VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Integrace distribuované laboratoře počítačových sítí se simulátorem PacketTracer

Integration of Distributed Virtual Network Laboratory with PacketTracer Simulator

2010 Bc. Jiří Knapek

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

V Ostravě 7. května 2010 .....

Rád bych na tomto místě poděkovala všem, kteří mi pomohli, protože bez nich by tato práce nevznikla. Nerad bych někoho opomenul, proto jen děkuji vedoucímu své práce Petru Grygárkovi, který rovněž koordinuje veškerý vývoj Virtlabu.

Abstrakt

Smyslem této práce je umožnit komunikaci mezi reálnými síťovými prvky a simulátorem Packet Tracer. Zahrnuje tedy vše potřebné pro splnění tohoto cíle. Popisuje protokoly použité při komunikaci mezi simulátory PT a rovněž obsahuje jejich implementaci. Dále jsou zde popsány změny, které bylo potřeba provést pro integraci tohoto řešení do Virtlabu.

Klíčová slova: PTMP, MU, PT API, Packet Tracer, Virtlab, PTServer, PTBridge

Abstract

The purpose of this work is to allow communication between real network equipment and simulator Packet Tracer. Includes everything you need to meet this target. Describes the protocols used for communication between simulators PT and also includes its implementation. It also describes changes that had to be made to integration this solution into Virtlab.

Keywords: PTMP, MU, PT API, Packet Tracer, Virtlab, PTServer, PTBridge

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
IPC	–	Inter Process Communication
MU	–	Multi-User connection
PT	–	Packet Tracer
PTMP	–	Packet Tracer Messaging Protocol
TCP	–	Transmission Control Protocol
TCP/IP	–	Transmission Control Protocol/Internet Protocol
UDP	–	User Datagram Protocol
VLAN	–	Virtual Local Area Network
WSDL	–	Web Services Description Language
XML	–	eXtensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce bylo umožnit komunikaci mezi simulátorem počítačových sítí Packet Tracer a distribuovanou síťovou laboratoří – Virtlab. V první kapitole tedy krátce popisuji co tato dvě řešení umožňují a nabízejí. Následující kapitola popisuje dostupné Packet Tracer API a jeho možnosti. Dále jsou pak popsány protokoly, které umožňují vzájemnou komunikaci mezi více simulátory Packet Tracer. Toto propojování podporují pouze novější verze programu Packet Tracer (5.0 a novější). Není však možné pro tuto komunikaci použít žádné již hotové řešení, jelikož takovéto buď neexistuje nebo není nikde zmiňováno.

Vše z čeho finální implementace vychází jsou rady přímo od vývojářů simulátoru a zdrojové kódy API. To umožňuje rozšiřovat samotný Packet Tracer. Ke komunikačním protokolům použitým při propojování programů Packet Tracer není dostupná téměř žádná dokumentace.

Pro to aby mezi sebou mohly komunikovat simulované a reálné síťové prvky, bylo potřeba vytvořit jakýsi síťový most (dalo by se snad i říci překladač). Tento most zajišťuje převody reálného provozu na simulovaný a naopak.

Dále je zde popisována analýza řešení a také implementovaná integrace vzdáleného Packet Traceru a Virtlabu. Popisovány jsou všechny nezbytné úpravy, které jsou pro mnou implementované rozšíření Virtlabu potřeba.

2 Popis souvisejících projektů

Tato kapitola obsahuje základní popis programu Packet Tracer a také projektu distribuované laboratoře počítačových sítí Virtlab.

Tyto dva projekty si kladou za cíl zjednodušit a zlepšit výuku zaměřenou na počítačové sítě. Takovýchto projektů existuje více, ovšem není mi znám žádný jiný, který by umožňoval vzájemné propojování v rámci daného projektu.

2.1 Virtlab

Tento projekt se zabývá zpřístupněním síťových prvků pro praktickou výuku vzdáleně prostřednictvím sítě Internet. Pro uživatele je zde možnost pomocí webového prohlížeče vzdáleně konfigurovat různé síťové prvky, které jsou propojovány automaticky dle úlohy. Každý uživatel si zde může rezervovat čas pro určité zapojení. Je možno vybrat si již připravené topologie, ale také definovat si topologii vlastní.

Současná verze dovoluje propojení více nezávislých lokalit a vzájemné zapůjčování prvků mezi lokalitami.

Tento projekt je aktivně vyvíjen na katedře informatiky Vysoké školy báňské. Je určen především pro studenty této vysoké školy a dalších spolupracujících škol a institucí. Další podrobnosti jsou dostupné na stránkách projektu [4].

2.2 Packet Tracer

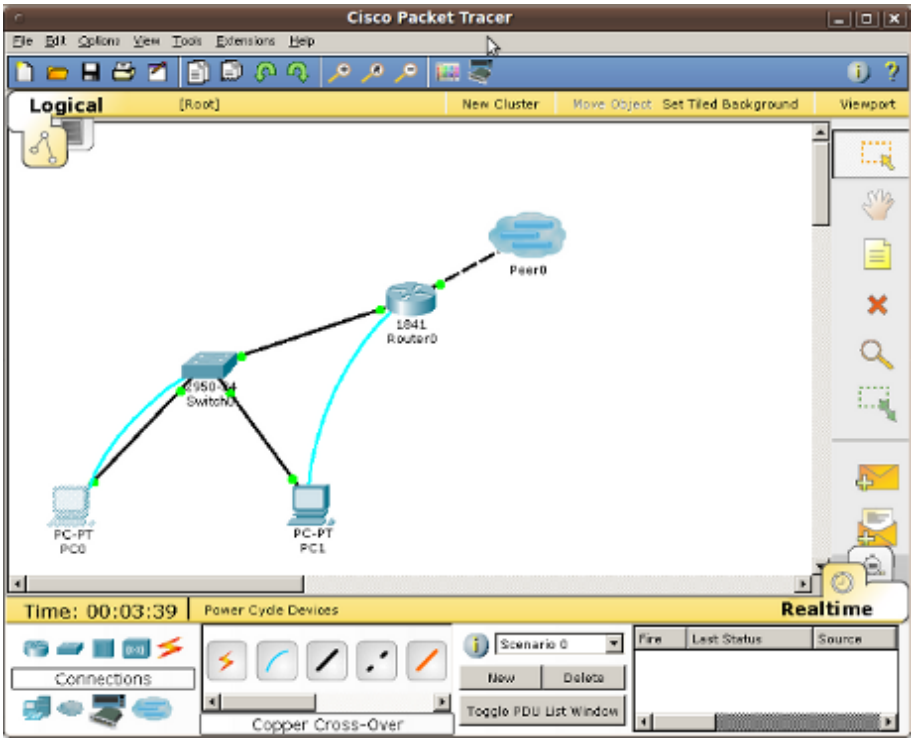
Jedná se o softwarový simulátor, který dovoluje simulovat rozličné síťové prvky. Co se týče nabídky prvků, jsou jeho možnosti omezeny tím co vše implementují vývojáři. Nejsou zde však zatím dostupné všechny protokoly a ne úplně vše funguje zcela správně. Tento simulátor je zaměřen jen na prvky firmy Cisco a Linksys.

Tento projekt je velmi aktivně vyvíjen společností Cisco, která jej vyvíjí pro potřeby svého výukového a certifikačního programu. Umožňuje tímto způsobem svým studentům vyzkoušet si probírané technologie na specifickém hardware. Prozatím je použitelný spíše pro jednodušší úlohy, protože simulované prvky neposkytují všechny funkce dostupné na reálném síťovém hardware. Napojují však tento program na své kurzy, aby docílili lepšího pochopení daného problému studentem.

Od verze 5.0 je možno propojit mezi sebou jednotlivé instance tohoto programu. Toto propojení funguje nad protokolem TCP/IP.

Současná nejnovější verze PT je 5.3 a tato zachovává verzi komunikačního protokolu obsaženou již v 5.0. Tato nová verze opravuje několik chyb a rozšiřuje podporované protokoly. Od verze PT 5.0 přibyla další podporovaná platforma operačního systému. Na platformě Windows byl PT funkční již v dřívějších verzích. K této nejrozšířenější platformě se připojil systém Linux, který je dosti často používán právě pro různé síťové aplikace. Více o této aplikaci se dozvíte například na webových stránkách Cisco Systems [5].

Jak vypadá okno spuštěného programu Packet Tracer je vidět na obrázku 1. V tomto okně je vidět aktivní MU spojení (modrý obláček) a také několik síťových prvků.



Obrázek 1: Okno programu Packet Tracer 5.3

3 Packet Tracer API

Při vydání PT 5.0 došlo také k uvolnění jeho API ve třech různých programovacích jazycích. Jedná se o jazyk Flash, Java a C++. Poslední jmenovaný má dvě verze. Ta starší je provázána s frameworkem Qt, novější již neobsahuje žádnou závislost na Qt. Tyto API dovolují všechny bez výjimky různě rozšiřovat lokálně spuštěný PT. Pro komunikaci s PT jsou zde použity IPC zprávy. Jejich přenos je však prováděn pomocí TCP/IP soketu. Pro přenos zpráv a vyjednání parametrů tohoto přenosu je použito protokolu PTMP. Tento komunikační protokol popisuji v následující kapitole.

Původní verze C++ API obsahuje zakomentovaný kód, který částečně implementuje i stranu serveru PTMP. Obě verze tohoto API mohou být používány na operačním systému Microsoft Windows a Linux. V době dokončování tohoto textu se objevila nová verze toho API, která je rozšířena pro vyvíjenou verzi PT 5.3. Nedošlo k žádným úpravám PTMP, ale rozšiřuje možnosti IPC komunikace.

Verze napsaná v jazyce Flash je jediná, která obsahuje implementaci protokolu MU, který rozšiřuje možnosti PTMP o vzájemnou datovou komunikaci PT. Umožňuje tedy vytvářet virtuální kabelová spojení mezi programy PT a po nich předávat simulovaný provoz. Je zde však implementováno jen několik málo síťových protokolů (CDP, ARP, ICMP), které PT umí. Je však možné, že tyto implementace nejsou zcela funkční. Jediná veřejná aplikace používající toto API je PtBridge vytvořená Gregem Neujarhem. Popis této aplikace jsem umístil do samostatné kapitoly 3.1. Jedná se totiž o program, který provádí to co je potřeba pro úspěšnou integraci PT do prostředí Virtlabu.

API napsané v jazyce Java poskytuje rovněž IPC komunikaci s programem PT (v době dokončování tohoto textu je dostupné pouze pro verzi 5.2.1). Neobsahuje žádné rozdíly proti standardní verzi C++ API, avšak dle PT portálu se jedná o nejpoužívanější API pro rozšiřování PT.

Více se zde o možnostech tohoto API rozepisovat nebudu. Použil jsem jen určité kousky ale jako celek není žádné vhodné pro splnění mého zadání.

3.1 PtBridge

Tato aplikace se skládá ze dvou částí jedna je napsaná v jazyce Flash a přeložena jako Adobe Air aplikace. Greg Neujarh ji pojmenoval AirBridge. Tato část využívá Flash API pro navázání PTMP a MU spojení s Packet Tracerem. Také zajišťuje předávání simulovaných rámců mezi PT a druhou částí – CppBridge, která je již v jazyce C++.

