda**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: **Implementace konektorů pro dynamické propojování distribuovaných virtuálních topologií v systému Virlab**  
**Implementation of Connectors for Dynamic Interconnection of Distributed Virtual Topologies in Virlab System**

### Zásady pro vypracování:

Implementujte systém konektorů umožňující logicky přemostňovat paralelně existující síťové topologie v systému Virlab podle okamžitých požadavků uživatelů. Systém konektorů koncipujte tak, aby jej bylo možné v budoucnu použít i pro propojování s dalšími externími systémy.

1. Rozšiřte schema popisu požadované (logické) topologie o nový typ prvku "konektor".
2. Implementujte modul tunelovacího serveru realizující přemostnění provozu mezi propojenými konektory.
3. Rozšiřte GUI pro rezervaci tak, aby uživatel mohl u všech konektorů rezervované topologie stanovit, kteří uživatelé budou mít právo se na tyto konektory napojit.
4. Vytvořte GUI a příslušnou distribuovanou softwarovou infrastrukturu pro dynamické určení propojení konektorů rezervované topologie s konektory jiných paralelně rezervovaných topologií.
5. Systém důkladně otestujte a podle pokynů vedoucího DP integrujte do produkčního prostředí.

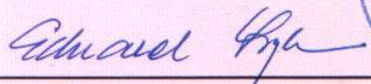
### Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

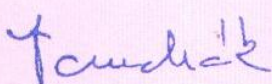
Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Grygárek, Ph.D.**

Datum zadání: 20.11.2009  
Datum odevzdání: 07.05.2010

  
\_\_\_\_\_  
doc. Dr. Ing. Eduard Sojka  
vedoucí katedry



  
\_\_\_\_\_  
prof. Ing. Ivo Vondrák, CSc.  
děkan fakulty

## **Poděkování**

Na tomto místě bych rád poděkoval všem, kteří mi s touto prací pomohli. Poděkování patří celému vývojovému týmu Virlabu, se kterým jsem spolupracoval. Především pak koordinátorovi týmu a mému vedoucímu Ing. Petru Grygárkovi, Ph.D.

## **Prohlášení**

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

V Ostravě dne 25. dubna 2010

.....

Bc. Pavel Burda

## **Abstrakt**

Cílem této diplomové práce je rozšířit funkcionalitu virtuální laboratoře počítačových sítí o logický prvek konektor. Ten umožňuje propojovat rezervované úlohy mezi sebou podle požadavku uživatelů a vytvářet tak nové a větší síťové topologie. Součástí práce je rozšíření webové aplikace (GUI), VirlabDia a také vytvoření několika ukázkových úloh demonstrujících použití konektorů.

## **Klíčová slova**

Virlab, virtuální síťová laboratoř, konektor, tunelovací server, dia

## **Abstract**

The purpose of this thesis is to extend the functionality of virtual laboratory of computer networks with logical component connector. It allows to connect reserved tasks with each other according to user requests and to create new and larger topologies. The part of this thesis are the extension of web application (GUI), VirlabDia and several tasks demonstrating the usage of connectors.

## **Keywords**

Virlab, virtual network laboratory, connector, tunnel server, dia

## Seznam použitých symbolů a zkratek

GDB	- The GNU Project Debugger
GNU	- GNU's Not Unix
GPL	- General Public License
GUI	- Graphical User Interface
HTTP	- Hypertext Transfer Protocol
I/O	- Input/Output
ISP	- Internet Service Provider
OS	- Operation System
RELAX NG	- REGular LAnguage for XML Next Generation
SOAP	- Simple Object Access Protocol
SQL	- Structured Query Language
TCP	- Transmission Control Protocol
TTL	- Time To Live
UDP	- User Datagram Protocol
UML	- Unified Modeling Language
WAN	- Wide Area Network
XML	- eXtensible Markup Language

# Obsah

1. Úvod .....	1
2. Virtuální síťová laboratoř (Virtlab) .....	2
2.1 Co je to Virtlab .....	2
2.2 Lokality .....	2
2.3 Princip funkce tunelovacího serveru .....	3
3. Cíle řešení .....	4
4. Analýza .....	5
4.1 Use case diagram .....	5
4.2 Cesta rámce přes konektory .....	6
4.2.1 Typy rozhraní konektorů .....	6
4.2.2 Použití virtuálního rozhraní iint .....	6
4.2.3 Popis cesty rámce .....	7
4.2.4 Problém se smyčkami .....	9
4.3 Nastavení konektoru .....	9
4.3.1 Přístupová práva .....	10
4.3.2 Maximální počet připojení .....	10
4.4 Grafické uživatelské rozhraní (GUI) .....	10
4.4.1 Úvod .....	10
4.4.2 Grafické znázornění konektoru .....	10
4.4.3 Seznamy konektorů .....	11
4.4.4 Notifikace o připojení konektoru .....	12
5. Implementace .....	13
5.1 Úvod .....	13
5.2 Logické vytvoření nového prvku – connector .....	13
5.2.1 RELAX NG schémata .....	13
5.2.2 XML soubor s vybavením lokality .....	13
5.2.3 Rezervační server .....	14
5.2.4 Generátor konfigurací .....	14
5.3 Implementace konektorů do tunelovacího serveru .....	14
5.3.1 Modul Port_connector .....	14
5.3.2 Hledání cílových konektorů v Redirect table .....	14
5.3.3 Problém jednoznačnosti názvu rozhraní .....	15
5.4 Implementace konektorů do řídicí aplikace .....	15

5.4.1	Zavedení prvku konektor do konverzního algoritmu.....	15
5.4.2	Rezervace úlohy s konektory.....	16
5.4.3	Dotazy na dostupné konektory ze všech lokalit .....	16
5.4.4	Nastavení konektoru .....	17
5.4.5	Filtr na dostupné konektory.....	17
5.4.6	Funkce připojit/odpojit konektor .....	17
5.4.7	Notifikace o připojení konektoru.....	19
5.4.8	Jazykové mutace.....	19
5.5	Integrace konektorů do aplikace VirlabDia.....	20
5.6	Zdrojové soubory .....	21
6.	Práce s konektorem v GUI.....	23
7.	Testování.....	25
7.1	Příprava .....	25
7.2	Propojení trunk linky mezi dvěma přepínači.....	26
7.3	Propojení směrovače (router-on-the-stick) a přepínače .....	28
8.	Závěr.....	31
9.	Přílohy.....	33
A.	Obsah CD .....	33
B.	Instalace konektorů na novou lokalitu .....	34
C.	Struktura databáze.....	35



## Seznam tabulek

Tabulka 1: Redirect table 1 .....	15
Tabulka 2: Redirect table 2 .....	15
Tabulka 3: Redirect table 3 .....	15
Tabulka 4: Zdrojové soubory tunelovacího serveru .....	21
Tabulka 5: Zdrojové soubory webového rozhraní .....	21
Tabulka 6: Zdrojové soubory VirlabDia .....	22
Tabulka 7: Adresace rozhraní pro router-on-the-stick .....	29
Tabulka 8: Obsah CD .....	33
Tabulka 9: SQL tabulka connectors .....	35

## Seznam obrázků

Obrázek 1: Základní distribuovaná architektura Virlabu.....	2
Obrázek 2: Use case diagram.....	5
Obrázek 3: UML Activity diagram .....	7
Obrázek 4: Ukázková topologie pro popis cesty rámce.....	8
Obrázek 5: Cesta rámce přes tunelovací server.....	8
Obrázek 6: Tok dat mezi násobnými konektory.....	9
Obrázek 7: Konektor .....	10
Obrázek 8: Kontextové menu konektoru .....	11
Obrázek 9: Upozornění na nepřčtenou poštu .....	12
Obrázek 10: Generátor konfigurací .....	14
Obrázek 11: Rozšířené vytvoření nové rezervace .....	16
Obrázek 12: Odkaz pro připojení konektoru.....	18
Obrázek 13: Odkaz pro odpojení konektoru .....	18
Obrázek 14: Notifikace o připojení konektoru.....	19
Obrázek 15: Software VirlabDia.....	20
Obrázek 16: Tlačítko pro práci s konektorem.....	23
Obrázek 17: Kontextové menu pro práci s konektorem .....	24
Obrázek 18: GUI pro práci s konektorem.....	24
Obrázek 19: DIA obrázek úlohy Konektor ISP.....	25
Obrázek 20: DIA obrázek úlohy Konektor VLAN.....	26
Obrázek 21: Dvě úlohy propojené konektory .....	26
Obrázek 22: Aktivní rezervace Konektor VLAN uživatele Pat .....	27
Obrázek 23: Seznam konektorů k dispozici u uživatele Pat .....	27
Obrázek 24: Notifikace o připojení ke konektoru uživatele Mat .....	28
Obrázek 25: Router-on-the-stick s konektory .....	29
Obrázek 26: Směrovací tabulka router-on-the-stick.....	29
Obrázek 27: Router-on-the-stick nastavení konektoru .....	30

## 1. Úvod

Tato diplomová práce se věnuje problematice dynamického propojování úloh v systému Virlab [1].

Uživatelé virtuální síťové laboratoře Virlab mohou nyní pracovat pouze na svých úlohách, popřípadě na těch, ke kterým jsou přizváni jako kolegové. Cílem mé práce je překonat toto omezení a poskytnout uživatelům možnost libovolně dynamicky (za běhu) spojovat své úlohy do rozsáhlých topologií.

Principem řešení je logické rozšíření úloh o nový prvek „konektor“. Ten lze připojit k libovolnému laboratornímu prvku pomocí rozhraní Ethernet nebo Serial. Konektory pak mohou uživatelé vzájemně propojovat i mezi různými lokalitami.

Tento systém poskytne možnost vytvářet z uživatelů týmy, v rámci kterých budou spolupracovat a propojovat své úlohy navzájem pomocí konektorů. Více uživatelů pak zvládne řešit složitější a rozsáhlejší úlohy než jednotlivci.

Součástí práce je integrace do webového GUI tak, aby uživatelé Virlabu mohli pohodlně využívat veškeré funkce konektorů.

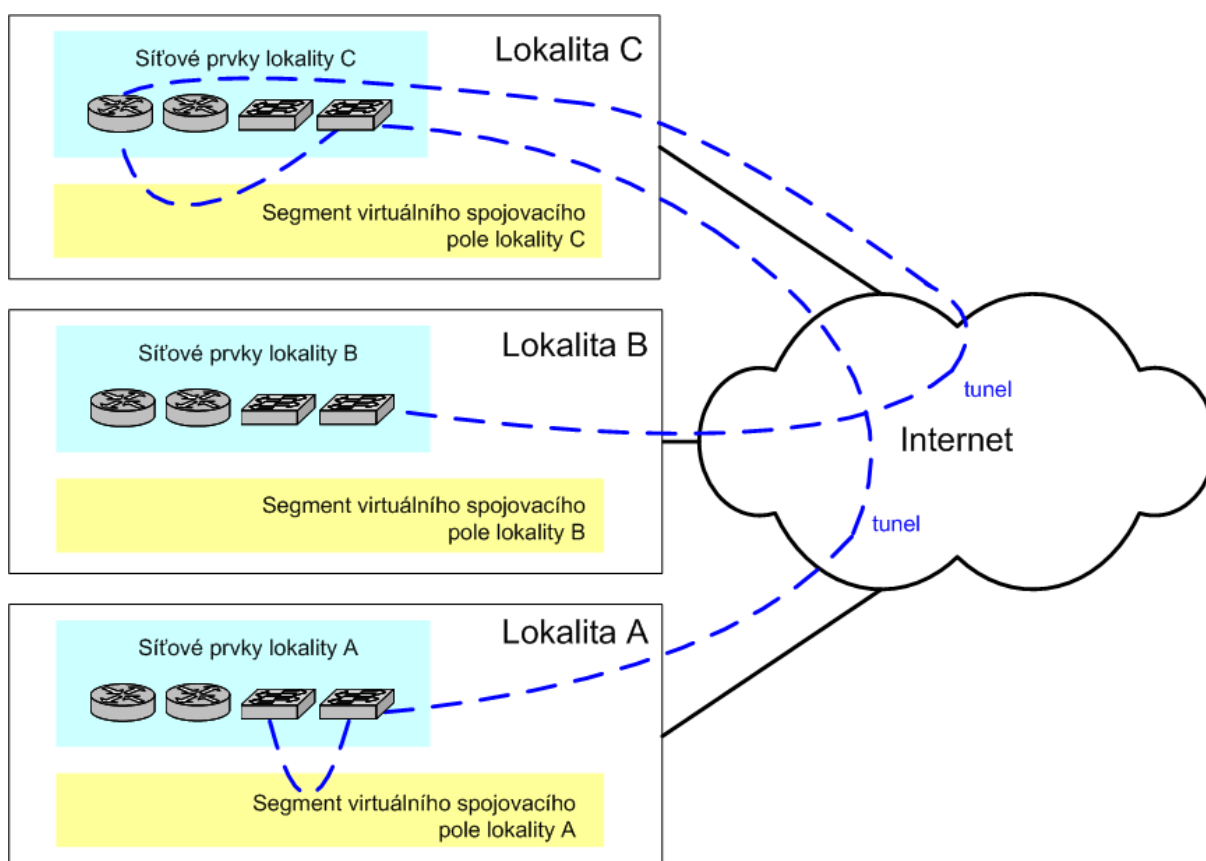
## 2. Virtuální síťová laboratoř (Virtlab)

### 2.1 Co je to 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.

Základní architektura Virtlabu je na obr. 1.

