VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace pokročilého generátoru síťového provozu

Implementation of Advanced Network Traffic Generator

2014

Ondřej Jaroš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Jaroš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace pokročilého generátoru síťového provozu
Implementation of Advanced Network Traffic Generator

Zásady pro vypracování:

Cílem práce je implementace genrátoru síťového provozu s využitím moderních metod simulování reálného síťového provozu. Vytvořené řešení musí splňovat požadavky na modularitu a možnosti vytváření profilů pro různé typy generovaného provozu. V řešení již bude zahrnuta podpora pro generování provozu protokolů rodiny TPC/IP. Předpokládá se použití OS GNU/Linux.

1. Seznamte se s existujícími řešeními pro generování síťového provozu.
2. Navrhněte řešení s ohledem na modularitu a rychlost. Navržené řešení implementujte. Součástí implementace bude grafické uživatelské rozhraní.
3. Výsledné řešení otestujte a srovnejte s vybranými existujícími generátory. Vypracujte řádnou dokumentaci hotového díla.

Seznam doporučené odborné literatury:

[1] KABELOVÁ, Alena a Libor DOSTÁLEK. Velký průvodce protokoly TCP/IP a systémem DNS. 5., aktualiz. vyd. Brno: Computer Press, 2008, 488 s. ISBN 978-80-251-2236-5.
[2] KERNIGHAN, Brian W a Dennis M RITCHIE. The C programming language. 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, c1988, xii, 272 s. ISBN 01-311-0362-8.
[3] PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z. 2. aktualiz. vyd. Brno: Computer Press, 2006, 430 s. ISBN 80-251-1278-0.
[4] VIRIUS, Miroslav. Jazyky C a C: Komplexní průvodce, aktualizované vydání. Praha: Grada, 2011, 367 s. Knihovna programátora. ISBN 978-80-247-3917-5.

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

Vedoucí bakalářské práce: **Ing. Daniel Stříbný**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

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

V Ostravě 7. května 2014

Rád bych zde poděkoval vedoucímu mé práce Ing. Danielovi Stříbnému za jeho rady a odbornou pomoc. Dále především své přítelkyni za její podporu během mého studia a také kolegům za jejich schovívavost a rady.

Abstrakt

Cílem této bakalářské práce je navržení a následná implementace nového řešení generátoru síťového provozu. Program překonává nedostatky stávajících řešení, zjištěné analýzou volně dostupných generátorů. Součástí práce je popis teoretických východisek a technických problémů, ze kterých návrh vychází. Taktéž je zde v rámci popisu implementace programu zahrnut výběr vhodného programovacího jazyka, výčet použitých algoritmů, knihoven a nástrojů.

Klíčová slova: generátor, počítačová síť, síťový provoz, testování sítě

Abstract

Goal of this bachelor's thesis is to design and implement new network traffic generator solution. The program overcomes shortcomings of existing solutions, that were discovered during analysis of freely available generators. This work describes the theoretical basis and technical issues, which were considered before the new design proposal was made. Also, in the scope of the implementation description, the choice of appropriate programming language is described there, as well as the list of used algorithms, libraries and tools.

Keywords: generator, computer network, network traffic, network testing

Seznam použitých zkratek a symbolů

ACL	–	Access Control List - seznam řízení přístupu pro zvýšení bezpečnosti částí sítě
BPDU	–	Bridge Protocol Data Unit - rámce odesílané přepínači s nakonfigurovaným protokolem STP
DSCP	–	Differentiated Services Code Point - pole v IP hlavičce pro určení priority v architektuře DiffServ
GUI	–	Graphical User Interface - grafické rozhraní pro ovládání programu
LAN	–	Local Area Network - označení síťe na geograficky limitovaném území
MAC	–	Media Access Control adresa - unikátní identifikátor síťového rozhraní
OSI	–	Open Systems Interconnections model charakterizující komunikaci na poč. síti
PHB	–	Per Hop Behavior - označuje architekturu QoS, kdy probíhá prioritizace na každém aktivním zařízení
STP	–	Spanning Tree Protocol - protokol zajišťující bezsmyčkovou topologii na druhé vrstvě
TCP	–	Transmission Control Protocol - spojově orientovaný transportní protokol
UDP	–	User Datagram Protocol - nespojový, rychlý protokol transportní vrstvy
UTP	–	Unshielded Twisted Pair - kroucená dvojlinka, nejběžnější typ kabeláže na místní síti
WAN	–	Wide Area Network - označení pro rozsáhlou celosvětovou síť