CppBridge se stará o převod CDP, ARP a ICMP protokolů mezi simulovaným provozem PT a reálným. Pro tento převod používá vlastní funkce, ty mi umožnily pochopit jakým způsobem jsou simulované rámce vytvářeny. Bohužel je ze zdrojového kódu CppBridge patrné, že tento formát není shodný s reálnými rámci. Jde z nich vyčíst také, že například protokol CDP má v PT provozu přidán parametr, který se u jeho reálné varianty nevyskytuje. Přesnější popis těchto rámců jsem zahrnul do kapitoly o protokolu MU.

Ve výpise 5, který je mezi přílohami, jde vidět pořadí jednotlivých kroků pro navázání spojení a také přijatý simulovaný CDP rámec. Tento výpis jsem převzal z blogu Grega

Neujarha na PT portálu [10]. V tomto případě se jedná o spojení ve kterém PT figuruje jako server.

4 Packet Tracer Messaging Protocol

Tato kapitola pojednává o komunikačním protokolu PTMP. Jde o přeložený a doplněný text [2]. Tento protokol přidaný do PT od verze 5.0 umožňuje IPC a MU komunikaci s PT. Protokol MU je popisován v následující kapitole. Popis IPC protokolů však tato práce neobsahuje jelikož je použit pouze pro lokální volání při použití API.

Díky němu se mohou spojit a vzájemně komunikovat dvě nezávislé instance PT. Pro datovou komunikaci je použit protokol MU, který je popsán v následující kapitole. PTMP protokol je rovněž použit pro inicializaci IPC komunikace. Tento protokol obstarává nezbytné aspekty pro návazní spojení (vyjednání parametrů, autentizace, ...).

4.1 Architektura PTMP protokolu

Protokol pracuje nad TCP/IP a je navržen jako obecný protokol pro předávání zpráv. Formát zpráv použitých u navázání spojení popisuji v další části této kapitoly. Aplikace využívající PTMP si mohou přidávat nové zprávy a jejich definice. Tyto zprávy mohou být speciálními rámci použitými v MU nebo IPC volání v případě IPC. Závisí tedy na každé komponentě, která využívá tento protokol, jaké zprávy si doplní a jakým způsobem je bude zpracovávat. Obecně můžeme PTMP považovat za další vrstvu nad TCP, kterou mohou využívat aplikace pro komunikaci s PT.

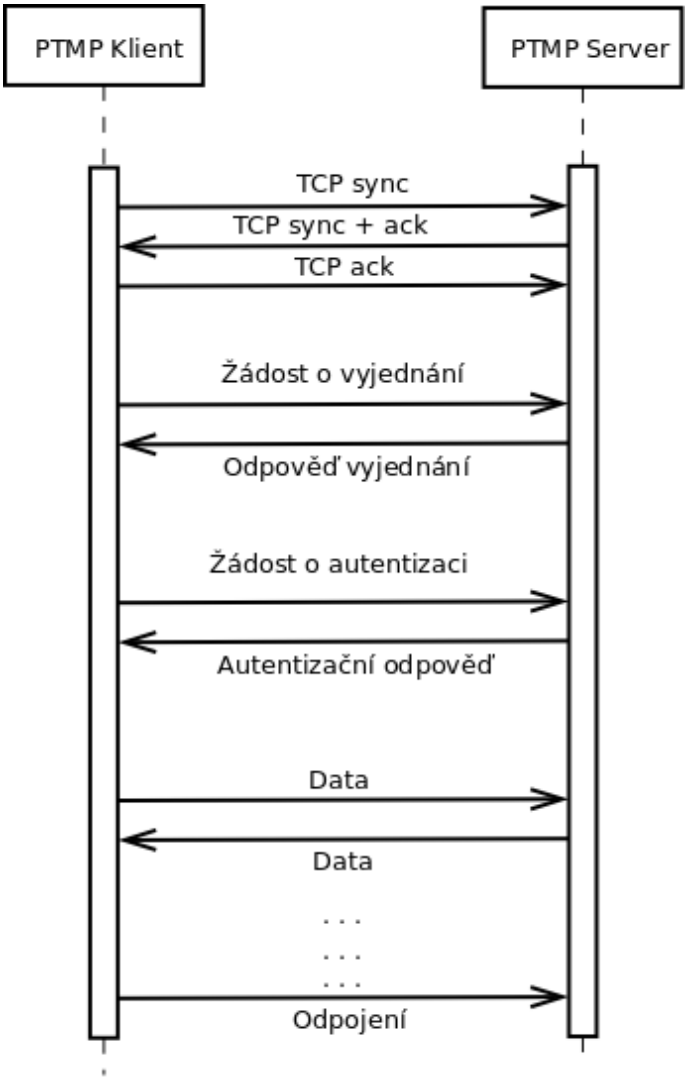
Komunikační schéma pro PTMP aplikace je klient-server, založené na TCP. Aplikace může naslouchat na TCP portu a dovolit jiným instancím navázat spojení. V našem případě jsem právě toto využil a implementovaný PTServer naslouchá na zvoleném portu a obsluhuje klienty. Výchozím portem pro MU je 38000 a pro IPC 39000. Tyto porty nejsou zatím přiřazeny žádné jiné aplikaci. Komponenty mohou samozřejmě tento port měnit. Jedna PTMP aplikace se může spojit s jinou, pokud zná její IP adresu a port.

Když je spojení navázáno, přichází na řadu vyjednání parametrů. Obě strany PTMP musí vědět, jaké parametry pro spojení použijí. Domlouvá se způsob autentizace, režim přenosu zpráv, komprese a šifrování. Pak se autentizuje klientská PTMP aplikace. Je zde tedy možný bezpečnostní problém, identita server není nijak zajištěna.

Po navázání TCP spojení, vyjednání parametrů a autentizaci se již nedá hovořit o konceptu klient-server. Obě PTMP aplikace mají stejné role a funkce. Mohou si tedy navzájem vyměňovat zprávy díky PTMP. Jak probíhají jednotlivé kroky je vidět na obrázku 2.

Jak dovoluje samotné TCP, je možno na jednom příchozím portu navázat více spojení. Pro každé TCP spojení se v Packet Traceru vytvoří (nebo si jej vytvoříme sami) MU spojení, které je reprezentováno graficky vlastním obláčkem.

V dokumentu [2] je napsáno, že pokud si přeje PTMP aplikace ukončit spojení, zašle o tom náležitou zprávu. Ta je popisována dále a je dokonce implementována i v API. Avšak při uzavření spojení v PT k odeslání této zprávy nedojde a je zasláno pouze nějaké nesmyslné číslo.



Obrázek 2: PTMP – architektura protokolu

4.2 Režimy přenosu zpráv

Protože PTMP aplikace mohou být napsány v různých programovacích jazycích, je nutné řešit i režim přenosu zpráv. Například pro aplikace napsané v jazyce Flash je nezbytný textový režim. Nicméně binární režim umožňuje efektivnější převody a kratší zprávy, proto je podporován také binární režim.

Režim přenosu zpráv je vyjednán na začátku každého spojení.

Dokument specifikující PTMP definuje základní datové typy pro zprávy. Tyto typy jsou použity pro základní PTMP zprávy a jsou implementovány i v API Packet Traceru. Je možno definovat i vlastní datové typy, jejich specifikace je již na aplikacích využívajících PTMP. Příkladem by mohly být rámce, které jsou přenášeny v MU. Tyto jsou podobné reálným, avšak jsou upraveny pro snazší zpracování v Packet Traceru (dále budou nazývány pseudorámci).

Formát těchto základní typů a jejich reprezentace v obou typech kódování je ukázán v tabulce 1.

Všechny uvedené typy nejsou povinné. Jsou však užitečné pro udržení kompatibility při přenosu zpráv. Datové typy bez znaménka nejsou zahrnuty, protože jazyk Java nemá podporu pro datové typy bez znaménka.

Binární režim určuje délku nebo ukončovací znak každého datového typu.

4.3 Obecný formát zpráv

Zprávy mohou být přenášeny mezi dvěma PTMP aplikacemi jakmile je navázáno TCP spojení. Tyto zprávy nemusí být poslány v rámci jednoho TCP segmentu. Mohou mít téměř nekonečnou délku ($2^{31} - 1$ = maximální velikost integeru) a mohou být v mnoha TCP segmentech. Jsou však zpracovány pouze v případě, že je přijata celá zpráva. Všechny zprávy musí splňovat následující formát zpráv, aby bylo možno odlišit různé typy zpráv a také kde zpráva končí.

Zpráva obsahuje tyto pole:

- Délka (int) - počet bajtů nebo znaků ve zprávě, popisuje délku typu a obsahu zprávy, není komprimována ani šifrována
- Typ (int) - určuje typ zprávy
- Obsah - data proměnné délky

V poli typu může být číslo, které nabývá hodnoty podle tabulky 2. Formáty jednotlivých zpráv PTMP jsou popsány v následujících částech.

4.4 Stavový diagram

Při navazování PTMP spojení prochází následujícími stavy:

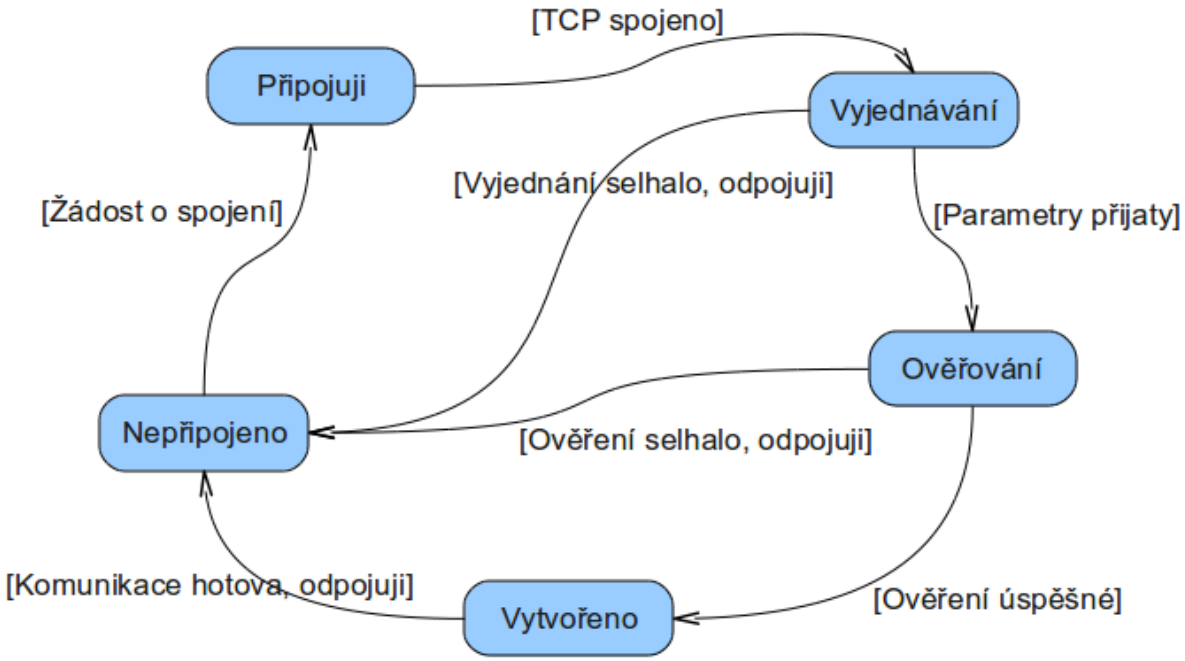
- Nepřipojeno: neinicializováno TCP spojení
- Připojuji: inicializace TCP spojení

