VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace sledování provozu na VLAN ve virtuální laboratoři počítačových sítí pomocí tethereal/tcpdump

2008 **Radek Novák**

Rád bych na tomto místě poděkoval mému vedoucímu Ing. Martinu Milatovi za cenné připomínky a nápady, které mi pomohly vylepšit a zdokonalit výslednou podobu mé práce.

Prohlášení

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

V Ostravě 6. května 2008 …….

Abstrakt

Smyslem této práce je rozšíření virtuální laboratoře počítačových sítí o schopnost sledování síťového provozu, umísťování virtuálních sond a prezentování výsledků pomocí webového rozhraní. Součástí této práce je vytvoření programů pro umísťování virtuálních sond, zachytávání síťového provozu a vytvoření uživatelského rozhraní pro ovládání virtuálních sond a prezentaci výsledků.

Klíčová slova: Virtuální síťová laboratoř, Virtlab, Sledování síťového provozu

Abstract

Purpose of this work is extends the Virtual Computer Network Laboratory with ability to monitor network traffic, positioning of virtual probes and presentation of results through web interface. This work includes creation of programs for virtual probes positioning, capturing of network traffic and creating of user interface for controlling of virtual probes and presentations of the results.

Keywords: Virtual network laboratory, Virtlab, Network traffic monitoring

Seznam použitých zkratek a symbolů

IP	Internet Protocol
PHP	Personal Home Pages
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WWW	World Wide Web
IDS	Intruders Detection System
IPS	Intruders Protection System
ASSSK	Automatizovaný systém správy síťových konfigurací
Q-in-Q	Tunelování 802.1q rámců v 802.1q rámcích
GUI	Graphical User Interface
API	APplication Interface
NIC	Network Interface Card
HTTP	HyperText Transfer Protocol

Obsah

1 Úvod

1.1 Virtuální síťová laboratoř – Virtlab

Projekt s názvem Virtuální síťová laboratoř (zkráceně Virtlab) původně vznikl z důvodů potřeby vzdáleně zpřístupnit zařízení, která se nacházejí ve školních laboratořích pro výuku předmětu zabývajících se počítačovými sítěmi, zejména pak pro potřeby studentů, kteří nemají možnost osobně navštěvovat zmíněné laboratoře.

Umožňuje studentům pomocí webového rozhraní vytvořit si vlastní síťovou topologii, umístit do této topologie aktivní prvky nebo koncové stanice a pomocí téhož webového rozhraní k těmto prvkům přistupovat a nastavovat je.

1.2 Původní návrh Virtuální síťové laboratoře

Nejstarší verze Virtlabu měla jedno veliké omezení. Jestliže bylo potřeba změnit topologii, musel člověk, který byl zodpovědný za provoz, jednotlivé prvky v dané lokalitě ručně propojit do nově požadované topologie.

Toto omezení se nakonec podařilo překonat pomocí zařízení ASSSK1. Toto zařízení dokáže propojovat sériové linky na základě příkazů, které jsou mu předávány. Zapojení určité topologie lze tedy zautomatizovat a není nutné topologii ručně přepojovat vždy, když někdo zažádá o její změnu.

Ačkoli zařízení ASSSK1 dokáže přepojovat sériové linky, byla nakonec použita jiná technika, aby se množství propojených linek nesnižovalo. Zmíněné propojování ethernetu se realizuje pomocí techniky zvané Q – in – Q na Cisco přepínačích řady 3550. Tato technika, jak už zkratka napovídá, je založena na technologii 802.1q, která umožňuje použití VLAN.

VLAN – neboli virtuální lokální síť je možnost, jak logicky od sebe oddělit broadcastové domény. Díky tomu můžeme ušetřit za aktivní prvky při stavbě sítě a nebo v našem případě umožnit nesnižování počtu sériových linek na zařízení ASSSK1.

Vzniká přidáním dvou bajtů do ethernetové hlavičky, pomocí níž lze poté identifikovat jednotlivé VLAN na trunk – linkách nebo v koncových zařízeních.

Díky technologii Q – in – Q dokážeme taktéž rozpoznat jednotlivá logická zařízení.

1.3 Nový návrh Virtuální síťové laboratoře

Jedním z omezení původní verze Virtbalu je, že v jeden okamžik může běžet pouze jediná úloha a to i kdyby byl dostatek prostředků pro provedení více podobných úloh.

V původní architektuře Virtlabu se nemluví o logickém zařízení které má daný počet rozhraní, ale o fyzickém zařízení, které daná úloha používá. Není tudíž možno toto zařízení využít více studenty. Proto je pro nás výhodnější mít zařízení popsané jako logické a až při vytváření topologie tato zařízení namapovat na zařízení fyzická tak, aby byly splněny všechny požadavky topologie. Popřípadě oznámit uživateli, že se daná topologie nedá v daný okamžik vytvořit.

Dalším omezením původní verze Virtlabu je, že zařízení (např. Cisto Catalyst 3550), která Virtlab používá, jsou poměrně drahá a ne každá lokalita si je může dovolit. Popřípadě, pokud se v jedné lokalitě nachází takovéto drahé zařízení a není téměř používáno a v další lokalitě se nachází někdo, kdo by toto zařízení mohl použít do své topologie, pak by bylo vhodné, kdyby mohly lokality toto zařízení sdílet.

Všechna tato omezení vyústila ve vytvoření nové verze Virtlabu, která nese název *Distribuovaná virtuální síťová laboratoř*. Tato nová architektura s sebou přinesla nové možnosti, ale bylo potřeba vyřešit některé problémy. Například jak budou probíhat rezervace mezi více lokalitami nebo jak směrovat provoz z jedné lokality do lokality druhé. Většina z těchto problémů byla vyřešena v rámci jiných diplomových prací a v současné době je Distribuovaný Virtlab funkční mezi lokalitami v Ostravě a v Karviné a dokáže plnit požadavky studentů.

1.4 Cíl této diplomové práce

Cílem této práce je implementovat do Distribuovaného Virtlabu novou možnost pro uživatele. Jedná se o sledování síťového provozu, což je velice užitečná vlastnost, která se využívá v reálných sítích nebo při výuce, kdy se dá analyzovat průběh síťové komunikace (např. stahování webových stránek pomocí protokolu HTTP).

Cílem práce je implementovat sondy, pomocí kterých mohou uživatelé monitorovat provoz na rozhraních zařízení, se kterými pracují a tento zachycený provoz později analyzovat v některém z mnoha dostupných analyzátorů.

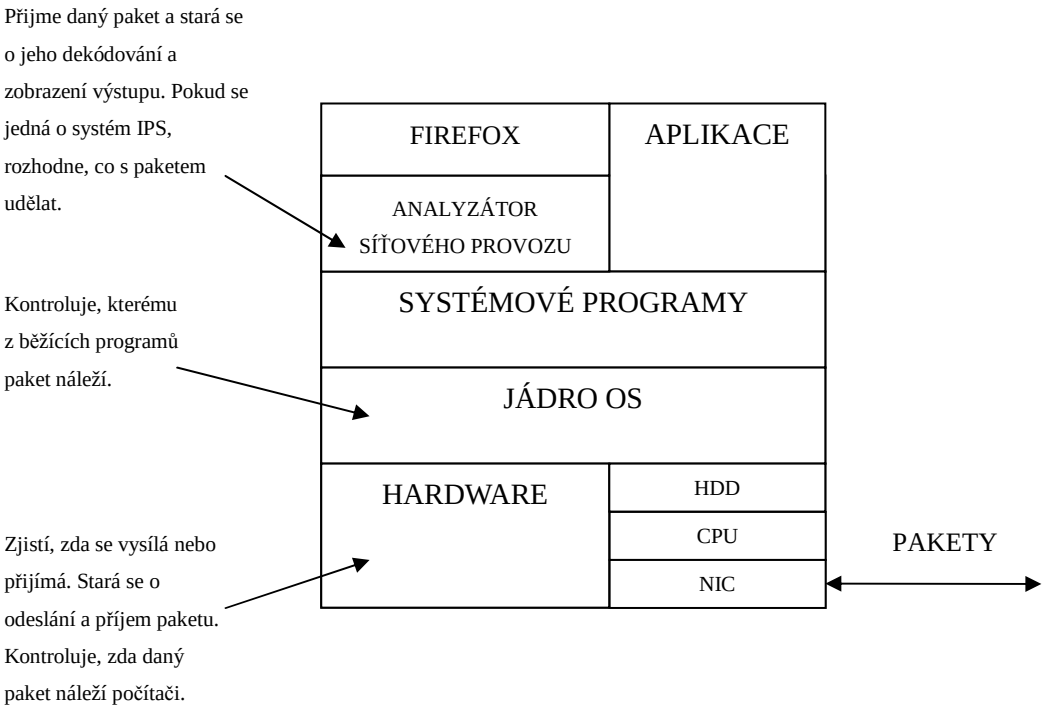
2 Sledování síťového provozu

2.1 Využití sledování síťového provozu

Sledování síťového provozu je velice užitečná věc, pomocí níž můžeme zjistit, jaká data putují v síťových paketech. Díky tomuto můžeme odhalit chyby v počítačové síti, dozvědět se více o různých síťových protokolech, odhalit chyby v našich programech, které mají komunikovat přes síť, odhalit různé škodlivé programy, které se pokoušejí z našeho počítače vysílat důvěrná data. Jako správci sítě můžeme kontrolovat, jak uživatelé využívají síť a jestli ji například nevyužívají k jiným účelům než mají. Sledování síťového provozu taktéž využívají programy, které fungují jako IDS a sledují různé pokusy o proniknutí do naší sítě nebo programy, které fungují jako IPS, které se těmto průnikům snaží zabránit.