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tématem této práce je návrh a implementace vlastního řešení modulárního generátoru síťového provozu. Ten je nezbytným nástrojem pro každého, kdo se zabývá návrhem a realizací počítačových sítí, neboť umožňuje efektivně napodobit běžná data odesílaná uživateli. Existují situace a případy, ve kterých je opravdový provoz na síti nezbytnou podmínkou pro její otestování, popř. pro analýzu funkčnosti implementovaných protokolů a nastavení. Generátory provozu nám tuto situaci ulehčují, jelikož umožňují simulovat provoz z jediné pracovní stanice bez nutnosti fyzického připojení několika (desítek, stovek) počítačů a uživatelů. Celá problematika počítačových sítí a datového provozu je podrobněji rozebrána ve 2. kapitole.

Existuje několik volně dostupných alternativ, pokud potřebujeme výše zmíněnou simulaci iniciovat. Analýza těchto programů, jejich možností a nedostatků je uvedena v kapitole 3. Většina z nich ale lpí na striktně deterministických přístupech a je proto velmi obtížné nastavit je tak, aby byla výsledná simulace podobná reálnému provozu. Jejich ovládání je však až příliš složité a komplikuje tak běžnému uživateli jejich použití.

Cílem této práce je vytvoření modulárního programu, který bude dle uživatelem definovaných parametrů vytvářet a odesílat síťový provoz z koncové stanice (počítače). Na základě zvolených profilů bude program umožňovat simulaci komplexního multiprotokolového datového proudu. Vzhledem k omezením a bezpečnostním opatřením, se kterými se můžeme setkat v místě připojení koncové stanice, nebude tento program umožňovat generování a odesílání datových jednotek, které by mohly narušit chod sítě.

Návrh tohoto řešení popisuji v kapitole 4. Vlastní implementaci programu, zvolený programovací jazyk a vývojové prostředí, zakomponované algoritmy a popis uživatelského rozhraní následně najdete v kapitole 5.

2 Teoretická východiska práce

V této kapitole čtenáře seznámím s teoretickými východisky a technickými specifikacemi, o které se budou opírat následující části. Definuji, co počítačová síť je, jak na ní probíhá výměna informací a která zařízení se tohoto procesu účastní.

2.1 Počítačová síť

Za počítačovou síť můžeme označit skupinu počítačů a jiných zařízení (tiskáren, telefonů, serverů, atp.) navzájem propojených bezdrátově nebo pomocí metalických či optických vodičů (kabelů). Vzájemné propojení v síti přináší uživatelům řadu výhod. Mezi ně patří především rychlé a jednoduché sdílení informací, souborů a prostředků (např. tiskáren či datových úložišť). Dále samozřejmě stojí za zmínku bezpečnost, společný přístup k firemním programům a zjednodušená správa.

Stejně jako se za účelem využívání dostupných výhod propojují jednotlivé počítače a síťová zařízení, propojují se také samotné sítě mezi sebou. Vzhledem k tomu, že při dodržení určitých pravidel můžeme takto spojovat sítě prakticky neomezeně, je potřeba zmínit nejběžnější dělení počítačových sítí, a to podle velikosti:

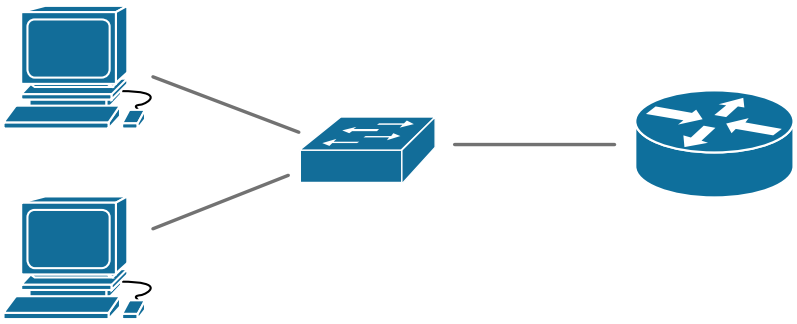
Vzájemné propojení zařízení na geograficky omezené oblasti se nazývá **LAN** (z anglického Local Area Network = *místní síť*). Jako LAN označujeme osobní, školní či firemní sítě, rozprostřené v rámci jedné nebo několika budov. Počet připojených zařízení se pohybuje v nejmenších sítích od jednotek (dva navzájem propojené počítače už můžeme považovat za LAN síť), až po několik stovek v případě místní sítě napříč kancelářskou budovou. Rychlost spojení mezi uživateli místní sítě je velmi vysoká (řádově desítky až stovky Mb/s), kterou nám poskytuje v drtivé většině případů Ethernetová[1] kabeláž.