Jméno	Binárně	Textově (vše ukončeno pomocí $\backslash 0$)
byte	8-bitová hodnota se znaménkem	hodnota v rozsahu -128 až 127
bool	8-bitová hodnota - true/false	”true” nebo ”false”
short	16-bitová hodnota se znaménkem	hodnota v rozsahu -2^{15} až 2^{15}–1
int	32-bitová hodnota se znaménkem	hodnota v rozsahu -2^{31} až 2^{31}–1
long	64-bitová hodnota se znaménkem	hodnota v rozsahu -2^{63} až 2^{63}–1
float	32-bitová hodnota s jedním desetinným místem	desetinné číslo
double	64-bitová hodnota se dvěma desetinnými místy	desetinné číslo
string	unikódové znaky různé délky ukončené $\backslash 0$	unikódové znaky různé délky ukončené $\backslash 0$
QString	unikódové znaky různé délky ukončené $\backslash 0$	unikódové znaky různé délky ukončené $\backslash 0$
IP adresa	32-bitová hodnota	IP adresa ve formátu x.x.x.x
IPv6 adresa	32-bitová hodnota	IPv6 adresa ve formátu x:x:x:x:x:x:x:x
MAC adresa	48-bitová hodnota	MAC adresa ve formátu xxxx.xxxx.xxxx
uuid	128-bitová hodnota	UUID ve formátu {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHHHHHH}

Tabulka 1: Tabulka PTMP datových typů a způsob jejich reprezentace

Hodnota	Název	Obsah zprávy
0	NEGOREQ	žádost o vyjednání parametrů
1	NEGORESP	odpověď na vyjednání parametrů
2	AUTHREQ	požadavek na autentizaci
3	AUTHCHAL	výzva k autentizaci
4	AUTHRESP	odpověď na autentizační výzvu
5	AUTHSTATUS	status autentizace
6	KEEPALIVE	keep-alive
7	DISCONNECT	odpojení
$>= 100, < 200$	–	IPC zpráva
$>= 200, < 300$	–	Multi-user zpráva

Tabulka 2: Hodnoty pole typ pro PTMP zprávy



Obrázek 3: PTMP – zjednodušený stavový diagram

- Vyjednávání: dochází k vyjednání parametrů použitých pro spojení
- Ověřování: dochází k autentizaci
- Vytvořeno: autentizováno a plně funkční PTMP spojení

Zjednodušený stavový diagram je vidět na obrázku 3. Výchozím stavem je Nepřipojeno.

4.5 Vyjednání spojení

Když je navázáno TCP spojení, tak první krok, který dvě PTMP aplikace udělají je, že si domluví parametry spojení. Je potřeba aby si komunikující komponenty dohodly parametry, které použijí pro toto vzájemné propojení.

PMTP aplikace si vyměňují informace při vyjednávání pomocí zpráv v tomto formátu:

- PTMP identifikátor (string): konstant identifikátor, ”PTMP”
- Verze PTMP (int): 1
- ID PTMP aplikace (uuid): UUID aplikace odesílající tuto zprávu
- Kódování (int): 1 = text, 2 = binární
- Šifrování (int): 1 = žádné, 2 = XOR

- Komprese (int): 1 = ne, 2 = zlib
- Autentizace (int): 1 = čistý text, 2 = jednoduchá, 4 = MD5
- Časové razítko (string): místní čas, kdy bylo zahájeno spojení, ve formátu YYYYMMDDHHMMSS
- Keep-alive (int): keep-alive dobu v sekundách
- Vyhrazeno (string): v současné době nevyužité

Klient posílá žádost o vyjednání spojení na server (zpráva typu 0 - NEGOREQ) se svými navrhovanými údaji, ve zprávě specifikuje své parametry, které navrhuje pro spojení. Server v odpovědi (zpráva typu 1 - NEGORESP) posílá své údaje, které se použijí pro komunikaci. Takto je vyjednání implementováno v současné verzi API.

4.6 Autentizace

Pro autentizaci je použito CRAM (Challenge Response Authentication Mechanism), autentizační mechanizmus založený na požadavcích a reakcích na ně.

Po navázání TCP spojení a vyjednání parametrů spojení začíná autentizační proces. Předpokladem je, že klientská i serverová aplikace využívající PTMP používá stejné ověřovací údaje (jméno a heslo nebo ID a klíč). Je to nezbytné pro úspěšné provedení autentizace.

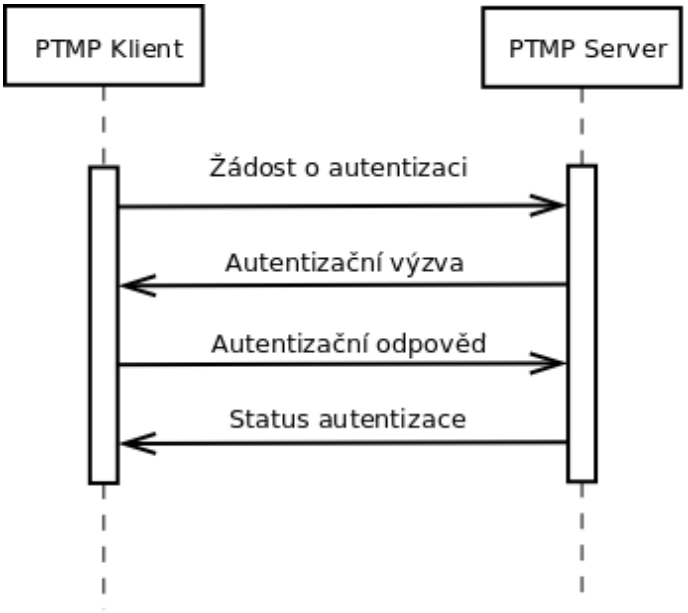
Klient nejprve zašle zprávu se žádosti o autentizaci (typ 2 - AUTHREQ). Server reaguje odesláním výzvy k autentizaci (typ 3 - AUTHCHAL). Výzva obsahuje náhodně generovaný řetězec obsahující 32 znaků.

Pokud byla domluvena autentizace v nešifrované podobě (čistý text), uživatel zašle své heslo server jako nešifrovaný text. V případě použití jednoduché autentizace se použije pro šifrování hesla algoritmus popsaný v části 4.6.5. Pokud se má využít MD5 autentizace, tak se pomocí hašovací funkce MD5 otisk hesla spojeného s vygenerovaným textem z výzvy. MD5 metoda je v PT zvolena jako výchozí. Takto upravené heslo se zašle serveru v autentizační odpovědi (typ 4 - AUTHRESP). Server použije dohodnutou metodu autentizace pro ověření hesla. Poté je klientovi zaslána zpráva o stavu autentizace (typ 5 - AUTHSTATUS). Pokud není autentizace úspěšná, server zašle zpět zprávu o odpojení (typ 7 - DISCONNECT) a ukončí spojení. V PT však uživatel neobdrží žádné upozornění o tom, že autentizace selhala. Pořadí kroků je vidět na obrázku 4.

4.6.1 Formát zprávy typu 2 - žádost o autentizaci

V této zprávě jsou následující pole:

- Uživatelské jméno (string): uživatelské jméno nebo ID klienta (např. ”Peer0”)



Obrázek 4: PTMP – sekvence autentizačního protokolu

4.6.2 Formát zprávy typu 3 - autentizační výzva

Tato zpráva obsahuje pole:

- Text výzvy (string): náhodně generovaný 32 znaků dlouhý text, je použit při šifrování hesla

4.6.3 Formát zprávy typu 4 - odpověď na autentizační výzvu

Pole této odpovědi jsou tyto:

- Uživatelské jméno (string): uživatelské jméno nebo ID klienta (např. ”Peer0”)
- Heslo (string): obsah tohoto pole je závislý způsobu autentizace
- Obecné (string): rezervováno, zatím nevyužito

4.6.4 Formát zprávy typu 5 - status autentizace

Tato zpráva obsahuje pole:

- Stav (bool): true = úspěšná autentizace, false = neúspěšná autentizace

4.6.5 Jednoduchá metoda autentizace

Pro zašifrování hesla je použita následující jednoduchá funkce. Toto je implementace získaná ze zdrojových kódů C++ API.

```
string simple_hash(string password)
{
    string hash;
    for (int i = 0; i < password.length; i++)
        hash[i] = 158 − password[i];

    return hash;
}
```

Výpis 1: Jednoduchý algoritmus pro šifrování hesla

4.7 Šifrování

Z bezpečnostních důvodů je možno komunikaci mezi PTMP aplikacemi šifrovat. Každá zpráva po vyjednání parametrů může být šifrována, pokud se tak aplikace dohodnou. Pro šifrování je použita jednoduchá metoda XOR dat se symetrickým klíčem (tj. server i klient používají stejný klíč pro šifrování a dešifrování).

Šifrovací klíč je odvozen od UUID a časové značky serveru a klienta. Všechny tyto údaje jsou vyměněny při vyjednávání spojení. Unikátní identifikátory UUID jsou náhodně generovány při spuštění aplikace. Jejich formát odpovídá datovému typu uuid definovaném výše. Pro klíč jsou pak použity všechny tyto údaje, které jsou za sebou poskládány v tomto pořadí:

1. UUID server
2. UUID kliena
3. ”PTMP”
4. Časové razítko serveru
5. Časové razítko klienta

4.8 Komprese

Pro kompresi je použita knihovna zlib [3]. Všechny zprávy po vyjednávacím procesu mohou být komprimovány pokud je komprese zpráv domluvena při vyjednávání spojení.

4.9 Pořadí operací PTMP

V popsaném pořadí jsou prováděny jednotlivé výše popsané operace při příjmu a odesílání zpráv.

Když PTMP odesílá zprávu, je použito toto pořadí operací:

1. Ve zprávě je vyplněn typ a obsah zprávy
2. Pokud je PTMP ve stavu autentizování nebo je již spojeno

   (a) Pokud je vyjednáno, je zpráva zkomprimována
   (b) Pokud je vyjednáno, zpráva se zašifruje
3. Spočítá se a vyplní velikost zprávy

Když PTMP přijme zprávu používá toto pořadí úkonů:

1. Získá ze zprávy její velkost
2. Pokud je PTMP ve stavu autentizování nebo je již spojeno
   (a) Pokud je vyjednáno, dešifruje zprávu
   (b) Pokud je vyjednáno, dekomprimuje zprávu
3. Předá zprávu patřičné komponentě využívající PTMP

4.10 Udržování spojení

Keep-aplive mechanizmus je v PTMP používán pro detekci části, která o odpojení neposlal patřičnou zprávou. Časové prodlevy mezi keep-alive zprávami jsou vyjednány na začátku spojení. Klient zasílá ve stanoveném čase zprávy, na které v tomto čase server čeká. Prodleva mezi zprávami je v sekundách. Pokud je při vyjednávání nastavena na nulu, tak keep-alive zprávy nebudou vůbec zasílány. Tyto PMTP zprávy nemají žádný obsah a jsou typu 6 - keep-alive. Pokud nejsou obdrženy tři za sebou jdoucí zprávy, tak PTMP prohlásí spojení za ukončené a informuje o tom aplikaci.

