VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Podpora vytváření virtuálních topologií ve virtuální laboratoři počítačových sítí s využitím technologie tunelování

Diplomová práce

2007 Tomáš Hrabálek

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

V Ostravě 9. května 2007 ……

Abstrakt

Smyslem této práce je rozšíření virtuální laboratoře počítačových sítí o schopnost propojení více lokalit s umožněním tvorby distribuovaných síťových topologií. Součástí této práce je vytvoření programů pro vlastní řešení virtuálního propojení síťových prvků, distribuovaný systém rezervací prostředků a s tím souvisejícího systému nahrávání a mazání konfigurací síťových prvků.

Klíčová slova: virtuální síťová laboratoř, Virtlab, rezervační systém, VLAN tunely

Abstract

The purport of this work is to extend the Virtual Computer Network Laboratory with an ability to interconnect more localities to enable generation of distributed network topologies. This work includes creation of programs for a solution of network devices connection, a distributed reservation system of resources and related uploading and erasing network devices configurations.

Keywords: virtual network laboratory, Virtlab, reservation system, VLAN tunnels

Seznam použitých zkratek a symbolů

ASCII	American Standard Code for Information Interchange
ASSSK	Automatizovaný systém správy síťových konfigurací
BASH	Bourne Again Shell
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MTU	Maximum Transmission Unit
PHP	Professional Home Pages
SSL	Socket Secure Layer
STP	Spanning Tree Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTF	Universal Character Set Transformation Format
VLAN	Virtual Local Area Network
WWW	World Wibe Web
XML	eXtensible Markup Languague

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

1.1 Nová koncepce Virtlabu

Projekt s názvem *Virtuální síťová laboratoř* (*Virtlab*) byl iniciován potřebou vzdáleně zpřístupnit specializovaná zařízení, která se nacházejí ve školních laboratořích pro výuku předmětů zabývajících se počítačovými sítěmi, zejména pro potřeby studentů, kteří nemají možnost osobně navštěvovat zmíněné laboratoře, a tak zajistit možnost dálkové práce s těmito zařízeními z Internetu.

Původní návrh a realizaci nedistribuované verze po softwarové stránce vytvořil Ing. Pavel Němec v rámci své diplomové práce *Virtuální síťová laboratoř*, kterou obhájil v roce 2005[1]. Zařízení umožňující vzdálené propojování laboratorních prvků vytvořil Ing. David Seidl jako součást své diplomové práce *Automatizovaný systém správy síťových konfigurací*, která byla rovněž obhájena v roce 2005[2]. V průběhu provozu Virtlabu se ukázala potřeba systém více zabezpečit a doplnit některé funkce. Proto byla virtuální laboratoř o tyto prvky vylepšena Ing. Romanem Kubínem v rámci diplomové práce *Zajištění bezpečnosti a implementace nových prvků řídícího systému virtuální laboratoře* obhájené roku 2006[3]. Schéma tohoto původního návrhu ukazuje obrázek 1.

Virtuální síťová laboratoř dnes umožňuje jednoduchou vzdálenou konfiguraci a vzájemné propojení síťových prvků (změny síťové topologie) z libovolného počítače připojeného k Internetu a to i více uživatelům připojeným současně.

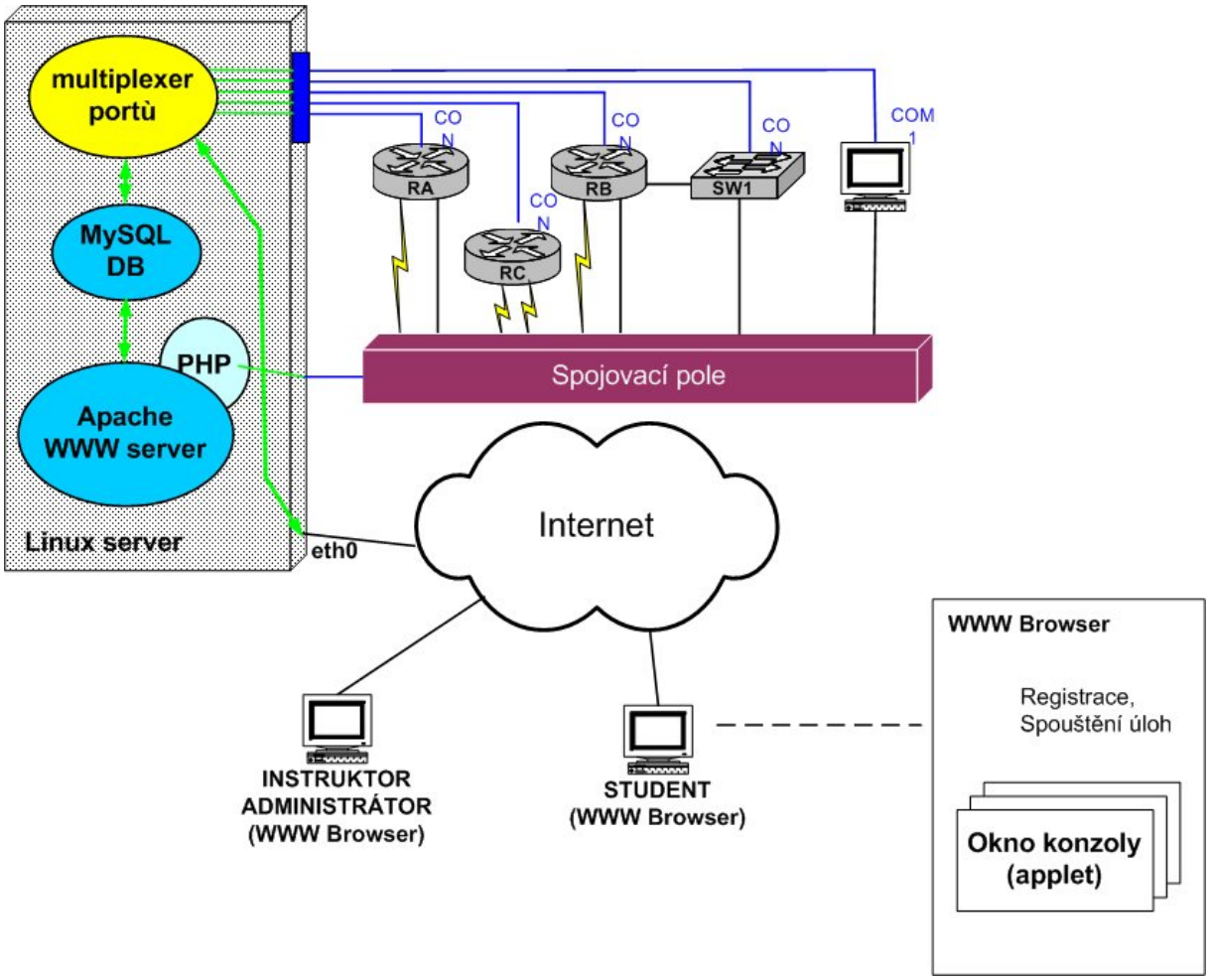
Distribuovaná virtuální síťová laboratoř, vytvořená v rámci této diplomové práce, rozšířila původní architekturu tím, že umožnila propojení více vzdálených lokalit (vzdálené propojení původních virtuálních laboratoří) tak, aby bylo možno vytvářet virtuální síťové topologie s použitím síťových prvků v zúčastněných lokalitách prostřednictvím tunelů skrze Internet. Navíc umožnila tvorbu více topologií najednou, takže může několik uživatelů, z lokalit takřka z celého světa, naráz vytvářet a používat různé virtuální síťové topologie s využitím libovolných dostupných síťových prvků, aniž by je muselo zajímat, je-li tento prvek v jejich místní laboratoři nebo v cizí laboratoři na vzdáleném místě. A to vše navíc lze ovládat jednoduše prostřednictvím webového prohlížeče třeba z domu či ze zaměstnání.

1.2 Omezení původní verze Virtlabu[1]

Omezením nejstarší verze Virtlabu byla pevná fyzická topologie. Jestliže bylo potřeba změnit fyzickou topologii, musel zodpovědný člověk jednotlivé prvky lokality ručně propojit do nově požadované topologie. Tento výrazně omezující faktor se podařilo překonat pomocí ASSSK-1[2]. Toto zařízení dokáže propojovat sériové linky na základě příkazů, které jsou mu předávány. Zapojení určené fyzické topologie, lze tedy jednoduše automatizovat. I když ASSSK-1 dokáže propojovat sériové linky i

1 Tato kapitola je společným dílem autora s řešitelem přidružené diplomové práce[5], Janem Vavříčkem.

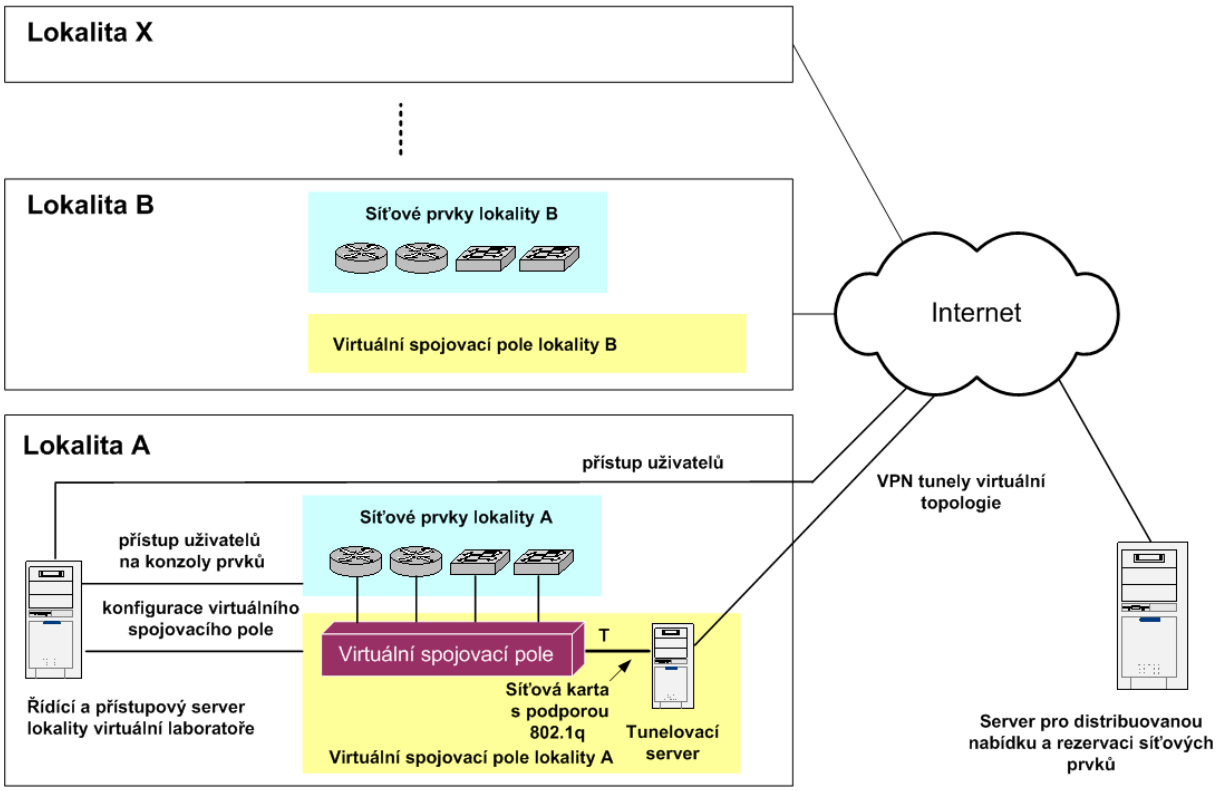
ethernet, byla pro propojování ethernetu časem použita jiná technika, aby se množství propojitelných linek nesnižovalo[2] – když pro propojování sériových linek nemáme žádné jiné řešení.



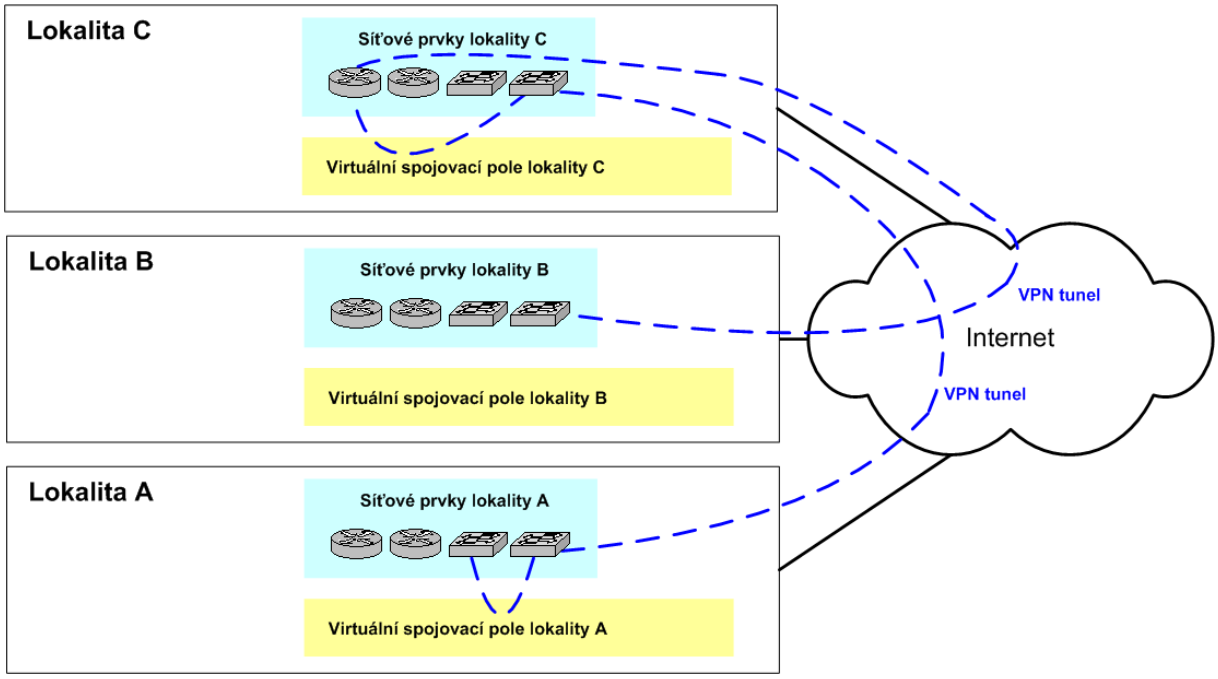
Obrázek 1: Původní architektura virtuální síťové laboratoře
(Převzato z webových stránek projektu Virtlab s laskavým svolením autora Ing. Petra Grygárka, PhD.)

Zmíněné propojování ethernetu bylo časem realizováno technikou zvanou Q-in-Q na Cisco přepínačích řady 3500. Tato technika, jak už zkratka napovídá, je založena na technologii 802.1q, která umožňuje použití VLAN. Přidáním dvou bajtů dat do hlavičky ethernetového rámce lze na takzvané trunk lince identifikovat jednotlivé VLAN. V našem případě se ono přidání provádí ve skutečnosti dvakrát. Jedno označení mohou provádět propojovaná zařízení (musíme jim umožnit se propojovat ethernetovou trunk linkou), druhé označení provede zařízení, které propojuje jednotlivé

2 ASSK-1 má jako každé fyzické zařízení jen konečný počet rozhraní. Ke kazdému z nich se může připojit sériová linka nebo Ethernet – nemohou ale fungovat zároveň. Propojování Ethernetových linek by tak zmenšovalo množství propojitelných sériových linek.



Obrázek 2: Základní architektura distribuované virtuální síťové laboratoře



Obrázek 3: Diagram propojení fyzických zařízení

(Obrázky převzaty z webových stránek projektu Virtlab s laskavým svolením autora Ing. Petra Grygárka, PhD.)

prvky (vytvoří tunel simulující trunk linku), aby odlišilo jednotlivé toky dat, které odpovídají logickým spojům mezi propojovanými zařízeními.

Pro úplnost je třeba dodat, že musíme tunelovat i protokoly druhé vrstvy ISO modelu (STP, CDP, VTP), aby propojovaná zařízení nepoznala, že jsou propojena pomocí přepínače.