Opakem místní LAN sítě jsou dálkové kabelové spoje nebo velké skupiny spojených sítí, které nám umožňují přenášet data na velké vzdálenosti. Ať už se jedná o sdílení informací mezi kancelářemi v sousedících městech nebo datový provoz na druhou polovinu planety, označujeme tyto sítě jako **WAN** (z anglického Wide Area Network = *rozsáhlá, celosvětová síť*). Nejznámějším příkladem WAN sítě je Internet.

Nehledě na to, zda se chystáme propojovat jen několik uživatelů v jedné kanceláři nebo několik poboček napříč Evropou, neobejdeme se bez síťových zařízení. Pro účely této práce není zapotřebí úplný a ucelený seznam, proto zde nejsou zmíněna bezdrátová zařízení, pasivní prvky, přístroje na modulaci signálu a ani historické (v moderních sítích nepoužívané) komponenty. Prvním síťovým zařízením je **přepínač**, který umožňuje rychlé a spolehlivé spojení mezi koncovými stanicemi, telefony či tiskárnami v rámci místní sítě. Přepínač je základním stavebním blokem, bez kterého by se žádná síť neobešla.

Pokud ale nechceme budovat infrastrukturu místní sítě, nýbrž ji chceme propojit se sítí jinou (nebo např. k Internetu), budeme potřebovat **směrovač**. Ten, narozdíl od přepínače, umožňuje sítě dělit, spojovat a zajišťovat jejich vzájemnou konektivitu. Také přináší široké možnosti, jak můžeme ovlivňovat procházející provoz.

[1] Ethernet - soubor technologií pro místní sítě, používá kroucenou dvojlinku (UTP) nebo optický kabel, realizuje 1. a 2. vrstvu OSI modelu



Obrázek 1: Přepínač jako stavební blok lokální sítě

2.2 OSI model a zapouzdření

Přenos dat mezi dvěma uživateli (zařízeními) není triviální proces. Na cestě od jednoho uživatele k druhému se do něj zapojují nejen samotné počítače a výše zmíněná aktivní zařízení, ale také množství protokolů a pravidel. Pro snadnější pochopení a oddělení jeho logických částí byl v rámci organizace ISO (International Organization for Standardization) vytvořen Open Systems Interconnection model (dále jen OSI model). OSI model popisuje komunikaci na počítačové síti pomocí abstrakce do sedmi vrstev, přičemž každá vrstva poskytuje své služby vrstvě nadřazené, využívá možnosti podřazené vrstvy a je zodpovědná za určitou část procesu při přenosu dat. Na cestě datové jednotky směrem od uživatele k přenosovému médiu jednotlivé vrstvy připojují k obdržené jednotce svá data, svou hlavičku. Tomuto procesu se říká zapouzdřování (angl. encapsulation).

Pokud tedy přenášíme data mezi dvěmi stanicemi, jsou aplikací předána nejvyšší (sedmé) vrstvě a poté postupně na všechny podřízené vrstvy. Na první - fyzické vrstvě - probíhá samotný přenos ve formě elektrického signálu (na kabelovém vedení) nebo např. světelných impulsů (na optickém vedení).