Ve výchozím stavu Packet Traceru pro MU spojení je volba pro udržování spojení nastavená na 0 a tudíž nejsou žádné zprávy tohoto typu zasílány. Toto chování je však pro nás výhodné, jelikož se šetří přenosovou kapacitou.

5 Multi-user connection

Tento protokol je nadstavbou nad protokolem PTMP a právě on umožňuje výměnu pseudorámců mezi instancemi PT. Tento protokol má poněkud jednoduší stavbu než výše popisované PTMP. Využívá jím vytvořené a obsluhované spojení a přidává do PTMP nové typy zpráv, které pak sám zpracovává.

Pseudorámce jsou přenášeny po virtuálních spojích - linkách, které jsou odlišeny jedinečným číslem linky. V PT jsou MU spojení zobrazována formou bleděmodrého obláčku se třemi pruhy (jeho vzhled je na obrázku 5). První dva pruhy indikují úspěšné PTMP spojení a poslední pruh pak signalizuje úspěšnost MU spojení. Tento prvek pak zobrazuje i informace o linkách. Každý připojený port zařízení má vlastní linku. Jedná se o dvoubodové spojení. Každý takovýto obláček má své jméno, které je jedinečné v rámci jedné instance PT.

5.1 Architektura MU protokolu

Tento protokol využívá již vytvořené PTMP spojení i se všemi parametry. Jakmile je pomocí protokolu PTMP spojení vytvořeno a protokol se dostane do stavu spojeno, tak se započne s navázáním MU spojení. Zašlou se zprávy, kterými si server a klient vymění uživatelské jméno a své UUID. Poté si vymění informace o již vytvořených linkách. Ty jsou číslovány od nuly, ale při odebírání dochází k přepočítání jejich čísel. Vždy se jedná o souvislou řadu čísel, proto se musí informovat obě strany o změnách a udržovat si synchronizované informace o linkách.

Poté může být zaslána zpráva o změně jména MU spojení (PT ji posílá vždy). Jedná se o jméno, které je použito pro popis obláčku v PT. Dále následuje zpráva s informací typu (202 - MUPORTADV). Tato zpráva je obvykle prázdná, její smysl a obsah je popsán dále v této kapitole.

Po výměně všech potřebných zpráv je MU spojení úspěšně navázáno a pokud je vytvořena i nějaká aktivní linka, tak již od této chvíle může přenášet pseudorámce.

V tabulce 3 jsou jednotlivé čísla zpráv přidaných do PTMP.

5.2 Vyjednání MU spojení

Aby bylo MU spojení funkční, je nezbytné, aby klient zaslal zprávu MUNEGOREQ a server mu odpověděl zprávou MUNEGORESP. Tyto zprávy mají shodný obsah, liší se jen jejich typ. Tato zpráva obsahuje tyto pole:

- Uživatelské jméno (string): uživatelské jméno, obvykle nastaveno na ”Guest” což je výchozí hodnota v PT
- Uuid aplikace (uuid): UUID spuštěné aplikace

Tato zprávy jsou tedy nezbytné pro korektní fungování celého MU spojení. Uživatelské jméno použité touto zprávou je v případě PT nastavitelné v uživatelském profilu. Pro úspěšné spojení tato hodnota může být jakákoliv, jelikož není nikde použita.

Obrázek 5: PT – vzhled odpojeného a připojeného MU obláčku

Hodnota	Název	Obsah zprávy
200	MUNEGOREQ	žádost o vyjednání MU spojení
201	MUNEGORESP	odpověď na vyjednání MU spojení
202	MUPORTADV	informace o portech
203	MULINKUPDATE	informace o lince
204	MULINKUPDATESTATUS	status proběhnuté operace z informační zprávy
205	MUPDU	pseudorámec
206	MUSAVENETREQ	požadavek na uložení sítě
207	MUSAVENETRESP	odpověď na uložení sítě
208	MUCONIN	vstup konzole
209	MUCONOUT	výstup na konzoli
210	MUNAMEUPDATE	nové jméno

Tabulka 3: Hodnoty pole typ pro MU zprávy

5.3 Informace o portech

Program PT posílá zprávu MUPORTADV, vždy při navázání MU spojení i když je prázdná. Tato zpráva slouží k informování protistrany o dostupných portech. Uživatel může v nabídce PT nastavit, které porty jsou viditelné přes MU spojení. Díky tomuto nastavení pak tyto porty mohou být použity pro vytvoření linky. Tato zpráva pak obsahuje základní pole a na ně dynamicky navázané informace o portech. Vypadá takto:

- Délka (int): délka následující zprávy o portech
- Zpráva o portech: obsahuje informace o portu v tomto formátu, tyto informace mohou byt obsaženy vícekrát
  - ID (int): identifikační číslo portu
  - Typ (int): typ portu viz. tabulka 4
  - Jméno (string): jméno portu

Tato informační zpráva není pro správné fungování MU spojení nezbytná.

5.4 Informace o lince

Celý přenos pseudorámců mezi PT je založen na přenosech po linkách. Toto by se mohlo porovnat se systémem VLANů 802.1Q. Aby obě strany měly stejné informace o připojených

Hodnota	Význam
1	konzole (v prvcích obvykle RJ45)
2	10BASE-T (RJ45)
3	100BASE-TX (RJ45)
4	1000BASE-T (RJ45)
5	100BASE-FX (SC)
6	1000BASE-LX (SC)
7	serial (DB 60)
8	serial (smart serial)
18	modem (RJ11)
19	serial (RS 232)
21	koax (BNC)

Tabulka 4: Hodnoty pole typ portu pro MU zprávy

linkách, je zde typ zprávy, která výměnu těchto informací umožňuje. Jsou podporovány operace vytvoření nové linky, změna parametrů linky, odpojení a smazání linky. Každá linka má přiřazeno číslo, které je počítáno od 0 a je závislé na pořadí vytvoření linky. Čísla linek jsou v souvislé řadě a při mazání tedy musí dojít k jejich přepočítání. Formát zpráv typu 203 - MULINKUPDATE je takovýto:

- ID operace (int): ID operace s linkou (nyní vždy 0)
- Typ operace (int): typ operace, která se má s linkou provést, 0 = vytvoření nové linky, 1 = změna informací o lince, 2 = odpojení linky, 3 = smazání
- ID linky (int): ID linky
- UUID linky (uuid): unikátní identifikátor linky
- ID portu (int): ID portu
- Typ kabelu (int): typ kabelu viz. tabulka 5
- Jméno portu (string)
- Typ portu (int): typ portu viz. tabulka 4
- Stav portu (bool): 1 = aktivní, 0 = neaktivní
- Přímé zapojení (bool): 0 - křížený kabel, 1 - přímý
- Automatické křížení (bool)
- Rychlost portu (int)
- Fullduplex (bool)

Hodnota	Význam
0	standardní měděný UTP kabel
1	přímý UTP kabel
2	křížený UTP kabel
3	konzolový kabel
4	optický kabel
5	sériový kabel
6	telefonní kabel pro modem
7	koaxiální kabel

Tabulka 5: Hodnoty pole typ kabelu pro MULINKUPDATE

- Autonegociace (bool)
- Vyjednání rychlosti (bool)
- Vyjednání duplexity (bool)
- Clockrate (int)
- DCE Port (bool)

V tabulce 5 jsou uvedeny typy kabelů, jak jsou definovány v programu PT.

Na tuto zprávy by měla odpovídat zpráva typu 204 - MULINKUPDATESTATUS. Tato má obsahovat ID operace a její status, avšak v Informačních zprávách o lince (203 - MULINKUPDATE) je ID operace vždy 0 a zpráva o statusu není zatím využita a odesílána. Možná je s ní počítáno do budoucna ale současný stav je taktéž plně funkční. Pro správnou funkci se mi tato zpráva nejeví ani příliš potřebná.

S těmito zprávami souvisí i to, že si díky nim může aplikace PT vytvořit tabulku přehledu jednotlivých linek, která obsahuje informace o lokálním a vzdáleném stavu připojeného portu (up/down) a také k jakému zařízení je linka připojena. Všechny tyto informace se zobrazují v PT.

5.5 Pseudorámec

Zpráva typu 205 - MUPDU obsahuje samotné pseudorámce, pomocí kterých dochází k výměnám dat mezi propojenými simulovanými zařízeními. Její základní formát je velmi jednoduchý. Obsahuje tato pole:

- ID zprávy (int)
- ID linky (int): ID linky kterou pseudorámec prochází
- PDU: pole proměnné délky obsahující pseudorámec

ID zprávy je na straně PT zřejmě generováno náhodně. Čísla na sebe nenavazují a nemusí jít nutně ani za sebou. Momentálně nejspíše toto pole není nijak využito ke zpracování. Číslo linky však ke správnému doručení zprávy nezbytné je. Pro to aby pseudorámce byly posílány linkou je nezbytné, aby byly oba konce linky aktivní. Samotný obsah PDU je různý, avšak základní informace o typu přenášeného rámce jsou obsaženy textově. Tyto informace jsem získal při analýze PTMP potažmo MU provozu pomocí síťového analyzátoru Wireshark. Jedná se vždy o textových soupis všech částí, které jsou v rámci přenášeny. Například pro ARP rámec vypadá tato textová hlavička takto "CEthernetIIHeader CArpPacket".

Obvykle pseudorámce obsahují velmi podobná pole jako v případě reálných rámců. Rozdíl který však u všech pseudorámců je, že téměř všechny délkové části a kontrolní součty jsou nulové. Také je zachováno obrácené pořadí jednotlivych síťových protokolů v pseudorámcích. Po textové hlavičce tedy následuje nejhlouběji zanořený obsah rámce (například samotný obsah protokolu ARP) a na konci se nachází část spojové vrstvy (např. ethernetové adresy).

U některých protokolů jsou však přidány do pseudorámců parametry, které se v reálných rámcích daného protokolu nevyskytují. Také formát některých parametrů neodpovídá zcela reálným rámcům. Z tohoto důvodu je potřeba pro každý podporovaný protokol použít funkci, která se postará o korektní převod dat.

5.6 Uložení sítě

Zprávy typu 206 - MUSAVENETREQ a 207 - MUSAVENETRES umožňují uložení simulované sítě. Jsou zde z důvodu dvou režimů, které PT obsahuje. Tím prvním je režim reálného času, který funguje jako standardní síť a dovoluje simulovat chování v reálném čase. V tomto režimu PT funguje při spuštění.

Druhý režim je simulační. V něm dochází k zobrazování provozu, který se sítí šíří. Je možno prohlížet si v okně obsah rámce. Zobrazení těchto rámců odpovídá tomu jak jsou rámce prezentovány v reálné síti. V tomto simulačním režimu lze filtrovat protokoly, pokud se zajímáme jen o určitý síťový protokol.