Více informací o projektu Virtlab naleznete na jeho domovských stránkách [1].



Obrázek 1: Základní distribuovaná architektura Virtlabu

### 2.2 Lokality

Jelikož cílem distribuovaného řešení je zpřístupnění síťových zařízení různých síťových laboratoří ve světě, je systém rozdělen na základní jednotky nazývané lokality, které reprezentují právě tyto síťové laboratoře. Tedy lokalita je místo, kde jsou fyzicky umístěna síťová zařízení, která se zpřístupňují místním uživatelům i uživatelům z ostatních lokalit. Každá lokalita má svá pravidla, jež definují, která zařízení a ve kterém čase jsou k dispozici vybraným vzdáleným lokalitám. V každé

lokalitě proto běží Rezervační server, který obhospodařuje práva uživatelů a lokalit a zabezpečuje půjčování prvků <sup>1</sup>.

Pro vzájemnou komunikaci mezi sebou lokality využívají SOAP protokol <sup>2</sup>.

### 2.3 Princip funkce tunelovacího serveru

Každá lokalita má vlastní modulární tunelovací server.

Po přijetí dat musí server zjistit zdrojové rozhraní. Toto rozhraní je textový řetězec v předem daném formátu. Podle zdrojového rozhraní se z tabulky přesměrování zjistí cílové rozhraní, které obsahuje informaci, kam se mají data přesměrovat. Cílové rozhraní se může nacházet buď v místní, nebo také v jiné lokalitě.

Pokud se nachází v místní, předají se data určenému modulu. Ten už je přepošle v závislosti na své implementaci. Tímto se vytvoří virtuální most v rámci jedné lokality.

Pokud se cílové rozhraní nachází v jedné z ostatních lokalit, předají se data modulu internetového portu. Ten je přepošle i s informacemi o přesměrování druhému tunelovacímu serveru enkapsulovaná do UDP paketu. Na druhé straně modul internetového portu přijatá data dekapsuluje a dále s nimi pracuje jako u prvního tunelovacího serveru. Pomocí tabulky přesměrování ze zdrojového rozhraní zjistí cílové a odešle je odpovídajícímu modulu. Tímto způsobem se vytvoří virtuální most v rámci více lokalit <sup>3</sup>.

Všechny moduly, které se aktivují při spuštění tunelovacího serveru, naslouchají na ethernetových rozhraních podle registrace jednotlivých modulů. Každý modul si může zaregistrovat libovolný počet rozhraní – pokud cílové rozhraní rámce odpovídá registrovanému, pak jsou data tomuto modulu předána ke zpracování.

---

<sup>1</sup> Tato podkapitola byla převzata z diplomové práce Ing. Tomáše Hrabálka [8]

<sup>2</sup> Simple Object Access Protocol slouží k výměně zpráv založených na XML přes síť, především pomocí HTTP protokolu na aplikační vrstvě

<sup>3</sup> Tento odstavec byl převzat z diplomové práce Ing. Tomáše Hrabálka [8]

### 3. Cíle řešení

V rámci této diplomové práce vytvořím nový typ laboratorního prvku – konektor. Ten bude sloužit k propojování různých úloh kdykoli během času jejich rezervace podle přání uživatelů.

Systém je potřeba navrhnout takovým způsobem, aby bylo možné propojovat i konektory mezi různými lokalitami. Dalším požadavkem je možnost připojit několik (nastavitelný maximální počet) konektorů k jedinému.

Realizace nového laboratorního prvku se bude skládat z následujících kroků:

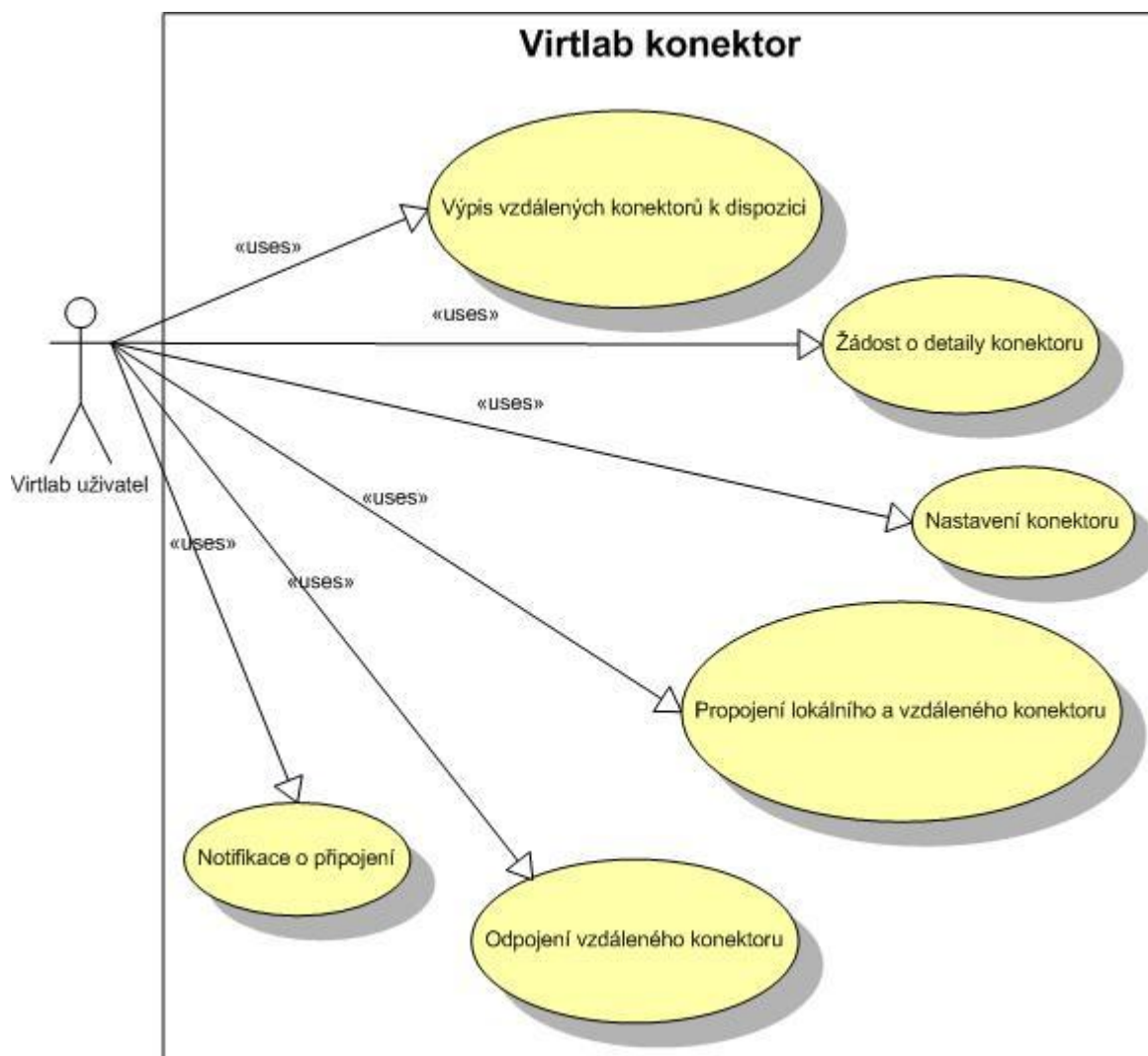
- **Logické vytvoření prvku**  
Tím je myšleno zavedení nového typu prvku do popisu úloh Virlabu tak, aby tento prvek byl rozpoznáván jako konektor.
- **Vytvoření nového modulu tunelovacího serveru**  
Prvky typu konektor budou mít vlastní speciální rozhraní, pomocí kterých se budou připojovat k ostatním laboratorním prvkům. Nový modul tunelovacího serveru si tato rozhraní zaregistruje. Při přijetí rámce s cílovým rozhraním konektoru se pak modul postará o úpravu jeho hlaviček (zdroj, cíl) tak, aby byl doručen jinému konektoru nebo laboratornímu prvku, který je k němu připojený.
- **Rozšíření grafického uživatelského rozhraní**  
Stávající webové GUI Virlabu se rozšíří o funkce, které uživatelům umožní pohodlně a intuitivně s konektory pracovat.
- **Integrace konektorů do aplikace VirlabDia**  
Tato aplikace slouží k návrhu topologií pro Virlab. Konektory budou přidány do kolekce, která obsahuje další laboratorní prvky podporované Virlabem.

## 4. Analýza

### 4.1 Use case diagram

Chování systému konektorů z pohledu uživatele distribuované virtuální laboratoře Virlab jsem definoval pomocí diagramu případů užití na obr. 2.

Pojmem lokální je označen konektor, který je vlastněný daným uživatelem (nachází se v jeho úloze). Vzdálené konektory jsou pak součástí úloh ostatních uživatelů.



Obrázek 2: Use case diagram

Případy užití:

- **Výpis vzdálených konektorů k dispozici** – zobrazí seznam všech konektorů, ke kterým se uživatel může připojit
- **Žádost o detaily konektoru** – zobrazí vlastnosti vybraného vzdáleného konektoru
- **Nastavení konektoru** – umožní nastavit oprávnění a max. počet připojení k lokálnímu konektoru
- **Propojení lokálního a vzdáleného konektoru** – spojí dva konektory z různých úloh

- **Odpojení vzdáleného konektoru** – odpojí vzdálený konektor od lokálního (je-li připojeno více konektorů, pak ostatní spojení zachová)
- **Notifikace o připojení** – upozorní uživatele, že někdo se připojil k jeho konektoru

## 4.2 Cesta rámce přes konektory

Veškeré rámce ve virtuální topologii procházejí přes „tunelovací server“ (obr. 5). Jeho součástí je `redirect table` – tabulka obsahující spoje mezi jednotlivými prvky. Podle této tabulky se tunelovací server rozhoduje, kam bude rámce odesílat. Záznamy se do `redirect table` přidávají vždy při aktivaci úlohy a mají omezenou platnost (délka trvání úlohy).

Konektory se tedy musí implementovat do tunelovacího serveru. Díky bakalářské práci Václava Bortlíka [3] je tento server modulární, takže místo úpravy samotného serveru lze napsat nový modul pro konektory.

Tento modul se u tunelovacího serveru zaregistruje tak, že bude přijímat veškeré rámce s cílovým rozhraním konektoru (viz. kap. 2.3):

- `eint`: Ethernet interface pro připojení k libovolnému prvku s Ethernet rozhraním
- `sint`: Serial interface pro připojení k libovolnému prvku se Serial rozhraním
- `iint`: virtuální rozhraní sloužící pouze k vzájemnému propojování konektorů

### 4.2.1 Typy rozhraní konektorů

Laboratorní prvky Virlabu používají dva typy rozhraní:

- Serial (např. WAN linky mezi směrovači)
- Ethernet (např. linka mezi přepínačem a směrovačem)

Konektory lze připojit k libovolnému laboratornímu prvku. Proto mohou mít ethernetové (`eint`) nebo sériové (`sint`) rozhraní. Nikoli však obě současně. Typ rozhraní se určuje během přidávání konektorů do seznamu vybavení lokality.