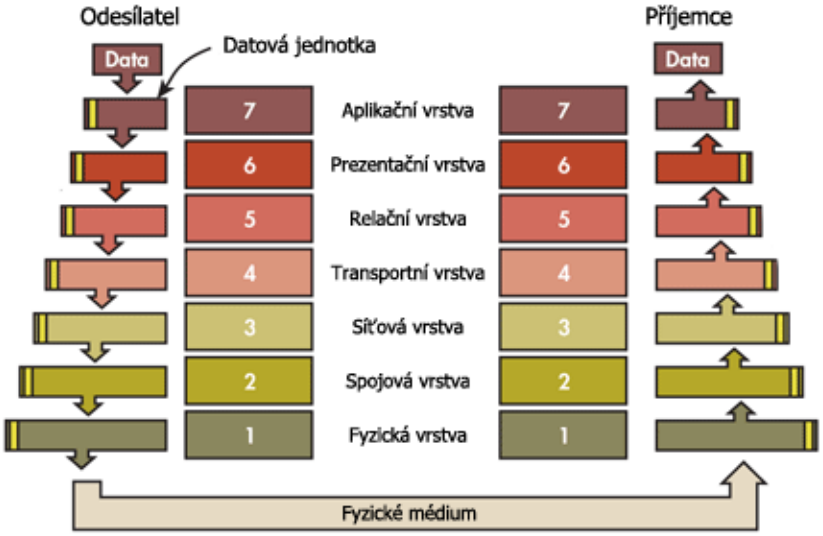
Toto abstraktní rozdělení je klíčové pro pochopení konceptu datového přenosu na počítačové síti a jeho následné simulaci pomocí generátoru.

2.2.1 První vrstva - fyzická

Fyzická vrstva je, jak již bylo uvedeno výše, zodpovědná za přenos signálu daným prostředím (metalickým kabelem, optickým kabelem, popř. vzduchem). Z pohledu datového přenosu je na první vrstvě důležité zvolit, které síťové rozhraní bude použito pro odeslání datových jednotek. Jako příklad lze uvést přenosný počítač, kde můžeme pro připojení do místní sítě využít jak kabelové, tak bezdrátové síťové rozhraní.

2.2.2 Druhá vrstva - spojová

Druhá vrstva zodpovídá za spolehlivý přenos datových jednotek mezi dvěmi sousedícími zařízeními (tj. od stanice ke stanici nebo k směrovači v rámci dané sítě). O tuto komunikaci uvnitř lokální sítě se starají převážně přepínače, které pracují právě na druhé vrstvě OSI



Obrázek 2: Vrstvy referenčního modelu OSI

modelu. Komunikace na druhé vrstvě v jedné síti je velmi rychlá, neboť není potřeba žádné směrování - datové jednotky jsou jen přepínány mezi jednotlivými stanicemi. K jejich identifikaci slouží MAC adresa[2].

2.2.3 Třetí vrstva - síťová

Na síťové vrstvě zařízení pracují s IP adresami[3]. Ke každé datové jednotce je přiřazena IP hlavička obsahující mimo jiné následující informace:

- Zdrojová IP adresa (odesílatel)
- Cílová IP adresa (příjemce)
- Verze (IPv4 nebo IPv6)
- DSCP hodnota - více v kapitole 2.4.3 a v příslušném RFC[8]
- Délka paketu, kontrolní součet a další...

Podle informací na třetí vrstvě tedy lze určit, jestli je příjemce tohoto paketu v lokální síti nebo je třeba tuto datovou jednotku směrovat do jiné sítě. Z toho plyne již očividné, že směrovače pracují s třetí vrstvou OSI modelu.

[2]MAC adresa je neměnná fyzická adresa daného síťového rozhraní, přiřazená již při výrobě.
[3]IP adresa je unikátní logický identifikátor zařízení v síti. Je doprovázena maskou sítě, která určuje velikost přiřazené sítě.

2.2.4 Čtvrtá vrstva - transportní

Předešlé vrstvy definovaly logickou adresaci zařízení, přenos v lokální síti a přenos signálu samotného. Tato vrstva se zaměřuje více na způsob a kvalitu datového přenosu, než na jeho provedení. Máme možnost využít jeden ze dvou transportních protokolů:

- **TCP** - spojově orientovaný, spolehlivý, ale v praxi pomalejší
- **UDP** - nespojovaný, nespolehlivý, ale velmi rychlý

Převážně je dnes pro veškerý provoz využíván protokol TCP. UDP naopak slouží v případech, kdy je rychlost transportu dat důležitější, než jeho kvalita. Najde tedy uplatnění hlavně při přenosu videa a zvuku, kde je nutné data odesílat ihned v daném pořadí, v reálném čase.

Oba tyto protokoly umožňují definovat čísla portů (zdrojový port a cílový port), které oddělují provoz více aplikací mezi stejnými stanicemi a zároveň poskytují jedinečnou možnost identifikovat, která aplikace či protokol je zodpovědný za daný přenos.