Tento režim nabízí automatické zachytávání a přehrávání. V tomto simulovaném režimu je však přenos rámců značně zpomalen z důvodu názornosti.

Pokud si uživatel přeje při navázaném MU spojení přepnout svou síť do simulovaného režimu, tak dojde ke spuštění nové instance PT. V ní má uživatel celu svou topologii a základní informace o vzdáleném konci. Tyto zprávy jsem však podrobněji nezkoumal, nepovažuji je za potřebné pro účely integrace s Virtlabem. Jejich formát zde také nebudu rozepisovat jelikož nejsou v implementaci mé práce využity.

5.7 Vstup a výstup konzole

Jelikož je možné připojit skrze MU i konzole vzdálených prvků, jsou v MU definovány i zprávy pro přenos těchto dat skrze MU spojení. Nejsou v nich však přenášeny příkazy a jejich výstupy. Podrobněji jsem se jimi nezabýval, pro funkčnost mé implementace ne-

jsou potřeba. Připojení na konzolová rozhraní síťových prvků je ve Virtlabu umožněno pomocí rozhraní v internetovém prohlížeči.

5.8 Změna jména MU spojení

Zpráva pro změnu jména je aplikací PT zaslána ihned po úspěšném navázání MU spojení. Není nezbytná pro funkčnost a je spíše o informaci pro uživatele. Díky nim ví i o změnách jména vzdáleného konce spojení. Jedná o název MU obláčku. Tento si uživatel může měnit jak chce během chodu programu, a proto je zde zpráva o změně jména.

Tyto zprávy mohou být zasílány jak serverem tak klientem. Tato zpráva má následující formát:

- Jméno (string) - jméno konce MU spojení

6 Návrh řešení

Po prozkoumání všech možností PT API a protokolů, které fungují při propojení mezi dvěma programy Packet Tracer byly uvažovány dvě možné varianty řešení.

První variantou by bylo vytvoření programu, který by si spustili uživatelé Virtlabu přímo na stejném počítači jako Packet Tracer a tento program by mohl už pak přímo komunikovat s Virtlabem, kde by se pro tyto účely vytvořil další modul v tunelovacím serveru. Toto řešení by mělo tu výhodu, že zátěž ohledně překladu paketů by byla na straně klienta. Také by bylo možno více využít již hotové kódy PT API. Nejspíše by se však hůře odstraňovaly problémy při spojení této aplikace s Virtlabem. Rovněž by nejspíše nastávaly problémy s aplikací na počítačích klientů (studentů). Také by bylo potřeba mít tento systém schopný provozu na obou podporovaných platformách. Ať již by to bylo docíleno přenositelným kódem v jazyce Java, nebo rozdílnými binárními kódy pro obě podporované platformy. Je tedy pravděpodobné, že by se takovéto řešení hůře udržovalo a rozšiřovalo.

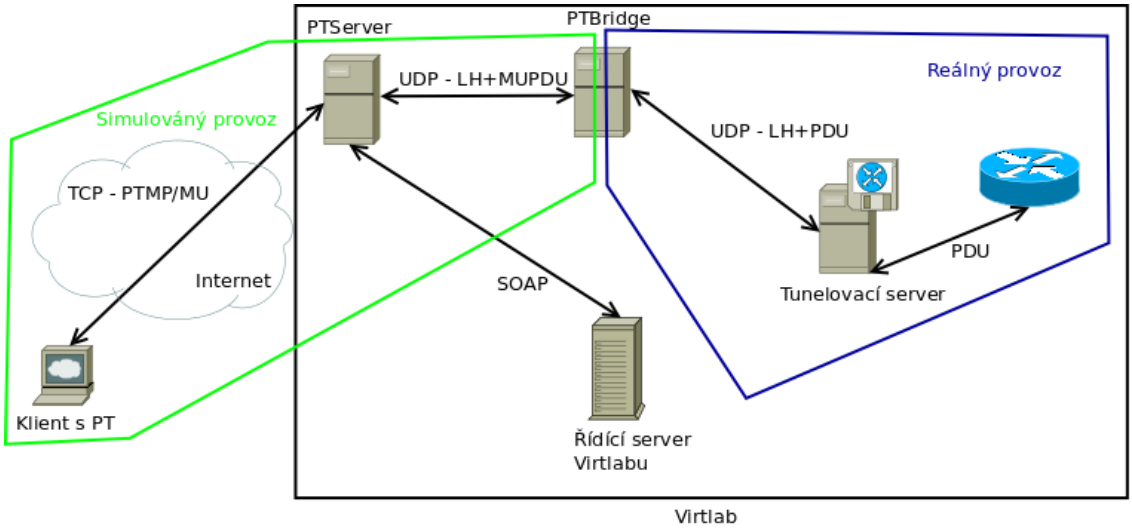
Druhou cestou je využít vyvíjenou aplikaci jako server pro připojování klientů s programem Packet Tracer. Toto řešení má rozhodně výhodu v tom, že je možno celý systém lépe spravovat a udržovat. Také bude jistě snazší odstranit chyby aplikace a vyrovnat se s případnými zásahy do protokolu.

Pro implementaci bylo rozhodnuto, že využijeme řešení, kdy je serverová strana (dále jen PTServer) umístěna na serveru Virtlabu. K tomu nás vedla především možnost lepší správy tohoto řešení.

Toto řešení umožní připojení klientů, kteří jsou v dnešním Internetu často schování za nějakým typem NATu a také umožní mnohem lepší kontrolu a detekci poruch při případných změnách protokolu. Dá se však očekávat, že PTMP v nejbližší době žádných velkých změn nedozná, jelikož požadavky na tento protokol kladené plní dobře. Co se týče protokolu MU a v něm přenášených rámců, tak bych také neočekával zásadní změny (tyto rámce se snaží napodobovat zhruba obsah reálných rámců). Pravděpodobně půjde vývoj PT stejným směrem jako nyní. To znamená, že bude vylepšována stabilita celé aplikace a přidána podpora pro další protokoly a příkazy, tak aby rozšířily možnosti tohoto simulátoru.

Celé řešení integrace je možno rozdělit do dvou komponent. Toto rozdělení je implementováno úplně. Každá z komponent je schopna samostatného běhu. Pro funkčnost implementace jsou však potřeba obě. Jak jsem uvedl výše, komponenta přijímající a obsluhující spojení s PT je nazvána PTServer. Druhou část, která se stará o překlad pseudorámců a rámců jsem pojmenoval PTBridge.

Toto oddělení je zde proto, aby bylo případně možno provozovat je odděleně, v případě, že by se jednalo o velmi zatíženou komponentu. Celkový koncept návrhu je vidět na obrázku 6. Mezi PTServerm a PTBridgem jsou posílány pomocí UDP pouze pseudorámce rozšířené o hlavičku (LH), její obsah a práce s ní je popsána v následující kapitole, která se zabývá implementací.



Obrázek 6: Celkový koncept řešení

6.1 PTServer

Jak název této komponenty napovídá jedná se o server, který obsluhuje příchozí spojení s klientskými PT. Autentizuje klienty a celkově jeho primární zaměření je na PTMP. Jeho práce je také komunikovat s řídícím serverem Virtlabu, jelikož od něj potřebuje získat informace o uživatelích, kteří mohou být připojeni. Sám řídícímu serveru předává informace o linkách vytvořených v MU spojení. Tato komunikace bude probíhat pomocí protokolu SOAP, který je již nyní ve Virtlabu nasazen na komunikaci mezi řídícími servery lokalit. Na základě informací o připojených linkách vytváří řídící server potřebné konektory.

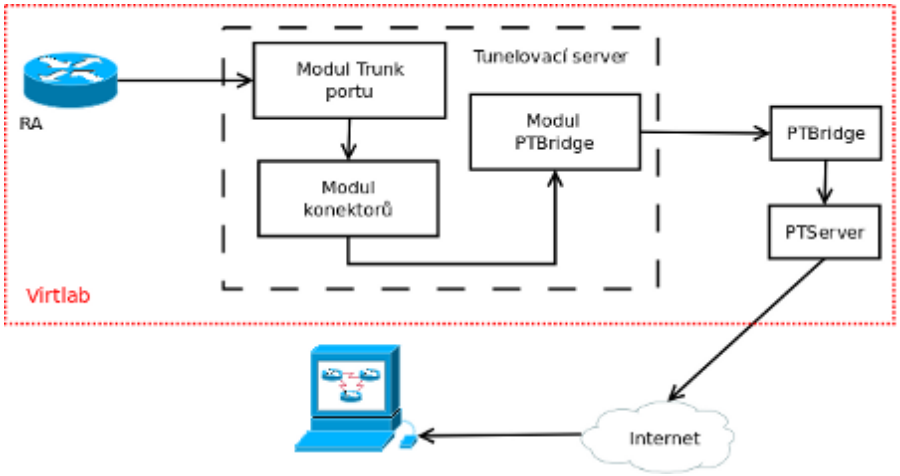
Implementaci konektorů realizoval Pavel Burda v rámci své diplomové práce [1]. Umožňují jejich vzájemné dynamické propojování. Přes takto propojené konektory se pak mohou normálně přenášet data stejně jako by se jednalo o kabelové spojení. Propojování konektorů je možné měnit za běhu aplikace, ale toto již zajišťuje obslužná logika konektorů.

Také na straně PTServeru funguje SOAP server, který přijímá informace o propojení konektoru. Pokud je propojen konektor vytvořený PTServerem s konektorem přidaným v konkrétní úloze je již možné přenášet přes celou soustavu data.

Každá z lokalit Virtlabu bude provozovat vlastní instanci PTServeru pro své uživatele.

6.2 PTBridge

Aby bylo možno oddělit překlad reálných rámců z Virtlabu na pseudorámce v PT, je zde samostatná komponenta PTBridge. Ta bude zajišťovat jen vzájemný překlad rámců. Toto oddělení je zde pro případ, že by se jednalo o vytíženou komponentu. Díky vyjmutí z hlavního programu ji bude možno provozovat například na samostatném serveru.



Obrázek 7: Cesta rámce z reálného prvku na počátač s PT

Podobnou úlohu plní i aplikace navázaná na PT API zmiňovaná v kapitole 3.1. Nám však poslouží dobře jako odrazový můstek, a také pro základní pochopení systému jakým jsou přenášený rámce mezi instancemi PT.

Tato komponenta musí býti stejně jako PTServer spuštěna na každé lokalitě.

Na obrázku 7 je jednoduše zakresleno jak rámec z laboratorního prvku Virtlabu putuje k počítači, který má spuštěn a připojen PT.

7 Implementace

Pro implementaci celé práce jsem použil jazyk C++. Pro daný úkol se nabízí jako nejlepší možný. Jelikož jsme se rozhodli pro řešení založené na PTServeru, který bude umístěn v architektuře Virtlabu. Servery této laboratoře jsou postaveny na operačním systému Linux. Z hlediska výkonu a požadovaného nasazení se jazyk C++ jeví jako nejlepší varianta. První verze PTServeru navíc používala části zdrojového kódu z PT API, který byl napsán právě v C++, ovšem obsahovalo i vazby na knihovny Qt. Současná verze již tyto vazby neobsahuje a používá část kódů z novější verze C++ PT API.