Uživatel při vytváření úlohy pak pracuje pouze s obecným prvkem typu konektor a nemusí rozlišovat typ jeho rozhraní. To zajistí mapovací algoritmus při rezervaci úlohy, kdy vybere konektor se správným typem rozhraní podle toho, ke kterému prvku je připojen.

### 4.2.2 Použití virtuálního rozhraní `iint`

Seznam propojených konektorů jsem mohl v tunelovacím serveru uložit do nové datové struktury. Tu bych však musel udržovat a to nejen ve vlastní, ale i v ostatních lokalitách. Konektory totiž lze propojit i mezi různými lokalitami.

Místo vlastní datové struktury jsem se rozhodl využít již stávající – `redirect table`. Při propojení konektorů do této tabulky přidám záznam lokální->vzdálený konektor. Pro rozlišení těchto záznamů od ostatních jsem vytvořil nový typ rozhraní – `iint`. Při přijetí rámce (tunelovacím serverem) s cílovým rozhraním `iint` pak modul konektorů najde v této tabulce odpovídající připojený konektor, kterému je potřeba rámec přeposlat.

Propojení konektorů je tedy realizováno vložením nového záznamu do `redirect table`. Ten obsahuje dvojici propojovaných konektorů s jejich virtuálním rozhraním `iint`. Odstraněním záznamu se konektory rozpojí. Tyto akce budou vyvolávány uživatelem za běhu úloh.

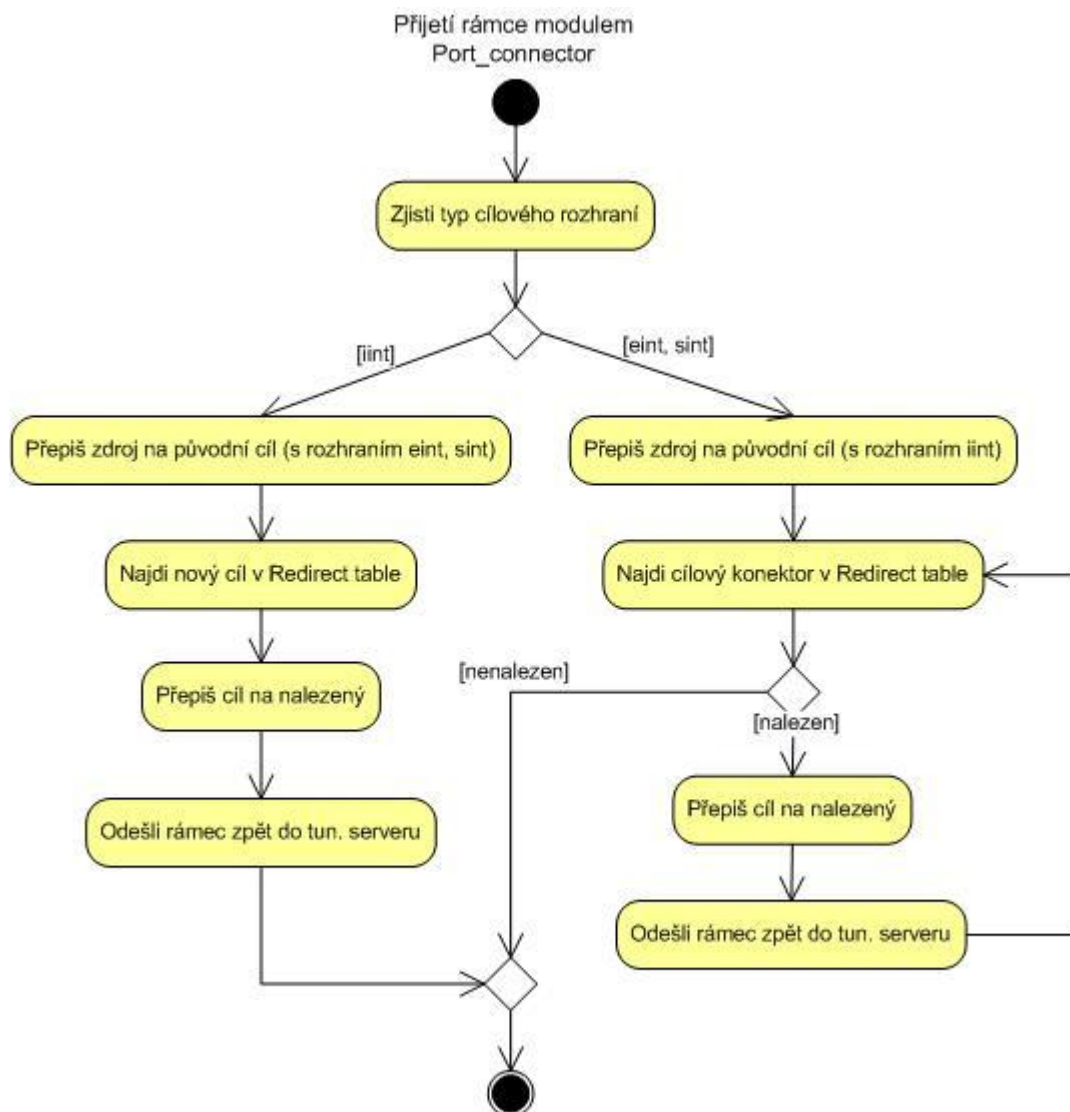
Pro práci s `redirect table` lze využít konzoli tunelovacího serveru na portu 40001. Přes tuto konzoli lze pohodlně spravovat tabulky i v cizích lokalitách.



Parametrem záznamů v `redirect table` je i pole TTL, které určuje dobu jejich platnosti. Pro konektory je potřeba nastavit konec rezervace, která končí dříve, protože konektor bude propojovat dvě úlohy s různými časy konce rezervace.

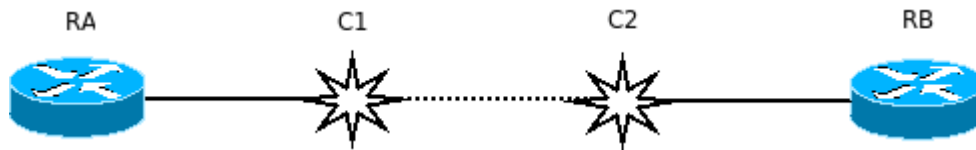
### 4.2.3 Popis cesty rámce

UML Activity diagramem (obr. 3) jsem popsal algoritmus modulu konektorů (třída `Port_connector` v tunelovacím serveru). Ten přijímá rámce s cílovým rozhraním konektoru (`eint`, `sint`, `iint`) a upravuje zdroj a cíl tak, aby konektory plnily svou funkci, tedy přeposílání rámce cestou „lokální prvek->lokální konektor->vzdálený konektor->vzdálený prvek“. Po každé úpravě je rámec odeslán zpět do jádra tunelovacího serveru.



Obrázek 3: UML Activity diagram

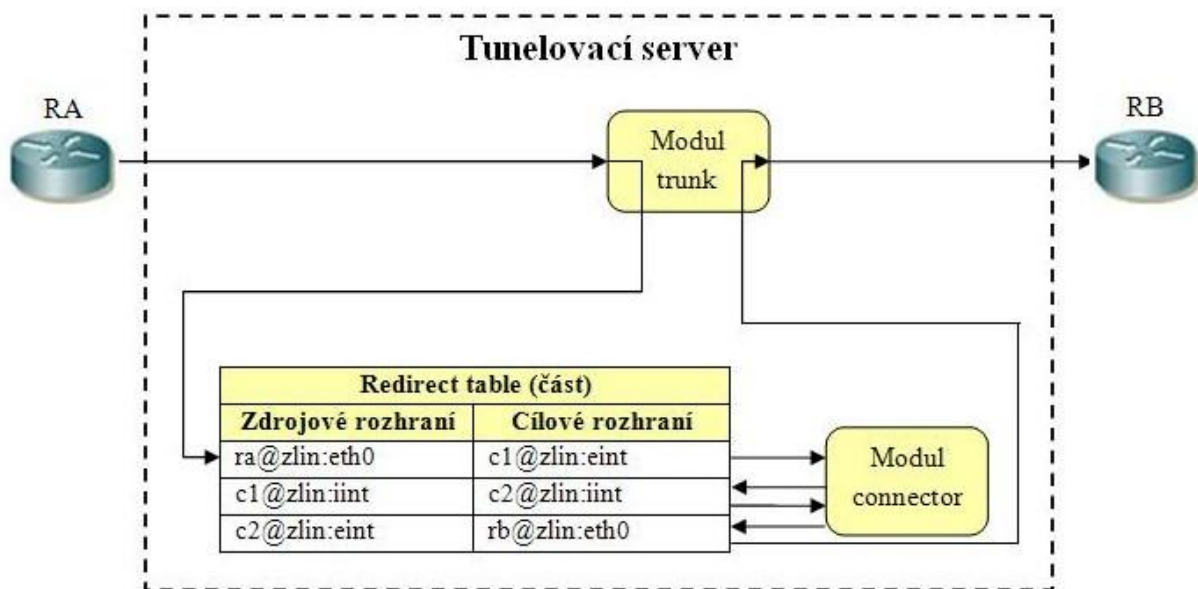
Jako příklad je použita ukázková topologie na obr. 4. Znáznorňuje dva směrovače propojené pomocí konektorů <sup>4</sup>.



Obrázek 4: Ukázková topologie pro popis cesty rámce

Obr. 5 zobrazuje cestu rámce ze směrovače RA přes tunelovací server a jeho moduly do směrovače RB (v rámci jedné lokality). Je zde uvedena i část `redirect table` – obsahuje pouze záznamy potřebné k přenosu rámce cestou RA-C1-C2-RB (nikoli zpět).

Modul trunk slouží k tunelování ethernetových rozhraní přes trunk. Využívá k tomu označování rámcu pomocí standardu IEEE 802.1q. Podrobnosti o tomto modulu jsou uvedeny v bakalářské práci Václava Bortlíka [3].



Obrázek 5: Cesta rámce přes tunelovací server

Textový popis cesty rámce odeslaného ze směrovače RA s konektorem C1 do směrovače RB s konektorem C2, tedy RA:fa0->C1:eint->C2:eint->RB:fa0:

- RA odešle rámec s cílem C1:eint
- rámec s cílovým rozhraním eint nebo sint je přijat modulem tunelovacího serveru `Port_connector`
  - o v `redirect table` jsou nalezeny všechny konektory připojené k C1:iint, pro který je rámec určen (v tomto případě je nalezen pouze C2:iint)
  - o pro každý z těchto připojených konektorů se:
    - zdroj přepíše na původní cílový konektor, ale s virtuálním rozhraním iint, tedy C1:iint

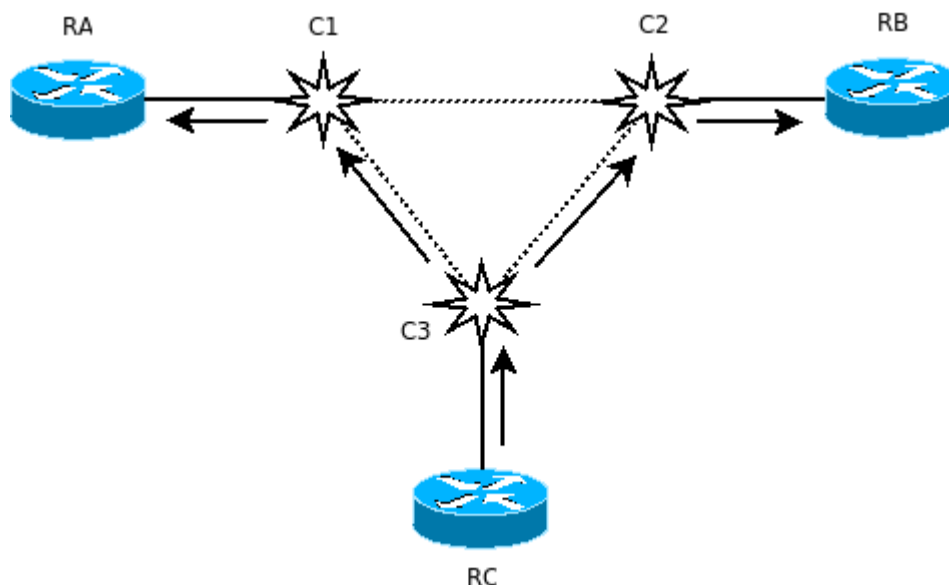
<sup>4</sup> Grafické znázornění konektoru je dále popsáno v kap. 4.4.2

- cíl se přepíše na nový cílový konektor s virtuálním rozhraním `iint`, tedy `C2:iint`
- rámec se odešle zpět do jádra tunelovacího serveru
- rámec s cílovým rozhraním `iint` je opět přijat modulem `Port_connector`
  - současným cílem je konektor `C2` – ten se nyní zapíše jako zdroj a rozhraní se změní na `eint` nebo `sint`, protože tímto rozhraním je konektor připojen v jiném laboratornímu prvku; nový zdroj je tedy: `C2:eint`
  - cíl je nalezen v `redirect table` – je to prvek připojený k `C2:eint`, v tomto případě `RB:fa0`
  - rámec se odešle zpět do jádra tunelovacího serveru, který se již postará o doručení do směrovače `RB`

#### 4.2.4 Problém se smyčkami

Vzhledem k tomu, že konektory umožňují násobné připojení, je potřeba zajistit, aby nevznikaly smyčky, ve kterých by se rámce zacyklily.