2.2 Zachycení paketu a jeho přesun k analyzátoru

K sledování síťového provozu potřebujeme program, který je schopen komunikovat s jádrem operačního systému a zajistit, aby byl každý příchozí nebo odchozí paket celý zkopírován a odeslán aplikaci, která se stará o analýzu síťového provozu. Cestu paketu operačním systémem od zachycení na NIC až po jeho doručení analyzátoru lze shrnout do tří částí, jak lze vidět na obrázku č.1.



Obrázek 1: Cesta paketu operačním systémem

2.2.1 Zachycení paketu na NIC

Síťová karta se musí postarat o správný příjem nebo odeslání paketu a jeho předání jádru operačního systému. Pokud rámec přijímáme, tak musíme poznat, kdy vůbec na síti někdo vysílá a rozpoznat, kde začíná a kde končí paket.

Další věc, kterou síťová karta musí udělat, je rozlišit, zda daný paket skutečně náleží síťové kartě a nebo se jedná o paket který patří někomu jinému. S takovým rámcem pak nemáme co dělat a síťová karta takovéto rámce ignoruje. Pokud analyzujeme síťový provoz, pak není takovéto chování pro nás zrovna ideální. Často potřebujeme sledovat provoz mezi různými stroji, tj. komunikaci, která se nás vůbec netýká. A pokud by síťová karta pakety ignorovala, tak bychom neměli co analyzovat. Tento problém se řeší přepnutím síťové karty do tzv. *promiskuitního módu.* V takovémto módu přijímá síťová karta všechny pakety. Musíme však počítat s tím, že operačnímu systému vznikne větší režie, protože bude muset zpracovávat i pakety, které danému počítači nenáležejí.

2.2.2 Zpracování paketu jádrem OS

Po kontrole, zda daný paket patří skutečně nám, předá síťová karta paket jádru operačního systému. Jádro operačního systému poté dekóduje hlavičky komunikačních protokolů, které mohou být v paketu obsaženy. Podle protokolů poté jádro OS rozhodne, co s paketem dále udělá. Pokud je v paketu obsažen protokol transportní vrstvy, pak musí jádro zkontrolovat, kterému z běžících programů data, která jsou nesena v paketu, patří a tomuto programu data předat. Popřípadě, pokud se jedná o transportní protokol, který zabezpečuje doručení dat ve správném pořadí a kontroluje, zda nejsou data poškozena, pak musí jádro OS tyto kontroly korektně provést. Cílová aplikace pak dostane jen data, která jí byla určena a to bez hlaviček síťových protokolů.

Pokud analyzujeme síťový provoz, pak potřebujeme paket v nezměněné formě, tj. v takové, v jaké jej dostalo jádro OS od síťové karty. Musíme tedy upozornit jádro OS, že máme zájem, aby nám byly přeposlány všechny pakety, které jádro OS dostalo ke zpracování a abychom je dostali v původní formě. Jednou z možností, které k tomu můžeme využít, je naslouchání na tzv. *raw socketech*, což je speciální druh socketu, pomocí kterého říkáme jádru OS, že chceme, aby nám poslal všechny pakety a to v původní formě, v jaké je dostalo od síťové karty.

2.2.3 Analýza přijatého paketu

Po přijetí paketu z jádra OS se program pro analýzu síťového provozu postará o dekódování paketu. Zopakuje celou proceduru, kterou udělalo jádro OS a síťová karta. Tj. postupně odstraňujeme hlavičky protokolů, tyto hlavičky analyzujeme a podle informací v nich obsažených pokračujeme dále. Pokud je daný program IPS, pak musí po analýze paketu rozhodnout, co s paketem dál a upozornit zpětně jádro OS, co má s paketem udělat.

2.3 Dekódování paketu analyzátorem

Úkolem analyzátoru je paket správně dekódovat a dekódované informace správně zobrazit. Dalším z úkolů je, správně si uspořádat a ukládat příchozí pakety. Pokud například analyzujeme 100 Mb linku, pak za vteřinu můžeme očekávat až 144 000 paketů, což už je velmi vysoké číslo a bez důkladných optimalizací analyzátoru je téměř nemožné toto množství analyzovat v reálném čase. Samotná analýza paketu by se dala rozdělit do dvou částí.

2.3.1 Dekódování nižších protokolů

Pod pojmem dekódování nižších protokolů si můžeme představit dekódování všech protokolů krom dekódování protokolů aplikační vrstvy. Toto dekódování dokážeme provést velice rychle. Jelikož hlavičky většiny těchto protokolů jsou pevně dány, takže nám stačí postupně od počátku odebírat potřebné počty bitů a na tyto bity pak použít šablonu námi požadovaného protokolu.

2.3.2 Dekódování protokolů aplikační vrstvy

Protokolů aplikační vrstvy je velmi mnoho od známých až po téměř neznámé. Mnohé z těchto protokolů nejsou veřejně přístupné, mnohé nejsou téměř vůbec popsané, některé v sobě mohou dokonce obsahovat další protokoly a některé jsou tak barvité, že je velice těžké analyzovat všechny možnosti.

Právě tato část síťové analýzy je nejvíce náročná a je nejvíce oceňovaná. Díky ní se můžeme dozvědět velkou spoustu informací o různých síťových protokolech nebo oddělit hlavičky protokolů aplikační vrstvy a podívat se na samotná data, která jsou v paketu přenášena.

Tuto část nejčastěji analyzují systémy IPS, které se pokoušejí zjistit nejasnosti v hlavičkách aplikačních protokolů nebo najít vzor v přenášených datech.

Podívejme se nyní, jaké metody můžeme využít pro dekódovaní obsahu:

- **Metody založené na získání informací z předchozích protokolů**

  Díky informacím, které získáme dekódováním protokolů nižších vrstev, můžeme odhadnout, jaká data aplikační vrstvy očekáváme. Například pokud dekódujeme jako protokol transportní vrstvy protokol TCP nebo UDP, tak ten obsahuje tzv. *číslo portu*, což je vlastně číslo, které identifikuje program, kterému mají být data doručena. Pomocí tohoto čísla a díky tzv. *dobře známým portům*, což jsou čísla nejčastěji používaných služeb, víme, jaká data budeme dekódovat. Problém nastane, pokud na daném portu přenáší data jiná služba než čekáme. Pak může dojít k chybám při dekódování.

- **Metody založené na regulérních výrazech a hledání vzorů**

  Pokud netušíme, jaký protokol je přenášen, můžeme vyzkoušet najít na začátku paketu vzor, který by nás navedl k tomu, jaký protokol je v datech obsažen. Lze toho například docílit používáním regulérních výrazů. Tato analýza je však časově náročnější a bez optimalizace se nehodí pro reálný provoz. Dalším případem takové analýzy je, pokud se pokoušíme v paketu najít vzor, který odpovídá námi danému předpisu. Toho s oblibou využívají systémy IPS, které pomocí toho zabraňují doručení nebo odeslání paketů se škodlivým obsahem.

2.4 Dostupné analyzátory

Analyzátory se dají rozdělit na komerční a volně šiřitelné. Komerčními se v této práci zabývat nebudeme. Pro potřeby Virtlabu potřebujeme volně šiřitelný analyzátor, který si mohou uživatelé sami opatřit a pracovat s ním.

Mezi největší výhody komerčních analyzátorů patří jejich veliká podpora protokolů aplikační vrstvy, veliká optimalizace, díky které je možné je nasadit na reálný provoz a to i na velice rychlé linky, webové rozhraní, možnost sledovat odděleně více lokalit nebo například možnost vzdáleně tyto analyzátory spravovat. Velikou nevýhodou je jejich cena, která se běžně může pohybovat kolem částky 5000 €.

Volně šířitelné analyzátory oproti tomu nenabízejí velkou podporu protokolů aplikační vrstvy, nejsou optimalizovány pro reálný provoz na rychlých linkách a podobně. Avšak jsou zdarma a mají často volně šířitelné zdrojové texty. Díky tomu je můžeme přizpůsobit svým požadavkům nebo sami přidat podporu pro protokoly aplikační vrstvy. Drtivá většina volně šiřitelných analyzátorů je postavena na volně šiřitelné knihovně PCAP nebo její obdobě WinPCAP.

2.4.1 Knihovna PCAP

Knihovna PCAP je volně šiřitelnou knihovnou, která nám umožňí zachytávat pakety. Nabízí nám API, pomocí které se můžeme jednoduše napojit na síťovou kartu a zachytávat všechny pakety, které na tuto síťovou kartu přijdou nebo z ní mají být odeslány.

Velice užitečnou věcí, kterou knihovna dále nabízí, je definování takzvaného *PCAP formátu* souboru. Což vlastně znamená, že nemusíme pakety analyzovat hned v reálném čase, ale můžeme je uložit do souboru a tento soubor analyzovat později, třeba i na úplně jiném počítači. *PCAP formát* se mezi volně šiřitelnými analyzátory rozmohl a drtivá většina analyzátorů *PCAP formátu* rozumí a umí pakety v něm uložené dekódovat. Tímto je vlastně zajištěna určitá přenositelnost mezi různými programy.

Další věci, které knihovna PCAP zvládá, jsou tzv. *BFP filtry*, které pomocí jednoduchých příkazů umožňují říct, jakou podmínku musí splňovat pakety, které chceme analyzovat. Například jednoduchý příkaz “tcp port 80“ nám umožní analyzovat pouze pakety, které používají TCP protokol a směřují na port 80.