I když byl vyřešen problém automatického zapojování topologií, stále bylo třeba vyřešit několik věcí. Jedním z omezujících faktorů je skutečnost, že v jeden okamžik může ve Virtlabu běžet daná úloha jen v jedné instanci, i kdyby byl dostatek prostředků pro více stejných úloh současně. Odpovědnost za toto nese popis úlohy, ve které jsou definovány fyzické prvky, které se musejí použít. V popisech úloh se nemluví o logických směrovačích s daným počtem rozhraní, ale o konkretním fyzickém prvku. Proto bylo potřeba definovat způsob popisu topologií na logické úrovni, popsat fyzické prvky každé lokality, při rezervaci úlohy zpracovat informace z těchto dokumentů a následně realizovat namapování jednotlivých logických prvků na prvky fyzické tak, aby byly splněny všechny požadavky topologie.[3]

Dalším omezením současné verze je nemožnost sdílení prvků mezi několika Virtlab lokalitami. V realitě se objevují situace, kdy jedna z Virtlab lokalit zakoupí nějaké nákladné zařízení, ale toto zařízení nebude uživateli této lokality plně využíváno. V čase, kdy není používáno v lokalitě, v níž je nainstalováno, by jej ale plně mohla využít lokalita jiná, která si zakoupení stejného zařízení dovolit nemůže. Tento stav vyústil v nový návrh Virtlabu - přesněji v návrh distribuovaného Virtlabu. Tento nápad s sebou přinesl řadu problémů, které bylo třeba vyřešit. Týkají se způsobu provádění rezervací, kdy je potřeba, aby o určité úloze v daný čas věděly všechny zúčastněné lokality, dále bylo třeba nalézt způsob propojení několika síťových prvků v různých lokalitách, jak provádět automatizované zapojování topologií napříč distribuovaným systémem, jak je automatizovaně konfigurovat či vzdáleně zpřístupnit uživatelům jejich ovládací konzole.[4]

1.3 Cíle této diplomové práce

Cílem této práce je navrhnout podstatná rozšíření a přepracování stávajících komponent virtuální laboratoře a kompletní návrh komponent nových, aby tímto byl umožněn vznik distribuované virtuální síťové laboratoře a to v těsné návaznosti na diplomovou práci Jana Vavříčka[5]. Jeho práce se zabývá kompletním přepracováním webového rozhraní implementovaného v PHP a oddělením logických identit prvků použitých v definicích úloh od fyzických prvků vybraných pro danou rezervaci.

Společným úsilím byl pod vedením Ing. Petra Grygárka, PhD. realizován kompletní návrh celé architektury distribuované virtuální síťové laboratoře, hlavních principů i terminologie, a nakonec celkové propojení obou diplomových prací v jeden celek. Po kompletaci návrhů bylo cílem tyto komponenty naprogramovat a navrhnout mechanizmy jejich komunikace jak mezi sebou, tak s okolím.

Tato práce je rozdělena do několika sekcí, které se věnují jednotlivým komponentám distribuované virtuální laboratoře. První sekce se zabývá problematikou systému rezervací jednotlivých

3 Celou touto problematikou se zabývá přidružená diplomová práce Jana Vavříčka[5]
4 Tato problematika je hlavním tématem této diplomové práce

laboratorních prvků, které mohou být rozmístěny v kterýchkoli lokalitách, ze kterých je složena distribuovaná virtuální laboratoř. Rezervační systém je složen z mnoha serverů a tyto musejí spolu vzájemně komunikovat, přičemž musí být zajištěna konzistence dat.

Další sekce se zabývá dálkovým propojením ethernetových rozhraní laboratorních prvků pomocí technologie tunelování rámců v datagramech UDP. Při programování příslušného softwaru byly použity specializované postupy, které umožňují příjem rámců značkovaných podle normy IEEE 802.1q, proto se jedna z kapitol věnuje jejich podrobnému popisu.

Dále se práce zabývá zpřístupněním konzolí laboratorních prvků uživatelům distribuovaného systému, což obnáší úpravu dvou programů: Konzolového serveru a Konzolového java appletu. Konzolový server přitom musí zajistit přístup k laboratorním prvkům všech spřízněných lokalit.

V další rozsáhlé části je popsán systém samočinného zapojení virtuálních topologií. Konkrétně pojednává o Aktivátoru konfigurací, generátorech konfiguračních souborů pro virtuální spojovací pole a Konfiguračním serveru a způsobu přidělování čísel VLAN pro účely propojení ethernetových rozhraní laboratorních prvků.

1.4 Základní principy distribuovaného systému

1.4.1 Lokality

Jelikož cílem distribuovaného řešení je zpřístupnění síťových zařízení různých síťových laboratoří ve světě, je systém rozdělen na základní jednotky, které jsou nazvány *lokality*, a které reprezentují právě tyto jednotlivé síťové laboratoře. Tedy lokalita je místo, kde jsou fyzicky umístěna síťová zařízení, která se zpřístupňují uživatelům místním i 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 zapůjčování prvků. Dále je zde spuštěn Konzolový server, Konfigurační server a webový server, který je určen vždy pro místní uživatele. Názorný diagram znázorňující princip distribuované virtuální laboratoře je zobrazen na obrázku 2.

1.4.2 Logické topologie

Jestliže uživatel chce vytvořit jistou úlohu, musí vytvořit návrh žádané síťové topologie. Ten neobsahuje přímo fyzická zařízení, ale určuje pouze typy síťových zařízení a způsob jejich propojení, proto jej nazýváme *logická topologie*. Teprve až je vytvořena tato logická topologie, zjišťuje se, je-li ji v daném čase možno realizovat, vyberou se vhodná síťová zařízení - a to jak lokální, tak i vzdálená, a prostřednictvím Rezervačních serverů se tyto prvky pro danou topologii vyhradí.

1.4.3 Princip propojení fyzických zařízení

Ke spojení zařízení v rámci jedné lokality jsou určena automatická spojovací zařízení - virtuální spojovací pole, která dokáží samočinně spojit libovolná sériová rozhraní i ethernetové porty.

V současně době je používáno zařízení ASSSK-1, které umožňuje přímé propojení signálových vodičů sériových a ethernetových 10BaseT portů, a přepínač Cisco 3550, kterým se vytvářejí virtuální ethernetové segmenty pomocí technologie VLAN. Vzdálené propojení dvou segmentů distribuovaného virtuálního spojovacího pole má na starosti Tunelovací server, který ze dvou těchto různých segmentů v různých VLAN udělá jeden logický ethernetový segment. Takto můžeme propojit například jeden ethernetový port na přepínači v jedné lokalitě s ethernetovým portem směrovače v jiné lokalitě, přičemž tato zařízení pracují, jakoby byla spojena přímo přes ethernetový bridge, jehož existence je, pro protokoly přes něj běžící, utajena. Princip je názorně zobrazen na obrázku 3.

1.4.4 Aktivace topologií

Při započetí definované rezervace vyvolá Rezervační server prostřednictvím Aktivačního skriptu generování konfigurací topologie, příslušné k dané rezervaci, pro spojovací pole a tunelovací servery. Poté odešle soubory s vygenerovanou konfigurací všem konfiguračním serverům v definovaných lokalitách. Tyto pak zajistí jejich nahrání na fyzické prvky, respektive Tunelovací server, v dané lokalitě.

1.5 Systém distribuovaných topologií

1.5.1 Definice pojmů[5]

Lokalita

Lokalita je jedna lokální, autonomní instance Virtlabu schopná samostatného provozu i spolupráce s jinými lokalitami prostřednictvím připojení k volnému Internetu. Spolupráce spočívá jednak v nabízení svých laboratorních prvků pro fyzické topologie požadované uživateli z jiných lokalit (tím vznikají distribuované topologie pomocí propojovacích tunelů) a jednak v používání laboratorních prvků nabízených jinými lokalitami pro fyzické topologie požadované uživateli vlastní lokality. Lokality jsou pojmenovávány textovými řetězci[6].

Lokalita obsahuje:

- laboratorní prvky lokality, mohou volitelně zahrnovat jeden či více serverů simulujících stanice (XEN[7]) a Cisco 7200 (Dynamips[8])
- řídící server lokality (Virtlab server)
- konzolový server
- segment virtuálního spojovacího pole
- konfigurační server spojovacího pole

5 Terminologie je výsledkem dlouhodobé diskuze vývojářského týmu. Přesně ji definoval Ing. Petr Grygárek, PhD., jako hlavní vedoucí celého projektu Virtlab[4]. V této práci je uvedena s jeho laskavým svolením.
6 Pro standardní kódování je použito UTF-8
7 XEN – simuluje koncové počítačové stanice, ke kterým se student může vzdáleně připojit a konfigurovat je.
8 Emulátor procesoru MIPS. Pomocí něj lze na PC simulovat směrovač Cisco 7200.

- rezervační server
- server pro mazání konfigurací síťových prvků
- tunelovací server

Laboratorní prvky

Laboratorní prvky jsou síťové prvky v jednotlivých lokalitách, připojené k lokálnímu segmentu virtuálního spojovacího pole, které jsou k dispozici pro práci studentů na jednotlivých úlohách. Může jít o fyzické síťové prvky nebo prvky simulované (XEN, Dynamips). Laboratorní prvky jsou globálně pojmenovávány ve tvaru `jmeno@lokalita`[9]. Vlastnosti jednotlivých laboratorních prvků jsou popsány v XML.

Uživatelé

Uživatelé jsou zaváděni v jednotlivých lokalitách. Mají jména jednoznačná v rámci lokalit. Globálně jednoznačné jméno uživatele vznikne spojením se jménem domovské lokality a má tvar `jmeno@lokalita`[9]. Uživatelé se autentizují v lokalitě, do které náleží, kde jsou také definována jejich práva či role v systému. V rámci lokality mohou být definovány lokální skupiny uživatelů. Přiřazení uživatele požadujícího sestavení úlohy do skupiny uživatelů může ovlivnit práva na výběr prvků při mapování logické topologie úlohy na fyzickou (toto mapování sestavuje mapovací algoritmus běžící v lokalitě uživatele, takže má definice skupin uživatelů k dispozici).

Logická topologie úlohy

Logická topologie úlohy je popis požadavků na laboratorní prvky žádaných pro řešení určité úlohy včetně popisu požadavků na jejich vzájemné propojení. Každý laboratorní prvek je v popisu zastoupen jedním logickým laboratorním prvkem. Logická topologie je popsána v XML.[10]

Fyzická topologie úlohy

Fyzickou topologií úlohy rozumíme soubor laboratorních prvků-zařízení (i z různých lokalit) namapovaných algoritmem mapování logické topologie na fyzickou na jednotlivé logické prvky odpovídající logické topologii úlohy, včetně přiřazení fyzických rozhraní laboratorních prvků ke spojům logické topologie. Fyzická topologie může být distribuovaná, tedy obsahovat laboratorní prvky z různých lokalit a spoje mezi nimi realizovat prostřednictvím propojovacích tunelů.

Úloha

Úlohou rozumíme definici zadání úkolu pro uživatele popisující mimo jiné logickou topologii úlohy. Úloha může dále obsahovat i vzorové řešení v podobě konfigurace jednotlivých zařízení.

9 Může obsahovat kódování UTF-8

10 V rámci logické topologie mluvíme o vrcholech (logický prvek topologie) a hranách-linkách (propojení dvou logických prvků). Terminologie byla převzata z teorie grafů.

Spuštění úlohy

Spuštěním úlohy rozumíme v čase vymezené propojení laboratorních prvků pro práci studentů podle požadavků popsaných logickou topologií úlohy. Časový interval sestavení úlohy (timeslot) je chápán obecně a není vázán na žádný fixní časový rastr. Zdrojem logické topologie může být buďto tabulka předdefinovaných úloh nebo GUI (u topologie na přání studenta).

Timeslot

Timeslotem je nazýván časový úsek rezervovaný studentem pro řešení určité úlohy. Začátek ani konec není vázán žádným pevným časovým rastrem. Prvních 5 minut timeslotu je vyhrazeno na vymazání předchozí konfigurace z laboratorních prvků použitých v úloze a spojení fyzické topologie, student během nich nemůže přistupovat na laboratorní prvky.

Virtuální spojovací pole

Distribuovaný spojovací systém založený na technologii VLAN a tunelování VLAN (802.1q rámců) pomocí UDP (vlastní enkapsulační formát) přes volný Internet. Spojování laboratorních síťových prvků v lokalitách se děje jejich zařazováním do stejných VLAN (případně VLAN tunelů QinQ při spojování trunků) na přepínačích Cisco 3550, tunely přes Internet jsou zajišťovány vlastním SW – tunelovacím démonem běžícím na tunelovacím serveru.

Řídící skripty virtuálního spojovacího pole

Řídící skripty virtuálního spojovacího pole na základě konfiguračních souborů a textového popisu propojení požadované fyzické topologie vygenerují konfigurační příkazy pro všechny segmenty virtuálního spojovacího pole (včetně tunelovacích serverů) ve všech lokalitách. Textový popis propojení fyzické topologie obsahuje v jednotlivých řádcích dvojice jmen fyzických laboratorních prvků a jejich rozhraní, které mají být propojeny. Skripty spouští před zahájením úlohy Rezervační server lokality, jejíž uživatel rezervaci úlohy vyžádal. Vygenerované konfigurační příkazy jsou zaslány do segmentů virtuálních spojovacích poli všech lokalit zúčastněných v distribuované topologii dané úlohy prostřednictvím jejich Konfiguračních serverů.

1.5.2 Hlavní části architektury [11]

Rezervační server

Aby ovládací software (PHP skript) věděl, které síťové prvky z celého distribuovaného systému může uživateli nabídnout bylo třeba implementovat i funkci vyhledávání volných zařízení v systému. A v neposlední řadě vznikl požadavek na to, aby určité prvky lokalit byly zpřístupněny jiným lokalitám jen po určitý časový interval, nejlépe v týdenním časovém plánu.

11 Tato kapitola je společným dílem autora s řešitelem přidružené diplomové práce[5], Janem Vavříčkem.

Rezervační server je démon, který běží vždy na jednom serveru v každé lokalitě distribuovaného systému a tedy v celém systému běží tolik instancí, kolik je definováno lokalit. Tento server má vždy určeno jméno své lokality, jména vzdálených lokalit a IP adresy jejich rezervačních serverů. Dále má ke každé lokalitě uveden seznam síťových prvků místní lokality, které může vzdáleným lokalitám poskytnout k zarezervování a to podle daného týdenního rozvrhu.

Když ovládací software chce zarezervovat určité prvky, pošle rezervačnímu serveru dotaz, které prvky jsou globálně v celém distribuovaném systému pro něj v určeném čase k dispozici. Rezervační server žádost zpracuje a odešle i do vzdálených lokalit. Každá lokalita má definován XML soubor s popisem vybavení místní laboratoře. Z něj rezervační server vybere nezarezervované a povolené prvky a odešle výsledný XML soubor tazateli. Rezervační server, který rozdistribuovával dotaz ovládacího software, poskládá všechny přijaté XML popisy dostupného vybavení do jediného souboru, který vrátí tazateli. Není-li k dispozici žádný síťový prvek, je vrácen platný soubor, ovšem bez zařízení.

Ovládací software XML soubor od Rezervačního serveru zpracuje, vybere z něj vhodné prvky, a může dále žádat o zarezervování seznamu síťových prvků v daném časovém rozmezí. K tomu vygeneruje globálně unikátní rezervační id ve tvaru 'celé číslo @ název lokality'. Rezervační server žádost rozdělí a pošle každé vzdálené lokalitě v žádosti jen ty prvky, které jí patří. Aby se eliminovaly konflikty, řekne místní rezervační server vzdáleným kolegům, aby onu rezervaci považovali za dočasnou. Povede-li se dočasná rezervace u všech žádaných lokalit, je všem rezervace potvrzena trvale. Může ovšem dojít k tomu, že si někdo před námi již daný síťový prvek zarezervoval a není možné rezervaci u některých z rezervačních serverů provést. Pak k potvrzení samozřejmě nedojde, po dané časové prodlevě dočasné rezervace vyprší a žádost o rezervaci selže.

Konzolový server

Konzolový server zprostředkovává přístup k sériovým nebo telnetovým konzolím síťových zařízení prostřednictvím protokolu TCP. Je využíván Java Appletem, který běží ve webovém ovládacím rozhraní, což umožňuje uživateli jednoduchý přístup k síťovým zařízením. Zároveň slouží i jako proxy server, který zprostředkovává přístup k zařízením, která jsou připojena ke vzdáleným konzolovým serverům, kam nemá místní uživatel přímý přístup. Také prostřednictvím speciálního PHP skriptu ověřuje, jsou-li požadavky uživatele oprávněné a tím konzoly prvků zabezpečuje od neautorizovaného přístupu.

Tunelovací server

Tunelovací server je démon běžící na zvláštním serveru, jež je součástí každého segmentu distribuovaného virtuálního spojovacího pole a jeho úkolem je zajistit propojení různých VLAN[12] mezi jednotlivými lokálními virtuálními laboratořemi a také různých VLAN v rámci jednoho segmentu distribuovaného spojovacího pole. Takto lze zajistit, aby síťová zařízení, která jsou

12 Podle standardu IEEE 802.1q.

připojena na tunelované VLAN, byla zapojena jakoby v jediném zdánlivém ethernetovém segmentu, přestože je každé zařízení v jiné lokalitě a zde dokonce v jiné VLAN.