K této situaci však nemůže dojít, protože po přijetí rámce z virtuálního rozhraní `iint` se cíl vždy přepisuje buď na fyzické rozhraní `eint` nebo `sint`. Nikdy se tedy nemůže stát, že by se rámec přeposlal na další konektor s virtuálním rozhraním `iint`. A proto může vést cesta rámce nejvýše přes dva konektory za sebou.



**Obrázek 6: Tok dat mezi násobnými konektory**

Na obr. 6 je příklad, kde směrovač `RC` odesílá data na konektor `C3`, který je připojen současně k `C1` i `C2`. Přesto, že konektory `C1` a `C2` jsou také propojeny, nevytvorí smyčku a nebudou si data přijatá z `C3` navzájem přeposílat, protože jejich zdrojem je konektor a nemohou být tedy odeslána na další konektor. `C1` a `C2` tedy data odešlou pouze na směrovače, které jsou k nim připojeny (data jsou na obr. 6 znázorněna šipkami).

### 4.3 Nastavení konektoru

Každému konektoru v úloze může uživatel (vlastník úlohy) nastavit následující vlastnosti, které se ukládají do tabulky v SQL databázi. Záznamy s výchozími hodnotami pro každý konektor

jsou vždy uloženy ihned po úspěšné rezervaci úlohy s konektory. Tyto výchozí hodnoty může uživatel změnit během výběru času rezervace.

### 4.3.1 Přístupová práva

Přístupová práva jsou definována regulárním výrazem. Jelikož uživatelské jméno je jedinečné pouze v rámci jedné lokality, musí tato práva určovat uživatele včetně lokality, ze které se k danému konektoru mohou připojit. Odpovídá-li uživatelské jméno a lokalita regulárnímu výrazu, pak se tento konektor zobrazí v tabulce konektorů k dispozici. Výchozí hodnotou je „\*“, tzn. že se může připojit libovolný uživatel z jakékoli lokality.

Uživatelské jméno a název lokality jsou odděleny znakem „@“. Například regulárním výrazem „bur254\@vsb.\*“ je povoleno připojení pouze uživateli bur254, který je přihlášen z lokality začínající názvem „vsb“.

### 4.3.2 Maximální počet připojení

Tento parametr určuje, kolik připojení ke konektoru může existovat. Výchozí hodnotou je jedna. Dosáhne-li počet připojení této hodnoty, pak se již konektor žádnému uživateli nezobrazí v tabulce konektorů k dispozici.

## 4.4 Grafické uživatelské rozhraní (GUI)

### 4.4.1 Úvod

Virtlab má své vlastní webové grafické uživatelské rozhraní, které uživatelé využívají pro rezervace a práci s úlohami. Součástí mé diplomové práce je rozšíření tohoto rozhraní o nové funkce, které konektory nabízejí.

Vlastní úprava GUI se skládá z následujících částí:

- Grafické znázornění konektoru (značka laboratorního prvku konektor)
- Seznamy konektorů (rozhraní pro práci s konektory)
- Notifikace o připojení konektoru
- Rozšíření formuláře pro vytvoření nové rezervace (nová pole sloužící k nastavení konektorů v úloze během rezervace)

### 4.4.2 Grafické znázornění konektoru

Každá úloha ve Virtlabu má svůj klikací bitmapový obrázek topologie generovaný z DIA souboru. Pokud má uživatel úlohu právě aktivní (v čase rezervace), pak může kliknutím do tohoto obrázku vyvolat kontextovou nabídku laboratorního prvku, na který kliknul.

Všechny prvky mají svůj symbol, který je jednoznačně identifikuje. Pro konektor jsem zvolil následující obrázek:



Obrázek 7: Konektor

Dalším krokem je integrovat tento obrázek do aplikace `VirtlabDia`, která slouží k návrhu (kreslení) topologií. Cílem je přidat nový prvek s tímto obrázkem do palety `Virtlab`, aby mohl být použit při vytváření nebo úpravě `Virtlab` topologií. Zavedení nového prvku bude vyžadovat změny ve zdrojovém kódu `VirtlabDia` a jeho následnou kompilaci.

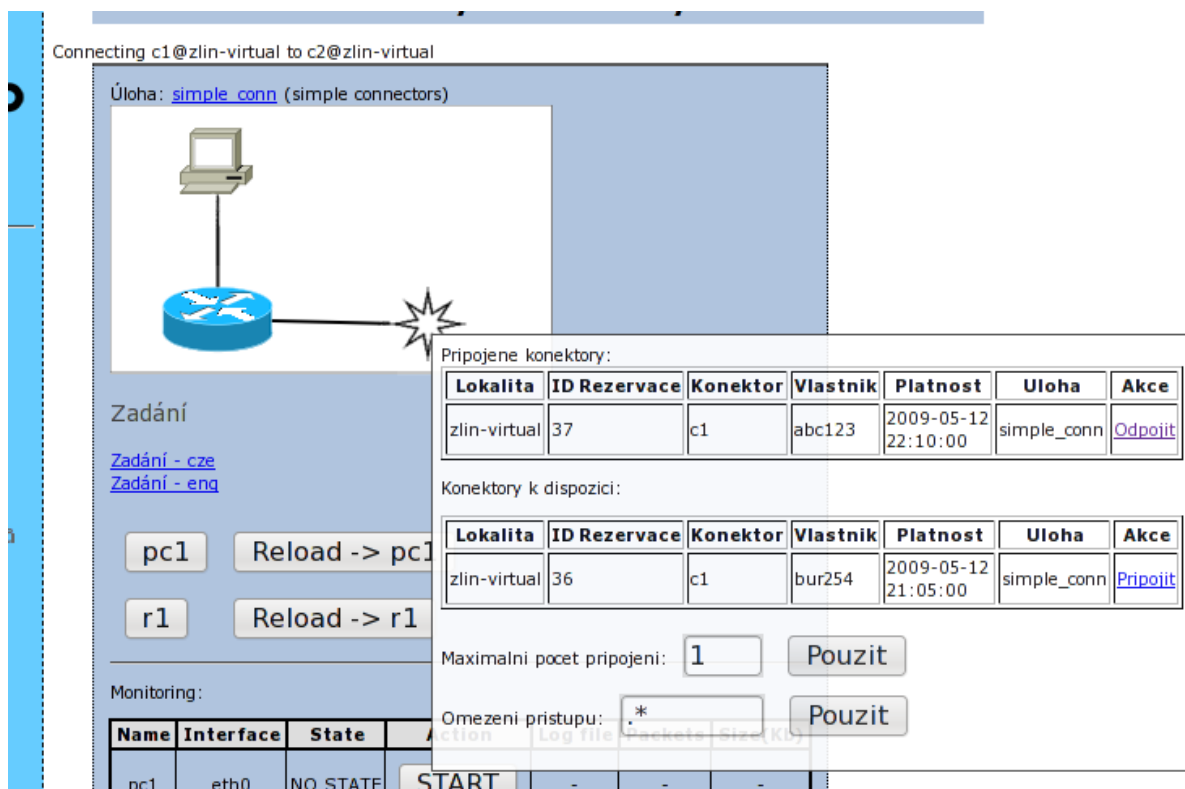
Poté se prvek s tímto symbolem zavede do webového rozhraní, aby byl správně interpretován jako prvek typu `connector`. Tím se zajistí, že kliknutí na konektor vyvolá jeho vlastní kontextovou nabídku.

Tyto kroky jsou podrobně popsány v části `Implementace`.

### 4.4.3 Seznamy konektorů

Původním návrhem pro grafické uživatelské rozhraní konektorů bylo využít kontextovou nabídku v klikacím bitmapovém obrázku topologie, který vytvořil Jan Rudovský v rámci své diplomové práce [4]. Prvek konektor by však měl své vlastní speciální menu, viz. obr. 8. To by obsahovalo:

- seznam připojených konektorů s možností odpojení
- seznam konektorů k dispozici s možností připojení
- nastavení konektoru



Obrázek 8: Kontextové menu konektoru

Vzhledem k tomu, že v budoucnu budou tabulky s konektory obsahovat desítky až stovky dostupných konektorů, je tato varianta nevhodná, protože menu by bylo příliš velké.

Zvolil jsem proto jinou variantu. Podobně jako se po kliknutí na laboratorní prvek (směrovač, přepínač) otevře samostatné okno s konzolí, tak se při kliknutí na prvek konektoru otevře samostatná

stránka obsahující výše zmíněné 3 části (dva seznamy konektorů a nastavení). Její snímek obrazovky (obr. 18) je uveden v kap. 6.

#### 4.4.4 Notifikace o připojení konektoru

Uživatel, který právě pracuje na úloze obsahující konektory, by měl být upozorněn v případě, že se na jeho konektor někdo připojí.

Virtlab již obsahuje svůj vlastní systém pro posílání elektronických zpráv mezi jeho uživateli. Ten byl vytvořen J. Vavříčkem v jeho diplomové práci [5]. Oznámení o nové poště se uživatelům okamžitě zobrazí v záhlaví Virtlab GUI, viz. obr. 9.



Obrázek 9: Upozornění na nepřečtenou poštu

Použití pošty k notifikaci je dále popsáno v kap. 5.4.7.

## 5. Implementace

### 5.1 Úvod

V této kapitole bude následovat popis implementace na základě analýzy z kap. 4. Skládá se z částí:

- **Logické vytvoření nového prvku** – zavedení konektoru do validačních a konfiguračních souborů
- **Implementace konektorů do tunelovacího serveru** – vytvoření nového modulu pro práci s konektory
- **Implementace konektorů do webového rozhraní** – vytvoření funkcí pro ovládání konektorů uživatelem
- **Zdrojové soubory** – popis vytvořených/upravených souborů

### 5.2 Logické vytvoření nového prvku – connector

#### 5.2.1 RELAX NG schémata

Tato schémata slouží k validaci XML souborů. Ve Virlabu se používají k popisu seznamu zařízení, která jsou k dispozici, a dále k popisu požadovaných topologií. Jelikož potřebuji vytvořit nový prvek typu konektor, musím jej nejdříve přidat do schématu pro popis zařízení:

`/web/relax/equipment.rng:`

```
...
<element name="equipment">

  <zeroOrMore>
    <element name="device">
      <attribute name="type">
        <choice>
          ...
          <value>connector</value>
        </choice>
      </attribute>
    ...
  ...

```

Podobným způsobem jsem přidal konektor i do schématu `/web/relax/topology.rng`, který slouží k validaci XML popisujícího požadovanou topologii.

#### 5.2.2 XML soubor s vybavením lokality

Dále se musí konektory přidat do vybavení Virlabu v souboru `/etc/virtlab/vybaveni.xml` (konfigurační soubor rezervačního serveru), aby je uživatelé mohli používat. Měli bychom si však uvědomit, že konektory jsou pouze virtuální prvky, tzn. neexistují fyzicky jako například směrovače a přepínače. A proto jich lze přidat libovolné množství.