Nevýhodou knihovny PCAP je, že ačkoliv je velice kvalitní, tak nepatří mezi špičku. Pokud se pokoušíme zachytávat pomocí této knihovny pakety na velmi rychlé lince tj. 100 Mb/s a více, tak bude knihovna některé pakety zahazovat, protože je nestihne zpracovat.

2.4.2 Analyzátor Tcpdump

Tcpdump je analyzátor, který pochází od tvůrců knihovny PCAP a je na ni tudíž postaven. Umožňuje analyzování paketů až do transportní vrstvy, aplikování BFP filtrů nebo ukládání souborů v PCAP formátu. Tcpdump je často využíván Linuxovými administrátory pro rychlou analýzu provozu nebo používán jako sonda pro zachytávání paketů a jejich ukládání do souborů v PCAP formátu, přičemž tyto soubory jsou dále analyzovány v jiných detailnějších analyzátorech.

2.4.3 Analyzátor Wireshark/Tshark

Analyzátor se skládá ze dvou programů. První z nich se jmenuje Wireshark, který nabízí GUI pro analýzu paketů. Druhý se jmenuje Tshark, což je textová obdoba. Oba programy jsou postaveny na knihovně PCAP a nabízejí stejné možnosti jako analyzátor Tcpdump.

Analyzátor Wireshark však na rozdíl od Tcpdumpu umožňuje analyzovat i protokoly aplikační vrstvy. A podpora protokolů není malá. Obsahuje jich úctyhodných 130. A to i méně používané protokoly jako třeba SNB. K tomu všemu Wireshark obsahuje GUI, pomocí kterého lze názorně zobrazovat jednotlivé části paketu a nebo zkoumat jednotlivé atributy různých protokolů.

Analyzátor Wireshark je pravděpodobně nejoblíbenějším volně šiřitelným analyzátorem. Je ke stáhnutí ve verzi pro Linux i ve verzi pro Windows. Nejčastější scénář využití Wiresharku je takový, že se nejdříve pakety zachytí a uloží pomocí Tcpdumpu a později se zanalyzují pomocí Wiresharku.

K dispozici jsou samozřejmě i zdrojové kódy. Samotný Wireshark má velmi pěknou nápovědu pro vývojáře. A proto pokud se vyznáme v programování a síťových protokolech, tak pro nás není problém si do Wiresharku dopsat nové funkce. Například podporu nových protokolů, možnost číst soubory s uloženými pakety ve vlastních formátech a nebo změnit GUI tak, aby vyhovovalo našim potřebám.

Jelikož je Wireshark postavený na knihovně PCAP, dědí některé její neduhy. Analyzátor se nehodí pro reálné sledování síťového provozu na rychlých linkách. V takovýchto případech Wireshark nestačí pakety analyzovat a spousta jich je zahozena.

2.4.4 IDS/IPS systém Snort

IPS systém Snort je další analyzátor síťového provozu postavený na knihovně PCAP. Umí tedy ukládat data v PCAP formátu, využívat BFP filtry a podobně. Je schopen analyzovat podobně jako Tcpdump protokoly až do transportní vrstvy.

Jelikož se však jedná o systém IPS, umí i další vymoženosti. Mezi velkou výhodou je možnost si psát pravidla, která se netýkají jen protokolů, ale týkají se i obsahu samotných paketů. Můžeme například hledat pakety, které v sobě obsahují slovo “ahoj“, pomocí regulérních výrazů získat paket, jehož data zažínají písmenem ‘n’ a končící slovem “ahoj“ a nebo najít paket, který má na 24 bitu 1. Možnosti jsou skutečně široké a díky tomu, že je Snort postaven na knihovně PCAP, lze všechny takto zachycené pakety uložit v PCAP formátu a detailně je zanalyzovat například pomocí Wiresharku. Což je taky často doporučovaná možnost.

Jelikož se jedná o systém IDS/IPS, není nastavení Snortu a psaní pravidel nejjednodušší záležitostí a chce trošku cviku. Nelze jej tudíž doporučit úplným začátečníkům.

2.4.5 Analyzátor NGrep

Analyzátor NGrep je další z analyzátorů postavený na knihovně PCAP . Umí tedy opět ukládat data v PCAP formátu, využívat BFP filtry a podobně. Jak už název napovídá, analyzátor umožňuje filtrovat pakety dle námi napsaných regulérních výrazů. Můžeme tedy stejně jako u IDS/IPS systému Snort najít pakety, které ve svých datech nesou slovo “ahoj“. NGrep umožňuje i specifikovat složitější pravidla. Například, že slovo „ahoj“ musí být v pátém zachyceném paketu. Je také mnohem jednodušší na ovládání než IDS/IPS systém Snort a lze říct, že pokud umíme ovládat program „grep“, pak zvládneme i použití analyzátoru NGrep.

2.5 HW podpora sledování síťového provozu

Je zřejmé, že množství zanalyzovaného provozu často závisí na umístění sondy v naší počítačové síti. Nejčastěji se používá scénář, kdy se sonda umísťuje jako další PC, které je připojeno na Switch/Hub a veškerý provoz, který těmito zařízeními projde, je směrován k sondě. Zde však nastává drobný problém. Hub vždy přepošle každý příchozí paket na všechny porty, tj. každý paket se dostane i směrem k naší sondě. Switch naopak pošle paket jen na port, na kterém se nachází cílová stanice. Díky tomu vznikly některé zajímavé technologie, pomocí kterých můžeme dosáhnout toho, aby byly pakety přeposlány k sondě. Dále uvedené technologie jsou nejčastěji používány na zařízeních firmy Cisco. Od této firmy pochází pak i přístroje použité ve Virtlabu.

2.5.1 SPAN/VSPAN

SPAN/VSPAN znamená Switched port analyzer / Virtual switched port analyzer. Jinak se této technologii říká také Port Mirroring. Jedná se o technologii, pomocí které řekneme switchi, jaký provoz má přeposílat(zrcadlit) na ostatní porty. Díky tomu můžeme dosáhnout přeposílání veškerého provozu k naší sondě. Technologie VSPAN navíc umí přeposílat i VLAN pakety, což nám umožňuje rozpoznat i číslo VLAN, do které daný paket náleží.

2.5.2 RSPAN

RSPAN znamená Repote Switched Port Analyzer. Jedná se o technologii, která nám umožňuje sledovat provoz na více přepínačích jedinou sondou. Princip je v tom, že mezi přepínači vytvoříme virtuální síť – VLAN, do které jsou zrcadleny všechny pakety na sledovaném přepínači a jsou dále záplavou poslány až k cílové sondě.

2.6 Závěr

Pro prozkoumání a otestování dostupných analyzátorů jsme jako vhodný analyzátor vybrali analyzátor Wireshark. Ze všech volně dostupných analyzátorů nabízí největší možnost dekódování, je šířen zdarma a je uživatelsky příjemný. Jelikož drtivá většina analyzátorů analyzuje provoz uložený v PCAP formátu, rozhodli jsem se, že námi vyvíjená sonda bude ukládat zachycené pakety v tomto formátu.

Jako zařízení, na kterém budeme provádět analýzu provozu, bylo zvoleno Cisco Catalyst 3550, ke kterému bude pomocí trunk linky přivedeno PC s monitorovací sondou.

.

3 Analýza architektury virtuální laboratoře

Každá lokalita distribuované virtuální laboratoře se skládá z PC, kterému se říká Řídící server a který má veřejnou IP adresu a je tudíž přístupný z Internetu. Na tomto PC běží servery, které se starají o správu lokality, rezervaci zařízení, mazání a aktivování konfigurací, vytváření virtuálních spojení mezi více lokalitami apod. Dále zde musí být přístupno zařízení ASSSK2, které se stará o propojování sériových linek a switche od firmy Cisco, které se starají o spojování ethernetových linek na bázi VLAN.

3.1 Servery starající se o běh Virtlabu

3.1.1 Rezervační server

Rezervační server se stará o správu rezervací v dané lokalitě. Kontroluje, jaké jsou momentální rezervace, zařízení, na které může být rezervace vytvořena, popřípadě tyto rezervace vytváří a ruší. Na rezervační server vždy přijde požadavek na rezervování a čas, na jak dlouho má být daná topologie rezervována. Pokud je rezervace možná, pak rezervační server provede na tato zařízení dočasnou rezervaci, která se však musí do určitého časového intervalu potvrdit. Pokud potvrzení přijde, pak rezervační server tuto rezervací potvrdí a daná zařízení zarezervuje.

3.1.2 Konfigurační server

Konfigurační server se stará o příjem a následné uložení konfigurací do aktivních prvků, tunelovacího serveru apod. Přijímá konfigurace spolu s názvem prvku, do kterého má být konfigurace uložena. Samotný server si uchovává informace o zařízeních k němu připojených a způsobech, jak se k nim dostat.

3.1.3 Mazací server

Mazací server se stará o vyčištění konfigurací prvků po ukončení rezervace a práce s prvky.

3.1.4 Konzolový server

Konzolový server se stará o zpřístupnění ovládacích konzolí jednotlivých prvků tak, aby s prvky bylo možno pracovat pomocí Java appletu. Nezáleží, jestli se konzole nachází na sériovém nebo ethernetovém rozhraní. Konzolový server si pamatuje cesty k zařízením a stará se o veškerou komunikaci.