Konfigurační server

Konfigurační server je jednoduchý síťový démon, jehož úkolem je příjem konfiguračních souborů, které vytvořil spojovací skript, pro spojovací pole síťových prvků, tunelovací servery a také nahrání těchto konfigurací do příslušných spojovacích zařízení. Tímto dochází k aktivaci žádané síťové topologie.

Aktivátor konfigurací

Hlavním prvkem Aktivátoru konfigurací je Aktivační skript. Jeho je zajistit spuštění skriptu generujícího konfigurace pro spojovací pole, dále skriptu pro vytvoření konfigurace pro tunelovací servery a nakonec spuštění skriptu, který tyto konfigurace pošle určeným konfiguračním serverům. Aktivační skript je spuštěn před začátkem dané úlohy rezervačním serverem.

Webový server[13]

Webový server lokality (taky označován jako řídící server lokality) je fakticky jediným místem, kde běžný uživatel přichází s Virtlabem do styku. Kromě řadových uživatelů jsou v systému rozlišování i administrátoři, správci úloh a tutoři. Webový server umožňuje uživatelům si na určitou dobu zarezervovat logickou topologii, kterou si vyberou. Server komunikuje s rezervačním serverem, který mu nabízí fyzická zařízení, která jsou k dispozici. Seznam dostupných zařízení a logická topologie, jsou vstupními argumenty mapovacího procesu, který pro jednotlivé vrcholy logické topologie, což jsou žádaná logická zařízení, vybere adekvátní fyzická zařízení a vygeneruje jejich fyzického propojení, které slouží dále k vygenerování konfigurací spojovacích prvků.

Konzolový applet

Klíčovou součástí klientského webového rozhraní je applet, jehož úkolem je zajistit uživateli vzdálený přístup k sériovým, případně telnetovým konzolám síťových prvků. Konzolový applet čte vstup uživatele a posílá ho konzolovému serveru (CServer),který zpět mu odesílá výstupy ze zařízení. Konzolový applet se rovněž umí připojit v módu tutora, což umožňuje tomuto speciálnímu uživateli sledovat práci obyčejných uživatelů, případně jim pomáhat s konfigurací.

13 O tomto pojednává přidružená diplomová práce Jana Vavříčka

2 Systém distribuovaných rezervací

Aby jednotliví uživatelé měli v daných časových rozmezích zajištěné exkluzivní přístupy k síťovým zařízením a zároveň, aby jeden prvek nebyl využíván pro různé síťové topologie současně, je naprosto nezbytné zavést určitá pravidla a omezení přístupu k síťovým prvkům. Při jedné virtuální laboratoři a velmi malém počtu uživatelů i síťových zařízení bylo toto možné řešit osobní domluvou nebo ručně psanými rozvrhy. Větší počet prvků a složitější topologie si však již vynucují zavedení automatické správy pravidel přístupu.

Proto byl již v původní verzi virtuální laboratoře navržen systém rezervací, kdy si registrovaný uživatel mohl zarezervovat určité pevně dané časové rozmezí – *timesloty*[14] a k němu zvolit topologii a prvky, se kterými chce pracovat. Aby uživatel nevybral veškeré existující timesloty, byly zavedeny týdenní uživatelské kvóty, které určovaly, kolik timeslotů týdně smí mít daný uživatel rezervováno.

2.1 Nové požadavky na rezervační systém

Původní systém rezervací měl mnohá omezení, které nedovolovaly použít jej v distribuované variantě. Na podobných principech byl proto navržen zcela odlišný rezervační proces, který musel splňovat navíc následující požadavky:

- Podpora více spuštěných úloh současně
- Rezervace možné v libovolných časových rozmezích, ne v pevném časovém rozsahu
- Podpora distribuovaných topologií s možností rezervace vzdálenou lokalitou
- Každá lokalita bude pojmenována unikátním jménem
- Každý nabízený síťový prvek musí mít unikátní identifikátor pro celý distribuovaný systém
- Přístupová práva k síťovým prvkům pro různé lokality podle týdenních rozvrhů
- Možnost poskytnout seznam existujících místních i vzdálených síťových prvků
- Podpora zjištění, které prvky jsou volné pro dané časové rozmezí a danou lokalitu
- Seznam s popisy prvků v XML formátu[15]
- Každá rezervace má globálně platný identifikátor
- Možnost rušení rezervací
- K rezervacím bude možno přikládat data s popisem fyzického zapojení topologie

14 Jeden timeslot trval obvykle jednu vyučovací hodinu, což je jeden a půl hodiny.
15 Návrhem DTD pro XML formát popisu prvků se zabývá diplomová práce Jana Vavříčka[5].

2.2 Varianty řešení

Při plánování distribuované virtuální síťové laboratoře se nabízely dva možné způsoby, jak bude systém rezervací fungovat. Byla zde možnost centralizovaného řešení, kde by byl jeden rezervační server, který by spravoval celý distribuovaný systém. Tento návrh měl velkou výhodu v jednoduchosti implementace, ale vzhledem k navrhovanému rozsahu distribuované laboratoře by práce skrze něj nebylo zcela efektivní. To by mohly být zapříčiněno nenadálým výpadkem serveru, což by znamenalo nefunkčnost celého systému. Dále by zde byla nevýhoda pro místní administrátory, kteří by o každou změnu v seznamu půjčovaných laboratorních zařízení nebo v přístupových právech museli žádat centrálního správce. Velkým mínusem by také byla přístupová rychlost, kde by musel centrální server musel byt připojen k velmi rychlé lince Internetu.

Proto byl navržen decentralizovaný distribuovaný rezervační systém, který umožňuje nezávislý běh jednotlivých lokalit a je odolný proti dlouhodobým výpadkům kterékoli ze vzdálených lokalit.

2.3 Princip

Základním kamenem distribuovaného systému rezervací je nově vytvořený démon „rsvsrv“ – neboli rezervační server. Ten je spuštěn na právě jednom určeném serveru v každé lokalitě distribuované virtuální síťové laboratoře. Klient rezervačního serveru, což je téměř výhradně rezervační webová aplikace[16], komunikuje vždy jen se svým místním rezervačním serverem. Pomocí speciálního protokolu mu posílá dotazy a příkazy, které rezervační server vyhodnotí a vrátí výsledek. Jestliže se dotaz nebo příkaz týká celého distribuovaného rezervačního systému, potom se místní server připojí na předdefinované rezervační servery ve vzdálených lokalitách a stejným protokolem jim tento příkaz nebo dotaz dodá, ovšem s upravenými daty, která jsou určeny jen právě tomuto serveru. Výsledky od vzdálených rezervačních serverů dá místní rezervační server poté dohromady a odešle zpět klientovi.

Takovýmto způsobem může klient žádat informace o existujících síťových zařízeních, dostupných síťových zařízeních v daném časovém období, dále může žádat o rezervaci, rušit rezervace a přikládat k rezervacím datové soubory s popisem žádané topologie.

2.3.1 Proces rezervace

2.3.1.1 Příprava rezervace

Chce-li klient zkonstruovat v daném časovém úseku nějakou topologii, nejprve pošle místnímu rezervačnímu serveru dotaz, které prvky jsou v celém distribuovaném systému k tomto časovém rozmezí k dispozici. Místní rezervační server dotaz zpracuje a požadavek rozešle vzdáleným rezervačním serverům. Každý z nich má na souborovém systém uložen XML soubor, ve kterém jsou sepsány vlastnosti veškerých síťových prvků, které daná lokalita může nabízet. Tento soubor musí odpovídat DTD popisu dle souboru „equipment.dtd“ (viz Příloha B, strana 38), který je také k serveru

16 Webovým rozhraním a aplikacím se věnuje diplomová práce Jana Vavříčka[5].

přiložen. Jeho návrh je součástí diplomové práce Jana Vavříčka[5]. Dále je zde umístěn konfigurační soubor, který se jmenuje „rsvsrv.conf“, kde je řečeno, která lokalita, co a kdy si může rezervovat (přesný popis je v kapitole 2.4.2). Podle této konfigurace vybere rezervační server povolené síťové prvky a XML dokument s popisem prvků upraví a vyfiltruje z něj to, co není pro žadatele určeno.

Takto upravený XML soubor je v případě vzdáleného rezervačního serveru odeslán zpět lokálnímu rezervačnímu serveru. Místní rezervační server načte všechny přijaté XML soubory a vytvoří z nich jediný popis všech prvků dostupných v žádaném časovém rozmezí, které odešle původnímu žadateli.

Ten se na základě logické topologie, kterou chce vytvořit, a nabízených prvků, pokusí namapovat fyzické prvky místo logických, podle zvolených kritérií.[17] Tímto vygeneruje konfiguraci pro fyzické spojení prvků, která se v případě úspěšného zarezervování prostřednictvím místního rezervačního serveru přiloží k této rezervaci.

2.3.1.2 Samotná rezervace

Po namapování klient zažádá místní rezervační server o zarezervování síťových prvků pro daný časový úsek. Ten opět žádost distribuuje vzdáleným rezervačním serverům, ovšem k žádosti přiloží i požadavek, aby server čekal na potvrzení rezervace. To je z toho důvodu, že by mohlo dojít ke konfliktu, kdy nebude aspoň na jednom vzdáleném rezervačním serveru již možné provést tuto rezervaci, protože například byla už část časového úseku mezitím zabrána jinou rezervací. Nemůže-li provést klient aspoň na jednom ze vzdálených rezervačních serverů zarezervování, potom u ostatních první fáze žádosti o rezervaci vyprší a je neplatná. Místní server poté klientovi pošle chybové hlášení.

V případě, že žádost o rezervaci proběhne u vzdálených serverů v pořádku, je všem zúčastněným rezervačním serverům odesláno potvrzení[18]. Při potvrzení si všechny rezervační servery informace o rezervovaných prvních ze své lokality uloží do databázové tabulky. Klientovi pak je odesláno kladné potvrzení.

2.3.1.3 Finální fáze

Aby měla rezervace smysl, je nezbytné k ní přiložit již zmíněný vygenerovaný soubor s popisem, jak se mají fyzické prvky navzájem propojit. Klient tedy svůj vygenerovaný soubor zvláštním příkazem odešle místnímu rezervačnímu serveru, který soubor k rezervaci přiloží.

2.3.2 Aktivování rezervace

Nastane-li čas, kdy je třeba spustit úlohu, zavolá speciální vlákno rezervačního serveru, které se nazývá „Aktivátor konfigurací“, takzvaný *aktivační skript*, kterému předá soubor s propojením síťových prvků, který byl přiložen k rezervaci a čas, kdy končí platnost rezervace. Podrobněji se Aktivátorem konfigurací i celým způsobem jakým probíhá automatické zapojování uživatelských virtuálních topologií zabývá kapitola 5 „Samočinné zapojení topologií“.

17 Mapovacím algoritmem a samotnou mapovací aplikací se zabývá diplomová práce Jana Vavříčka[5].
18 Takzvaný „second commit“.

2.4 Rezervační server

Samotný rezervační server je mnohovláknová aplikace určená pro operační systém GNU/Linux. Hlavní proces aplikace po spuštění spustí další vlákno, kde běží „Aktivátor konfigurací“. Dále pak samo čeká na připojení klientů. Připojí-li se klient je obsluhován nově vytvořeným vláknem, což přináší výhodu připojení více klientů rezervačního serveru najednou. Zároveň to přináší řadu možných problémů, které souvisejí především v přístupu do sdílených prostředků, jakými je přístup do tabulek rezervací – ať je to už tabulka dočasných rezervací („first commit table“) ve sdílené operační paměti nebo tabulka uložená v databázovém systému. Tyto záležitosti byly vyřešeny použitím semaforů (*mutex*), respektive zavedením transakcí.

2.4.1 Způsob ukládání rezervací

Úspěšně provedené rezervace se ukládají to databázových tabulek. Pro ukládání je použit databázový systém *MySQL* s podporou tabulek *InnoDB*, ve kterých je možno použít transakce a cizí klíče. K databázovému serveru totiž, jak bylo zmíněno, může najednou přistoupit i několik vláken rezervačního serveru současně, a proto bylo nutno vyřešit některé problémové situace pomocí transakcí.

Rezervační server používá dvě databázové tabulky: „reservations“, kde se ukládá identifikátor rezervace, čas začátku a konce rezervace, a „reserved_devices“, kde jsou potom k příslušnému identifikátoru rezervace uvedena jména zarezervovaných zařízení.

2.4.2 Konfigurace

Nastavení různých parametrů rezervačního serveru, včetně rozvrhu a přístupových práv nebo adres vzdálených rezervačních serverů se nacházejí v souboru „rsvsrv.conf“.

Konfigurační soubor je rozdělen do sekcí, které se uvozují definovanými příkazy. Tento příkaz, spolu se svými argumenty, je vždy prvním řádkem sekce a je to buď první řádek souboru, nebo je před ním prázdný řádek. (Což znamená, že mezi sekcemi je vždy prázdný řádek.) V sekci se poté píší příkazy pro ni specifické. Některé z definovaných sekcí jsou tvořeny jen úvodním příkazem a specifické příkazy definovány nejsou. V následující kapitole jsou popsány typy zmíněných sekcí a příklad konfigurace je uveden v příloze A na straně 46.

2.4.2.1 Typy sekcí konfiguračního souboru

Název místní lokality

Povinným příkazem je **location**, který definuje název místní lokality. Tento název je jediným argumentem příkazu a musí být jednoslovný.

Týdenní rozvrh

Týdenní rozvrh definujeme příkazem **timetable**, jehož argumentem je jeho název. Specifickými příkazy jsou *sunday*, *monday*, *tuesday*, *wednesday*, *thursday*, *friday* a *saturday*. Každým z těchto příkazů definujeme povolenou dobu v daný den. Povolená doba se zadává jako argument ve formátu *hh-HH*, neboli od hodiny hh do hodiny HH (hh:00 - HH:00). První číslo může nabýt rozsahu 0-23, druhé 1-24. Chceme-li uvést pro jeden den více časových rozsahů, napíšeme tento den vícekrát, pokaždé s jiným rozsahem rozsahem.

Definice vzdálených lokalit

Vzdálené lokality definujeme příkazem **virtlab**. Tento má dva argumenty: její název (jednoslovný) a adresu jejího rezervačního serveru. Specifickými příkazy jsou unikátní identifikátory lokálních laboratorních prvků, které tato vzdálená lokalita smí použít. Každý řádek musí být ve tvaru *prvek@nase_lokalita*. Jako argument se u každého prvku musí uvést název časového rozvrhu, který byl definován příkazem *timetable*. Jako vzdálenou zde bereme i naši místní lokalitu. Proto je vhodné přidat řádek *virtlab nase_lokalita 127.0.0.1* se seznamem prvků, které sami sobě nabízíme.

Konfigurace připojení k MySQL serveru

V kódu programu je již implicitně řečeno, k jakému MySQL serveru se má rezervační server připojit (*root@localhost*, databáze *virtlab*, bez hesla). Pokud je alespoň jeden z parametrů nutno změnit, je nutné uvést sekci **mysql**, která má jediný povinný argument ve tvaru *uživatel@adresa_mysql:jméno_databáze*. Pokud neuvedeme žádný další příkaz, bude se toto připojení realizovat bez hesla. Heslo můžeme uvést jako první specifický příkaz (přímo). Ostatní specifické příkazy jsou ignorovány.

Nastavení aktivátoru konfigurace (topologie)

Aktivátor nastavujeme příkazem *conf-activator*, jehož jediným argumentem je název (i s cestou) skriptu, který provede transformaci popisu fyzického zapojení virtuální topologie na konfigurační soubory spojovacích prvků virtuálního spojovacího pole a tyto soubory do nich nahraje. Skriptu bude jako argument předán čas konce rezervace v sekundách a cesta k datovému souboru s popisem fyzického zapojení virtuální topologie. Specifické příkazy zde nejsou.

2.4.3 Komunikační protokol

Komunikační protokol mezi klientem a rezervačním serverem nebo mezi rezervačními servery navzájem pracuje skrze TCP a je založen na principech podobných HTTP. Je to protokol textově orientovaný a lze jej lze jej používat i skrze program *telnet*. Standardně běží na portu 50001.