### 5.2.3 Rezervační server

Existující konektory se ještě musí přidat do konfigurace rezervačního serveru v souboru `/etc/virtlab/rsv-server.conf`, aby si je uživatelé mohli rezervovat jako ostatní laboratorní prvky.

### 5.2.4 Generátor konfigurací

Obsah souborů s vybavením a konfigurací rezervačního serveru jsem podrobně nepopisoval, protože pro práci s nimi existuje webové GUI – Generátor konfigurací. Prvek typu konektor je již zaveden i do něj a nyní lze generovat konfigurační soubory s konektory v sekci Vybavení – Přidat nové zařízení, viz. obr. 10:

Obrázek 10: Generátor konfigurací

Z formuláře na obr. 10 jsou vynechány údaje, které se konektorů netýkají (verze OS, RAM, FLASH, ...). Pomocí nové funkce „klonování“ lze přidat větší množství konektorů současně.

Dalším krokem je konfigurace rezervačního serveru. Je potřeba povolit rezervaci nových laboratorních prvků v sekci `TimeTable`.

Vygenerované konfigurační soubory se poté pomocí skriptu nahrají na příslušnou lokalitu<sup>5</sup>.

## 5.3 Implementace konektorů do tunelovacího serveru

### 5.3.1 Modul `Port_connector`

Podle UML Activity diagramu z analýzy (kap. 4.2.3) jsem implementoval nový modul tunelovacího serveru – `Port_connector`. Ten jsem zaregistroval tak, aby přijímal veškeré rámce s cílovým rozhraním konektoru (`eint`, `sint`, `iint`).

### 5.3.2 Hledání cílových konektorů v `Redirect table`

Součástí algoritmu modulu `Port_connector` je hledání připojených konektorů v `redirect table`. Implementoval jsem tedy metodu `get_destination_connectors` do třídy

<sup>5</sup> Podrobnou dokumentaci ke Generátoru konfigurací naleznete zde [1]



`Redirect_table`. Ta přijímá parametr „zdrojový konektor“ a vrací seznam všech konektorů, které jsou k němu připojené.

### 5.3.3 Problém jednoznačnosti názvu rozhraní

Tunelovací server byl navržen s předpokladem, že zdrojové resp. cílové rozhraní bude v `redirect table` vždy nejvýše jednou. To znamená, že ke konkrétnímu rozhraní nelze připojit více než jedno další (vytváří se pouze dvou-bodové spoje). Takto vypadá část `redirect table` obsahující propojení konektorů C1 a C2:

Lokální konektor	Vzdálený konektor
<code>c1@zlin:iint</code>	<code>c2@zlin:iint</code>

**Tabulka 1: Redirect table 1**

K jednomu konektoru by však mělo jít připojit libovolný počet dalších stejného typu (Ethernet, Serial). To nyní nelze, protože pokud se do této tabulky přidá další řádek propojující konektory C1 a C3, bude přepsán záznam propojení C1 a C2, jelikož zdroj `c1@zlin:iint` již existuje.

Tento problém jsem vyřešil doplněním názvu virtuálního rozhraní `iint` o identifikátor připojovaného konektoru (část za mřížkou):

Lokální konektor	Vzdálený konektor
<code>c1@zlin:iint#c2@zlin:eint</code>	<code>c2@zlin:iint#c1@zlin:eint</code>

**Tabulka 2: Redirect table 2**

Tímto jsem vytvořil jednoznačný název virtuálního rozhraní `iint` pro každé připojení k C1. Například pro připojení konektoru C3 k C1 by byl vytvořen záznam se zdrojovým rozhraním `c1@zlin:iint#c3@zlin:iint`, který již lze vložit do `redirect table`:

Lokální konektor	Vzdálený konektor
<code>c1@zlin:iint#c2@zlin:eint</code>	<code>c2@zlin:iint#c1@zlin:eint</code>
<code>c1@zlin:iint#c3@zlin:eint</code>	<code>c3@zlin:iint#c1@zlin:eint</code>

**Tabulka 3: Redirect table 3**

Identifikátor za mřížkou je však pouze umělé rozšíření – na funkci tunelovacího serveru nemá vliv a je ignorován.

## 5.4 Implementace konektorů do řídicí aplikace

### 5.4.1 Zavedení prvku konektor do konverzního algoritmu

Konverzní algoritmus řídicí aplikace má za úkol převádět prvky z DIA souboru na laboratorní prvky Virtlabu. Využívá k tomu pole `$virtlab_objects` ze třídy `XMLTransform`. Zde jsem přidal konektory:

```
private $virtlab_objects = array(
    ...
    'Virtlab - Connector' => 'connector'
```

Po této úpravě mohu provést převod DIA souboru na XML soubor s popisem topologie. K tomu lze využít nástroj „Transformace logické topologie“, který je mezi Podpůrnými nástroji v hlavním menu.

#### 5.4.2 Rezervace úlohy s konektory

Uživatel, který chce rezervovat úlohu, si ji vybere ze seznamu. Poté se objeví formulář pro výběr časového období. Tento formulář jsem rozšířil o výchozí nastavení konektorů, viz. obr. 11.

V této chvíli již znám rezervovanou úlohu (její ID), takže jsem schopen zjistit použité typy prvků. Toho docílím SQL dotazem do tabulky `files`, která obsahuje XML soubory popisující topologie všech úloh v systému. Rozšíření GUI o nastavení konektorů se tedy objeví pouze v případě, že v úloze vybrané k rezervaci je detekován alespoň jeden konektor.

Obrázek 11: Rozšířené vytvoření nové rezervace

Nastavení při rezervaci je použito jako výchozí pro všechny konektory v dané úloze. Jednotlivé konektory pak uživatel může nastavovat nezávisle až v aktivní úloze. Tyto možnosti včetně popisu nastavení jsou uvedeny níže.

#### 5.4.3 Dotazy na dostupné konektory ze všech lokalit

Lokality Virlabů spolu komunikují pomocí SOAP protokolu. V SOAP serveru ve všech lokalitách jsem implementoval funkci `get_connectors()`. Ta pomocí SQL dotazu do tabulek `reservations`, `reserved_devices` a `connectors` vrací pole všech konektorů, které jsou v dané chvíli k dispozici – tzn. úlohy obsahující tyto konektory jsou právě aktivní. Toto pole obsahuje pro každý konektor i jeho vlastnosti:

- název konektoru – fyzický (`device`)
- název konektoru – logický (`vertex`)
- ID rezervace (`resID`)
- ID úlohy (`task_id`)

- název úlohy (`name_short`)
- vlastník (`creator`)
- doba platnosti (`to`)
- přístupová práva (`access`)
- max. počet připojení (`max_connections`)
- výsledek mapovacího algoritmu (`mapping_result`)

Fyzický název je skutečné jméno zařízení, které je uvedeno v seznamu vybavení lokality (tzn. je jedinečné v rámci lokality). Logický název vznikne až po rezervaci úlohy a je určen mapovacím algoritmem (je jedinečný pouze v rámci jedné rezervace). Běžný uživatel Virlabu se setkává pouze s logickými názvy.

Každý konektor lze jednoznačně určit pomocí ID rezervace a fyzického názvu konektoru. Přístupová práva a max. počet připojení slouží k nastavení konektoru. Výsledek mapovacího algoritmu využívám pro zjištění typu rozhraní konektoru (Ethernet, Serial). Ostatní vlastnosti jsou pouze informačního charakteru – aby uživateli usnadnily výběr konektoru, k němuž se chce připojit.

V případě, že uživatel potřebuje procházet konektory, ke kterým by mohl připojit svůj vlastní, stačí se pomocí SOAP rozhraní dotázat své i ostatních lokalit na jejich seznam.

#### 5.4.4 Nastavení konektoru

Pro uchování nastavení konektoru byla vytvořena nová SQL tabulka `connectors`. Ta obsahuje na každém řádku jeden konektor určený jednoznačně svým fyzickým názvem (`device`) a ID rezervace (`resID`). Nastavení konektoru je pak ve sloupcích:

- `access` (přístupová práva)
- `max_connections` (maximální počet připojení)

Přístupová práva se ukládají jako řetězec ve formátu regulárního výrazu. Maximální počet připojení je celé kladné číslo (hodnota NULL znamená neomezený počet).

Data z této tabulky jsou získávána SOAP rozhráním popsaným výše.

#### 5.4.5 Filtr na dostupné konektory

Funkce `get_connectors` ze SOAP serveru vrací všechny právě aktivní konektory.

V grafickém rozhraní jsem dále implementoval funkci, která na tento seznam aplikuje filtr. Ten ponechá pouze ty vzdálené konektory, které splňují všechny následující požadavky:

- konektory nejsou totožné (konektor se nemůže připojit sám k sobě)
- stejný typ rozhraní jako lokální konektor (Ethernet, Serial)
- vzdálený konektor ještě nevyčerpal max. počet připojení
- přihlášený uživatel a název jeho lokality odpovídají regulárnímu výrazu přístupových práv ke konektoru

#### 5.4.6 Funkce připojit/odpojit konektor

Poté, co si uživatel ze seznamu konektorů jeden vybere, může kliknout na jeho odkaz `Připojit`, viz. obr. 12.

**Připojené konektory:**  
Tento konektor není momentálně nikam připojen.

**Konektory k dispozici:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
most-virtual	17	c1	abc123	2009-12-30 15:39:00	simple_conn	<a href="#">Připojit</a>
most-virtual	18	c1	bur254	2009-12-30 15:05:00	simple_conn	<a href="#">Připojit</a>

**Nastavení konektoru:**

Maximální počet připojení:

Omezení přístupu:  (Regulární výraz na uživ. jméno)

Obrázek 12: Odkaz pro připojení konektoru

Tímto se provedou akce v následujícím pořadí:

- zjistím IP adresy tunelovacích serverů ze všech lokalit
- zjistím čas konce obou rezervací (každý konektor má svou) a kratší zvolím jako TTL parametr příkazu `redir`
- do příslušných tunelovacích serverů (jimž lokální/vzdálený konektor patří) založím TCP spojení na portu 40001 a odešlu jim `redir` příkaz, kterým zapíšu záznam do jejich `redirect table`, např.:

```
redir c1@zlin:iint#c2@most:eint c2@most:iint#c1@zlin:eint \
2009-12-12 19:58:00
```

Po připojení se vzdálený konektor objeví v druhé tabulce – Připojené konektory, viz. obr. 13.

**Připojené konektory:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
most-virtual	17	c1	abc123	2009-12-30 15:39:00	simple_conn	<a href="#">Odpojit</a>

**Konektory k dispozici:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
most-virtual	18	c1	bur254	2009-12-30 15:05:00	simple_conn	<a href="#">Připojit</a>

**Nastavení konektoru:**

Maximální počet připojení:

Omezení přístupu:  (Regulární výraz na uživ. jméno)

Obrázek 13: Odkaz pro odpojení konektoru

Pokud nyní uživatel klikne na odkaz `Odpojit`, pak se provedou stejné akce jako při připojení s tím rozdílem, že do tunelovacích serverů se nyní pošle příkaz `noredir`, např.:

```
noredir c1@zlin:iint#c2@most:eint
```

### 5.4.7 Notifikace o připojení konektoru

V analýze jsem se rozhodl použít k notifikaci systém elektronických Virlab zpráv. Ty jsou založeny na SQL tabulce `mail`. Každá zpráva představuje jeden řádek v této tabulce. Obsahuje následující sloupce:

ID, čas, odesílatel, příjemce, předmět, tělo zprávy, příznak stavu (ne/přečtená)

Pomocí SQL dotazu tedy mohu odeslat zprávu. Problém však nastane při odesílání zprávy do jiné lokality, protože každá lokalita má svou vlastní SQL databázi. Využil jsem proto opět SOAP rozhraní, kde jsem implementoval funkci `send_notification` s parametry „příjemce, předmět, tělo zprávy“.

Po připojení vzdáleného konektoru zjistím podle názvu vzdálené lokality adresu příslušného SOAP serveru a zavolám tuto funkci. Odesílatelem zprávy je uživatel `system`, příjemcem je vlastník vzdáleného konektoru a text zprávy je na obr. 14.