3.1.5 Tunelovací server

Tunelovaní server se stará o přeposílání paktů z jedné VLAN do druhé VLAN. A to i pokud se VLAN nenachází ve stejné lokalitě. V obou případech je celý VLAN paket zabalen do UDP paketu a poslán přes internet do Tunelovacího serveru druhé lokality, popřípadě zachycen stejným Tunelovacím serverem. Zde je paket přijat, rozbalen a poslán na rozhraní do příslušné VLAN.

3.2 Zařízení starající se o běh Virtlabu

3.2.1 ASSSK2

Jedná se o zařízení vyvinuté na VŠB – TUO. Jeho účelem je propojit dvě různé sériové linky a to za pomoci udílených příkazů. Díky tomuto je možné vzdáleně vytvořit topologii a pomocí tohoto zařízení vhodně propojit linky tak, aby byla daná topologie funkční.

3.2.2 Cisco Catalyst 3550

Jedná se o switch firmy Cisco, který zajišťuje spojování ethernetových linek. Dále pak umožňuje rozdělení jednotlivých zařízení do VLAN a díky tomu nám umožní jejich snadnou identifikaci. Switch je připojen pomocí trunk linky k řídícímu serveru lokality a umožňuje tak přeposílání VLAN paketů i mezi více lokalitami.

3.2.3 Server s virtuálními stroji

Na tomto PC běží virtuální stroje XEN, které se starají o simulaci jednotlivých PC. Každé PC je přístupné pomocí Telnet konzole, která je přístupná z Java appletu pomocí konzolového serveru.

3.3 Algoritmus pro vytváření topologie

Běh algoritmu začíná ve chvíli, kdy je potřeba pro uživatele vytvořit danou topologii. Nejdříve musíme zjistit, jaké zařízení budeme pro naši topologii potřebovat. Po zjištění počtu a typu zařízení zjistíme, zda naše lokalita může v danou chvíli potřebné zařízení nabídnout. Pokud ano zařízení rezervujeme. Pokud naše lokalita v danou chvíli nemůže všechna zařízení poskytnout, pak přichází na řadu kontaktování okolních lokalit a zjišťování, zda lokality obsahují dané zařízení. Pokud ano, pak na tato zařízení provádíme dočasné rezervace. Což znamená, že si zařízení rezervujeme na určitý kratičký čas, aby nám je nemohl zarezervovat někdo jiný, zatímco budeme pátrat po ostatních zařízeních. Pokud se nám podaří zarezervovat všechna potřebná zařízení, pak tyto dočasné rezervace potvrdíme. V opačném případě se dočasné rezervace zruší.

Pokud máme všechna zařízení rezervovaná, pak může přijít na řadu další krok. Musíme nyní vytvořit konfigurace dle dané topologie a námi rezervovaná zařízení propojit. Což v důsledku znamená, že musíme pomocí konfiguračních serverů daných lokalit uložit do spojovacích zařízení námi požadované konfigurace. Díky rozdistribuovanosti Virtlabu musíme taky kontaktovat Tunelovací servery a zažádat je o vytvoření virtuálních tunelů pro určité VLAN mezi určitými lokalitami.

Nyní, když už máme všechna potřebná zařízení rezervována a správně nakonfigurována, můžeme přistoupit k poslednímu kroku našeho algoritmu. Tím je uložení informací do konzolových serverů, které se nám budou starat o zpřístupnění konzolí. Jako poslední část tohoto kroku vytvoříme webové rozhraní, které umožní naším uživatelům komunikovat s prvky a nastavovat je.

3.4 Včlenění námi vyvíjené sondy

Po důkladné analýze a otestování různých možností nasazení a vytvoření naší sondy jsme vybrali jako vhodné místo pro nasazení Tunelovací server. Důvod je poměrně jednoduchý. V tomto místě dochází k přeposílání paketů z jedné VLAN do ostatních VLAN, a to i v případě, že se nenacházejí ve stejných lokalitách.

Jelikož má každé zařízení svou VLAN, tak díky zachytávání provozu v celé VLAN můžeme analyzovat veškerý provoz, který se týká daného zařízení. Díky tomu, že se Tunelovaní server stará o přeposílání paketů i mezi různými lokalitami, nezáleží na tom, zdali je naše zařízení a zařízení, s nimž právě komunikujeme v dané lokalitě.

Tunelovaní server ve své současné podobě však umožňuje pouze přeposílání paketů mezi jednotlivými VLAN a jinými lokalitami. Vnitřní architektura Tunelovacího serveru nám však umožňuje pouze přeposílání paketu z jedné VLAN do jediné další VLAN. Neumožňuje přeposílání z jedné VLAN do více VLAN. Což nás může omezovat při konfigurování topologií.

3.4.1 Návrh re-implementace Tunelovacího serveru

Nyní, když známe místo, kde umístíme naši sondu, promyslíme množství a dosah změn, které ve struktuře Tunelovacího serveru provedeme. Jelikož server v současné době neumožňuje přeposílání paketů z jedné VLAN do více VLAN, bude se jednat o první věc, kterou na Tunelovacím serveru změníme. Jako další budeme muset implementovat zachytávání paketů v požadované VLAN a jejich prezentaci. Jelikož hlavním úkolem Tunelovacího serveru je přeposílání paketů, pak musí toto zachytávání paketů probíhat odděleně a nesmí ovlivnit rychlost přeposílání.

Bohužel v průběhu implementace požadovaných změn se ukázalo, že Tunelovací server obsahuje chyby, kterých si původní autor nebyl vědom. Jedná se nejčastěji o chyby špatného návrhu datových struktur pro uložení přesměrování VLAN, dále pak souběhu při vytváření a rušení klientských vláken a nakonec použití špatného druhu vláken, což je chyba, která postupem času vede k zablokování ovládací konzole. Jelikož budeme upravovat architekturu Tunelovacího serveru, pak tyto chyby opravíme.

3.4.2 Návrh prezentace zachyceného provozu

Po zvážení, jak nejlépe prezentovat uživatelům výsledky, jsme došli k závěru, že nejlepší bude zachycený provoz ukládat do souboru v PCAP formátu. Tyto soubory pak budou moci uživatelé stahovat a analyzovat je v některém z mnoha dostupných volně šiřitelných analyzátorů. Jako nejvhodnější analyzátor jsme zvolili analyzátor Wireshark.

3.4.3 Návrh ovládacího rozhraní

Jako nejvhodnější rozhraní pro ovládání námi vyvinuté sondy jsme zvolili webové rozhraní. V něm budou moci uživatelé spouštět a zastavovat sondy na rozhraních určitých zařízení a samozřejmě stahovat soubory s uloženým provozem a tyto soubory dále analyzovat. Vhodnou částí, kterou doimplementujeme, je i zobrazování počtu zachycených paketů, aby měli uživatele přehled o počtu paketů, které prochází daným zařízením.

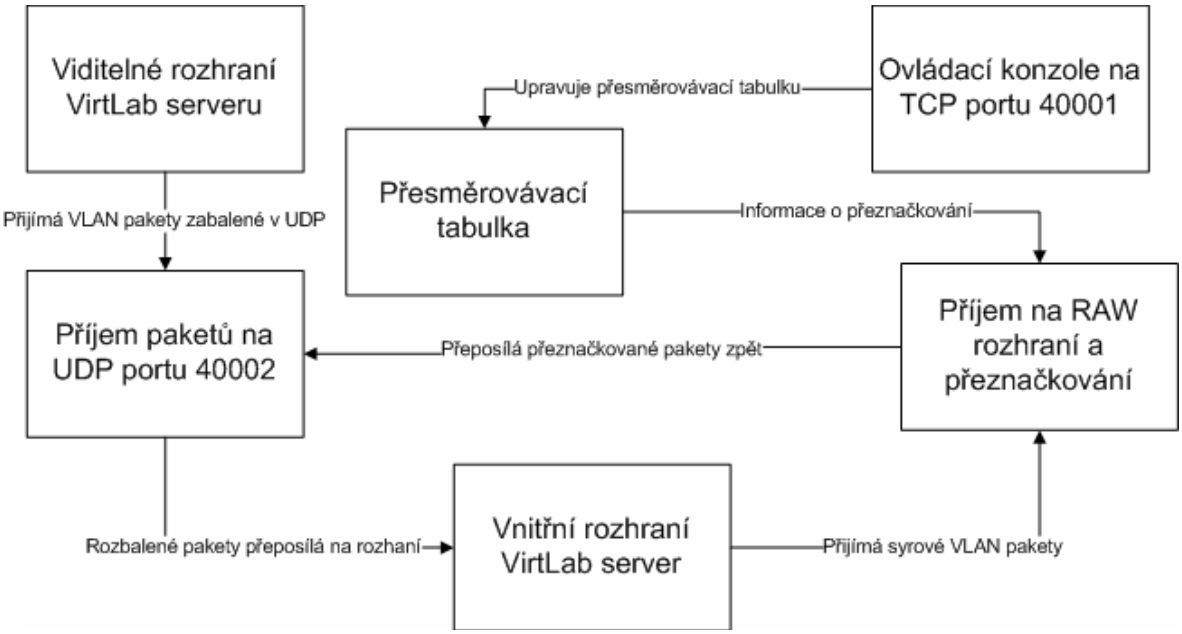
Jako poslední implementujeme část, která po odhlášení uživatele vyčistí všechny soubory se zachyceným provozem a uvolní tak místo na disku serveru.

4 Implementace monitorovací sondy

4.1 Popis stávající implementace Tunelovacího serveru

Celý Tunelovaní server je napsán v jazyce C pod OS Linux. Jeho hlavním úkolem je přijímání VLAN paketů, jejich přeznačkování do určité VLAN a odeslání k cílovému Tunelovacímu serveru nebo lokálnímu spojovacímu prvku. Celou funkčnost lze rozdělit do několika částí. Jak lze vidět na obrázku č.2.