Základním prvkem protokolu je textový řádek. Ten musí být ukončen některou ze standardních sekvencí *<LF>* nebo *<CR><LF>*. Ve směru od uživatele k serveru je prvním řádkem řádek příkazový. Zde je napsán název příkazu. Na dalších řádcích jsou potom argumenty tohoto příkazu

(každý opět na zvláštním řádku). Argument je psán způsobem „*název_argumentu:hodnota*“ (kolem dvojtečky není dovolena žádná mezera ani jiný oddělovač). Po zapsání všech argumentů následuje řádek ukončovací, což je prázdný řádek. Ten říká zpracujícímu procesu, že veškeré argumenty již byly napsány. Po prázdném ukončovacím řádku následuje opět příkazový řádek pro možný další příkaz a další řádky pokračují popsaným způsobem. Názvy příkazů i argumentů nejsou citlivé na velikost písma, argumenty většinou ano.

Ve směru opačném, tedy od serveru k uživateli, je vždy jako první vrácen řádek s chybovým kódem a jeho popisem. Chybový kód je trojmístné číslo. Čísla *200-299* značí, že předchozí příkaz byl úspěšný, kódy *300-399* označuje jeho neúspěšnost. Za tímto číslem následuje mezera a slovní popis kódu. Za koncem řádku mohou následovat data, ovšem jejich formát a význam je specifický pro dané příkazy.

Definované příkazy jsou *get-offer*, *reserve*, *commit*, *attach* a *cancel*. Blíže jsou popsány v následujících kapitolách.

2.4.3.1 Příkaz pro zjišťování dostupných prvků

Rezervační server nabízí příkaz „***get-offer***“, pomocí kterého můžeme zjistit, jaké jsou dostupné laboratorní prvky pro uživatele místní lokality, a to buď obecně nebo v daném časovém úseku. V případě úspěšného provedením, vrací příkaz XML soubor s popisem prvků a to tak, že za řádkem návratového kódu je další řádek, kde je uvedena velikost dat v posílaného souboru (v bajtech) a hned za koncem toho řádku následují data posílaného souboru.

Pro příkaz je několik možných argumentů: ***for*** – povinný argument, kde se uvádí jméno lokality, pro kterou se tyto informace zjišťují, dále argument ***time*** – za ním následuje časový rozsah ve formátu „`from{yyyy-mm-dd hh:MM}to{yyyy-mm-dd hh:MM}`“, kde yyyy je rok, mm je měsíc, dd je den, hh je hodina a MM minuty. Pokud uvedeme jen tyto dva argumenty, zjišťují se informace pouze v rámci místní lokality. Pokud chceme zjistit informace o celém distribuovaném systému, použijeme argument ***distrib*** a za dvojtečkou uvedeme hodnotu „yes“. Druhou možnou hodnotou, ovšem s významem opačným, je „no“. Tato hodnota je implicitní, pokud argument *distrib* není uveden. Hodnoty tohoto argumentu nejsou citlivé na velikost písmen.

2.4.3.2 Příkaz pro provedení rezervace

Zarezervování se provádí příkazem „***reserve***“. Jeho povinnými argumenty jsou opět ***for***, potom ***id***, které určuje identifikátor rezervace, a povinný je i argument ***time***. Volitelným argumentem je opět ***distrib***. Při komunikaci mezi rezervačními servery je použit navíc argument ***need-commit***. Jeho hodnotami může být „yes“ a „no“. Záporná varianta je implicitní a znamená, že ihned po provedení rezervace jsou data uložena do databázových tabulek. Ve druhém případě je tímto vzdálenému rezervačnímu serveru oznámeno, že bude použito dvoufázového potvrzení rezervace a provedená rezervace je proto pouze dočasná (takzvaný „first commit“) a je třeba ji potvrdit

příkazem *commit* (takzvaný „second commit“). Teprve po jeho zavolání pak je rezervace trvale uložena do databázových tabulek. Tento příkaz nevrací za řádkem návratového kódu žádná další data.

2.4.3.3 Potvrzení rezervace

Bylo-li při příkazu k rezervaci zažádáno o počkání na potvrzení (argumentem „*need-commit:yes*“), provede se toto potvrzení příkazem ***commit***. Ten nařídí rezervačnímu serveru, aby rezervaci natrvalo zapsal do databáze. Jeho jediným a povinným argumentem je ***id***, identifikátor rezervace. Tento příkaz rovněž nevrací žádná data za návratovým kódem.

2.4.3.4 Přiložení popisu topologie k rezervaci

Máme-li vytvořen soubor s popisem fyzického propojení laboratorních prvků, je nutné ho k rezervaci přiložit příkazem ***attach***. Ten má dva povinné argumenty: standardní, již zmíněný identifikátor rezervace ***id*** a navíc argument ***data-length***. Jeho hodnotou musí být celé číslo udávající velikost přikládaného souboru v bajtech. Samotná data se přikládají hned za prázdným řádkem, který ukončuje příkaz.

2.4.3.5 Zrušení rezervace

Vytvořenou rezervaci můžeme zrušit příkazem ***cancel***. Jeho jediným povinným argumentem je identifikátor rezervace ***id***. Nepovinným, ve stejném smyslu jako u ostatních příkazů, je argument **distrib**.

3 Tunelování provozu mezi lokalitami

Již při návrhu distribuované virtuální síťové laboratoře bylo evidentní, že bude nutné najít softwarové řešení, jakým způsobem propojit VLAN, kterými propojují spínací prvky virtuálního spojovacího pole ethernetová rozhraní síťových prvků v uživatelem definovaných topologiích mezi jednotlivými lokalitami navzájem. Dále se ukázala potřeba zavedení ethernetového mostu mezi jednotlivými VLAN v rámci lokality (kvůli propojování zařízení s rozhraními přivedenými do virtuálního spojovacího pole přes pevně danou VLAN). Software měl tedy splňovat tyto vlastnosti:

- Poběží na běžném počítači s OS GNU/Linux s jádry řady 2.6. se zapnutou podporou příjmu ethernetových rámců dle normy IEEE 802.1q[19]. Tento příjem, jakož i takzvaný promiskuitní mód bude podporovat alespoň jedna ethernetová karta tohoto počítače.
- Bude přijímat ethernetové rámce dle normy IEEE 802.1q z trunk linky, která bude propojovat zmíněnou ethernetovou kartu s *trunk* portem virtuálního spojovacího pole (konkrétně přepínače Cisco 3550).
- Příjme rámce určené k poslání do jiné, nebo vzdálené VLAN, po přijetí podle potřeby přečísluje číslo VLAN ID na číslo nové, celý rámec zabalí do UDP paketu a odešle vzdálenému tunelovacímu serveru. Při lokálním přečíslování VLAN ID dojde takto k vytvoření mostu mezi určenými VLAN.
- Software příjme rámec zabalený v UDP paketu, vybalí jej a odešle do své připojené trunk linky k virtuálnímu spojovacímu poli.

3.1 Nalezené řešení

Při hledání možných řešení se jako nejlepší varianta ukázalo napsání vlastního softwaru, který tunelování i bridging zajistí. Byl tedy napsán program v jazyce C, pojmenovaný *tunelovací server*, který je schopen přijímat syrové rámce z rozhraní určeného ethernetového zařízení a rozpoznat hlavičku dle již zmíněné normy. Umí zabalit přijaté rámce do UDP a odeslat druhé lokalitě a rovněž z cizí lokality přijmout, vybalit a v syrové podobě odeslat trunk linkou k virtuálnímu spojovacímu poli. Pro ovládání programu byla zvolena textová řádková konzole (CLI[20]).

3.1.1 Princip funkce

Tunelovací server po spuštění nejprve přepne uživatelem zadané síťové rozhraní do takzvaného promiskuitního módu, což umožní příjem všech ethernetových rámců, které ne rozhraní posílá trunk linkou připojené virtuální spojovací pole. Na rozhraní je otevřen komunikační socket, který je nastaven na příjem takzvaných „syrových“ ethernetových rámců. Takto je možné přijímat nejen data

19 Tato podpora není principiálně nutná, protože software přijímá rámce v syrové podobě. Při jejím vypnutí však ovladače některých ethernetových karet začaly rámce značkované podle normy IEEE802.1q zahazovat.

20 CLI – Command Line Interface

rámce, ale i jeho hlavičku s MAC adresami. Tunelovací server poté začne na tomto socketu poslouchat provoz.

U každého přijatého rámce je zkontrolováno v hlavičce IEEE 802.1q číslo VLAN, neboli VLAN ID. Není-li toto číslo nalezeno v tabulce přesměrování, kterou na tunelovacím serveru vytvořil administrátor, je rámec zahozen. Jinak je VLAN ID přečíslováno na cílové číslo, které je také v této tabulce definováno. Pak je celý rámec zabalen do UDP paketu a odeslán na IP adresu, která je posledním atributem tabulky přesměrování. Tunelovací server tedy potřebuje dvě síťová rozhraní – jedno pro trunk linku a druhé pro přístup k vnější síti (viz obrázek 4). Cílem může být i sám odesílatel, což způsobí vznik již zmíněného mostu mezi lokálními VLAN. Na druhé straně samozřejmě tunelovací server UDP paket příjme, rozbalí jej a opět v syrové formě odešle do připraveného socketu. Cílové číslo VLAN tedy určuje odesílající tunelovací server.

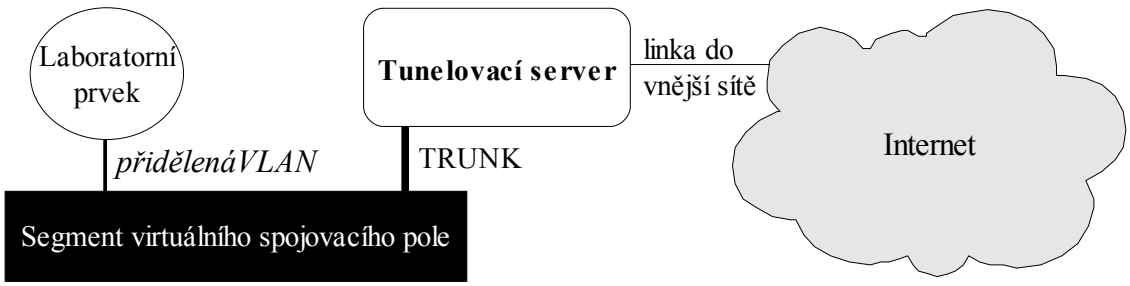
3.2 Ovládání a implementace tunelovacího serveru

3.2.1 Spuštění

Tunelovací server je stěžejní program pro vytváření virtuálních ethernetových spojení mezi prvky v různých lokalitách, ale i mezi prvky v různých VLAN ve stejné lokalitě. Jeho základem je spustitelný soubor „vlan_redirector“. Jeho jediným povinným argumentem je název síťového zařízení, z něhož vede trunk linka do virtuálního spojovacího pole. V případě potřeby před něj můžeme vložit argument „-d“, který způsobí, že bude program vypisovat ladící informace. Protože tunelovací server využívá nízkoúrovňový přístup k síťovému zařízení, je možné jej spustit pouze uživatelem root.

3.2.2 Konfigurace

Administrátor tunelovacího serveru musí po spuštění nastavit, co a kam se má přeposílat. Shromaždiště těchto informací se nazývá, jak již bylo zmíněno, *tabulka přesměrování*. Její nastavování probíhá přes textovou řádkovou konzoli, která běží standardně na portu 40001, a to skrze dané textové příkazy a jejich argumenty. Pro ladící účely je rovněž možné si vypsat celou tabulku přesměrování. Ke konzoli je možné se připojit i programem *telnet*. Příkazy, které je možné v konzoli použít uvádí Tabulka 1.



Obrázek 4: Zapojení rozhraní tunelovacího serveru

Příkaz	Argumenty	Popis
exit	*žádné*	Ukončí spojení s konzolovým klientem
redir	přijaté_vlan_id nové_vlan_id adresa_cílového_serveru	Do tabulky přesměrování přidá záznam, který říká, že rámce označené číslem *přijaté_vlan_id* se zkopírují a kopie přečíslují na číslo *nové_vlan_id* a rámec se odešle na tunelovací server daný polem *adresa_cílového_serveru*
noredir	vlan_id	Z tabulky přesměrování odstraní záznamy s určeným příchozím *vlan_id*
show	*žádné*	Vypíše tabulku přesměrování

Tabulka 1: Příkazy řádkové konzole tunelovacího serveru

3.2.3 Implementace

Jak bylo již zmíněno, je tunelovací server implementován v jazyce C. Kód je psán ve standardu POSIX, což umožňuje snadnou platformovou přenositelnost mezi operačními systému tohoto standardu. Ovšem primárně je určen pro běh v OS GNU/Linux. Uživatelské konzole jsou naprogramovány pomocí vláken (standard pthreads[21]), takže je nutné mít pro kompilaci i běh prostředí, které běh vláken podporuje.

3.2.3.1 Řešení příjmu rámců IEEE 802.1q

Přestože použitý operační systém Linux má standardně v jádře zabudovanou podporu pro VLAN standardu IEEE 802.1q, možnosti použití jsou omezené. Veškeré nastavování se totiž provádí příkazem „vconfig“, pomocí kterého se pro určené VLAN vytvářejí a mažou logické síťové rozhraní, kam jsou směřovány rámce označené touto VLAN. Na logickém rozhraní je možno poté přijímat rámce neznačkované. Ve Virtlabu je ale potřeba používat desítky různých VLAN najednou, které se navíc v čase dynamicky mění, což by v tomto případě znamenalo neustálé vytváření a mazání desítek logických rozhraní, přičemž na každém z nich by musel být poslouchán provoz. To je ovšem z hlediska rychlosti značně neefektivní.

Z těchto důvodů přijímá tunelovací server všechny ethernetové rámce z rozhraní, z něhož vede trunk linka do virtuálního spojovacího pole v syrové formě, tedy včetně hlaviček, které sám vyhodnocuje. K tomuto je použit socket, který je speciálně vytvořen tak, aby přijímal právě veškerý provoz včetně hlaviček. Pro přístup k datům hlavičky byla podle normy vytvořena nová paměťová struktura `vlanethhdr`, kterou nalezneme ve zdrojovém kódu v souboru „include/ieee8021q.h“.

Následuje stručný popis, jak lze v jazyce C pod OS GNU/Linux vytvářen sockety pro příjem syrových rámců. Podrobný popis je potom uveden v příloze C na straně 48.

21 POSIX threads

Vytvoření socketu pro příjem syrových rámců se provede voláním standardní funkce `socket`. Jako argumenty se přitom vloží hodnoty PF_PACKET pro rodinu socketu, SOCK_RAW pro typ socketu, nakonec ETH_P_ALL pro určení protokolu. Existuje i konstanta ETH_P_8021Q pro přímý příjem protokolu IEEE 802.1q, ale při nastavení této konstanty nebyly přijímány vůbec žádné rámce. Pravděpodobně se jedná o špatnou implementaci v jádře systému.

Poté se ze slovního názvu síťového zařízení (například „eth1“), z něhož vede trunk linka do virtuálního spojovacího pole, zjistí voláním funkce `ioctl` jeho systémové číslo, které se použije pro vyplnění parametru `sll_ifindex` struktury typu `sockaddr_ll`. Ta slouží jako argument pro příkaz `bind`, který je zmíněn dále. V této struktuře se ještě vyplní adresní rodina – sll_family na hodnotu AF_PACKET a dále protokol sll_protocol na hodnotu ETH_P_ALL.

Následně se voláním funkce `ioctl` nastaví rozhraní určené pro příjem značkovaných rámců do takzvaného promiskuitního režimu, který umožňuje, aby síťové zařízení přijímalo veškerý síťový provoz, který se vyskytuje na připojené trunk lince (tedy rámce s libovolnou cílovou MAC adresou).

Poté je dříve vytvořený socket svázán s určeným rozhraním, kde budeme přijímat syrové rámce, voláním funkce `bind`. Pokud by se tato funkce nezavolala, přijímal by socket provoz ze všech v operačním systému aktivních síťových rozhraní.

Samotný příjem rámců se provádí opakovaným voláním funkce `recvfrom`. Ta uloží data každého přijatého rámce do připraveného bufferu. Přetypováním na vytvořenou strukturu `vlanethhdr` lze snadno přistupovat k hlavičce rámce. Je zkontrolován typ dat, které je neseny v ethernetovém rámci. Jestliže typ neodpovídá 802.1q, je rámec ignorován. V opačném případě je zjišťováno, nachází-li se VLAN ID, které je uvedeno v hlavičce 802.1q, v tabulce přesměrování. Dále je zpracován již dříve popsaným postupem.