Celé řešení je v podstatě odděleno od stávajících komponent distribuované laboratoře. Toto řešení bude nejspíše využito spíše v budoucnu v případě, že bude docházet k většímu využívání implementovaného řešení.

Spojení, které se vytváří mezi PTServerm a klientem, který má spuštěn PT, je vytvářeno pomocí výše popsaného protokolu PTMP a MU. Oba fungují nad protokolem TCP/IP. Zde bylo nezbytné použít to, co je použito k vzájemnému propojení mezi programy Packet Tracer.

Mezi PTServerem a PTBridgem již není potřeba stavové spojení. Zatím se počítá s nasazením na jeden stroj a oddělení obou části je zde spíše pro možnou budoucnost. Je zde tedy použito UDP spojení. Po tomto jsou již přenášený jen rámce obsahující hlavičku linky a pseudorámce. Tato hlavička je pak použita i v rámci zaslaném tunelovacímu serveru, který se stará o předání na patřičné rozhraní laboratorního prvku Virtlabu (záleží již na subsystému konektorů). Hlavička linky (LH) obsahuje IP adresu klienta, číslo TCP portu jeho připojení a číslo linky po které je pseudorámec přenášen. Například tato hlavička může vypadat takto "127.0.0.1:5896@0". Poslední číslo je právě číslo linky. Před tímto polem je přenášena jen informace o délce dané hlavičky (celé číslo). Toto jsou veškeré potřebné informace potřebné pro správné doručení pseudorámce PT.

PTBridge zajistí překlad pseudorámce na formát, který je používán u standardních rámců. Samozřejmě musí doplnit nezbytné parametry, která pseudorámce postrádají.

Proto, aby bylo vše funkční, bylo potřeba provést i drobné změny ve stávajícím obslužném kódu Virtlabu. Ty jsou popsány v samostatné podkapitole.

7.1 PTServer

Komponenta PTServer má za primární cíl umožnit připojení PT klienta. Implementován je jako serverová část PTMP, takže obsluhuje a odesílá potřebné zprávy. Tato část taktéž komunikuje s řídícím serverem Virtlabu. Tato komunikace obsahuje možnost získat přihlašovací údaje pro účely autentizace a autorizace vzdálených PT. Přihlašovacími údaji se rozumí uživatelské jméno a heslo. Heslo je dočasně generováno pro aktivní úlohy.

Pro každé příchozí spojení (připojení klienta) je vytvořeno obslužné vlákno. Tudíž server může nadále obsluhovat další příchozí spojení. Počet vláken tedy odpovídá počtu připojených klientů. Mezi vlákny k žádné komunikaci nedochází.

PTServer zajišťuje jak již bylo výše napsáno autentizaci PT klienta. V případě, že autentizace selže (uživatel zadá špatné jméno nebo heslo) tak je řídícímu serveru zaslána zpráva o neúspěchu. Pro získávání údajů a posílání zpráv řídícímu serveru Virtlabu je

použito protokolu SOAP. Zpráva o neuspěšné autentizaci se uživateli v koncovém PT nezobrazí. Pak je spojení rozpojeno a vlákno končí svou činnost. V případě úspěšné autentizace je navázáno spojení a server může obsluhovat zprávy o linkách. Po připojení klienta není žádná linka v MU spojení mezi uživatelem a PTServerm vytvořena. Uživatel nejprve musí ve svém PT vytvořit novou. Jakmile tak učiní, je o tom informován pomocí SOAPu řídící server. Takto vytvořenou linku je pak možno propojit s konektorem v aktivní úloze. Pokud je vytvořený konektor propojen s konektorem v topologii, je informován PT, že vzdálený konec linky je zapojen (aktivní). Z tohoto důvodu se udržují informace o obou koncích linky.

Momentálně jsou podporovány pouze ethernetové linky. Virtlab v současné době podporuje pouze vzdálené propojování právě ethernetových portů. Zprávy o linkách, které nejsou ethernetového typu, jsou ignorovány.

Psedourámce jsou přenášeny pouze pokud jsou oba konce linky aktivní. Tak je tomu díky PT, která čeká na aktivaci obou konců linky. Toto ulehčí zátěži, která by mohla vzniknout při překladech všech rámců. Pokud dojde ke smazání nebo odpojení linky, tak je konektor představující tuto linku rovněž odstraněn. Při rozpojení jsou odstraněny všechny PT konektory daného spojení. Propojování pseudo-konektorů (linek zakončených ve Virtlabu) mezi sebou není zatím dovoleno. Pokud bychom stáli o to vytvořit server pro připojování PT mezi sebou (jakýsi proxy server), tak by bylo rozhodně vhodnější vynechat komponentu starající se o překlad rámců.

Pro překlad této aplikace je na cílovém systému potřeba gSOAP Toolkit [6]. Ten je zde pro podporu komunikace protokolem SOAP s řídícím serverem Virtlabu. Dále je potřeba knihovna *pthread*, protože jsou použita pro implementaci vlákna. Žádné další speciální požadavky na knihovny PTServer neobsahuje.

Výpis všech chyb a systém logování je použit dle zavadeného standardu v systému Virtlab. Pro tyto účely jsem převzal třídu z bakalářské práce Václava Bortlíka [7].

7.2 Popis tříd PTServeru

Nyní krátce popíši všechny důležité třídy, které se nacházejí ve zdrojových kódech PTServeru.

PTServer – primární třída, která naslouchá na nakonfigurovaných portech. Příchozí klientské spojení předává dále třídě **PTConnection**. Sama obsahuje metody pro komunikaci s PTBridgem. V samostatném vlákně spouští SOAP server obsažený v třídě **SoapServer**.

PTConnection – třída zajišťuje komunikaci s PT klientem. Obsahuje vše potřebné pro provoz a obsluhu PTMP a MU spojení. Používá třídu **SoapClient** pro SOAP komunikaci s řídícím serverem Virtlabu.

PTMPMessage – obsahuje logiku pro zpracování PTMP a MU zpráv. Jednotlivé zprávy pak dědí tuto základní, třídu ale jsou umístěny ve vlastních souborech.

Messages – součástí je implementace logování dle pravidel Virtlabu. Tuto třídu jsem převzal z již bežícího systému tunelovacího serveru.

MD5 – třída převzata z C++ API. Obsahuje implementaci MD5 hašovacího algoritmu.

MULinks – implementace tabulky linek. Obsahuje metody pro práci s touto tabulkou. Skladuje informace ve vektoru, která obsahuje objekty **MULink**.

PTBuffer – buffer použitý pro práci se zprávami a rámci. Nezbytná pro PTMP. Obsahuje i práci se základními PTMP datovými typy.

Všechny třídy jsou logicky rozděleny do složek. Složka PTMP obsahuje například všechny PTMP zprávy, jejich zpracováni a také obslužné třídy. Složka MU obsahuje zprávy protokolu MU. Ve složce soap se nachází implementace SOAP funkcí a rovněž všechny soubory generované gSOAPem. Ve složce zlib je pak kompresní knihovna zlib. Složka crypto obsahuje třídu **MD5**.

Pro lepší pochopení propojení těchto tříd jsem vytvořil jednoduchý sekvenční diagram 8. Na něm je však pouze obecné propojení, nejedná se o nějakou z konkrétních činností.

7.2.1 Konfigurace PTServeru

Pro to, aby bylo možné měnit důležité parametry je zde konfigurační soubor. Jeho formát je poměrně jednoduchý a příklad lze vidět na výpise 2. Mezi parametry a hodnotami je mezera. Je možné do tohoto souboru vkládat komentáře uvozené znakem mřížky – "#". Řádky začínající tímto znakem jsou ignorovány. Jednotlivé parametry pak musí být odděleny ukončením řádku.

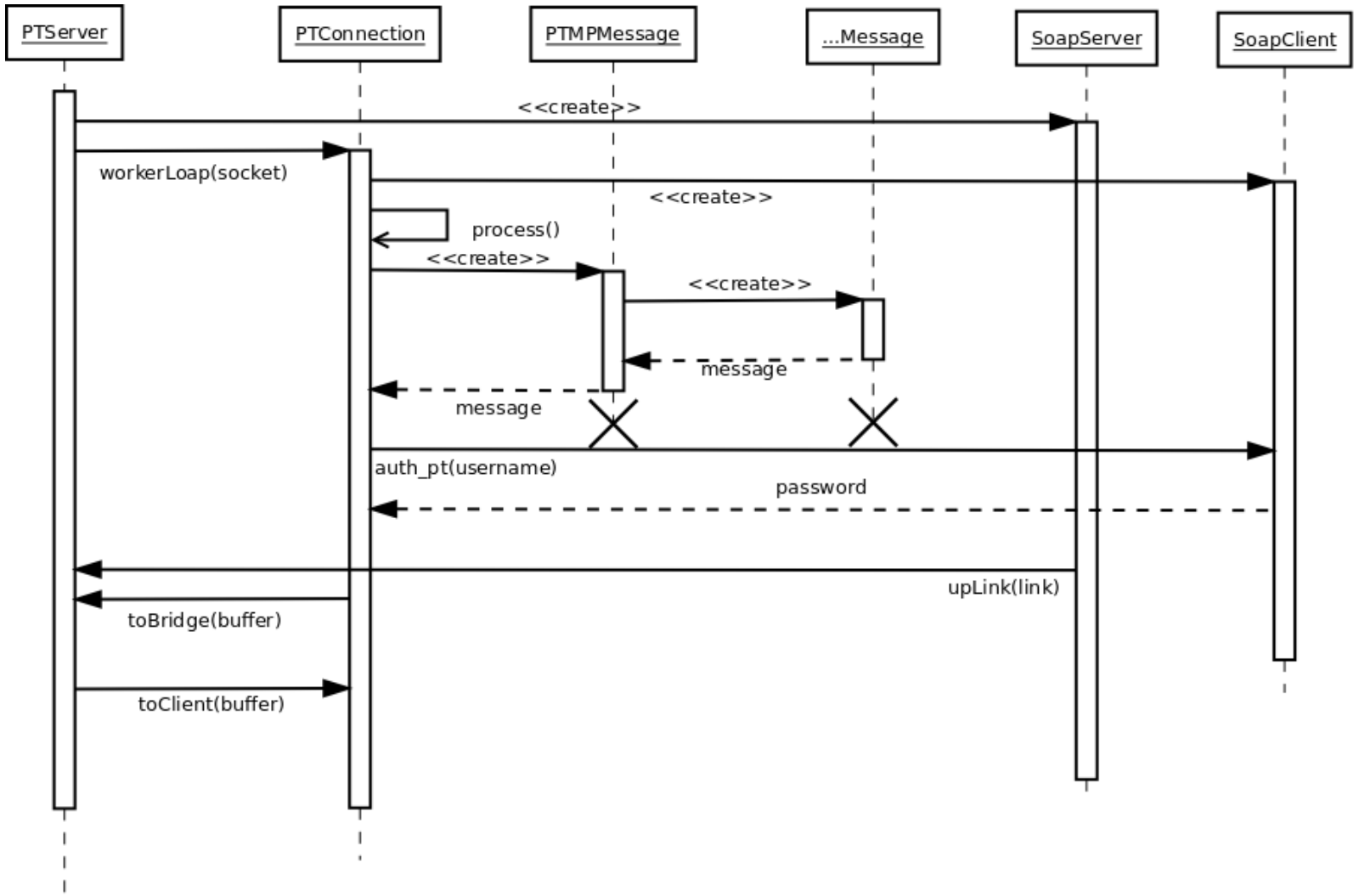
```
# Tyto parametry jsou pouzity pro poslani pseudoramce PTBridge
# Port na kterem PTBridge naslouchа komunikaci od PTServeru
PTBridge-port 38100
# IP adresa nebo jmno hostitele pro PTBridge
PTBridge-ip localhost

#############################################################################
# Tyto parametry jsou pouzity pro prijem pseudoramcu z PTBridge
# IP adresa nebo jmno hostitele pro pjem dat
PTServer-ip 127.0.0.1
# Port pro pjem pseudoramcu
PTServer-port 38101

#############################################################################
# Udaje pro prichozi spojeni od PT
# IP adresa nebo jmno hostitele
Listen-ip 127.0.0.1
# Port na kterem je naslouchano (toto je vychozi nastaveni pro MU spojeni)
Listen-port 38000
#############################################################################
# Parametry pro SOAP komunikaci mezi ridicim serverem a PTServerem
# Port na kterm naslouchа SOAP server na strane PTServeru
PTSsoap-port 8080
# Cilove misto SOAPu na Virtlabu
VtSoap-location http://localhost/soapserver.php
```