- **Část starající se o příjem všech paketů a jejich přeznačkování**
  Hlavním úkolem této části je poslouchání na *raw socketu* a příjem všech rámců na daném ethernetovém rozhraní. Pokud je rámec typu VLAN, podíváme se do přesměrovávací tabulky a pokud existuje pro daný rámec přesměrování, tak daný rámec přečíslujeme a pošleme ho zabalený v UDP paketu k cíli.
- **Část starající se o příjem paketů z jiných Tunelovacích serverů a jejich rozbalení a poslání**
  Úkolem této části je poslouchání na UDP portu 40002 a zachytávání rámců, které přicházejí od jiných Tunelovacích serverů. Pokud na daném portu přijmeme rámec, pak ho rozbalíme z UDP, ověříme si, že se jedná o rámec VLAN a v nezměněné formě ho pošleme na naše ethernetové rozhraní.
- **Konzole pro ovládání Tunelovacího serveru**
  Tato část se stará o funkčnost konzole pro ovládání Tunelovacího serveru. Poslouchá na TCP portu 40001 a přijímá klientská spojení.



Obrázek 2: Blokové schéma Tunelovacího serveru

4.2 Reimplementace přesměrovávací tabulky

4.2.1 Nevýhody původní implementace

V původní implementaci je přesměrovávací tabulka implementována jako jednosměrný spojový seznam. Jak lze vidět na obrázku č.3. Kde každý z prvků seznamu obsahuje informaci o zdrojové VLAN, cílové VLAN a adrese Tunelovacího serveru, na který má být přečíslovaný paket přeposlán.

Obrázek 3: Původní implementace přesměrovávací tabulky

Toto řešení však není příliš ideální. Pokud máme například přeposílat 5000 rámců, které mají VLAN 140, pak musíme projít celou přesměrovávací tabulku, než zjistíme číslo cílové VLAN a adresu cílového Tunelovacího serveru. Další nevýhodou je, že každý záznam v tabulce je identifikován číslem zdrojové VLAN. Což v praxi znamená, že po přidání dalšího záznamu se stejným číslem zdrojové VLAN vyústí v přepsání starého záznamu. Což není chování, které bychom si přáli.

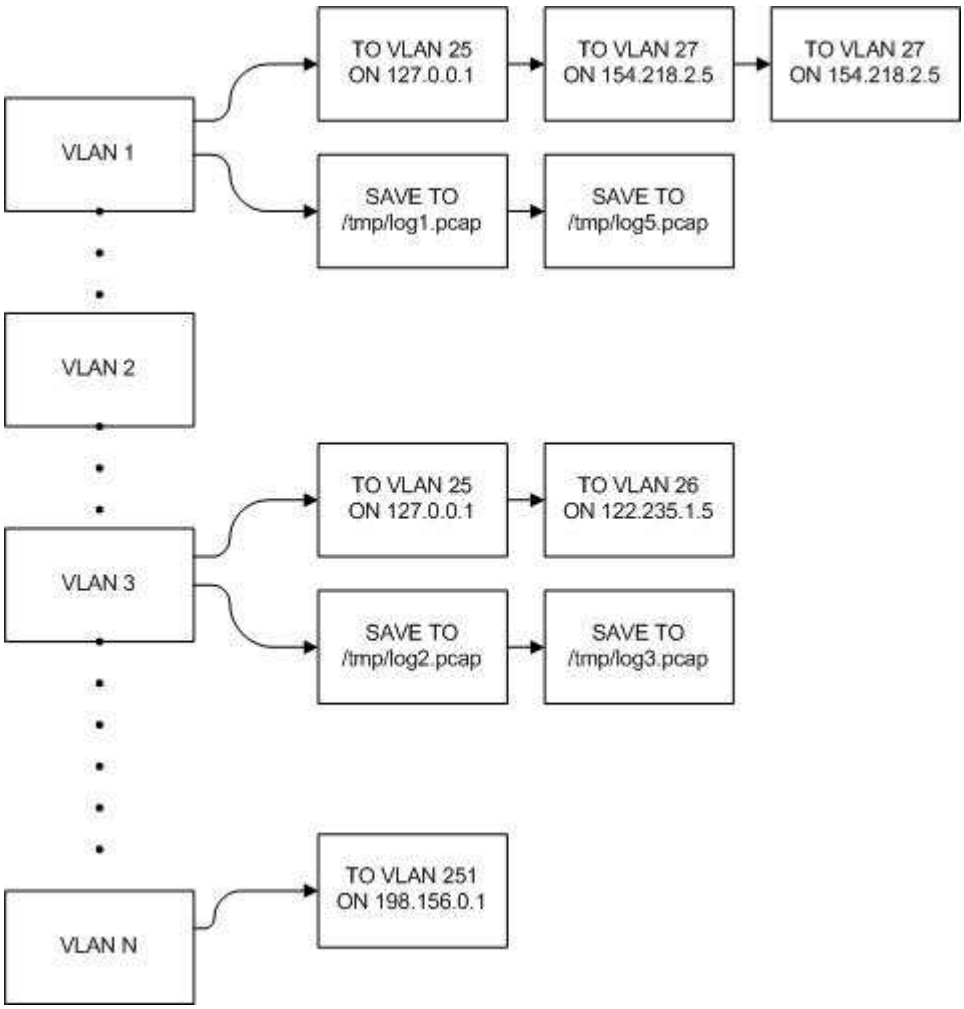
Jelikož nová implementace Tunelovacího serveru bude muset umět uložit informace o zachytávání provozu ve VLAN, což znamená, že budeme muset přidat nový typ záznamu který bude uchovávat informace o ukládání síťového provozu. Pokud přidáme tyto informace do přesměrovávací tabulky, pak zjistíme, že spojový seznam nebude vhodná struktura. Jak lze vidět na obrázku č.4.

Obrázek 4: Rozšíření původní přesměrovávací tabulky o záznamy o ukládání provozu

Jak jde vidět, pokud přidáme další strukturu pro ukládání informací o přesměrování, pak nám náš seznam bude narůstat a budeme muset procházet přes spoustu záznamů než najdeme ten, který potřebujeme. Nemluvě o tom, že mícháme informace které náleží rozdílným částem.

4.2.2 Nová implementace

Pro novou implementaci využijeme efektivnější datovou strukturu.Tou bude Hashovací tabulka[6]. Jak lze vidět na obrázku č.5



Obrázek 5: Nová implementace přesměrovávací tabulky

Naše Hashovací tabulka bude tvořena polem o *n* prvcích, kde každý z prvků bude obsahovat dva ukazatele. Jeden na začátek seznamu s přesměrováními a druhý na začátek seznamu s informacemi o ukládání. Jelikož bude ukládání a přesměrování odděleno, přesměrovávací část nemusí nikdy procházet seznam s informacemi o ukládání a naopak, ukládací část nikdy nebude muset procházet seznam s informacemi o přeposílání. Každé části vždy stačí znát jen číslo VLAN příchozího paketu a na základě tohoto indexu okamžitě vyhledají v poli příslušný záznam a podle toho, o jakou část se jedná použije daný seznam.

Toto řešení je velmi rychlé, zbavuje nás nutnosti procházet záznamy, které nepotřebujeme a díky tomu, že každý z našich záznamů bude identifikován číslem cílové VLAN a cílovou IP adresou, můžeme mít pro jednu VLAN více přesměrování a pro jedno přesměrování může existovat i více cílů, na které rámec zašleme.

4.3 Implementace ukládání provozu

4.3.1 Návrh implementace

Ukládání provozu je novou částí Tunelovacího serveru. Musíme jí však implementovat tak, aby nezpomalovala ostatní části. Příjem paketů se děje ve dvou částech. V části, která se stará o sběr syrových paketů a v části, která přijímá zabalené pakety v UDP paketech. V obou těchto částech budeme muset paket zachytit a předat ho naší části, která se bude starat o ukládání provozu.

Každá z těchto částí poběží v samostatném vlákně. Jelikož budou obě části zapisovat do společného prostoru, budeme muset řešit problém souběhu. Algoritmus na řešení tohoto problému se nazývá Problém sdíleného bufferu neboli také Problém výrobců a spotřebitelů.

Každý přijatý paket budeme ukládat do binárního souboru, který bude v PCAP formátu. Název pak budeme vytvářet podle jména uživatele, zařízení a jména rozhraní na kterém budeme zachytávat síťový provoz.

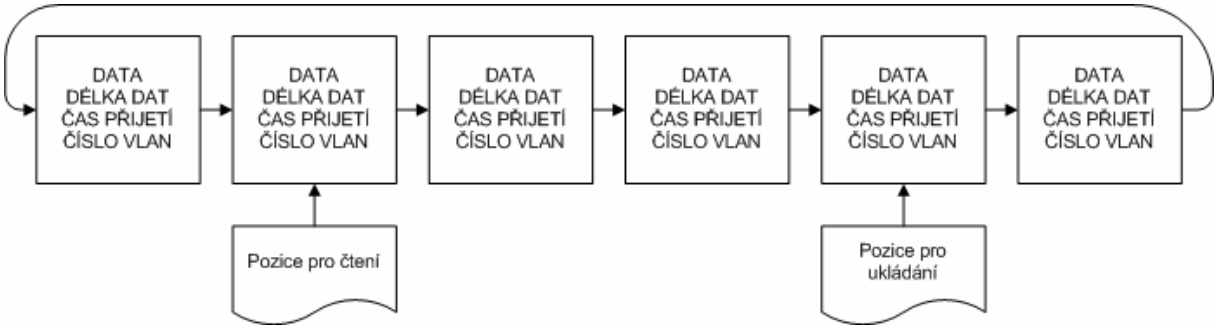
4.3.2 Problém sdíleného bufferu

Myšlenka algoritmu je jednoduchá[3]. Existují procesy, které do bufferu pouze data přidávají. Těm říkáme výrobci. Dále existují procesy, které z bufferu data pouze vybírají. Těm říkáme spotřebitelé. Další věcí kterou budeme potřebovat, jsou tři semafory. Jeden nám bude řídit množství uložených dat, druhý bude sloužit k řízení spotřebitele a třetí nám bude zajišťovat střídání mezi výrobci.