3.2.4 Generování konfigurace tunelovacího serveru ve Virtlabu

Když rezervační server v čase zahájení rezervované úlohy spustí aktivátor konfigurace, je v rámci aktivačního skriptu spuštěn i skript „make-conn-configs“[22], který mimo jiné vytvoří i soubor „tunelserver.out“. V tomto souboru je vždy na každém řádku uvedena definice jednoho propojení mezi lokalitami pomocí VLAN. Soubor slouží jako vstup pro další skript spouštěný aktivátorem, který se jmenuje „make-tun-conf“. Ten zpracuje zmíněný soubor společně ze souborem „site.conf“, který definuje propojení mezi názvem lokality a adresou tunelovacího serveru. Jako výstup je vygenerováno několik souborů „tun-*lokalita*“, kde se nachází konfigurační skript, který je prostřednictvím konfiguračního serveru odeslán do řádkové konzole tunelovacího serveru lokality, jejíž jméno se nachází v názvu souboru za spojovníkem.

22 Popsáno v kapitole 5 „Samočinné zapojení topologií“.

3.2.5 Zjištěné problémy a jejich možná řešení

Při testování tunelovacího serveru došlo ke zjištění několika provozních problémů, na které literatura[10][11][12] nepoukazovala. Prvním problémem je skutečnost, že přestože je možný příjem rámců značkovaných normou 802.1q s velikostí až MTU+4, implementace ethernetových rozhraní v jádře Linuxu neumožňuje odeslat rámce větší, než MTU[23], což znemožňuje, aby se prostřednictvím tunelovacího serveru tunelovaly rámce větší, než MTU-4. Z tohoto důvodu je nutné na laboratorních zařízeních, u nichž je předpokládán provoz skrze tunelovací servery snížit zmíněné nastavení právě na MTU-4[24].

Další problém se ukázal při příjmu rámců na některých gigabitových ethernetových kartách s vestavěnou podporou akcelerace VLAN. Tyto totiž odstraňují z ethernetového rámce hlavičku 802.1q, kterou předtím samy zpracují, a předají informace v ní obsažené operačnímu systému jiným kanálem. Ten se z těchto informací rozhodne, jestli rámec příjme a na které logické síťové rozhraní je pošle. Ovladače některých síťových karet toto chování vypínají v promiskuitním režimu, bohužel ne ve všech byla tato vlastnost implementována. S těmito kartami není v současnosti možné tunelovací server použít.

Řešením do budoucna je úprava implementace protokolu 802.1q v jádře operačního systému Linux o možnost vytvoření virtuálního síťového zařízení, kam byly posílány rámce z mnoha definovaných VLAN i s přidanou hlavičkou 802.1q a zároveň rámce, které by bylo poslány do tohoto zařízení, by byly odeslány ven ethernetovou trunk linkou beze změny. To by pravděpodobně vyřešilo oba zmíněné problémy.

3.2.6 Zabezpečení tunelů

Již od začátku návrhu distribuované virtuální laboratoře bylo počítáno s tím, že propojení tunelovacích serverů bude realizováno buď neveřejnou universitní sítí nebo bude provoz přenášen prostřednictvím šifrovaných kanálů (VPN skrz veřejný Internet), tudíž nebylo nutné explicitně implementovat jakékoliv zabezpečení.

23 MTU je u Ethernetu standardně 1500 bajtů.
24 Standardně 1496 bajtů

4 Vzdálený přístup ke konzolím laboratorních prvků

Každý laboratorní prvek užitý v distribuované virtuální síťové laboratoři nabízí možnost konfigurovat jej skrze textovou konzoli. Tato konzole může být přístupná buď přes sériový port nebo skrze telnet. Aby uživatelé Virtlabu mohli laboratorní prvky konfigurovat, byl už v původním Virtlabu navržen způsob, jak zprostředkovat přístup k těmto konzolím přímo z webových prohlížečů uživatelů[1][3]. Ve webovém prohlížeči byl spuštěn java applet simulující řádkový terminál, který se připojil ke konzolovému serveru, jehož úkolem bylo přemostit datový tok z/do java appletu přímo do konzole laboratorního zařízení. Přístup ke konzolím měli výhradně oprávnění uživatelé Virtlabu, kteří byli přihlášení do systému.

V distribuované virtuální síťové laboratoři může být ovšem mnoho lokalit, z nichž každá má své registrované uživatele. Bylo by obtížné ověřovat oprávněnost uživatele z cizí lokality, kdyby se ten chtěl připojit přímo ke konzolovému serveru místní lokality. Proto byl navržen způsob komunikace se vzdálenými konzolovými servery prostřednictvím místního konzolového serveru, který slouží jako proxy server. Pro místní server není problém ověřit právo uživatele ke konzoli zařízení přistoupit a vzdálený konzolový server pak jen ověřuje, že ten místní má právo se k němu připojit.

4.1 Konzolový server

4.1.1 Původní funkce

Konzolový server je speciální démon, který umožňuje, jak bylo zmíněno, vzdálený přístup k sériovým či telnetovým konzolím síťových prvků, či simulovaných síťových prvků prostřednictvím TCP spojení. Ve virtuální síťové laboratoři je tímto klientem výhradně java applet, který poskytuje webové rozhraní. Uživatelský vstup z klávesnice je jeho prostřednictvím předán konzoli určeného zařízení a výstup konzole je naopak zase odeslán zpět uživateli.

Tento server původně naprogramovali Ing. Pavel Němec[1] a Ing. Roman Kubín[3] pro potřeby původní síťové laboratoře. Ten uměl zprostředkovat jen sériové konzole, proto jej rozšířil Ing. Petr Grygárek, PhD. o možnost připojení skrze TCP na konzole telnetové. Uživatelská aplikace, která se k serveru připojila, měla možnost používat konzoli prvku, který měl uživatel právo používat, avšak ten musel být přímo dosažitelný serverovým počítačem. Navíc klient nepředával serveru obecný identifikátor zařízení, nýbrž písmena identifikující přímo čísla sériových portů, případně telnetových konzolí, což znemožňovalo obecnější použití. Proto byly zdrojové kódy původního serveru upraveny a některé funkce byly implementovány znovu, zcela odlišným způsobem, jiné byly jen rozšířeny za použití původní funkcionality.

4.1.2 Rozšíření pro potřeby distribuovaného systému

Jak bylo již poznamenáno, omezené schopnosti původního konzolového serveru nebyly dostačující pro potřeby nového distribuovaného systému. Proto musel být konzolový server podstatně upraven, aby splňoval požadované vlastnosti:

- Bude dovolovat se připojit ke konzolím i, jak bylo zmíněno, zprostředkovaně, což znamená, že musí umět fungovat i jako proxy server.
- Zařízení bude identifikováno podle globálního pojmenování „název@lokalita“. Podle lokality server pozná, bude-li se připojovat přímo ke konzoli zařízení ve své lokalitě, nebo slouží jen jako proxy server.
- Zabezpečení a uživatelská práva se budou ověřovat prostřednictvím „session id“, které identifikuje uživatele přihlášeného ve webové aplikaci.

Rovněž byl vysloven dodatečný požadavek, aby k témuž zařízení mohl kromě uživatele přistoupit i takzvaný „tutor“, což může být lektor onoho uživatele, který bude smět sledovat jeho práci, či zadávat vstup do konzole paralelně se studentem. Více v kapitole 4.1.6 Tutor.

Všechny tyto potřeby byly zváženy a na jejich základě byl upraven jak komunikační protokol, samotný server, tak i klientská aplikace – java applet. O těchto aspektech se zmiňují následující kapitoly.

4.1.3 Konfigurační soubory

Oproti starší verzi, kdy konzolový server neobsahoval žádné konfigurační soubory, je nutné před spuštěním virtuální laboratoře v nové verzi nastavit dva konfigurační soubory, kterými nastavíme jakým způsobem se konzolový server dostane ke konzolí síťových prvků.

4.1.3.1 Nastavení lokalit

Prvním konfiguračním souborem je „*domains.conf*“. Tímto textovým souborem se nastavuje, které lokality, v nichž jsou jednotlivá zařízení, jsou místní a tedy se bude přímo přistupovat ke konzolím, a které vzdálené, pro něž bude konzolový server fungovat jako proxy.

Každý řádek je uvozen klíčovým slovem „*local*“ nebo „*remote*“, přičemž první zmíněné se musí v souboru objevit právě jednou a druhé v libovolném počtu. Potom následují parametry, které jsou oddělené jednou mezerou. Prvním argumentem obou klíčovým slovem je název lokality.

Je-li řádek uvozen slovem „*local*“, potom je zmíněná lokalita místní. Znamená to, že globální identifikátory všech místních síťových prvků obsahují za zavináčem právě tuto lokalitu. Toto je jediný povinný argumenty tohoto řádku. Může ještě následovat jeden, což je celé číslo, které určuje port pro TCP spojení, na kterém místní server naslouchá, jinak naslouchá na standardním portu[25].

25 Standardní port pro připojení ke konzolovému serveru je 10000

Ostatní řádky, uvozené klíčovým slovem „*remote*“, říkají, že místní konzolový server bude pro určené lokality sloužit jako proxy server, tedy že veškerý provoz pro konzole prvků dané lokality bude přeposílán vzdálenému konzolovému serveru. Jeho IP adresa je druhým argumentem těchto řádků. Na konci může obsahovat dvojtečku, za kterou následuje port. Není li dvojtečka s portem uvedena, použije se port standardní[25].

Použijeme-li pro název doménu u „*remote*“ řádků hvězdičku „*“, potom bude provoz všech ostatních domén, které dosud nebyly určeny, přeposlán uvedenému konzolovému serveru. Není-li řádek s hvězdičkou v souboru uveden, je provoz na neznámých doménách automaticky odmítnut. Toho může být mimo jiné využito pokud konzolový server běží na počítači, ke kterému jsou připojeny laboratorní prvky, a který není dostupný z Internetu. Potom může být v lokalitě spuštěn ještě jeden „veřejný“ konzolový server, k němuž je přístup z Internetu, a který veškerý provoz jen přesměruje „hlavnímu“ konzolovému serveru. Adresu tohoto „hlavního“ serveru je pak uvedena za hvězdičkou v konfigurace „veřejného“ konzolového serveru. V této konfiguraci se ale nesmí uvést u definice místní lokality její skutečný název, ale nějaký název pomocný (například „naše_lokalita_public“).

Následující ukázka souboru „domains.conf“ ukazuje konfiguraci konzolového severu, který běží v lokalitě „nase_lokalita“ na portu 10000 a může sloužit jako proxy pro přístup ke konzolím prvků v lokalitách „vsb“ a „filjakovo“. Pro ilustraci je uveden i řádek s přesměrováním zbylého provozu.

```
local nase_lokalita 10000
remote vsb 158.196.135.23
remote filjakovo 174.15.24.68:10001
remote * 10.0.0.1
```

4.1.3.2 Nastavení přístupu ke konzolám místních zařízení

Druhým souborem pro konfiguraci konzolového serveru je „*devices.conf*“. Tento opět textový soubor určuje, jakým způsobem konzolový server dosáhne na konzole určených lokálních zařízení.

Každý řádek je uvozen jménem síťového zařízení, jehož konzoli zpřístupňujeme. Jelikož je místní lokalita už definována, píše se pouze prefix globálního jména, tj. část před zavináčem.

Za mezerou následuje klíčové slovo, které určuje typ připojení ke konzoli síťového prvku. Toto klíčové slovo může být „serial“ - pro přístup ke konzoli přes sériový port, nebo „telnet“ - pro přístup skrze TCP připojení.

U řádků pro sériové připojení následuje cesta k ovladači zařízení sériového portu. Soubor tohoto zařízení musí pochopitelně existovat a mít práva nastavená pro čtení i zápis tak, aby k němu mohl konzolový server přistupovat. Sériový port musí již být inicializován – to znamená musí být nastavena rychlost, řízení toku dat a podobně, aby byla zaručena kompatibilita s připojeným prvkem. Pro snadnou inicializaci sériového portu byl členy vývojového týmu Virtlab vytvořen skript „*init-moxa-port*“, který je možno použít.

V řádcích s klíčovým slovem „telnet“ za mezerou následuje IP adresa a port konzole. Port je od adresy oddělen podle standardních zvyklostí dvojtečkou.

Zařízení, která nejsou v souboru definována, jsou pochopitelně nepřístupná.

Následuje ukázka souboru „*devices.conf*“, který říká, že v místní lokalitě lze přistupovat ke konzolím pěti laboratorních prvků, ke třem přes sériový port a ke dvěma skrze telnet.

```
r1 serial /dev/ttyS0
r2 serial /dev/ttyM1
s4 serial /dev/ttyUSB0
c1 telnet 127.0.0.1:12345
c2 telnet 158.196.135.13:23
```

4.1.4 Komunikační protokol

Klient, který se připojí ke konzolovému serveru, s ním komunikuje prostřednictvím velmi jednoduchého textově orientovaného protokolu. Ihned po připojení odešle klient serveru čtyři povinné argumenty, každý je psán textově a ukončen koncem řádku[26].

Prvním argumentem (jakoby prvním řádkem) je globální identifikátor síťového zařízení, k jehož konzoli se chce klient připojit, ve formátu „*jméno@lokalita*“.

Argument druhý obsahuje řetězec „session ID“, což je jednoznačný identifikátor uživatele přihlášeného ve webovém rozhraní virtlabu.[27] Pomocí tohoto řetězce probíhá zabezpečení, více dále.

Následuje argument, který určuje takzvaný mód „tutora“. Může nabývat hodnot „*off*“, „*observer*“, „*exclusive*“, nebo „*shared*“. Více informací o módu tutora je v kapitole 4.1.6 Tutor.

Posledním argumentem je globálně unikátní identifikátor rezervace. Ten je, jak již bylo zmíněno v kapitolách o rezervačním systému, ve formátu „*celé_číslo@lokalita*“. Tento argument, stejně jako session ID, rovněž slouží k zabezpečení konzolového serveru.

4.1.5 Zabezpečení

Každý uživatel přihlášený k webovému rozhraní získá, jak již bylo zmíněno, dočasný jednoznačný identifikátor, který se jmenuje *session ID*. K tomuto textovému řetězci a k rezervacím vytvořeným daným uživatelem se váží určitá práva přístupu ke konzolím síťových prvků, která daný uživatel má. Jedná se o to, ke kterým konzolím smí přistoupit, může-li být tutorem a podobně.

Proto je konzolovému serveru předán také argument *ID rezervace*. Všechny předávané argumenty tvoří dohromady spolu s IP adresou uživatelova počítače tvoří přístupový klíč ke konzoli. Což znamená, že pokud se chce někdo neoprávněně do konzole dostat a zadá jednu z hodnot nesprávně (například zadá prvek, ke kterému nemá přístup), je jeho připojení odmítnuto.

26 Konec řádku může být unixový <CR>, případně telnetový (DOS, MS Windows) <CR><LF>
27 O tomto pojednává přidružená diplomová práce Jana Vavříčka[5]

Zároveň se také může stát, že uživatel je připojen k nějaké konzoli, přičemž čas jeho rezervace už vypršel, potom je přístup rovněž zablokován, protože identifikátor rezervace již není platný.

Ověřování platnosti údajů zaslaných appletem konzolovému serveru je realizováno prostřednictvím speciálního ověřovacího PHP skriptu[28], který je součástí webového rozhraní a tudíž zná parametry, které byly vygenerovány pro applet a může je ověřit. Tento skript může ovšem spustit výhradně místní konzolový server.

Aby konzolový server poznal, že uživatel přestal mít právo přistupovat ke konzoli daného prvku, je platnost údajů ověřována periodicky. To znamená, že vyprší-li například čas rezervace, ověřovací skript údaje označí za špatné. Uživateli je potom posláno hlášení o této události a poté je odpojen.

4.1.5.1 Nedostatky a plánovaná vylepšení

K úplnému zabezpečení komunikace s konzolovým serverem chybí kontrola, zda komunikace mezi uživatelem a konzolovým serverem nebyla narušena či pozměněna. Tato záležitost bude řešena šifrováním komunikace vrstvou SSL. Prozatím, pokud se vyskytne potřeba tuto komunikaci šifrovat, může toto být zabezpečeno skrze šifrovaný VPN tunel to do sítě místního Virtlabu.

4.1.6 Tutor

Již v původní nedistribuované verzi Virtlabu se objevil požadavek, aby v něm existoval uživatel zvaný tutor, který by mohl sledovat práci jiného uživatele s konzolí určeného síťového zařízení, případně aby mohl také do této konzole zapisovat, či úplně nad ní převzít kontrolu.

Některé tyto funkce byly v původním systému zapracovány, ovšem z důvodu částečného přepsání konzolového serveru byla tato funkce reimplementována od začátku jiným způsobem a také rozšířena.

4.1.6.1 Režimy práce tutora

Jak bylo v úvodu zmíněno, tutor může pracovat v několika režimech. První režim se nazývá „*observer*“, neboli pozorovatel. Ten umožňuje, aby tutor sledoval práci uživatele s laboratorním prvkem, aniž mu do jeho práce mohl jakkoli zasáhnout. To znamená, že výstup, který je odesílán prvkem konzolovému serveru je kopírován a posílán tutorovi.