Výpis 2: Ukázka konfiguračního souboru pro PTServer

Umístění tohoto konfiguračního souboru je možno určit v hlavičkovém souboru. Je tedy nezbytné, aby bylo známo již v době kompilace.



Obrázek 8: Jednoduchý sekvenční diagram PTServeru

Několik parametrů je však možno nastavit i při spuštění programu. Pro výpis těchto parametrů stačí spustit program s parametrem *-h*. Ten zajistí výpis všech dostupných parametrů.

7.3 PTBridge

Tato část je klíčová pro vzájemný překlad rámců. Je nezbytné překládat pseudorámce, které jsou posílány z PTServeru na reálné rámce. Ty jsou pak posílány rovněž pomocí protokolu UDP tunelovacímu serveru. Při obou těchto přenosech je použita hlavička linky (LH), která není PTBridgem nijak měněna ani na ni není pohlíženo.

Má implementace funguje jako jeden proces, který obsluhuje obě tato spojení. Není zde potřeba udržovat TCP spojení jelikož je potřeba maximálně urychlit vzájemné přenosy. Každý přijatý rámec je zpracován k tomu vytvořenou funkcí. Současná verze zvládá převádět rámce protokolů CDP, RIP, ARP a ICMP. U protokolu ICMP však fungují jen echo zprávy. Jiné nejsou v programu PT podporovány.

Pro lepší možnost testování a vývoje překladových funkcí je možno spustit PTBridge v módu, ve kterém odesílá všechny rámce na vybrané síťové rozhraní (podporovány jsou jen ethernetové síťové karty). Taktéž na nastaveném rozhraní naslouchá provozu a ten pak předává pro zpracování. Po přepracování rámce na pseudorámec je odeslán dále na PTServer. Pro tuto funkčnost je využito knihovny pcap [14], která je známa z programů pro zachytávání a analýzu síťového provozu. Zachytávání rámců na síťové kartě se v tomto případě spouští v samostatném vlákně. Pro tuto funkčnost je nutné spouštět aplikaci s právy superuživatele. Pro spuštění tohoto režimu je nutno zadat parametr *-i=*, za kterým následuje číslo síťového rozhraní. Čísla rozhraní se vypíšou v případě, že je program spuštěn s právy superuživatele.

Pro každý protokol a směr je v této implementaci vlastní překladová funkce. Bohužel se mi zatím nepodařilo získat žádné lepší údaje o tom jak, jsou pseudorámce tvořeny. Zachoval jsem tedy systém, který je použit rovněž u PtBridge 3.1. Předpokládám však, že po tom co se podaří získat od vývojářů PT k tvoření pseudorámců slíbenou pomoc, bude moci být tento překladový systém předělán.

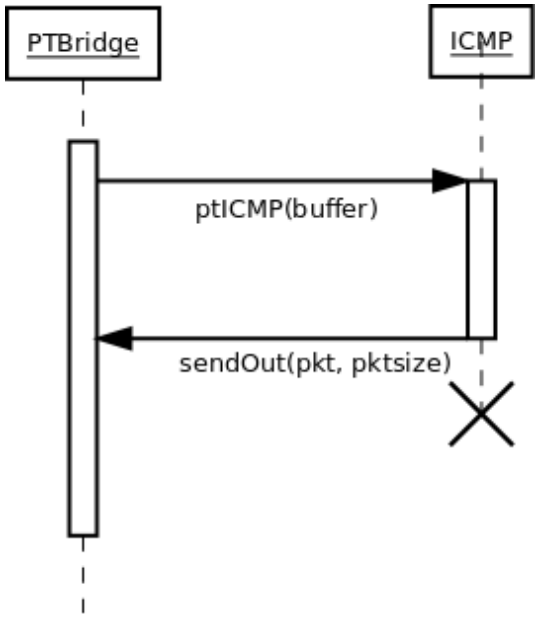
Pro analýzu práce výše popsaných protokolů a také pak následnou kontrolu práce překládací práce PTBridge se velmi osvědčil nástroj pro analýzu sítového provozu Wireshark [15] (dříve Ethereal). Umožňuje jak zachytávání tak i analýzu rámců a také zobrazuje obsah rámce v šestnáctkové soustavě. Toto je velmi užitečné při zkoumání obsahu zpráv protokolů PTMP a MU.

Také PTBridge používá systém logování a výpisů, který jsem již popsal v podkapitole zabývající se implementací PTServeru.

7.4 Popis tříd PTBridge

Tato část obsahuje stejnou třídu pro logování a výpisy jako PTServer. Dále je zde i třída **PTBuffer** popisovaná v části PTSertveru, je zde poněkud zjednodušena.

PTBridge – primární třída, která naslouchá na nastavených portech a zajišťuje komunikaci jak s PTServerm tak s tunelovacím serverem. V samostatném vlákně případně



Obrázek 9: Jednoduchý sekvenční diagram PTBridge

spouští případné zachytávání a odesílání provozu pomocí knihovny pcap. Předává rovněž přijatý provoz patřičným metodám tříd, které zpracovávají dané rámce.

Je zde pak ještě složka PDU s třídami, které obsahují metody pro převod konkrétního rámce.

Na sekvenčním diagramu 9 je postup, který se použije při příjmu pseudorámce ICMP. Ten je po přepracování odeslán opět za pomocí metody PTBridge. Záleží na způsobu spuštění, kam bude rámec doručen. Všechny ostatní překlady probíhají podobným způsobem.

7.4.1 Konfigurace PTBridge

I pro komponentu PTBridge je použito konfiguračního souboru, který obsahuje všechny nezbytné parametry. Formát souboru musí splňovat stejné předpoklady jako v případě PTServeru. Ukázkový výpis této konfigurace je vidět na výpise 3.

```
# Port, na kterem se naslouchа pro prenosy z PTServeru
PTBridge-port 38100
# IP adresa nebo jmeno hostitele pro naslouchani
PTBridge-ip localhost

##############################################################################
# IP adresa nebo jmeno hostitele PTServeru (pro komunikaci s PTBridgem)
PTServer-ip 127.0.0.1
# port PTServeru
PTServer-port 38101

##############################################################################
```

```
# IP adresa nebo jmno hostitele s tunelovacim serverem
Tunserver-ip 127.0.0.1
# port tunelovacho serveru
Tunserver-port 40002

#############################################################################
# Port pro naslouchani prenosu z Tunelovaciho serveru
PTBtunnel-port 40101
# IP adresa nebo jmeno hostitele pro naslouchani
PTBtunnel-ip localhost
```

Výpis 3: Ukázka konfiguračního souboru pro PTBridge

Rovněž zde je parametr *-h*, který na vypíše všechny parametry, se kterými lze PTBridge spustit. Není zde žádný povinný parametr.

7.5 Úpravy Virtlabu

Pro potřeby tohoto řešení bylo potřeba upravit i několik kódů. Těchto úprav však nebylo potřeba mnoho díky solidnímu stavu projektu Virtlab. Ten je dobře připraven na případné rozšiřování.

V databázi řídícího serveru jsem vytvořil tabulku pro uchování informací i PT linkách. V této tabulce jsou uchovávány informace o IP adrese a portu klienta. Tato informace slouží k unikátní identifikaci klienta. Dále jsou v této tabulce uloženy informace o UUID linky, jméno konce linky v PT (obvykle jako *jmeno_prvku jmeno_rozhrani*), stav vzdáleného portu (aktivní/neaktivní) a typ portu.

Do tabulky *reservations* jsem přidal sloupec *otp*. Tento slouží pro uložení generovaného hesla, které se použije při autentizaci PT. Toto pole je vyplňováno při aktivaci rezervace. Pro tyto účely jsem přidal do souboru *make-reservation.php* funkci 4, která toto heslo generuje. Funkci volám s parametrem *length* nastaveným na 8. Pro potřeby autentizace PT se mi jeví takto dlouhé heslo jako dostatečné.

```
function generatePassword($length=9)
{
    srand(makeSeed());

    $alfa = '1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';
    $token = '';

    for($i = 0; $i < $length; $i ++)
    {
        $token .= $alfa[rand(0, strlen($alfa))];
    }
    return $token;
}
```

Výpis 4: Funkce pro generování náhodného hesla

Toto heslo je však nutno uchovávat v tomto čistě textovém formátu a stejně jej i předávat pomocí SOAPu do aplikace PTServer. Zde je teprve s heslem pracováno dle vyjednané metody autentizace.

Proto komunikaci s řídícím serverem je jak jsem již psal použito gSOAPu. Aby však vše řádně fungovalo, vytvořil jsem pro tuto komunikaci soubory WSDL. Tyto pomocí jazyka XML popisují dostupné služby SOAP serveru a také formát komunikačních zpráv. Jelikož se chystá nová implementace řídícího serveru, tak jsem prozatím vytvořil vlastní PHP skript obsahující jen služby, které potřebuji. Tento skript používá formát zpráv definovaný souborem WSDL. Na straně řídícího serveru je použito SOAP prostředků, které jsou zabudovány v prostředí PHP [17].

Propojení mezi PTBridgem a tunelovacím serverem jsem původně chtěl udělat jako v případě připojení vzdáleného počítače do topologie Virtlabu [18]. Raději však než zasahovat do modulu internetového portu tunelovacího serveru, jsem vytvořil vlastní modul pro komunikaci s PTBridgem.

Tento modul je postaven na již hotových modulech. Naslouchá na svém vlastním UDP portu a předává rámce obdržení z tohoto portu dále do přepínacího jádra tunelovacího serveru. Rovněž posílá rámce, které obdrží od přepínacího jádra pomocí UDP soketu na další zpracování do PTBridge. Tento modul dědí třídu **Port_internal**. Načítá rovněž konfiguraci, která obsahuje IP adresu a port, na kterém má naslouchat a také informace kde naslouchá PTBridge.