Vždy, když nám přijde paket, výrobce ho převezme. Pak zkontroluje počet paketů v bufferu a pokud je tato hodnota menší než velikost bufferu, pak zde paket uloží. Potom zvedne hodnotu semaforu pro počet paketů v bufferu o jedničku a zvedne hodnotu semaforu pro spotřebitele, čímž ho uvede do provozu. Pokud je spotřebitel uveden do provozu, pak převezme paket z bufferu, sníží počet paketů v bufferu o jedničku a tento paket zpracuje.

4.3.3 Implementace sdíleného bufferu

Náš sdílený buffer implementujeme jako spojový, kruhový seznam o přesně daném počtu prvků. Při používání našeho bufferu si budeme muset zapamatovat dvě pozice. První je pozice, na kterou zrovna můžeme ukládat přijatý paket. Druhá je pozice, ze které můžeme uložený paket číst. Díky vhodné implementaci procesu výrobce se nám nikdy nestane, aby pozice pro čtení byla dál než pozice pro ukládání. A díky tomu, že je náš seznam kruhový, můžeme znova použít staré pozice. Viz. Obrázek č.6.



Obrázek 6: Implementace bufferu pro ukládaní paketů

4.3.4 Implementace procesů výrobce a spotřebitele

Oba procesy využívají některé společné proměnné. Mezi ně patří tři semafory potřebné pro správné řízení a dva ukazatele, z nichž jeden ukazuje na první pozici pro čtení a druhý na pozici pro zápis. Následující popis algoritmu je popsán v pseudo-jazyce podobnému implementačnímu jazyku C.

```
#define       BUFFER_MAX = 500;        //Buffer na 500 paketů

sem_t         producer_sem = 1;        //Semafor pro výrobce
sem_t         consumer_sem = 0;        //Semafor pro spotřebitele

buffer_pos    *writing_pos = buffer_start;  //Pozice pro čtení. Inicializována na začátek buff.
buffer_pos    *reading_pos = buffer start;  //Pozice pro zápis. Inicializována na začátek buff.
```

Funkci výrobce, který má na starosti ukládání paketů, volá část, která se stará o příjem zabalených paketů a část, která se stará o příjem syrových paketů. Každá z nich běží v samostatném vlákně a díky semaforům nikdy nedojde k souběhu. Jako parametry funkce se dodávají data paketu, počet těchto dat a čas, v jakém jsme paket zachytili. Tato data jsou pak následně zkopírována do bufferu a část, která tuto funkci zavolala, se opět může věnovat své práci a přenechat buffer druhé části.

```
/* Funkce výrobce */
void append_packet(char *data, short data_count, time_t capture_time, int vlan_num)
{
        sem_down(producer_sem);                                  //Vstup do kritické sekce

        if( get_sem_value(consumer_sem) <  BUFFER_MAX )          //Pokud je v bufferu místo
        {
                /* Uložíme data na aktuální pozici v bufferu */
                memset(writing_pos->data, data, data_count);
                memset(writing_pos->data_count, data_count, sizeof(short) );
                memset(writing_pos->capture_time, capture_time, sizeof(time_t) );
                memset(writing_pos->vlan_num, vlan_num, sizeof(int) );

                sem_up(consumer_sem);                            //Zvětšíme počet paketů

                writing_pos = writing_pos->next;                 //Posuneme se v bufferu
        }

        sem_up(producer_sem);                                    //Opustíme kritickou sekci
}
```

Funkce spotřebitele běží v samostatném vlákně a jejím úkolem je přebírat data z bufferu a pomocí seznamu o ukládání ukládat uložené rámce do binárních souborů v pcap formátu. Jelikož funkce běží v samostatném vlákně, svým během nezatěžuje části pro příjem a přeposílání paketů. Důvod, proč jsme tuto část implementovali odděleně, je poměrně prostý. Zápis do souboru na disk, který je mnohem pomalejší než paměť, může být zdlouhavý. A pokud bychom jeden paket zapisovali do velkého množství souborů pak by na nás ostatní části musely čekat a mohlo by docházet ke ztrátě provozu.Díky oddělení vše však funguje správně a žádná z částí nečeká na tu druhou.

```
/* Funkce spotřebitele */
void *process_packet(void *thread_args)
{
        /* Seznam s informacemi o ukládání */
        login_list *logins = NULL;

        while(1)                                        //Nekonečná smyčka
        {
                sem_down(consumer_sem);                 //Pouze pokud máme pakety

                /* Zjistíme informace o ukládání */
                logins = get_login_for_vlan(reading_pos->vlan_num);

                /* Posuneme se na začátek seznamus přesměrováními */
                login_list *help = logins->begin;

                while(help != NULL)                     //Projdeme seznam
                {
                        /* Uložíme data v pcap formátu. Tj. Počet dat, čas přijetí a data */
                        write(help->fp, reading_pos->data_count, sizeof(short) );
                        write(help->fp, reading_pos->capture_time, sizeof(time_t) );
                        write(help->fp, reading_pos->data, reading_pos->data_count );

                        help = help->next;              //Posuneme se dále

                }

                reading_pos = reading_pos->next;        //Posuneme pozici pro čtení

        }

}
```

4.4 Reimplementace ovládací konzole

4.4.1 Stávající implementace

V původní podobě ovládací konzole zvládá příkazy na zobrazení přesměrování, smazání přesměrování, přidání přesměrování, zobrazení nápovědy a příkazu k ukončení. Ke konzoli je možno se připojit na TCP portu 40001 a konzole zvládá najednou obsloužit až 35 klientů. Souběh je řešen pomocí zamykání a algoritmu Čtenářů a spisovatelů.

V naší nové implementaci budeme potřebovat změnit strukturu příkazů pro správu přesměrování, pro zobrazování přesměrování a ukládání a jako poslední budeme muset implementovat příkazy pro správu ukládání.

4.4.2 Reimplementace příkazů pro správu přesměrování

Syntaxe pro přidání přesměrování zůstane v podstatě nezměněna. Pokud chceme přidat přesměrování, musíme zadat číslo zdrojové VLAN, číslo cílové VLAN a IP adresu cílového Tunelovacího serveru. Díky změně struktury přesměrovávací tabulky je nyní možno přidat pro jednu zdrojovou VLAN přesměrování do více jiných VLAN. Je samozřejmě také možné pro jednu zdrojovou a cílovou VLAN mít více různých cílových Tunelovacích serverů.

Syntaxe pro smazání přesměrování se změní. Díky tomu, že můžeme pro jednu zdrojovou VLAN přidat více cílových VLAN a pro stejnou zdrojovou a cílovou VLAN více cílových Tunelovacích serverů pak musíme přidat další parametry. Nyní budeme požadovat zdrojovou VLAN, cílovou VLAN a IP adresu cílového Tunelovacího serveru. Syntaxe se tím sice stala složitější, ale je nám umožněna mnohem bohatší konfigurace Tunelovacího serveru.

4.4.3 Implementace příkazů pro správu ukládání provozu

Příkazy pro ukládání provozu se v původní verzi nevyskytovaly. Je tedy na nás, jak implementujeme syntaxi pro správu. Rozhodli jsme se tady takto. Každé ukládání bude patřit uživateli a každý z uživatelů bude moci mít více ukládání. Ukládání se poté specifikuje aliasem zařízení, názvem zařízení a rozhraním, na kterém chceme zachytávat provoz. Číslo VLAN, kterou budeme sledovat, zůstane před uživatelem skryto a uživateli se bude jevit jako by sledoval provoz na skutečném zařízení. Po zavolání příkazu na ukládání se uživateli vytvoří binární soubor, který bude obsahovat zachycený provoz a který si po zastavení ukládání bude moci přes webové rozhraní stáhnout.

Příkaz pro zastavení má stejnou syntaxi jako příkaz pro přidávání. Jako parametry požadujeme jméno uživatele, alias zařízení, jméno zařízení a název rozhraní. Po vykonání příkazu se všechny pakety, které byly v bufferu, uloží do souboru a do tohoto souboru se dále nebudou žádné další pakety zapisovat. Stejný stav nastane, pokud velikost souboru přesáhne 10MB. Což je pro nás ještě únosná mez a pro uživatele únosné množství, které může stáhnout na svůj domácí počítač.

Příkaz pro vymazání má odlišnou syntaxi oproti dvěma předchozím. Jako parametr požaduje pouze uživatelské jméno a po jeho provedení vymažeme všechna uživatelova přesměrování a vymažeme binární soubory s uloženým provozem z disku. Tento příkaz vyvoláme, pokud se uživatel odhlásí nebo mu vyprší jeho časové kvóty.