Druhým možným režimem je takzvaný „exclusive“, neboli exkluzivní. Zde se vytvoří přesně opačná situace, než v prvním případě, kdy tutor může neomezeně pracovat s konzolí, zatímco uživatel může jeho práci jen pozorovat. Tudíž vstup od uživatele konzolový server zahazuje, zatímco místo něj posílá data přicházející od tutora.

Třetím režimem je „master“, což znamená řídící. Tento mód rozšiřuje předchozí o to, že se tutor stane jediným uživatelem, který může posílat data do laboratorního prvku a přijímat výstup. Vstup od uživatele je blokován, stejně jako výstup k uživateli.

28 V rámci své diplomové práce[5] jej napsal Jan Vavříček.

Posledním, čtvrtým režimem je speciální režim „shared“, neboli sdílený. V rámci něho může tutor i uživatel zapisovat do laboratorního prvku i přijímat jeho výstup a to najednou. Oba mají tedy stejná práva. Nevýhodou tohoto módu je skutečnost, že se tutor s uživatelem musí domluvit, kdo bude kdy zapisovat vstup, aby nedocházelo ke konfliktům.

Vždy, když se tutor připojí ke konzolovému serveru, ať je to jakýkoliv režim, je uživateli posláno s výstupem z prvku i upozornění, že se tutor připojil a jaký je jeho režim. V případě „master“ módu je uživatel rovněž upozorněn, že nic neuvidí do doby, než se tutor odpojí. Po odpojení tutora, se veškerá práva uživateli opět vracejí a uživatel je na tuto skutečnost rovněž upozorněn.

4.1.6.2 Princip napojení tutora na konzoli

Je zřejmé, že je-li proces konzolového serveru obsluhující uživatele, který pracuje s konzolí určitého laboratorního prvku, připojen k tomuto prvku (ať již sériovým portem, nebo telnetem), nemůže se stejným způsobem k zařízení připojit i proces obsluhující tutora. Proto bylo třeba zavést určitý systém sdílení přístupu ke konzoli zařízení.

Spravujícím procesem, který přímo přistupuje ke konzoli, v tomto případě zůstal ten, který obsluhuje uživatele. Rodičovský proces konzolového serveru obsahuje v paměti tabulku, která popisuje jednotlivé laboratorní prvky ke kterým se konzolový server smí přímo připojit. V této tabulce jsou mimo jiné pro každé zařízení také dva *file descriptory*, které každý ukazují na jeden otevřený socket. Tyto sockety jsou přitom spárovány, tudíž tvoří obousměrnou frontu, kterou nazvěme komunikační kanál tutora. Jeden socket je určen procesu obsluhujícímu uživatele a druhý pro proces obsluhující tutora.

Když se tutor připojí ke konzolovému serveru, obdrží jeden tento socket jako přístupový bod ke konzoli laboratorního zařízení. Protože není jisté, že je zrovna toto zařízení otevřeno nějakým uživatelem a na druhé straně komunikačního kanálu nějaký proces naslouchá, odešle tutorův proces skrze socket zprávu ve významu „*Jsi naživu?*“. Vyčká na odpověď několik vteřin a nedostane-li ji, potom se bere konzole žádaného zařízení jako odpojená a tudíž není komu dělat tutora, a proto proces obsluhující tutora končí, tutor je odpojen. Přijde-li odpověď s významem „*Jsem naživu!*“, potom je uživatel připojen a proces tutora mu odešle režim tutora.

Uživatelův proces po přijetí režimu nastaví, kdo a jakým způsobem může se zařízením pracovat, vždy podle typu režimu. Tak například může odesílat do konzole jak data přijatá od uživatele, tak ze socketu připojeného k tutorovi.

Jestliže se tutor odpojuje, pošle jeho proces uživatelskému procesu zprávu ve významu „*Umírám!*“. Ten zruší veškerá omezení pro uživatele a přestane s procesem tutora komunikovat. Čeká jen na dalšího tutora a jeho zprávu s významem „*Jsi naživu?*“.

Odpojuje-li se uživatel a tutor je aktivní, odešle proces uživatelův procesu tutorovu zpráva s významem „*Umírám!*“. Tutor je pak odpojen a jeho proces také skončí.

4.1.6.3 Komunikační protokol mezi procesem tutora a uživatele

Mezi procesem tutora a uživatele jsou posílána data stejně jako mezi procesem uživatele a konzolí laboratorního prvku – tedy textově. Aby bylo možno posílat speciální zprávy zmíněné v předchozí kapitole, byl vybrán jeden znak, který signalizuje, že znak následující za ním má význam nějaké zprávy. Nazvěme tento speciální znak jako „řídící“. Tento řídící znak spolu s následujícím významovým znakem tvoří řídící zprávu.

Jako řídící byl vybrán znak s ASCII kódem 7, který se nazývá *„alert“* (výstraha) nebo *„bell“* (zvonec). Jeho escape sekvence je '\a'. Aby bylo možno mezi konzolí a tutorem posílat i tyto znaky, jsou tyto znaky v komunikaci nahrazeny speciální sekvencí „\a\a“, která přebírá původní význam. Používané zprávy jsou uvedeny v tabulce 2.

Zpráva	Význam
\a\a	Náhrada za běžný znak '\a'
\aa	Běží proces? (*Jsi naživu? = Are you alive?*)
\aA	Tento proces běží (*Jsem živý! = I'm alive!*)
\aD	Tento proces končí (*Umírám! = I'm dying!*)
\aE	Je požadován exkluzivní mód (*Exclusive*)
\aM	Je požadován řídící mód (*Master*)
\aO	Je požadován pozorovací mód (*Observer*)
\aS	Je požadován sdílený mód (*Shared*)

Tabulka 2: Tabulka řídících znakových sekvencí mezi procesy tutora a uživatele

4.2 Java Applet pro vzdálený přístup na konzole laboratorních prvků

Nutnou věcí, bez které by Virtlab neměl smysl, je java applet, který prostřednictvím konzolového serveru, zprostředkovává uživateli přístup ke konzolím laboratorních prvků přímo v jejich webovém prohlížeči. Skrze něho tak mohou uživatelé konfigurovat veškeré síťové laboratorní prvky, které byly namapovány na jejich žádanou topologii.

4.2.1 Původní funkcionalita

Applet, který používá distribuovaný Virtlab, byl napsán již Ing. Pavlem Němcem, v rámci jeho diplomové práce[1], pro prvotní Virtlab. Poté byl ještě vylepšen a zabezpečen v rámci diplomové práce Ing. Romana Kubína[3]. Na úpravách se podílel i Ing. Petr Grygárek, PhD. Applet byl určen pro komunikaci s původním konzolovým serverem. Nepoužíval žádné globálně platné identifikátory síťových prvků, nýbrž přímo výrazy, jichž význam značil nejdříve přímo sériový port, kde byla připojena konzole, později značil i IP adresu pro telnetové konzole. Ovšem na druhou stranu, díky použití komunikační vrstvy SSL, byla komunikace s konzolovým serverem výborně zabezpečena.

Protože bylo potřeba lehce pozměnit komunikační protokol appletu s konzolovým serverem a některé věci, jako například parametry předané webovým prohlížečem, upravit, byla pro lepší laditelnost SSL vrstva vyjmuta. Následně byly provedeny potřebné úpravy, které zajistily, že tento applet je použitelný s novým konzolovým serverem v distribuovaném prostředí.

Jak bylo již poznamenáno v kapitolách o konzolovém serveru (4.1), komunikace s tímto serverem je již autentizována užitím řetězců „session ID“, které identifikuje uživatele ve webovém rozhraní, ale stále není dokonalé. Proto po řádném otestování appletu i konzolového serveru je doporučeno opětovné zabudování podpory SSL, aby nemohlo dojít k narušení komunikace útočníkem.

4.2.2 Propojení s novým Cserverem

Konzolový java applet se s konzolovým serverem spojuje skrze TCP na IP adresu, která je appletu předána webovým prohlížečem, který ji načte z webové stránky Virtlabu pro spouštění úloh. Po připojení odešle applet konzolovému serveru argumenty, jak to bylo popsáno v kapitolách o konzolovém serveru, tak, že za každým argument přidá konec řádku ('\n'), tedy každý argument je poslán na zvláštním řádku.

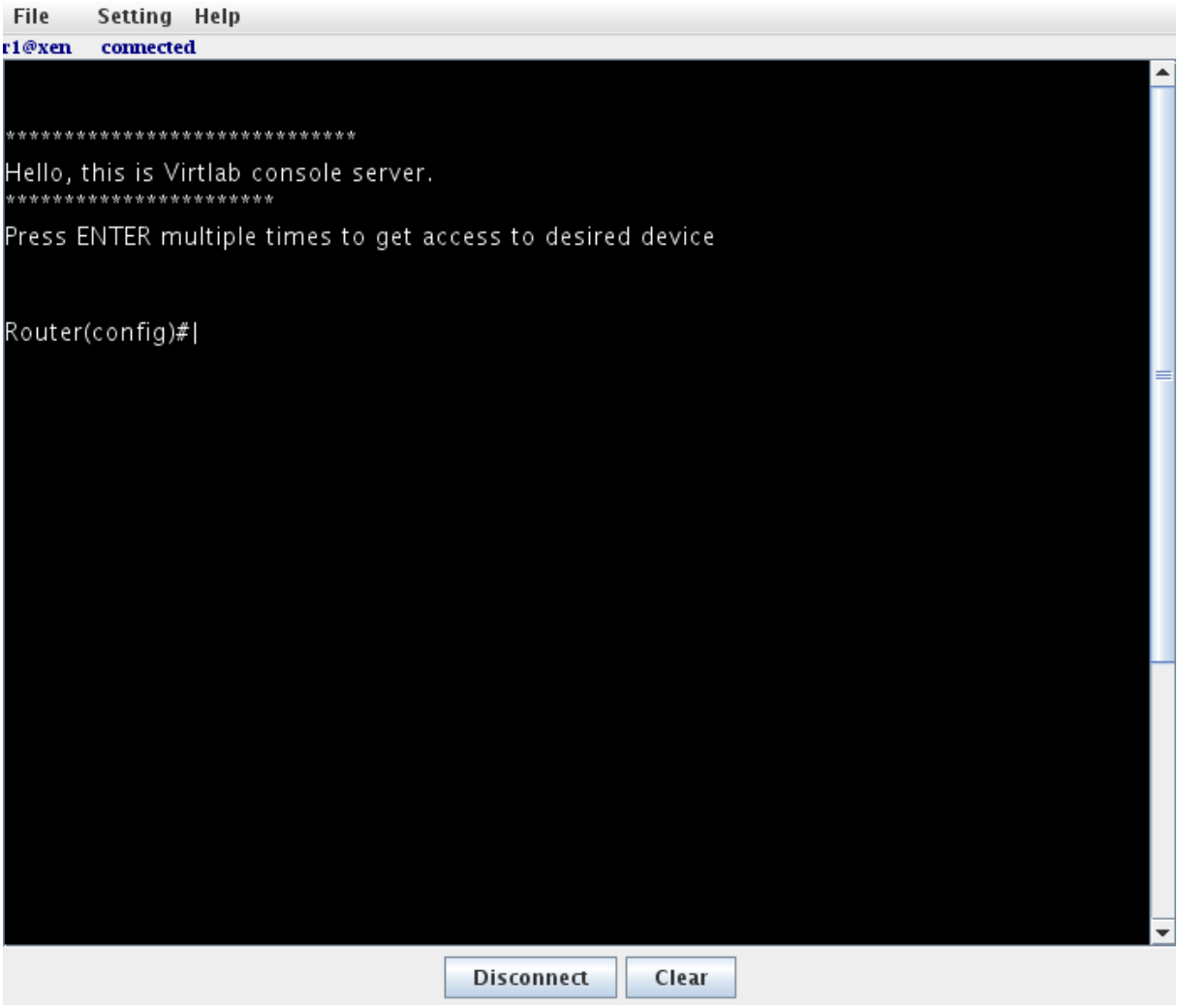
Konzolový server příjme tyto argumenty, ověří jejich platnost a to, že skutečně může k dané konzoli přistoupit applet z IP adresy, ze které je přihlášen, vše pomocí procesu popsaného v kapitole 4.1.5. Pokud není některý s argumentů platný, nebo daná IP adresa nemá povolen přístup, je applet od konzolového serveru odpojen.

Jinak začne probíhat znaková komunikace, kdy applet posílá uživatelské vstupy z klávesnice konzolovému serveru a ten naopak posílá výstup z konzole daného laboratorního prvku. Konzolový server může spolu s konzolovým výstupem posílat i svá vlastní hlášení uživateli.

Ihned po připojení je uživateli poslána uvítací hláška a když vyprší čas rezervace, je také touto cestou uživatel upozorněn. Rovněž připojí-li se tutor, je uživatel s touto událostí obeznámen a je mu posláno vysvětlení, co má očekávat. Po odpojení tutora následuje rovněž hlášení o této události.

4.2.3 Ovládání

Ovládání konzolového java appletu zůstalo stejné jako tomu bylo u starších verzí Virtlabu.[29] Rozdílem, který uživatel pocítí, je kompletní přeložení veškerých hlášení do anglického jazyka. To samozřejmě umožňuje nasazení v zahraničních laboratořích. Uživateli zůstala možnost přepnout do češtiny hlavní nabídku appletu. Dále, jak již bylo zmíněno, na blízké vypršení času rezervace není uživatel upozorněn výstražným dialogem, jak tomu bylo dříve, nýbrž přímo textem, který konzolový server přidá k výstupu laboratorního prvku. (Obrazovku s ovládacími prvky appletu ukazuje *obrázek 5*.)



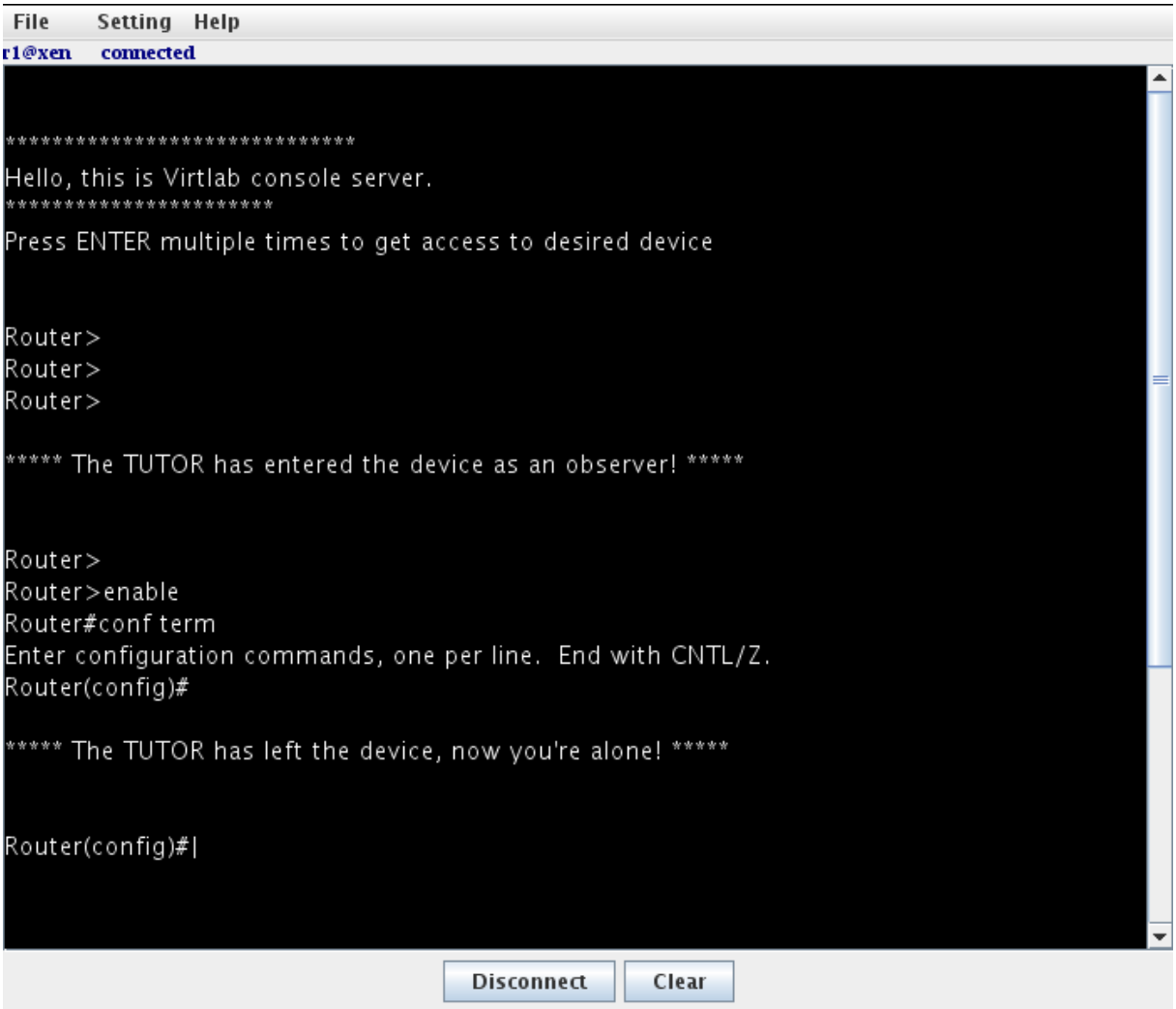
Obrázek 5: Počáteční obrazovka appletu

29 Tak, jak je popsáno v diplomových pracích Ing. Pavla Němce[1] a Ing. Romana Kubína[3]

4.2.4 Tutor

Tutor má applet zobrazen naprosto stejně jako uživatel, jen s tím rozdílem, že u režimu *observer* nemůže zadávat žádný vstup. Uživatel je při práci tutora na tuto činnost upozorněn textem přímo v emulovaném terminálu (viz *obrázek 6*).