Obrázek 14: Notifikace o připojení konektoru

Vlastníkovi vzdáleného konektoru tedy bylo oznámeno, že uživatel `abc123` se připojil k jeho konektoru `c2@most-virtual`. Z hlavičky Datum odeslání zná i čas připojení.

Funkci `send_notification` jsem navrhnul tak, aby byla obecně použitelná i pro případné další budoucí notifikace systému Virlab.

### 5.4.8 Jazykové mutace

Virlab podporuje více jazyků. V celém GUI jsem tedy pro výpisy uživateli nepoužíval texty přímo ve zdrojovém kódu, ale přidával jsem české a současně anglické řetězce do třídy `virtlabLanguage`:

```
case 'conn-settings': return 'Nastavení konektoru:';
```

Na texty jsem se pak odkazoval pomocí proměnné `$_lang` a jednoznačného klíče, např.:

```
print $_lang->Text('conn-settings') . "<br />\n";
```

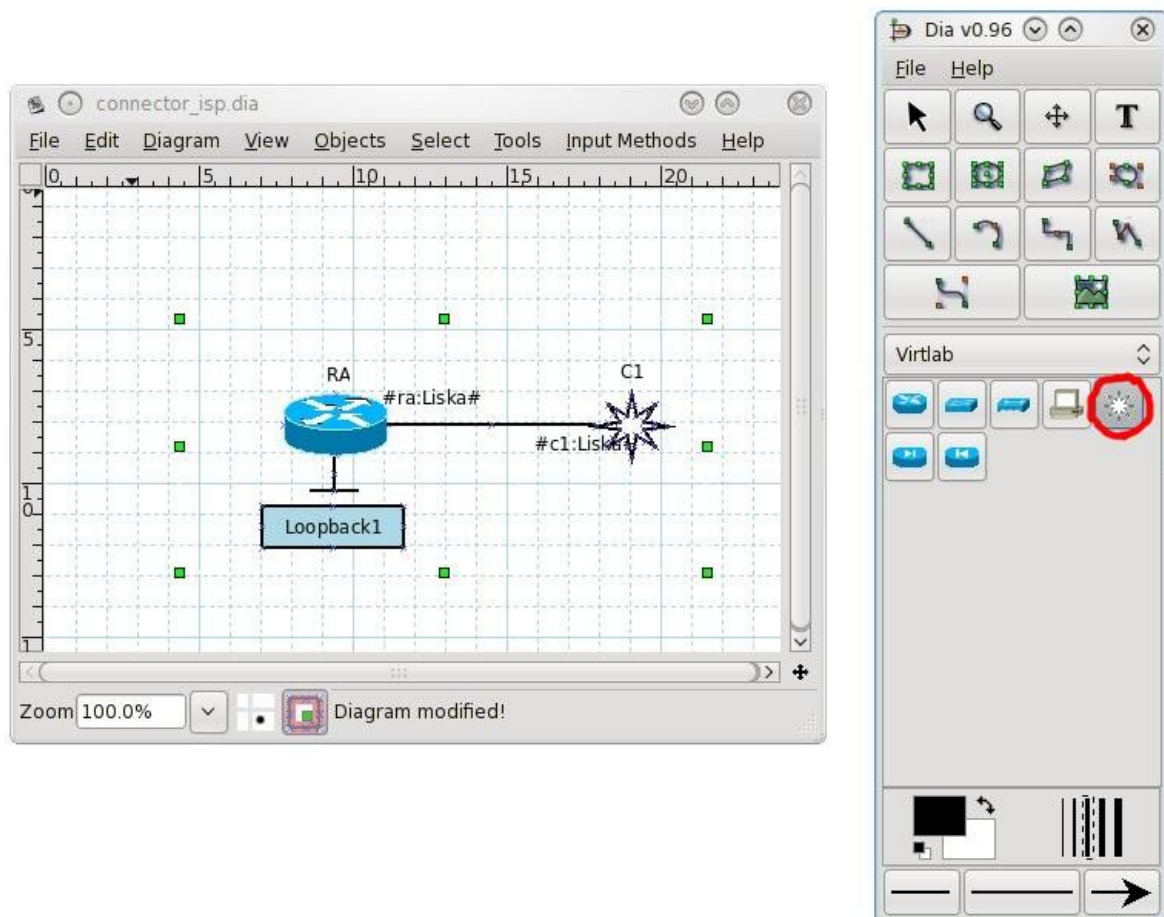
## 5.5 Integrace konektorů do aplikace VirlabDia

Každá úloha obsahuje DIA soubor, který slouží jako zdroj pro vygenerování obrázku topologie a současně jako XML popis zapojení prvků. Každý DIA soubor je tedy možné převést na XML obsahující textový popis použitých prvků a jejich zapojení včetně všech atributů.

Software VirlabDia<sup>6</sup> nabízí grafické uživatelské rozhraní k návrhu těchto souborů. Obsahuje speciální paletu symbolů Virlab s prvky *router*, *switch*, *hub*, *PC*, *firewall*. Do této palety jsem přidal nový prvek – *connector*:

1. Stáhnul jsem z SVN poslední verzi zdrojových souborů VirlabDia.
2. Do adresáře `shape/Virlab` jsem přidal soubor `connector.shape` s popisem konektoru a `connector.png` představující obrázek konektoru.
3. V adresáři `sheets` jsem do palety symbolů Virlab přidal prvek `connector`.
4. Zkompiloval jsem zdrojové soubory pro Linux (DEB i RPM balíčky) a pro Windows.

Po instalaci a spuštění VirlabDia lze do topologie přidávat konektory:



Obrázek 15: Software VirlabDia

<sup>6</sup> Podrobnější informace o software VirlabDia a jeho úpravách naleznete v diplomové práci J. Rudovského [4].

V příkladu na obr. 15 jsem nastavil jméno směrovače, konektoru i spojovací linky. Na konce linky jsem přidal také popisky, které se po aktivaci úlohy automaticky přepíšou na správné názvy namapovaných rozhraní.

## 5.6 Zdrojové soubory

Následuje seznam vytvořených a upravených zdrojových souborů (adresáře odpovídají struktuře v SVN Virlabů) a stručného popisu jejich úpravy nebo účelu:

Název souboru	Popis úpravy / účelu
/crossconnect_v3/src/Frame.cpp	Přidány metody pro získání a nastavení zdrojového rozhraní rámce.
/crossconnect_v3/src/Frame.h	Přidány hlavičky nových metod.
/crossconnect_v3/src/Observer_port_setter.cpp	Přidáno eint rozhraní do seznamu pro aktivaci portů.
/crossconnect_v3/src/Port_connector.cpp	Modul tunelovacího serveru pro obsluhu rozhraní konektorů.
/crossconnect_v3/src/Port_connector.h	Hlavičkový soubor modulu tunelovacího serveru.
/crossconnect_v3/src/Redirect_table.cpp	Přidána pomocná metoda pro získání dostupných konektorů.
/crossconnect_v3/src/Redirect_table.h	Přidána hlavička nové metody.
/crossconnect_v3/src/Run.cpp	Vytvoření instance třídy modulu konektorů.
/crossconnect_v3/src/Run.h	Vložení ukazatele na instanci třídy modulu konektorů.
/crossconnect_v3/Makefile	Přidána kompilace zdrojových souborů modulu konektorů.

**Tabulka 4: Zdrojové soubory tunelovacího serveru**

Název souboru	Popis úpravy / účelu
/web/class/XMLTransform.php.inc	Vložení nového záznamu pro mapování konektorů z DIA do XML.
/web/class/virtlabLanguage.php.inc	Vložení nových textových řetězců v českém a anglickém jazyce pro výpisy v GUI.
/web/class/virtlabWeb.php.inc	Vložení nového záznamu pro začlenění webové stránky s nastavením konektorů do menu Virlabů.
/web/virtlab/connector.php	Nová webová stránka sloužící k zobrazení konektorů a práci s nimi.
/web/virtlab/connector-functions.php	Podpůrné funkce pro práci s konektory v GUI.
/web/virtlab/reser-active.php	Úprava stránky zobrazující aktivní úlohu. Byl přidán nový typ prvku tak, aby se pro konektory generovalo vlastní menu a tlačítka vyvolávající jejich nastavení.
/web/virtlab/reser-new.php	Přidána detekce konektorů v úloze k rezervaci. Při jejich nálezů je zobrazena možnost výchozího nastavení konektorů již při rezervaci.
/web/index.php	Vložení záznamu s mapováním souboru connector.php na stránku s nastavením konektorů.
/web/soapserver.php	Přidány funkce pro získání seznamu dostupných konektorů a odeslání systémové notifikace uživateli.

**Tabulka 5: Zdrojové soubory webového rozhraní**

<b>Název souboru</b>	<b>Popis úpravy / účelu</b>
/dia/src/shapes/Virtlab/connector.jpg	Grafický symbol (obrázek) konektoru.
/dia/src/shapes/Virtlab/connector.shape	Textový popis symbolu konektoru.
/dia/src/shapes/Virtlab/Makefile.am	Vložení symbolu konektoru pro následnou kompilaci.
/dia/src/sheets/Virtlab.sheet.in	Přidání symbolu konektoru do palety Virtlab.

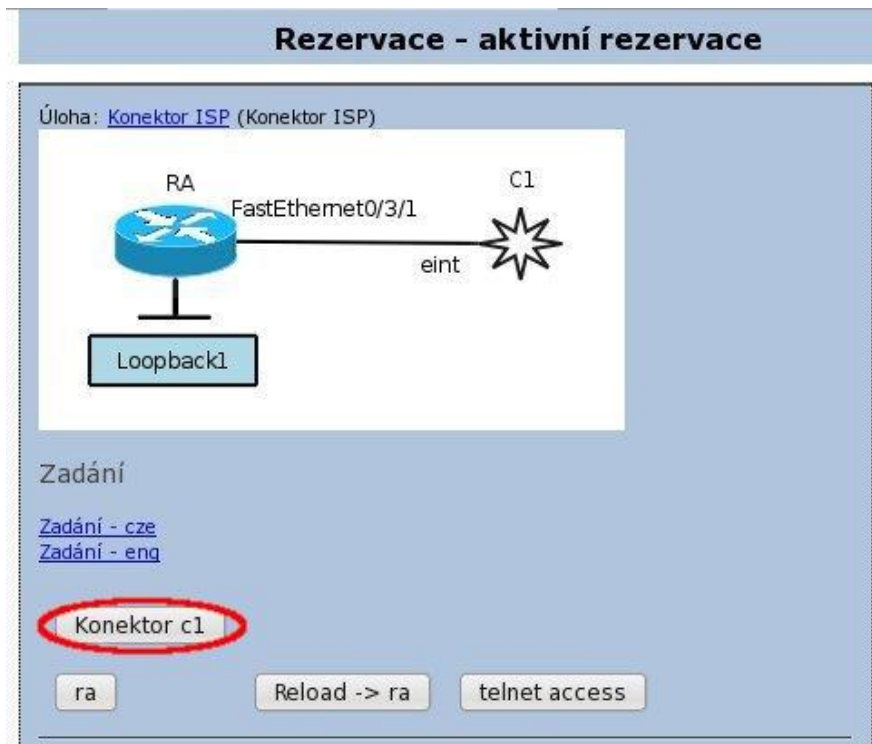
**Tabulka 6: Zdrojové soubory VirtlabDia**