Aby si uživatel mohl propojit konektor s pseudokonektorem (PT linkou) je potřeba poupravit funkci řídícího serveru, která uživateli vypisuje použitelné konektory. Je potřeba také přidat do kódu, který zajišťuje propojení konektorů aby odeslal SOAP zprávu PT-Serveru. Tato zpráva obsahuje právě hlavičku linky a dle ní se zašle patřičnému klientovi informace o aktivaci linky.

8 Závěr

Začátky vývoje tohoto řešení byly pro mě dosti obtížné, jelikož jsem neměl mnoho informací o protokolech použitých při spojení dvou PT. Navíc první verze, ve které jsem používal API závislé na Qt, jsem řešil mnoho problémů právě s Qt framewokem. Neměl jsem s ním vůbec žádné zkušenosti a tak jsem měl první funkční výsledky této práce až po velmi dlouhé době. Z tohoto důvodu jsem se rozhodl pokusů s tímto řešením zanechat a přepracovat vše do kódu používající jen C++.

Bohužel jsem doteď neměl zkušenosti s vývojem takto velkého projektu ani v jazyce C++. Rozhodně mi tato práce přinesla detailní pochopení použitých protokolů, ať těch použitých v PT nebo těch, které jsou přenášeny v reálné síti. Také použití gSOAP toolkitu si vyžádalo dosti času, ale tyto znalosti jistě využiji i v budoucnosti.

Snažil jsem se však celý systém navrhnout tak aby se dal snadno upravovat a rozšiřovat. Jen část pro převod rámců v PTBridge je zatím provedena pomocí metod, která provádí celý tento překlad. Nechtěl jsem zatím provádět návrh jiného systému, jelikož jsem stále čekal na kód od vývojářů PT. Ten by mohl pomoci vhodnému návrhu objektové struktury pro převod rámců.

Dále mě při dokončování tohoto textu napadlo jak vhodně použít PTMP zprávy Port Info pro lepší uživatelskou přívětivost celého řešení. Bohužel jsem je však zatím nestačil přidat do kódu aplikace. Toto rozšíření bude součástí dalšího dopracování programů, které mám v plánu nad rámec této diplomové práce.

Rovněž bych rád přidal překladové funkce do programu PtBridge 3.1 a rozšířil tím jeho možnosti.

Pevně doufám, že se tento projekt bude dále rozvíjet. Myslím, že se jedná o zajímavou problematiku, která může napomoci právě praktické výuce počítačových sítí. Zajímavým rozšířením by mohlo být třeba právě vyvinutí serverového řešení, které by umožnilo komunikaci mezi mnoha simulátory Packet Tracer.

9 Reference

[1] BURDA, Pavel. *Implementace konektorů pro dynamické propojování distribuovaných virtuálních topologií v systému Virtlab*. Ostrava, 2010. Diplomová práce. VŠB-TU Ostrava.

[2] WANG, Michael. *Packet Tracer Messaging Protocol (PTMP) : Specifications Document*. Cisco, 2008. 13 s.

[3] GAILLY , Jean-loup; ADLER, Mark. *Zlib* [online]. 2010 [cit. 2010-04-23]. Domovská stránka. Dostupné z WWW: http://www.zlib.net/.

[4] *Virtuální laboratoř počítačových sítí* [online]. 2010 [cit. 2010-04-23]. Dostupné z WWW: http://www.virtlab.cz/.

[5] *Cisco Systems* [online]. c2010 [cit. 2010-04-23]. Cisco Packet Tracer. Dostupné z WWW: http://www.cisco.com/web/learning/netacad/course_catalog/PacketTracer.html.

[6] *gSOAP : SOAP C++ Web Services* [online]. 2010 [cit. 2010-04-24]. Dostupné z WWW: http://gsoap2.sourceforge.net/.

[7] BORTLÍK, Václav. *Distribuované spojovací pole pro distribuovanou laboratoř počítačových sítí*. Ostrava, 2008. 34 s. Bakalářská práce. VŠB-TU Ostrava. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/images/9/92/Bortlik-diplomka.pdf.

[8] OLIVKA, Petr. *Operační systémy* [online]. c2005 [cit. 2010-04-23]. Dostupné z WWW: http://poli.cs.vsb.cz/edu/osy/.

[9] DOSTÁL, Radim. *Objektově orientované programování v C++* [online]. 06.12. 2000 [cit. 2010-04-23]. Dostupné z WWW: ¡http://www.builder.cz/art/cpp/cpp_oop.html.

[10] *Packet Tracer Collaboration Portal* [online]. c2007 [cit. 2010-04-23]. Dostupné z WWW: http://pt.netacad.net/.

[11] JANITOR, Jozef. *Jozjan's blog* [online]. 2008-02-28 [cit. 2010-04-23]. Packet Tracer 5.0 – New Features. Dostupné z WWW: http://blog.jozjan.net/2008/02/packet-tracer-50-new-features.html.

[12] BLAISE, Barney. *Lawrence Livermore National Laboratory* [online]. 01/14/2010 [cit. 2010-04-23]. POSIX Threads Programming. Dostupné z WWW: https://computing.llnl.gov/tutorials/pthreads/.

[13] *The C++ Resources Network* [online]. c2009 [cit. 2010-04-23]. Dostupné z WWW: http://www.cplusplus.com/.

[14] *TCPDUMP/LIBPCAP public repository* [online]. 2010-04-06 [cit. 2010-04-23]. Dostupné z WWW: http://www.tcpdump.org/.

[15] *Wireshark Wiki* [online]. 2010-04-11 [cit. 2010-04-23]. Dostupné z WWW: http://wiki.wireshark.org/.

[16] *Cisco Documentation* [online]. 2002-10-02 [cit. 2010-04-23]. Frame Formats. Dostupné z WWW: http://www.cisco.com/univercd/cc/td/doc/product/lan/trsrb/frames.htm.

[17] *PHP : SOAP* [online]. c2010, Last updated: Fri, 23 Apr 2010 [cit. 2010-04-27]. Dostupné z WWW: http://www.php.net/manual/en/book.soap.php.

[18] NOVÁK, Radek. *Připojení uživatelského PC do Virtuální topologie*. Ostrava, 2009. 4 s. Semestrální práce do předmětu Technologie počítačových sítí. VŠB-TU Ostrava.

A Výstup PtBridge

```
[-TCP-]
<-- Pripojovani k PT
Attempting connection to 127.0.0.1:38000
Connection established to 127.0.0.1:38000

[-PTMP-]
<-- Odeslani vyzvy k vyjednani spojeni
Negotiating: [NegotiationMessage:
          signature: PTMP
            version: 1
                uid: {453ce0fa-1454-f339-c6a2-80339ac4a472}
           encoding: binary
         encryption: none
        compression: none
    authentication: md5
          timestamp: 20081201165654
          keepalive: 0
           reserved:
]
Writing 77 bytes in text

--> Prijeti odpovedi pro vyjednani spojeni
Data Received: 77 bytes received.
Expecting 74 bytes and buffer is 74
Negotiated: [NegotiationMessage:
          signature: PTMP
            version: 1
                uid: {49e2a792-4439-41ba-b866-8f59f93c58c6}
           encoding: binary
         encryption: none
        compression: none
    authentication: md5
          timestamp: 20081201165654
          keepalive: 0
           reserved:
]
<-- Zadost o autentizaci
Authenticating: [AuthReqMsg:
    appID: Remote Network
]
Writing 23 bytes in binary

--> Autentizacni vyzva
Data Received: 41 bytes received.
Expecting 37 bytes and buffer is 37
Challenged: [AuthChalMessage:
    challenge: L6Sh488Bla584h99bH5p0hW8R14l4LA1
]

<-- vytvoreni a odeslani odpovedi
Hashing...
```

```
Hashed into 16 bytes
Responding: [AuthRespMessage:
        appID: Remote Network
    response: BF8AFF55F48BA21CB3B38CA99DD4C893
     message:
]
Writing 57 bytes in binary

--> Obdrzeni zpravy o uspesnosti autentizace
Data Received: 9 bytes received.
Expecting 5 bytes and buffer is 5
Authenticated: [AuthStatusMessage:
    status: true
]
Authentication Successful!

[-MU-]

<-- Odeslani MU zadosti o spojeni
Negotiating:
Writing 30 bytes in binary

--> Odpoved na MU vyjednani
Data Received: 65 bytes received.
Expecting 26 bytes and buffer is 61
[MUNegotiationMessage:
    username: Guest
        uuid: {4d8ff124-1116-4159-bbf0-46d440abe938}
]
Negotiated connection to remote user: Guest

<-- Odeslani zpravy o zmene jmena
Updating network name: [MUNameUpdMessage:
    name: Local Network
]
Writing 22 bytes in binary

--> Prijata zprava s informacemi o portech (prazdna)
Expecting 8 bytes and buffer is 31
Received: [MUPortAdvMessage:
    length: 0
]

--> Prijata zprava s novym jmenem vzdaleneho konce
Expecting 19 bytes and buffer is 19
[MUNameUpdMessage:
    name: Remote Network
]
The remote network is now known as: Remote Network

--> Prijata zprava o nove lince
Data Received: 176 bytes received.
Expecting 84 bytes and buffer is 172
Expecting 84 bytes and buffer is 84
```

```
Data Received: 88 bytes received.
Expecting 84 bytes and buffer is 84

<-- Vytvoreni nove linky (ethernet)
[MULinkUpdMessage:
                    linkOpId: 0
                      opType: 1
                      linkId : 0
                    linkUuid: {d2c811b0-4ba8-4fee-9ecb-67420b94a4f2}
                      portId: -1
                   cableType: 0
                    portName: FlashPort
                    portType: 3
                   portPower: true
                straightPins : false
                   autoCross: false
                   bandwidth: 10000
                  fullDuplex : true
              autoNegotiate: true
   bandwidthAutoNegotiate: true
      duplexAutoNegotiate: true
                   clockRate: 0
                     dcePort: false
]
Writing 74 bytes in binary
Data Received: 88 bytes received.
Expecting 84 bytes and buffer is 84

--> Prijaty pseudoramce, ktery obsahuje zpravu protokolu CDP
Data Received: 511 bytes received.
Expecting 507 bytes and buffer is 507
Received: [MUPDUMessage:
    msgId: 17040
   linkId : 0
      pdu: CIEEE802Dot3Header0CSnapLLCHeader0CCdpFrame
           02180000006CCdpDeviceId00100Router0CCdpAddress0020000000000CCdpPortId
           00300FastEthernet0/00CCdpCapability004001CCdpSoftwareVersion00500Cisco
           Internetwork Operating System Software10IOS (tm) C2600 Software
           (C2600-I-M), Version 12.2(28), RELEASE SOFTWARE (fc5)10Technical
           Support: http :// www.cisco.com/techsupport10Copyright (c) 1986-2005
           by cisco Systems, Inc.10Compiled Wed 27-Apr-04 19:01 by miwang
           0CCdpPlatform00600C26000170170303000080001200 00208151197177110
           12204204204000008
]
```

Výpis 5: Ukázka jednotlivých kroků PT spojení