Režim tutora si uživatel vybere ve webovém rozhraní Virtlabu. Režim je poté appletu předán jako jeden z parametrů.[30]



Obrázek 6: Hlášení v appletu po připojení a odpojení tutora

30 Webové rozhraní pro výběr režimu tutora je součástí diplomové práce Jana Vavříčka[5].

5 Samočinné zapojování topologií

V kapitole 2 Rezervační systém byl popsán způsob, jakým se systém dozví počátek a konec rezervace. Tento čas samozřejmě určuje i počátek, kdy má být zapojena uživatelem žádaná topologie, a čas konce platnosti této topologie. Popis topologie, jak bylo zmíněno, přiloží klient rezervačního serveru (výhradně webová aplikace) k dané rezervaci prostřednictvím místního rezervačního serveru. Ten tuto konfiguraci uchová pouze pro sebe a nedistribuuje ji vzdáleným rezervačním serverům, z čehož vyplývá, že on je jediným prvkem celé virtuální laboratoře, který může iniciovat samočinné zapojení topologie. Tedy iniciátorem spojení laboratorních prvků do žádané topologie je vždy rezervační server domovské laboratoře příslušného uživatele, který o rezervaci žádal.

Tato inicializace a zároveň kontrola času začátku rezervace je prováděna částí rezervačního serveru, která je nazvána „Aktivátor konfigurací“.

5.1 Aktivátor konfigurací

Rezervační server, jak již bylo zmíněno, obsahuje vlákno nazvané Aktivátor konfigurací. Toto vlákno po spuštění rezervačního serveru načte z databázové tabulky rezervací čas začátku nejblíže začínající rezervace. Poté si nastaví časovač[31] a vyčkává na tento čas.

Dojde-li v průběhu tohoto čekání v rezervacích k nějaké změně – to znamená že je přidána nebo zrušena nějaká rezervace – je aktivátor hlavním procesem rezervačního serveru z tohoto čekání vyrušen[32] a zkontroluje znovu, která rezervace začíná nejdříve. A pak znovu vyčkává na její začátek.

Není-li definována žádná rezervace, potom server pochopitelně čeká pouze na nějakou změnu provedenou v rezervacích.

Když nastane čas, na který Aktivátor konfigurací vyčkává, načte si Aktivátor z databázové tabulky podrobnosti o rezervacích, které v tento čas začínají (neboť jich pochopitelně může být více) a postupně, pro každou rezervaci zvlášť, volá speciální aktivační skript nazvaný „Activator-script“. Tomu předá jméno, čas konce dané rezervace a hlavně cesta k souboru se zapojením topologie.

Než je tento skript znovu volán pro další rezervaci, čeká se vždy na jeho ukončení. Paralelní spuštění několika instancí by totiž mohlo vést ke konfliktům.

Po spuštění aktivačního skriptu pro všechny rezervace, které začínají ve stejný čas, vyčká Aktivátor ještě nějakou chvíli v nečinnosti, aby byla jistota, že nahrávání veškerých konfigurací do všech prvků všech segmentů virtuálního spojovacího pole je již ukončeno. Potom se znovu načte čas nejbližší rezervace a Aktivátor vyčkává.

31 Časovač je implementován pomocí alarmového signálu SIGALARM
32 Čekání aktivátoru se přeruší pomocí prvního uživatelského signálu SIGUSR1

5.1.1 Aktivační skript

Aktivační skript je malý skript psaný pro BASH, jehož úkolem je na základě předaných argumentů, kterými jsou konec platnosti rezervace, název rezervace a cesta k souboru s popisem fyzického projení prvků, zavolat skripty, které vygenerují konfigurace pro virtuální spojovací pole, tunelovací servery a tyto konfigurace předat dále k nahrání do příslušných zařízení.

5.1.1.1 Generování konfigurací pro virtuální spojovací pole

Soubor se zapojením topologie, jehož cestu skript přijal jako argument, je zkopírován do souboru „topologie.conf“ umístěného v adresáři se skriptem, o němž je psáno dále. Tento soubor jako svůj hlavní vstupní soubor využívá skript pro generování konfigurací virtuálních spojovacích polí[33]. Tento skript je následně spuštěn. Jeho výstupem jsou soubory, které mají jméno ve tvaru „*název_virtuálního_spojovacího_pole-jméno_lokality*“, které mají být posléze nahrány do příslušných spojovacích prvků. Také vygeneruje soubor „tunelserver.out“, jenž slouží pro generování konfigurací tunelovacích serverů.

5.1.1.2 Generování konfigurace tunelovacích serverů

Do stejného adresáře jako předchozí skript byl v rámci této diplomové práce vytvořen skript „make-tun-conf“, jehož úkolem je pomocí souborů, které má buď předchozí zmíněný skript vytvořeny jako konfigurační – „site.conf“ nebo souborů, které vytvořil – tedy již zmíněný „tunelserver.out“, vygenerovat konfiguraci určenou tunelovacím serverům.

Výstupem toho skriptu jsou soubory se jmény ve tvaru „tun-*název_lokality*“, které mají být prostřednictvím řádkové konzole tunelovacího serveru každé z lokalit do nich nahrány.

5.1.1.3 Princip přiřazení čísel VLAN pro spojování laboratorních prvků

Protože virtuální spojovací pole pro spojování laboratorních zařízení provádí spojování ethernetových portů pomocí technologie VLAN, je nutné nějakým způsobem určit, jaká čísla VLAN mohou být pro tuto záležitost použita. Rozsah všech čísel, která mohou označovat VLAN[34] je rozdělen na menší souvislé rozsahy. Vždy jeden tento menší rozsah je přiřazen aktivačnímu skriptu do jedné lokality. Toto přiřazení je provedeno nastavením proměnných `VLANS_FROM` pro začátek rozsahu a `VLANS_TO` pro konec rozsahu přímo v těle skriptu. Tedy pro zapojení topologií, které byly vytvořeny nějakým uživatelem, jsou vždy použita čísla VLAN, která jsou přiřazena domovské lokalitě tohoto uživatele.

Pro přehled, která čísla jsou zrovna používaná a která nikoli, byl vytvořen speciální program nazvaný „Úložiště použitých čísel VLAN“, neboli „Vlanstore“. Ten je spuštěn na pozadí vždy těsně

33 Skript pro generování konfigurací virtuálních spojovacích polí „make-conn-configs“ pro Virtlab vytvořil vývojář Virtlabu Jiří Dvořák. Veškerá dokumentace k tomuto skriptu se nachází v úložišti zdrojových kódů (SVN), které je přístupné vývojářům Virtlabu.

34 VLAN ID je dvanáctibitové číslo, proto je možných 4094 kombinací. VLAN 1 je speciální, 0 nelze použít.

před spuštěním skriptu generujícího konfiguraci, neboť tento s ním musí komunikovat. Po skončení generátoru konfigurací je zase tento program ukončen. Toto úložiště je podrobněji popsáno v kapitole *5.2 Úložiště použitých čísel VLAN*. Komunikace mezi skriptem generujícím konfigurace a Úložištěm čísel VLAN se odehrává prostřednictvím souborů typu *roura*, jejichž cesta je uvedena v úvodní části těla skriptu v proměnných `FIFO_CONF2VLANSTORE`, pro komunikaci ve směru „skript do úložiště“, a `FIFO_VLANSTORE2CONF`, pro směr opačný. Tyto roury jsou kvůli dobré inicializaci na začátku smazány a vytvořeny znovu.[35] Způsob komunikace je rovněž blíže popsány ve zmíněné kapitole 5.2.

5.1.1.4 Nahrání konfigurace do virtuálních spojovacích polí

Poté, když jsou vytvořeny veškeré konfigurační soubory pro spojovací prvky virtuálního spojovacího pole i tunelovací servery, je spuštěn pomocný skript „uploadconfig.sh“, který je přiložen ve stejném adresáři, jako aktivační skript. Ten vezme veškeré určené konfigurační soubory a tyto posílá na konfigurační servery (viz kapitola 5.3) lokalit, jejichž jména jsou uvedena za spojovníky v názvech těchto souborů.

Kam mají být konfigurační soubory nahrány, určuje soubor „servers.conf“. Zde je na každém řádku uvedeno jméno jedné lokality a IP adresa konfiguračního serveru této lokality. Na tuto IP adresu jsou pak pomocí příkazu „netcat“[36] soubory odeslány. Popis komunikačního protokolu je uveden v kapitole 5.3.4.

5.1.1.5 Reinicializace laboratorních prvků

Jestliže má uživatel virtuální síťové laboratoře úspěšně používat nějaký laboratorní prvek, je důležité, aby v tomto prvku nezůstala jakákoli konfigurace po předchozích uživatelích a tedy aby měl nově příchozí uživatel k dispozici tento prvek „čistý“.

Proto v každé lokalitě běží mazač konfigurací („eraser“), který je v současnosti součástí Konfiguračního serveru. Tomu je vždy odeslán speciální příkaz, který řekne, že se bude mazat konfigurace, a dále název laboratorního prvku. Mazač pak sám zajistí, jakým způsobem bude dané zařízení reinicializováno. Podrobnější informace o této části Konfiguračního serveru v kapitole 5.3.3.

Aktivaci mazání pro jednotlivé prvky zajišťuje speciální skript zvaný Mazací aktivátor (Erase Activator), konkrétně soubor „activate-erase.sh“. Ten ze souboru s popisem topologie (onen soubor přiložený k rezervaci) nejprve zjistí jména použitých laboratorních prvků. Poté pomocí souboru „domains.conf“ přiřadí skript podle lokalit, kde se prvky nacházejí, k těmto prvkům IP adresy mazacích serverů. Formát tohoto souboru je jednoduchý: na každém řádku je uvedeno jméno lokality a mezerou IP adresa mazacího serveru.

Mazací server, jak již bylo zmíněno, je implementován v konfiguračním serveru. Lze zaznamenat, že zde jsou dva velmi podobné soubory, kde každý přiřazuje lokalitě IP adresu konfiguračního serveru a tedy by se mohlo zdát, že jde o informaci zdvojenou. Dva soubory jsou zde proto, poněvadž

35 Roura, neboli FIFO, je vytvořena příkazem *mkfifo*.
36 Příkaz „netcat“ musí být v prostředí operačního systému nainstalován!

konfigurační servery pro nahrávání konfigurací virtuálním spojovacím polím a pro mazání konfigurací nemusejí být obecně totožné a mohou běžet na různých strojích. Navíc je zde umožněna větší modularita pro případ, že bude mazací server vyčleněn z Konfiguračního serveru.

5.2 Úložiště použitých čísel VLAN

V kapitole 5.1.1.3 jsme se seznámili s principem přiřazování čísel VLAN pro účely virtuálních spojovacích polí a tunelovacích serverů. V každém okamžiku v každé lokalitě tedy dochází k situaci, kdy jsou některá čísla zabraná a jiná volná. Aby bylo možno tuto situaci zmapovat a skript generující konfigurace věděl která čísla je možno přiřadit, byl naprogramován program, který slouží jako úložiště použitých čísel VLAN a volná čísla umí nabízen zmíněnému skriptu. Jeho jméno je „Vlanstore“.

5.2.1 Spuštění a ukončení

Úložiště se spouští voláním binárního souboru „./vlanstore“. Spouští jej ovšem výhradně Aktivační skript. Je u něj nutno zadat čtyři povinné argumenty:

- *Datový soubor* – v něm jsou uložena použitá čísla VLAN a to včetně času konce používání[37]. Tento soubor může být textový i binární, což záleží na parametrech při kompilaci[38]. V textovém souboru je každá zápůjčka uvedena na novém řádku. Ten obsahuje vždy dvě čísla oddělená mezerou: číslo VLAN a čas konce platnosti zápůjčky.
- *Vstupní roura* – jedná se o soubor typu roura (fifo), pomocí které komunikuje skript generující konfigurace virtuálních spojovacích polí s úložištěm ve směru od skriptu do úložiště.
- *Výstupní roura* – opět soubor typu roura s podobným účelem jako předchozí, jen pro komunikaci v opačném směru od úložiště do skriptu.
- *Začátek rozsahu čísel VLAN* – první číslo v přiděleném rozsahu čísel VLAN, které může být použito a zapůjčeno.
- *Konec rozsahu čísel VLAN* – poslední čísla v přiděleném rozsahu čísel VLAN, které může být použito k zápůjčce.

Při spuštění programu je načten datový soubor s použitými čísly VLAN. Je-li v datech číslo, jehož platnost zápůjčky již vypršela, není načteno to tabulky použitých čísel a je ho tedy možné znovu použít.

Program se ukončuje pomocí signálu SIGTERM. Při přijetí tohoto signálu ukončí program otevřená spojení a uloží tabulku zapůjčených čísel VLAN do datového souboru. Čísla, jejichž platnost zápůjčky vypršela, nejsou uložena.

37 Celočíselný formát udávající počet vteřin od roku 1970
38 Je-li při kompilaci definováno makro `BINARY_DATAFILE`, bude soubor binární. Jinak je textový.

Aktivační skript spouští úložiště jen v době běhu skriptu generujícího konfigurace, protože jinak je jeho neustálý běh zbytečný. Toto navíc přináší lepší zabezpečení v případě náhlého výpadku elektrické energie, kdy je vlastně aktuální stav již uložen na disku a není tím pádem ohrožen. Další výhodu lze spatřit v možnosti využívat více rozsahů VLAN (aktivační skript třeba náhodně vybere jeden ze dvou možných rozsahů pro danou rezervaci apod.)[39], aniž by byly ohroženy stávající zápůjčky.

5.2.2 Zapůjčování čísel VLAN přes roury

Jak bylo několikrát naznačeno, probíhá zapůjčovaní čísel VLAN přes dva soubory typu roura, přičemž jeden je určen pro datový tok do úložiště a druhý pro směr opačný. Zapůjčení skrze tyto roury probíhá takto: klient, který chce zapůjčit nějaké číslo, odešle do roury čas, kdy končí rezervace, pro kterou má být číslo VLAN určeno. Tento čas je uveden v sekundách od 1. 1. 1970[40]. Za ním následuje konec řádku \n, ve formě LF. Na toto zareaguje úložiště odesláním čísla zjištěním nejmenšího volného čísla VLAN pro zapůjčení a textově jej odešle do druhé roury. Za číslem následuje také znak konce řádku. Zapůjčené číslo je přidáno do tabulky zápůjček.

Není-li možno zapůjčit žádné číslo VLAN, je zpět odeslána 0. Nula nemůže být nikdy použita jako VLAN id, a proto druhá strana pozná, že se jedná o oznámení chyby.

5.3 Konfigurační server

5.3.1 Stručný popis

Konfigurační server je ve Virtlabu naprosto novou komponentou, která byla vytvořena speciálně pro distribuovanou variantu. Úkolem konfiguračního serveru je převzetí konfiguračních souborů pro prvky místního segmentu virtuálního spojovací pole a skrze sériové či telnetové konzole jejich nahrání do příslušných zařízení. Rovněž umí inicializovat určené laboratorní prvky na implicitní konfiguraci – tedy vymazat konfiguraci, která v nich zbyla po předchozích uživatelích.

Konkrétněji se jedná o jednoduchý síťový démon, který naslouchá na daném portu[41], kde přijímá konfigurace prvků virtuálních spojovacích polí, čí příkazy k vymazání (reinicializaci) určených laboratorních prvků.