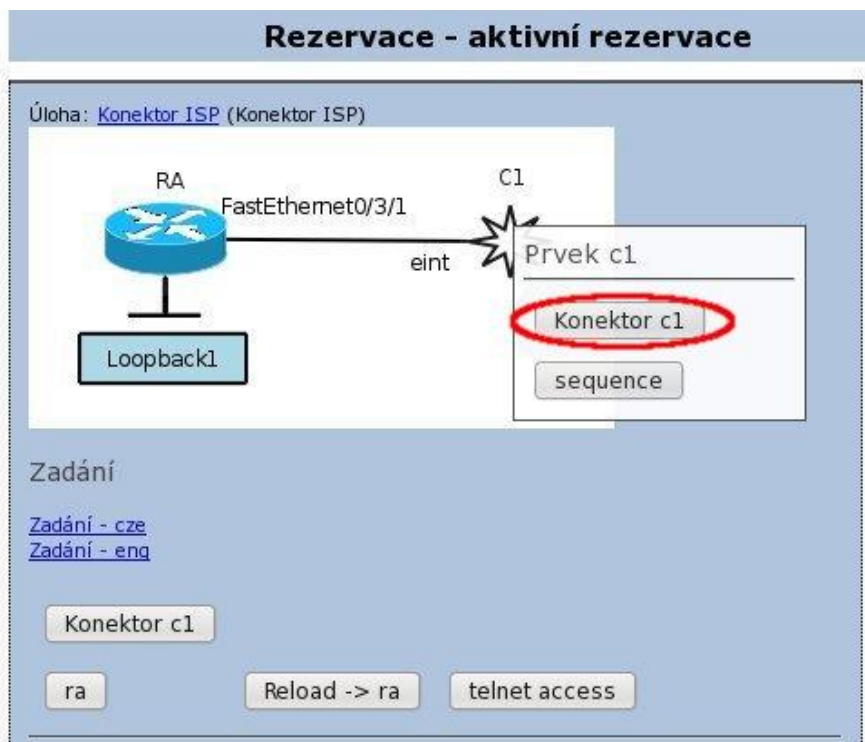
## 6. Práce s konektorem v GUI

S konektorem může uživatel pracovat přímo v aktivní úloze (menu Rezervace – Aktivní rezervace) dvěma způsoby:

- kliknutím na tlačítko `Konektor cX`, kde X je číslo konektoru v dané úloze, viz. obr. 16
- vyvoláním kontextového menu konektoru přímo v obrázku topologie, viz. obr. 17




Obrázek 16: Tlačítko pro práci s konektorem



Obrázek 17: Kontextové menu pro práci s konektorem

Po kliknutí se zobrazí stránka (obr. 18) se seznamy vzdálených konektorů, ke kterým se vybraný (lokální) může připojit, popřípadě ke kterým již je připojen. Dále je zde nastavení (maximální počet připojení a přístupová práva) vybraného konektoru.



**ZLIN-VIRTUAL**  
15:00:53

**Hlavní**

- hlavní stránka
- osobní údaje
- moje poznámky
- pošta

**Úlohy**

- školní úlohy
- editace
- nová úloha
- seznam

### Nastavení konektoru

**Připojené konektory:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
most-virtual	17	c1	abc123	2009-12-30 15:39:00	simple_conn	<a href="#">Odpojit</a>

**Konektory k dispozici:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
most-virtual	18	c1	bur254	2009-12-30 15:05:00	simple_conn	<a href="#">Připojit</a>

**Nastavení konektoru:**

Maximální počet připojení:

Omezení přístupu:  (Regulární výraz na uživ. jméno)

Obrázek 18: GUI pro práci s konektorem

## 7. Testování

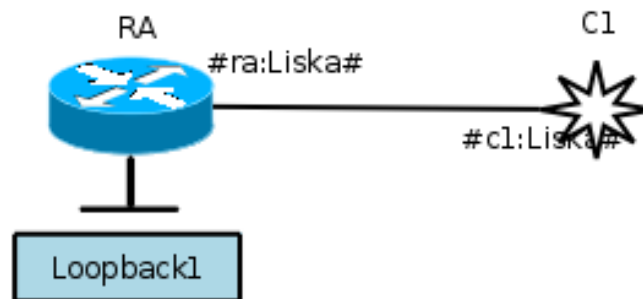
### 7.1 Příprava

Vývoj konektorů probíhal na virtuálních lokalitách ZLIN a MOST<sup>7</sup>. Potřeboval jsem obě současně k otestování propojení konektorů mezi různými lokalitami. Po odladění jsem řešení nasadil na lokalitu VSB-ADVANCED, která již nepracuje s virtualizovanými prvky, ale se skutečnými.

Vytvořil jsem dvě úlohy s konektory v kategorii Experimentální:

1. **Konektor ISP** (obr. 19)

Směrovač RA s nakonfigurovaným loopback rozhraním, který simuluje poskytovatele Internetu. FastEthernet rozhraním je směrovač připojen ke konektoru C1.

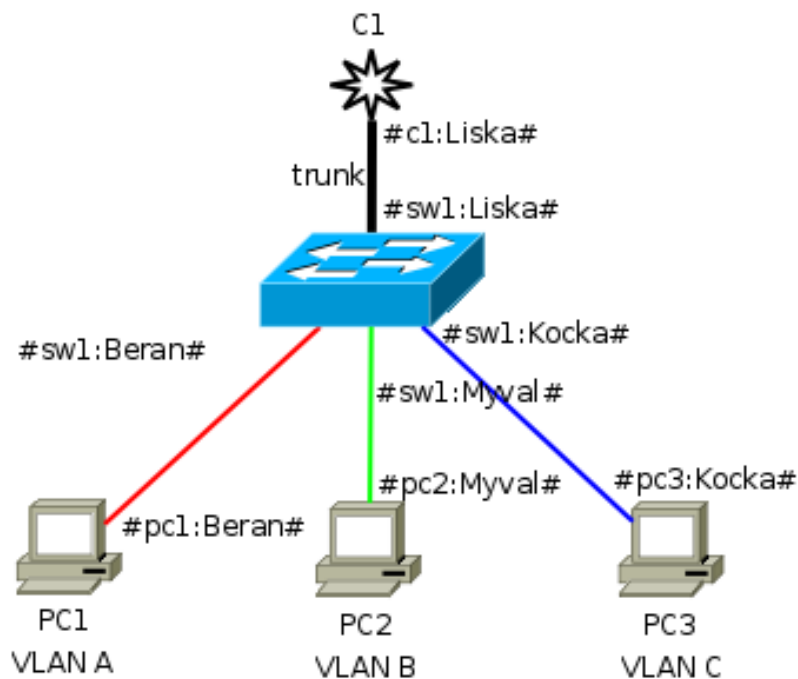


Obrázek 19: DIA obrázek úlohy Konektor ISP

2. **Konektor VLAN** (obr. 20)

Přepínač SW1 se třemi PC v různých VLAN. Konektor na trunk rozhraní je určen k připojení přepínače ke směrovači (router-on-the-stick) nebo k jinému přepínači.

<sup>7</sup> Virtuální lokality obsahují pouze virtualizované prvky a jsou určeny k vývoji a testování Virlabu



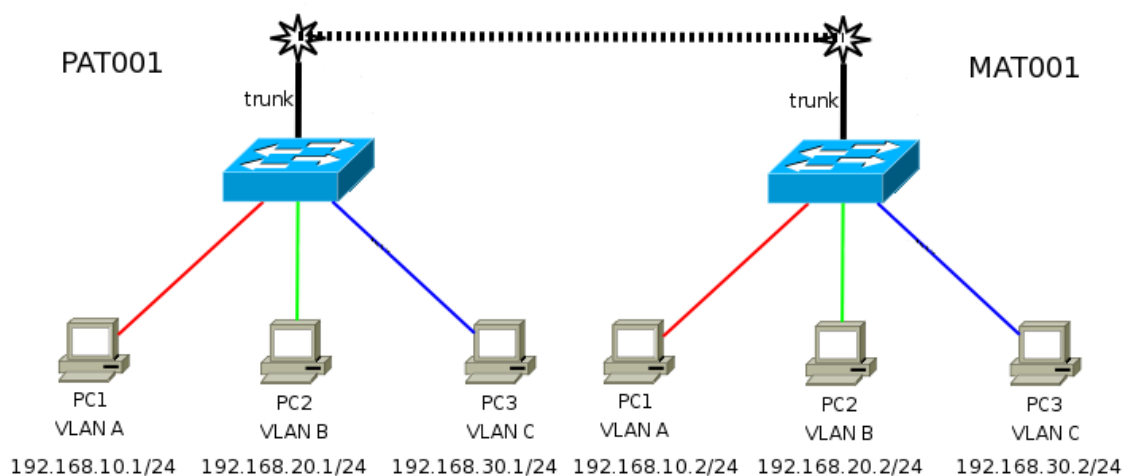
Obrázek 20: DIA obrázek úlohy Konektor VLAN

Dále jsem vytvořil dva nové uživatele, kteří budou úlohy rezervovat:

1. Pat - pat001
2. Mat - mat001

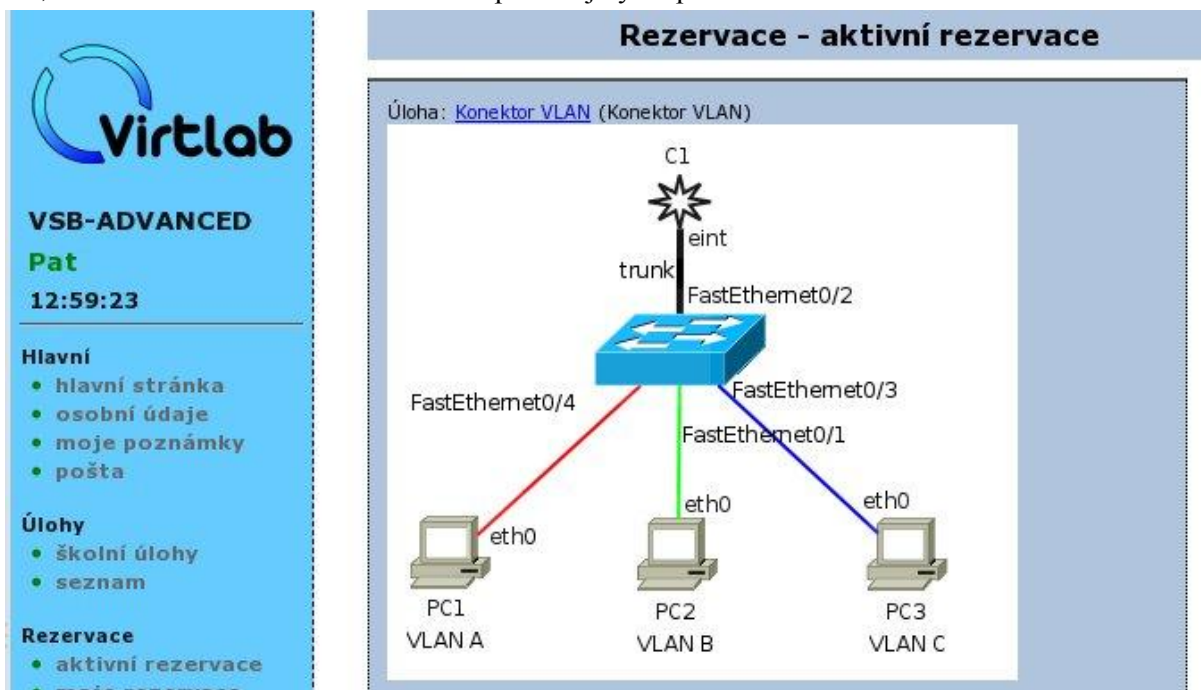
## 7.2 Propojení trunk linky mezi dvěma přepínači

Uživatelé Pat i Mat si zarezervovali svou úlohu Konektor VLAN. Cílem je nakonfigurovat a propojit dvě úlohy pomocí konektorů tak, aby počítače ve stejných VLAN měly mezi sebou konektivitu, viz. obr. 21.



Obrázek 21: Dvě úlohy propojené konektory

Po aktivaci úlohy se v DIA obrázku přepíšou názvy rozhraní. Na obr. 22 je úloha uživatele Pat, uživatel Mat může mít rozhraní namapována jiným způsobem.



Obrázek 22: Aktivní rezervace Konektor VLAN uživatele Pat

Po aktivaci úlohy jsem u uživatele Pat nakonfiguroval na přepínači VTP server, trunk linku a přiřazení VLAN na odpovídající rozhraní. Na PC stanicích jsem nastavil IP adresy podle schématu na obr. 21.

V úloze uživatele Mat jsem provedl podobné kroky s tím rozdílem, že jeho přepínač je v roli VTP klienta a rozhraní mohou tedy do VLANů přiřadit až po domluvě VTP protokolu.

Dále jsem si nechal zobrazit seznam konektorů k dispozici u uživatele Pat, viz. obr. 23.

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
vsb-advanced	35	c1	mat001	2010-01-24 13:40:00	Konektor VLAN	<a href="#">Připojit</a>

Obrázek 23: Seznam konektorů k dispozici u uživatele Pat

Objevil se pouze jeden konektor patřící Matovi. Kliknutím na odkaz Připojit se konektor přesunul do Připojených konektorů a uživateli Mat přišla zpráva, viz. obr. 24.