4.4.4 Implementace seznamu zařízení

Jak bylo zmíněno výše, každé ukládání se provádí na zařízení a jeho rozhraní. Pomocí těchto dvou atributů poté můžeme zjistit, do jaké VLAN dané zařízení a jeho rozhraní náleží. Všechny tyto informace jsou uloženy v souboru "/etc/virtlab/spoje.conf". Každá lokalita obsahuje tento soubor a jsou v něm popsána všechna zařízení, která v daných lokalitách existují spolu s číslem VLAN. V současné době jsou v souboru zapsána všechna zařízení, která všechny lokality Virtlabu obsahují. Toto se však v budoucnu může změnit a v souboru už budou jen zařízení, která náleží dané lokalitě. Proto náš Tunelovací server na začátku tento soubor zpracuje a uloží informace o zařízeních které náleží do lokality ve které Tunelovací server pracuje. Název této lokality může zjistit ze souboru "/etc/virtlab/sitename". Po zpracování souboru si Tunelovací server všechna tato zařízení uloží do seznamu a při jakémkoliv požadavku na ukládání tento seznam zkontroluje a zjistí z něho, zda dané zařízení existuje a pokud ano, pak zjistí, kterou VLAN má sledovat.

4.4.5 Reimplementace příkazu na zobrazování

V původní verzi příkaz zobrazoval pouze všechna přesměrování. V naší nové verzi jsme příkaz rozšířili o možnost zobrazovat i sledování. Syntaxe se díky tomu změní. Nyní si musíme vybrat, zda chceme zobrazit přesměrování anebo ukládání. Ke každému z těchto výběrů existuje nepovinný parametr. Pokud si vybereme, že chceme sledovat přesměrování, můžeme zvolit číslo zdrojové VLAN, pro kterou chceme vidět všechna přesměrování. Pokud si vybereme, že chceme zobrazit ukládání, pak můžeme zvolit jméno uživatele, od kterého chceme ukládání vidět.

Pokud zobrazujeme informace o ukládání, pak dostaneme informace o jménu uživatele, zařízení, rozhraní, počtu zachycených paketů a počtu bajtů, které jsme zatím zachytili.

4.5 Oprava chyb v Tunelovacím serveru

4.5.1 Oprava souběhu

Při přijímání nového klientského spojení může dojít k souběhu, a to pokud ve stejném čase vzniká nová klientská konzole a přitom jedna z konzolí končí a označuje, že je v seznamu místo pro novou konzoli. Bohužel při souběhu nedojde k zapsání informace o tom, že je pozice volná a tato pozice již zůstane napořád obsazená. Což snižuje maximální množství připojitelných klientů. Řešení je poměrně jednoduché. Stačí přidat binární semafor, pomocí kterého se budou střídat nově vznikající a zanikající konzole. Pak již k souběhu nebude docházet

4.5.2 Oprava vláken

V původní verzi autor použil normální vlákna, jež mají tu vlastnost, že po svém ukončení čekají, až si rodičovský proces převezme od vlákna návratovou hodnotu. Na to však autor nepomyslel a od vláken hodnotu nepřebírá. Což mělo za následek, že vlákna zůstala v paměti a po dosažení maximálního počtu vláken již nebylo možno další vlákna vytvářet. Řešení je opět jednoduché. Stačí vláknům nastavit vlastnost "*detached*"[6]. Díky této vlastnosti vlákna nečekají na převzetí návratové hodnoty a po ukončení jsou smazána z operační paměti.

5 Implementace uživatelského rozhraní

5.1 Stávající implementace

Celé uživatelské rozhraní je pojato jako webová aplikace a je implementováno s použitím jazyka PHP ve verzi 5. Uživatelé, kteří toto rozhraní využívají, se dělí na administrátory, tutory a studenty.

5.1.1 Rozdělení uživatelů

Administrátoři mají na starost tvorbu nových úloh, správu uživatelů, kontrolují stav serverů, popřípadě mohou provést rezervaci na přání, což znamená, že si mohou zažádat o svou vlastní topologii, se kterou mohou pracovat v rámci své kvóty.

Tutoři se mohou dívat, jak student na úloze pracuje a popřípadě studentovi napsat zprávu a poradit mu.

Studenti si rezervují úlohy, ve kterých mají určitý čas na jejich zpracování a úkoly, které musejí splnit. Po vypršení daného časového limitu se úloha sama ukončí.

5.1.2 Formulář pro řešení úlohy

Celé ovládání námi řešené úlohy se skrývá v menu pod volbou “aktivní úlohy“. Student může najednou mít zadáno více úloh, které řeší. Každá úloha je dána svým názvem, jejím popisem, zobrazením topologie a seznamem zařízení. Student může přistoupit ke konzoli každého zařízení pomocí Java appletu, který se mu otevře po kliknutí na tlačítko vedle zařízení. Jelikož úloh může být více, je ovládání každé úlohy umístěno v samostatném panelu.

5.1.3 Konzole zařízení

Ke každému zařízení existuje jeho ovládací konzole. Jelikož je Virtlab rozdistribuovaný, nemusí být zaručeno, že zařízení, které uživatel dostane, musí být v rámci jedné lokality. Je možné, že část zařízení bude v jedné lokalitě a druhá část zařízení v lokalitě druhé. Proto každé tlačítko skrytě uchovává informace o umístění a názvu daného prvku. Pomocí těchto informací pak kontaktuje konzolový server dané lokality a zprovozní ovládací konzoli.

5.1.4 Přemapování názvů zařízení

Jelikož každá topologie používá vlastní názvy zařízení, které se často liší od názvů zařízení, kterými jsou pojmenovány zařízení ve Virtlabu, je každému zařízení Virtlabu, které je v naší topologii přiřazen alias. Zařízení pak v topologii vystupuje pod tímto aliasem. Tlačítko pro konzoli pak obsahuje skutečný název zařízení spolu s lokalitou, ve které je dané zařízení umístěno.

5.2 Implementace ovládání sledování provozu

5.2.1 Návrh místa pro včlenění

Sledování síťového provozu souvisí s řešenou úlohou. Proto se jako ideální místo pro včlenění sledování provozu jeví formulář pro práci s aktuálními úlohami. Přidáme tudíž ke každé úloze seznam zařízení a jejich rozhraní, na kterých může uživatel sledovat síťový provoz. Jelikož uživatel může v daném čase řešit větší množství úloh, přidáme seznam možných zařízení ke sledování ke každé úloze.

5.2.2 Vzhled a možnosti našeho ovládacího prvku

Náš ovládací prvek bude vypadat následovně. Bude se jednat o tabulku, která ve svých řádcích bude obsahovat zařízení a jejich rozhraní. Sloupce tabulky pak budou tvořeny volbami, pomocí kterých můžeme sondu ovládat nebo informaci, pomocí které zjistíme aktuální stav sondy. Mezi ovládací prvky bude patřit možnost spuštění sondy, zastavení sondy a stáhnutí log souboru. Mezi informační sloupce pak budou patřit sloupce zobrazující stav sondy a sloupec zobrazující počet zachycených paketů. Stavy sondy pak mohou být spuštěná, zastavená a doposud neexistující. Poslední možnost nám oznamuje, že na daném zařízení jsme doposud neprováděli žádný sběr paketů.

5.2.3 Algoritmus pro zjištění stavu sond

Pro to, abychom zjistili, které zařízení můžeme sledovat a které Tunelovací servery musíme, kontaktovat musíme znát lokalitu, ve které se zařízení nachází, jeho pravé jméno a název rozhraní. K těmto informacím se dostaneme poměrně jednoduše. Každá úloha obsahuje informaci o přemapování názvů zařízení. V této informaci je vždy napsán alias zařízení, pod kterým vystupuje v topologii a jeho pravé jméno spolu s rozhraním a názvem lokality. Poslední věcí, kterou potřebujeme znát, je jméno uživatele, který dané úlohy právě řeší. A to z důvodu, že každé sledování musí patřit některému z uživatelů Virtlabu.

Nyní, když víme, kde najdeme potřebné informace, stačí nám pouze vytvořit seznam lokalit, které musíme kontaktovat a každé lokalitě přiřadit zařízení, o kterých budeme potřebovat informace. Když se nám podaří námi získané informace takto uspořádat, stačí nám již jen postupně kontaktovat Tunelovací servery daných lokalit, připojit se k jejich ovládacím konzolím na TCP portu 40001 a pomocí příkazu pro získání informací vypíšeme a uložíme informace o aktuálních sledováních pro našeho uživatele v daných lokalitách.

Když máme zjištěny informace o sledování, můžeme pomocí těchto informací korektně přidat do naší tabulky sondu se všemi potřebnými informacemi a přidat potřebná tlačítka pro ovládání sondy.

5.2.4 Algoritmus zastavování a spouštění sond

Každá sonda se na začátku nachází ve stavu nepoužitá. Což znamená, že jsme v informacích z Tunelovacího serveru nenašli o ukládání pro daného uživatele žádnou zmínku. Dalším ze stavů, ve kterém sonda bude, je stav spuštěná. Znamená to, že ukládání existuje a je funkční. Posledním ze stavů, ve kterém se sonda může nacházet, je stav zastavená. Což znamená, že ukládání existuje, ale je v současné době zastavené. K tomu může dojít z vlastní vůle uživatele anebo pokud binární soubor s uloženým provozem přesáhne hranici 10 MB.

Pokud je sonda ve stavu zastavená nebo nepoužitá, pak máme k dispozici tlačítko pro její spuštění. Po kliknutí na tlačítko se naváže spojení s Tunelovacím serverem dané lokality a spustí se příkaz pro přidání ukládání. Všechny potřebné informace, které jsme předtím zjistili a které jsou nutné pro spuštění sondy, obsahuje skrytý formulář, ze kterého odešleme informace pomocí tohoto spouštěcího tlačítka.