5.3.2 Nahrávání konfigurací prvků virtuálního spojovacího pole

Pro nahrávání konfigurací do prvků virtuálního spojovacího pole je třeba u konfiguračního serveru v každé lokalitě upravit konfigurační soubor „uploads.conf“. V něm jsou definovány prvky virtuálního spojovacího pole v místní lokalitě, jejichž konfigurace obsluhuje právě tento místní konfigurační server. Informace o každém tomto prvku jsou v souboru vždy na zvláštním řádku. Řádek začíná názvem místního prvku spojovacího pole. (Tento název je vždy totožný s názvem souboru,

39 Např. bude-li mít lokalita nedostatečný rozsah čísel VLAN pro úlohy a ten nebude možno spojitě rozšířit.
40 Standardní způsob počítání času pro počítače založené na platformě kompatibilní s IBM/PC
41 Implicitně 60001

který generují skripty pro vytváření konfigurací prvků spojovacích polí. Tedy ve formátu „jméno-lokalita“.[42]) Za jménem následuje mezera a adresa určující způsob připojení ke konzoli tohoto spojovacího zařízení. Způsob zápisu se shodný se způsobem, který požaduje skript „*uploader.sh*“, který zprostředkovává nahrávání konfigurací do spojovacích prvků. V použité verzi je syntaxe tato: buď může být přímo napsána cesta k sériovému portu, nebo IP adresa s dvojtečkou a portem, kam se má konfigurace odeslat.

Příjme-li konfigurační server konfiguraci pro některé virtuální konfigurační pole, spustí zmíněný skript „*uploader.sh*“, kterému je předán přijatý soubor spolu s cestou uvedenou v konfiguračním souboru. Není-li spojovací prvek, pro který je přijímána konfigurace, uveden ve zmíněném v konfiguračním souboru, neprovede se nic.

Výše zmíněný nahrávací skript ještě přidá na začátek přijatého souboru konfigurační příkazy, které jsou nutné pro nastavení daného prvku do konfiguračního režimu, kde je možno konfiguraci pak nahrát, a nakonec přidá příkazy, které konfiguraci ukončují[43]. Předkonfigurační a pokonfigurační příkazy jsou uloženy v souborech, jejichž název je stejný jako název daného prvku, na konec názvu je ještě přidán řetězec „*.pre*“ pro předkonfigurační příkazy a „*.post*“ pro pokonfigurační příkazy. Pro přímé nahrání výsledného souboru do zařízení je pak volán program „*upload-conndevice-cfg*“, který je převzat z předchozí verze Virtlabu. Jeho autorem je Ing. Petr Grygárek, PhD.

Ukázka souboru „uploads.conf“:

```
C3550-VSB1 /dev/ttyS2
C3550-VSB2 /dev/ttyS3
```

5.3.3 Mazání konfigurací síťových prvků

Funkce mazání konfigurací síťových prvků v rezervačním serveru je základním kamenem způsobu reinicializace laboratorních prvků, o které mluví kapitola 5.1.1.5. Aby po předchozích uživatelích nezůstávala v laboratorních prvcích dřívější uživatelská konfigurace, bylo nutné nalézt způsob, jak nahrát do všech prvků dané rezervace nějakou implicitní konfiguraci a to navíc s ohledem na typ prvku.

Prvkem může být jednoduchý přepínač nebo router, kde stačí do textové konzole odeslat několik příkazů, nebo tím prvkem může být virtuální počítač, případně i jiný emulátor nějakého prvku, jehož implicitní konfigurace se nastaví restartem.

Klient konfiguračního serveru, což je v tomto případě výhradně skript „erase-activator“, který je volaný aktivátorem konfigurací, zvláštním příkazem řekne, že chce vymazat konfiguraci nějakého laboratorního prvku a předá jeho jméno. Z důvodu modularity a možnosti semi-paralelního mazání, konfigurační server následně spustí skript, který se jmenuje „eraser.sh“, jemuž předá jméno

42 Podle oddělujícího znaménka lze takto jednoduše poznat, že se nejedná o laboratorní prvek.

43 Vygenerovaná konfigurace neobsahuje hesla ani příkazy pro nastavení konfiguračního režimu síťového prvku virtuálního spojovacího pole.

laboratorního zařízení a jehož úkolem je najít způsob vymazání, respektive reinicializace prvků, ale server nečeká na ukončení tohoto skriptu, což značně urychluje práci při vymazávání více prvků zároveň.

5.3.3.1 Mazací skript „eraser.sh“

Tento skript je standardně umístěn v podadresáři „utils“ adresáře konfiguračního serveru. Pro konfiguraci je použit soubor „device-erasers.conf“. Zde je vždy na každém řádku uvedeno jméno místního laboratorního prvku a za ním příkaz (celý zbytek řádku), kterým se konfigurace daného prvku vymaže na implicitní (reinicializuje se). Tento příkaz je skriptem zavolán.

Standardním způsobem reinicializace je nahrání nějakého inicializačního konfiguračního souboru skrze sériovou nebo telnetovou konzoli daného prvku. K tomu se používá již jednou zmíněný program „upload-conndevice-cfg“. V případě restartu nějakého virtuálního počítače se místo nahrávání zavolá skript, který tento restart provede.

Ukázka souboru „device-erasers.conf“:

```
r1@vsb utils/upload-conndevice-cfg /dev/ttyM1 erasers/erase-cisco 100
sw1@vsb utils/upload-conndevice-cfg 10.0.0.1:23 erasers/erase-cisco 100
uml1@vsb erasers/erase-uml 1
xen1@vsb erasers/erase-xen 1
```

5.3.4 Popis protokolu

Protokol, kterým komunikuje konfigurační server se svým klientem je velmi jednoduchý. Předávají se pouze textová data. První řádek určuje jméno prvku segmentu virtuálního spojovacího pole a další obsah za ním patří již konfiguračnímu souboru.

Pokud jako název prvku virtuálního spojovacího pole uvedeme slovo „erase“, přepne se konfigurační server do mazacího módu a očekává od klienta již příjem jen jediného řádku, kterým je název laboratorního prvku, jehož konfigurace má být vymazána.

Ukázka žádosti o nahrání konfigurace do prvku virtuálního spojovacího pole:

```
C3550-vsb
tajne_heslo
enable
tajne_heslo
conf_term
...
```

Ukázka žádosti o vymazání konfigurace laboratorního prvku:

```
erase
r1@vsb
```

6 Závěr

V práci jsem se zaměřil na obecné navržení principů distribuované virtuální síťové laboratoře a to s ohledem na jednotlivé komponenty celého systému podle jejich funkce, poté i na jejich reálnou implementaci. Nejprve jsem určil možná principiální řešení funkčnosti daných oblastí. Zejména se jednalo o princip vzdáleného propojení síťových ethernetových segmentů založených na technologii tunelování VLAN, dále způsob provádění rezervací prostředků – laboratorních prvků a nakonec zpřístupnění konzolí těchto prvků uživatelům. Zde bylo nutno analyzovat již hotové komponenty stávající virtuální laboratoře počítačových sítí a určit, je-li z nich možno vycházet při budování nového ovládacího softwaru, nebo jestli je nutné naprogramovat tyto prostředky nově. Při analýze bylo zjištěno, že je možno využít většinu zdrojového kódu stávájícího Konzolového serveru i Konzolového java appletu pro zpřístupnění konzolí laboratorních prvků bez nejmenších problémů. Jako problematický se ukázal původní rezervační systém, jehož základní filosofie se ukázala zcela nevhodná a bylo tedy nutno napsat naprosto nové softwarové řešení pro tento úkol. Po menších úpravách, které na žádost provedl člen vývojového týmu Virtlab Jiří Dvořák, se ukázal jako výborně použitelný původní skript generující konfigurace pro virtuální spojovací pole. Ovšem skripty, které tyto konfigurace nahrávaly do příslušných zařízení nebylo v distribuovaném systému možno použít a byly nahrazeny nově naprogramovaným Konfiguračním serverem. Tomu byla posléze přidělena i funkce zprostředkovatele mazání konfigurací v laboratorních prvcích. Při hledání řešení způsobu tunelování VLAN byla zkoumána různá řešení, ovšem po důkladné analýze jsem dospěl k závěru, že bude tento software nutno rovněž naprogramovat. Zde bylo nutno určit způsob, jakým bude tunelování probíhat a jak lze toto naprogramovat. Nalezené výsledky jsou rovněž předloženy v této práci.

Vše bylo nutno důkladně konzultovat s diplomantem Janem Vavříčkem, jehož úkolem byla realizace především uživatelského prostředí, návrh formátu pro popis virtuální topologie, vytvoření software, který provádí namapování této topologie na fyzické prvky. Dále vytvoření formátu pro popis laboratorního vybavení pomocí XML.

Pro reálné nasazení je potřeba důkladně všechny komponenty otestovat, jakož i prozkoušet vzájemnou komunikaci a interakci. Dále bude třeba se zaměřit na zabezpečení, jemuž nebyla přiřazena při implementaci nejvyšší priorita. Po tomto může dojít k reálnému nasazení v univerzitách a školících zařízeních, které projevily již zájem o účast ve Virtlabu. V budoucnu bude možná vhodné pozměnit stávající způsob generování konfigurací pro zapojení topologií a distribuci těchto konfigurací tak, aby konfigurace pro všechna virtuální spojovací pole nebyla vytvářena domovskou lokalitou uživatele a následně distribuována, ale aby každá lokalita si tyto konfigurace vytvářela sama, což přinese větší flexibilitu i nezávislost jednotlivým lokalitám.

Na závěr chci poděkovat všem, kteří přispěli k realizaci celého projektu, především jeho vedoucímu Ing. Petru Grygárkovi, PhD. Dále kolegovi Janu Vavříčkovi a také všem zúčastněným v projektu Virtlab děkuji za spolupráci.

Literatura

[1] Pavel Němec, *Diplomová práce-Virtuální síťová laboratoř*, VŠB-TU Ostrava, 2005

[2] David Seidl, *Diplomová práce-Automatizovaný systém správy síťových konfigurací*, VŠB-TU Ostrava, 2005

[3] Roman Kubín, *Diplomová práce-Zajištění bezpečnosti a implementace nových prvků řídícího systému virtuální laboratoře*, VŠB-TU Ostrava, 2006

[4] Ing. Petr Grygárek, PhD., Terminologie distribuovaného Virtlabu, VŠB-TU Ostrava, 2007, http://www.cs.vsb.cz/vl-wiki/index.php/Virtlab:DistrTerminologie

[5] Jan Vavříček, *Diplomová práce-Rozvoj řídícího software virtuální laboratoře počítačových sítí*, VŠB-TU Ostrava, 2007

[6] Ing. Petr Grygárek, PhD., *Počítačové sítě*, VŠB-TU Ostrava, http://www.cs.vsb.cz/grygarek/PS/index.html

[7] Richard Stones, Neil Matthew, *Linux – začínáme programovat*, Computer Press, 2000, ISBN: 80-7225-307-2

[8] Ing. Petr Olivka, *Operační systémy*, VŠB-TU Ostrava, http://poli.cs.vsb.cz/edu/osy/

[9] Institute of Electrical and Electronics Engineers, Inc., *IEEE 802.1q*, http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf

[10] Andreas Schaufler, Linux Network Performance: Raw ethernet vs. UDP, http://www.landshut.org/bnla01/members/Faustus/fh/linux/udp_vs_raw/index.html

[11] Linux manual pages, klíčová slova: *„ip“, „socket“, „raw“, „netdevice“*

[12] Radim Dostál, Sockety a C/C++, Intenet Info, http://www.root.cz/serialy/sokety-a-cc/

Příloha A

Ukázka konfiguračního souboru pro rezervační server „rsvsrv.conf“

```
location vsb

mysql root@localhost:virtlab
Tajne Heslo

timetable tt1
sunday 0-24
monday 0-24
tuesday 0-24
wednesday 0-24
thursday 0-24
friday 0-24
saturday 0-24
timetable tt2
tuesday 18-24
wednesday 16-24
friday 0-24

timetable tt2
tuesday 16-18
friday 10-20

virtlab vsb 127.0.0.1
r1@vsb tt1
r2@vsb tt1

virtlab fila 168.172.32.145
r1@vsb tt1
r2@vsb tt2

conf-activator /home/tomas/virtlab/activator-script/activate.sh
```

Příloha B

Výpis souboru „equipment.dtd“, který je použit rezervačním serverem (kapitola) pro validaci XML souborů s popisem laboratorních prvků dané lokality[44]

```
<!ELEMENT   equipment    (device*)>
<!ELEMENT   device       (os, interfaces?, special?)>
<!ELEMENT   os           (#PCDATA)>
<!ELEMENT   interfaces   (interface+)>
<!ELEMENT   interface    (max_bps?, int_feature*)>
<!ELEMENT   max_bps      (#PCDATA)>
<!ELEMENT   int_feature  (#PCDATA)>
<!ELEMENT   special      (feature+)>
<!ELEMENT   feature      (#PCDATA)>

<!ATTLIST   device
            type             (router | switch)          #REQUIRED
            name             CDATA                      #REQUIRED
            serial_number    NMTOKEN                    #REQUIRED
            platform         CDATA                      #REQUIRED>

<!ATTLIST   interface
            technology       (serial | ethernet)        #REQUIRED
            ether_type       (legacy | fast | gigabit)  #IMPLIED
            connect_group    NMTOKEN                    #REQUIRED
            name             CDATA                      #REQUIRED>
```

44 Soubor je vytvořen Janem Vavříčkem v rámci jeho diplomové práce[5].

Příloha C

Podrobné vysvětlení způsobu příjmu syrových ethernetových rámců v jazyce C pod OS Linux

Mějme definovány proměnné:

```
int socket;
char *interface; //obsahuje nazev rozhrani – napr. „eth1“
struct sockaddr_ll srv_addr,client_addr;
```

Vytvoření zmíněného socketu pro příjem „syrových“ rámců provedeme takto:

```
sock_listen = socket( PF_PACKET, SOCK_RAW, htons(ETH_P_ALL) );
```

Poté se ze slovního názvu síťového zařízení (například „eth1“) zjistí jeho systémové číslo:

```
struct ifreq eth_if, eth_flags;
strcpy( eth_if.ifr_name, interface );
strcpy( eth_flags.ifr_name, interface );

if(ioctl(sock_listen, SIOCGIFINDEX, &eth_if, sizeof(eth_if)) == -1)
      {/*...Zpracovani chyby ioctl...*/}
if(eth_if.ifr_ifindex == 0)
      {/*...Zpracovani chybne zadaneho rozhrani}
```

Rozhraní se nastaví do promiskuitního režimu:

```
if(ioctl(sock_listen, SIOCGIFFLAGS, &eth_flags, sizeof(eth_flags)) == -1)
      {/*...Osetreni chyby ioctl...*/}

original_interface_flags = eth_flags; //puvodni nastaveni sitoveho rozhrani

// Pridani priznaku pro promiskuitni rezim.
eth_flags.ifr_flags |= IFF_PROMISC;
if (ioctl(sock_listen, SIOCSIFFLAGS, &eth_flags,  sizeof(eth_flags)) == -1)
      {/*...Osetreni chyby při prepinani do promiskuitniho modu...*/}
```

Příprava struktury, která nastavuje parametry příjmu pro příkaz „*bind*“:

```
//nastaveni paketove rodiny pro prijem
srv_addr.sll_family = AF_PACKET;
srv_addr.sll_protocol = htons(ETH_P_ALL); //misto ETH_P_8021Q prijimame vse
//nastaveni cisla sitoveho rozhrani
srv_addr.sll_ifindex = eth_if.ifr_ifindex;
```

Nakonec je socket svázán s připraveným síťovým rozhraním a nastaven příjem „syrových“ paketů:

```
// prirazeni adresy a portu socketu
if(bind(sock_listen, (struct sockaddr*) &srv_addr, sizeof(srv_addr)) < 0 )
{/*...Osetreni chyby bind...*/}
```

Nyní máme socket připravený pro příjem „syrových“ ethernetových rámců i s hlavičkou. Příjem jednotlivých rámců se poté provádí pomocí:

```
bytesRead=recvfrom(sock_listen,buf,BUF_SIZE,0,(struct sockaddr*)
&client_addr,(socklen_t *) &fromLen);
```

To, že se jedná o rámec označený zmíněnou normou poznáme následovně:

```
if(client_addr.sll_protocol == htons(ETH_P_8021Q))
...
```

Vlastní zpracování rámce:

```
vlan_header = (struct vlanethhdr*)&buf;
```

Pro možnost porovnávání VLAN ID s hodnotou uloženou v typu „int“, musíme nad položkou struktury „vid“ provádět operace bitového posunu:

```
vlan_header->vid >> 4
```