Obrázek 24: Notifikace o připojení ke konektoru uživatele Mat

Od této chvíle je aktivní trunk linka mezi přepínači a vyjedná se synchronizace VLAN databáze pomocí VTP protokolu.

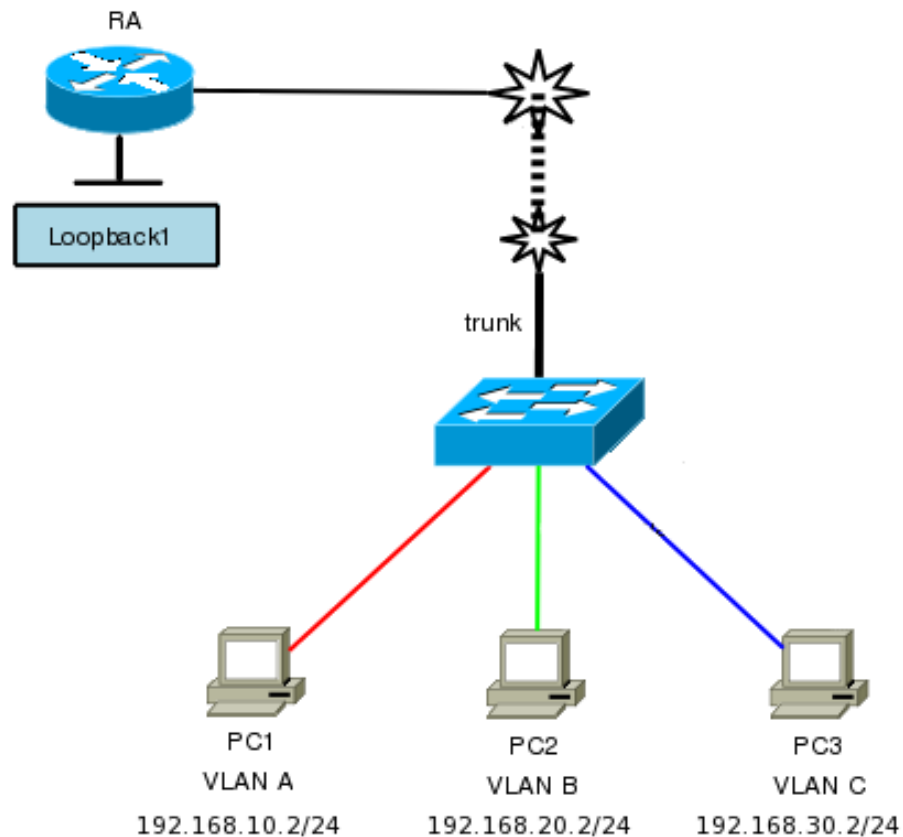
Pomocí příkazu ping jsem ověřil konektivitu v rámci jednotlivých VLAN.

### 7.3 Propojení směrovače (router-on-the-stick) a přepínače

Jelikož konektory byly již řádně otestovány také na lokalitě se skutečnými prvky (VSB-ADVANCED), bylo rozhodnuto nasadit je na produkční lokalitu VSB-BASIC, kterou používají studenti počítačových sítí. Provedl jsem potřebné úpravy a vytvořil na VSB-BASIC stejné úlohy, jaké jsou uvedeny v části 7.1. Pouze uživatelské účty jsem využil již existující:

1. bur254 – Pavel Burda
2. djendrys – Dagmar Jendryščíková

Úlohu „Konektor ISP“ si zarezervoval bur254 a „Konektor VLAN“ uživatelka djendrys. Cílem je nakonfigurovat tzv. „router-on-the-stick“ neboli směrovač, který bude zajišťovat inter-vlan routing. Konektivitu poté budou mít mezi sebou i počítače z různých VLAN. Výsledná topologie s propojenými konektory je na obr. 25.



Obrázek 25: Router-on-the-stick s konektory

Po aktivaci úloh `bur254` nakonfiguroval na směrovači RA subinterface pro každou VLAN s IP adresami:

VLAN ID	Rozhraní (subinterface)	IP adresa
VLAN A	FastEthernet 0/1.10	192.168.10.1/24
VLAN B	FastEthernet 0/1.20	192.168.20.1/24
VLAN C	FastEthernet 0/1.30	192.168.30.1/24

Tabulka 7: Adresace rozhraní pro router-on-the-stick

Tímto jsem naplnil směrovací tabulku potřebnými záznamy, viz. obr. 26.

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route


Gateway of last resort is not set

C    192.168.30.0/24 is directly connected, FastEthernet0/1.30
C    192.168.10.0/24 is directly connected, FastEthernet0/1.10
C    192.168.20.0/24 is directly connected, FastEthernet0/1.20
R1#
```

Obrázek 26: Směrovací tabulka router-on-the-stick

Uživatelka djendrys nakonfigurovala VLAN na přepínači a nastavila IP adresy pro počítače dle topologie na obr. 25 a současně také odpovídající výchozí brány.

V této chvíli se již může připojit konektor, viz. obr. 27.



**VSB-BASIC**  
Pavel Burda  
12:32:37

**Hlavní**

- hlavní stránka
- osobní údaje
- moje poznámky
- pošta

**Uživatelé**

- Aktivní uživatelé
- procházení / editace
- nový uživatel
- vymazání uživatelů
- skupiny uživatelů

**Úlohy**

- školní úlohy
- editace

### Nastavení konektoru

**Připojené konektory:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
vsb-basic	3672	c1	djendrys	2010-02-04 12:23:00	Konektor VLAN	<a href="#">Odpojit</a>

**Konektory k dispozici:**

Lokalita	ID rezervace	Konektor	Vlastník	Platnost	Úloha	Akce
vsb-basic	3664	c1	bur254	2010-02-04 11:34:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3665	c1	bur254	2010-02-04 11:37:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3666	c1	bur254	2010-02-04 11:42:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3667	c1	bur254	2010-02-04 11:57:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3668	c1	bur254	2010-02-04 12:03:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3669	c1	bur254	2010-02-04 12:11:00	Konektor ISP	<a href="#">Připojit</a>
vsb-basic	3671	c1	bur254	2010-02-04 12:18:00	Konektor ISP	<a href="#">Připojit</a>

**Nastavení konektoru:**

Maximální počet připojení:

Omezení přístupu:  (Regulární výraz na uživ. jméno)

**Obrázek 27: Router-on-the-stick nastavení konektoru**

Ihned poté je uživatelce djendrys odeslána notifikace a aktivuje se trunk linka. Konektivitu mezi VLAN jsem otestoval příkazem ping mezi všemi počítači.



## 8. Závěr

V této práci jsem navrhnul a implementoval do virtuální síťové laboratoře nový prvek konektor, který slouží k dynamickému propojování úloh podle okamžitých požadavků uživatelů.

Největší část spočívala v implementaci nového modulu do tunelovacího serveru a rozšíření webového GUI Virlabu. Dále je součástí práce rozšíření RNG schémat, nové funkce SOAP rozhraní a rozšíření aplikace VirlabDia.

V rámci práce jsem vytvořil několik nových úloh využívajících konektory. Pomocí těchto úloh byla funkce konektorů řádně otestována a to i na produkčních lokalitách.

Konektorům jsem implementoval podporu sériových rozhraní, avšak tato vlastnost nemohla být otestována, protože v současné době provoz mezi WAN rozhraními neprochází tunelovacím serverem. Nicméně konektory jsou již připraveny na dobu, kdy budou vytvořeny Ethernet-RS232 převodníky a WAN provoz bude tunelován.

Konektory jsem se snažil navrhnout tak, aby bylo možné s jejich pomocí k virtuální síťové laboratoři připojit i externí systémy. V současnosti se jedná o PacketTracer [6] a vzdálené PC [7].

Diplomová práce splnila všechny požadavky, které na ni byly kladeny v zadání. Konektory jsou již dnes nasazeny v produkčním prostředí na všech lokalitách. Studenti nyní mohou své úlohy mezi sebou propojovat a vytvářet tak libovolné topologie, na kterých můžou spolupracovat. Dále mají možnost své úlohy pohodlně spojovat s externími systémy.

Do budoucna bude potřeba vytvořit více nových úloh využívajících konektory a také je přidat do již existujících úloh.

---

## Reference

### Internet:

- [1] Virtuální laboratoř počítačových sítí, [online], [cit. 1.3.2010],  
URL: <<http://www.virtlab.cz>>
- [2] Dostál, Radim, Objektově orientované programování v C++, [online],  
[cit. 1.3.2010], URL: <[http://www.builder.cz/art/cpp/cpp\\_oop.html](http://www.builder.cz/art/cpp/cpp_oop.html)>
- [3] Bortlík, Václav, Bakalářská práce - *Distribuované spojovací pole pro virtuální laboratoř počítačových sítí*, VŠB-TU Ostrava, FEI, 2008
- [4] Rudovský, Jan, Diplomová práce – *Ergonomizace uživatelského rozhraní virtuální laboratoře počítačových sítí*, VŠB-TU Ostrava, FEI, 2009
- [5] Vavříček, Jan, Diplomová práce – *Rozvoj řídicího software virtuální laboratoře počítačových sítí*, VŠB-TU Ostrava, FEI, 2007
- [6] Knapek, Jiří, Diplomová práce – *Integrace distribuované laboratoře počítačových sítí se simulátorem PacketTracer*, VŠB-TU Ostrava, FEI, 2010 [předpokládaný rok odevzdání]
- [7] Novák, Radek, Semestrální projekt do předmětu TPS – *Připojení uživatelského PC do Virtlab topologie*, VŠB-TU Ostrava, FEI, 2010
- [8] Hrabálek, Tomáš, Diplomová práce – *Podpora vytváření virtuálních topologií ve virtuální laboratoři počítačových sítí pomocí ethereal/tcpdump*, VŠB-TU Ostrava, FEI, 2007

## 9. Přílohy

### A. Obsah CD

Příložený CD disk obsahuje následující adresáře:

Adresář	Obsah
/dia/src/shapes/Virtlab	connector.jpg connector.shape Makefile.am
/dia/src/sheets	Virtlab.sheet.in
/sql	www-connectors.sql www-create-db.sql
/tun-server/src	Frame.cpp Frame.h Observer_port_setter.cpp Port_connector.cpp Port_connector.h Redirect_table.cpp Redirect_table.h Run.cpp Run.h Tun_server.cpp
/tun-server	Makefile
/ulohy-s-konektory/Konektor ISP	connector_isp.dia connector_isp.xml task.xml
/ulohy-s-konektory/Konektor VLAN	connector_vlan.dia connector_vlan.xml task.xml
/web/class	virtlabDia.php.inc virtlabLanguage.php.inc virtlabWeb.php.inc XMLTransform.php.inc
/web/virtlab	connector.php connector-functions.php reser-active.php reser-new.php tasks-list.php
/web	index.php soapserver.php

**Tabulka 8: Obsah CD**

---

## B. Instalace konektorů na novou lokalitu

### Postup instalace:

1. Aktualizace zdrojových souborů na nejnovější revizi z SVN repozitářů DISTR a DIA. Instalačním skriptem z SVN /DISTR/instalace/install.sh nainstalovat (aktualizovat) web a tun-server.
2. V SQL databázi vytvořit tabulku connectors pomocí skriptu z SVN /DISTR/instalace/sql/www-connectors.sql – pouze při aktualizaci lokality (u nových je již tabulka zahrnuta ve skriptu www-create-db.sql).
3. Přidat do systému nového uživatele: `system` (odesílatel notifikací)
4. Z SVN repozitáře DIA nainstalovat na lokalitu nový virtlabdia ( $\geq 0.9-3$ )
5. Přidat libovolné množství konektorů pomocí Generátoru konfigurací na <https://config.viakis.net>

---

## C. Struktura databáze

Sloupec	Typ	Integritní omezení	Komentář
device	varchar(150)		Fyzický název
vertex	varchar(150)		Logický název
resID	int(11)	cizí klíč reservations (id)	ID rezervace
max_connections	int(3)		Max. počet připojení
Access	varchar(150)		Přístupová práva

**Tabulka 9: SQL tabulka connectors**