Pokud se sonda nachází ve stavu spuštěná, pak se nám zobrazí tlačítko pro její zastavení. Po kliknutí se stejně jako v předchozím případě naváže spojení s daným Tunelovacím serverem a spustí se příkaz pro zastavení sledování. Všechny potřebné informace jsou opětně skryty v neviditelném formuláři a tyto informace opětně pošleme pomocí tohoto tlačítka.

5.2.5 Stahování logovacích souborů

Pokud se sonda nachází ve stavu zastavená, můžeme stahovat logovací soubor, který je v PCAP formátu a obsahuje uložený provoz. Logovací soubor je uložen v prostoru webového serveru a uživatel zde má tudíž přístup a může daný soubor pomocí internetového prohlížeče stáhnout. Jelikož se soubor může nacházet i v jiných lokalitách, obsahuje odkaz pro stažení souboru úplnou adresu dané lokality.

Název souboru je pak tvořen jménem uživatele, aliasem zařízením na kterém se provoz zachytával a jeho rozhraním. Po stáhnutí může uživatel tento soubor otevřít v některém z mnoha analyzátorů.

5.2.6 Algoritmus pro zjišťování počtu aktuálně zachycených paketů

Každá monitorovací sonda obsahuje sloupeček, ve kterém se zobrazuje počet zachycených paketů. V každém řádku se nachází samostatný rámeček, ve kterém se pomocí HTML značek pravidelně obnovuje obsah. Zjištění počtu paketů se pak děje poměrně jednoduše. Každý takový rámeček obsahuje atributy, které obsahují adresu Tunelovacího serveru dané lokality, skutečný název zařízení, rozhraní a jméno uživatele. Pokaždé, když se stránka vygeneruje, se připojíme k danému Tunelovacímu serveru, spustíme příkaz pro zobrazení ukládání daného uživatele, v zobrazených informacích najdeme naše zařízení, odkud získáme počet zatím zachycených paketů. Díky HTML značkám se toto bude dít každé 2 vteřiny.

5.2.7 Algoritmus pro úklid logovacích souborů

Když zjišťujeme informace pro korektní zobrazení stavu sond, zjistíme i lokality, ze kterých daná zařízení pocházejí. Není proto pro nás nic jednoduššího, než si tyto informace ukládat a díky nim zjistit, ve kterých lokalitách uživatel ukládal provoz. Abychom tyto informace neztratili a měli je k dispozici po celou dobu přihlášení uživatele, uložíme je do SESSION proměnné.

Pokud se uživatel odhlásí anebo pokud mu vyprší časové limity, spustíme uklízecí funkci. Její princip je poměrně jednoduchý. Získá informace ze SESSION proměnné uchovávající adresy lokalit, ve kterých jsme sledování prováděli, tyto lokality postupně kontaktuje a spustí funkci pro vymazání všech ukládání daného uživatele. Tunelovací server se už pak sám postará o korektní ukončení ukládání a smazání souboru.

6 Závěr

Naším úkolem bylo přidat uživatelům novou funkci sledování síťového provozu. V průběhu zpracování našeho úkolu jsme prozkoumali některé volně šiřitelné analyzátory a vybrali pro uživatele ten nejpříjemnější a nejsnáze ovladatelný.

Dalším z bodů našeho úkolu byla analýza projektu Virtlab a vyhledání nejvhodnějšího místa pro včlenění námi vyvíjené sondy. Jako nejvhodnější místo jsme vybrali Tunelovací server. V tomto místě dochází k přesměrovávání veškerého provozu, tudíž je to ideální místo, kde můžeme námi požadovaný provoz zachytit.

Nejdůležitějším bodem úkolu byla implementace sondy. Jelikož jsme jako místo pro naší sondu vybrali Tunelovací server, museli jsme důkladně prozkoumat jeho implementaci a zjistit rozsah změn. Díky naším úpravám nyní Tunelovací server dokáže ukládat požadovaný síťový provoz a přeposílat jeden VLAN paket do více cílů. V průběhu reimplementace jsme našli v Tunelovacímu serveru chyby, které si původní autor neuvědomil. Tyto chyby měli po určitém čase za následek nefunkčnost Tunelovacího serveru. Díky našim opravám se toto již v budoucnu nestane.

Jako poslední část jsme implementovali webové rozhraní, pomocí kterého mohou uživatelé pohodlně vytvořené sondy ovládat. Naše rozhraní jsme včlenili do již vytvořeného webového rozhraní Virtlabu, což si vyžádalo prozkoumání stávající implementace a vyhledání míst, do kterých jsme doimplementovali naše změny. Námi vytvořené rozhraní je plně funkční a dokonce podává uživateli informace o aktuálním počtu zachycených paketů v reálném čase.

Závěrem lze snad jen říct, že se nám náš úkol podařilo splnit. Uživatelé nyní mohou při řešení svých úloh využít novou funkci pro sledování síťového provozu a pracovat na svých úlohách efektivněji a cítit se, jako by pracovali se skutečnou sítí.

Literatura

[1] Pavel Herout, *Programovací jazyk C první a druhý díl*, Computer Press 2005

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

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

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

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

[6] Piotr Wroblewski, *Algoritmy, Datové struktury a programovací techniky*, Computer Press, 2004

Obsah CD

/tun-server		adresář obsahující zdrojové texty Tunelovacího serveru
	/doc	adresář obsahující manuály Tunelovacího serveru
	/include	adresář obsahující hlavičkové soubory Tunelovacího serveru
	/src	adresář obsahující zdrojové texty Tunelovacího serveru
/web		adresář obsahující zdrojové texty webového rozhraní
	/virtlab	adresář obsahující zdrojové texty jednotlivých funkcí web. rozhraní
/root		
	abstrakt-CZ.txt	abstrakt v českém jazyce
	abstrakt-EN.txt	abstrakt v anglickém jazyce
/text		adresář s texty
	nov675-bc-prace.pdf	text bakalářské práce

Příloha A

Popis protokolu Tunelovacího serveru

- Jedná se o textově orientovaný protokol
- Komunikace přes TCP spojení port 40001
- Pro názvy příkazů se rozlišují velká a malá písmena (case sensitive)

S Tunelovacím serverem je komunikováno příkazy

Příkaz pro přidání přesměrování
redir <local VLAN> <remote VLAN> <destination IP>

Příkaz pro zrušení přesměrování
noredir <local VLAN> <remote VLAN> <destination IP>

Příkaz pro přidání/spuštění ukládání
log <username> <device alias>=<device real name>:<device interface>

Příkaz pro zastavení ukládání
nolog <username> <device alias>=<device real name>:<device interface>

Příkaz pro vymazání logovacích souborů od uživatele
dellog <username>

Příkaz pro zobrazení všech přesměrování
show redirs

Příkaz pro zobrazení všech přesměrování dané VLAN
show redirs <VLAN>

Příkaz pro zobrazení všech ukládání
show logs

Příkaz pro zobrazení všech ukládání daného uživatele
show logs <username>

Příkaz pro nápovědu
help <command>

Příkaz pro ukončení konzole
exit

Příklady použití jednotlivých příkazů

redir 102 101 127.0.0.1
- přesměruje VLAN 101 na VLAN 102 a pošle ji na lokální počítač

noredir 102 101 127.0.0.1
- zruší přesměrování z VLAN 101 na VLAN 102 a jejich přeposílání na lokální počítač

log nov675 pc1=pc12:eth0
- spustí ukládání pro uživatele nov675 pro zařízení pojmenovaném jako pc0 na reálném zařízení pc12 a jeho rozhraní eth0

nolog nov675 pc1=pc12:eth0
- zastaví ukládání pro uživatele nov675 na zařízení pojmenovaném jako pc0 na reálném zařízení pc12 a jeho rozhraní eth0

dellog nov675
- smaže všechny logovací soubory uživatele nov675

show redirs
- zobrazí všechna přesměrování

show redirs 101
- zobrazí všechna přesměrování pro VLAN 101

show logs
- zobrazí všechna ukládání

show logs nov675
- zobrazí všechna ukládání uživatele nov675

help redir
- zobrazí nápovědu pro příkaz redir

exit
- ukončí konzoli

Příloha B

Popis souborů Tunelovacího serveru a webového rozhraní, ve kterých jsme provedli změny

Tunelovací server – složka “src/tun-server“

/doc

manuals	- Manuály

/include

ieee8021q.h	- Definice VLAN
vlan_logger.h	- Hlav. funkcí pro práci s bufferem na ukládání paketů
vlan_over_udp.h	- Hlav. funkcí pro práci s příjmem zabalených paketů
vlan_redirect_console.h	- Hlav. funkcí pro práci s konzolí
vlan_redirect_table.h	- Hlav. funkcí pro práci s přesměrovávací tabulkou

/src

vlan_logger.c	- Funkce pro práci s bufferem na ukládání paketů
vlan_over_udp.c	- Funkce pro práci s příjmem zabalených paketů
vlan_redirect_console.c	- Funkce pro práci s konzolí
vlan_redirector.c	- Hlavní soubor který nám zaručuje konfiguraci a běh serveru
vlan_redirect_table.c	- Funkce pro práci s přesměrovávací tabulkou

Makefile - Soubor pro správné zkompilování

Webové rozhraní – složka “/web”

index.php - Obsahuje funkci pro úklid logovacích souborů

/virtlab

reser-active.php	- Stará se o práci s aktivní rezervací a o ovládání sond
packets.php	- Stará se o zobrazování počtu zachycených paketů