2.2.5 Další vrstvy

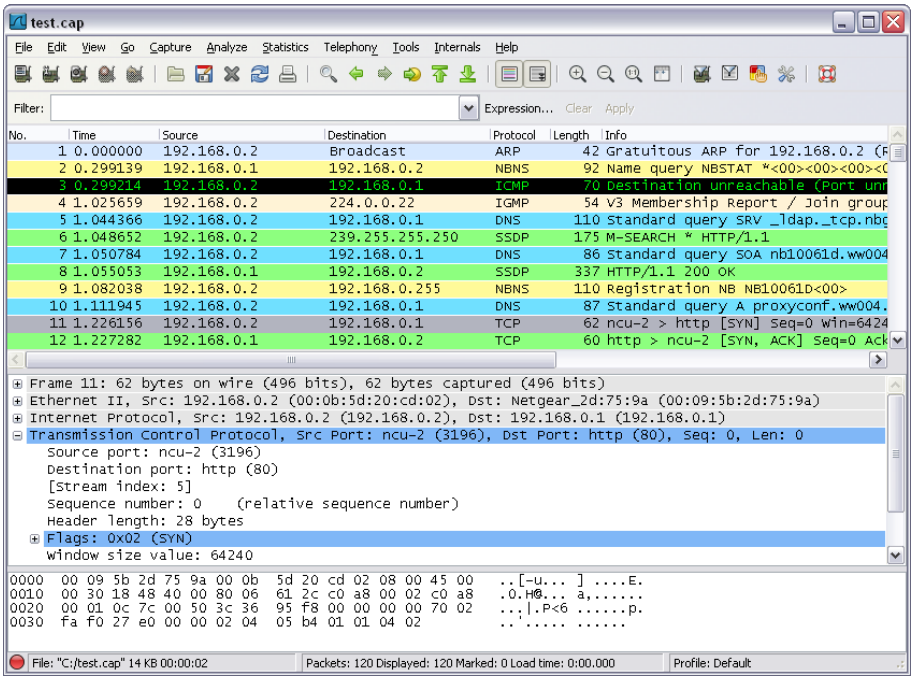
Relační, **prezentační** a **aplikační** vrstva tvoří zbytek modelu. Z pohledu sledování a generování síťového provozu již však nejsou důležité, neboť veškeré síťové zařízení, manipulující s datovými jednotkami, identifikují provoz na základě prvních čtyř vrstev. Obsah následujících třech závisí na dané aplikaci, použitém šifrování apod.

2.3 Datový provoz a jeho záznam

Chceme-li zachytávat, analyzovat či generovat datový provoz, musíme vzít v úvahu, jak již bylo uvedeno, informace z prvních čtyř vrstev OSI modelu. Na vyšších vrstvách už jsou samotná data, která z pohledu počítačové sítě nehrají roli, jelikož se nijak nepodílí na přepínání, směrování nebo identifikaci dané datové jednotky.

Datový přenos můžeme dělit na **uživatelský** a **kontrolní**. První z nich je právě ten, za který jsou zodpovědné jednotlivé aplikace a uživatelé přistupující k dostupným síťovým prostředkům, například při prohlížení webových stránek, stahování souborů, přístup na sdílená úložiště apod.

Druhý uvedený - kontrolní provoz - zastřešuje datové jednotky zasílané automaticky síťovými zařízeními a na nich běžícími protokoly. Tyto datové jednotky jsou transparentní z pohledu uživatele, avšak nezbytné za účelem správného fungování počítačové sítě. Můžeme zde zařadit výměnu informací mezi směrovacími protokoly (tzv. Hello pakety), kontrolní pakety udržující bezsmyčkovou topologii na přepínané síti (tzv. BPDU pakety), popř. pakety udržující stav dané linky či spojení (tzv. Keepalive pakety). Kontrolní provoz je na síti nezbytný, nicméně tyto pakety jsou v porovnání s uživatelskými daty velmi malé, takže nemají přílišný vliv na celkové zatížení sítě. V souvislosti s kontrolním provozem je třeba vzít v úvahu i další aspekt - bezpečnost. Kupříkladu generování BPDU paketů z koncové stanice by při nedostatečném zabezpečení mohlo ovlivnit připojený přepínač



Obrázek 3: Grafické uživatelské rozhraní programu Wireshark

a v důsledku mít fatální důsledek pro celý segment sítě. Z těchto zmíněných důvodů vyplývá, že pro úspěšné testování počítačové sítě není nezbytně nutné generovat kontrolní provoz, nýbrž je důležité zaměřit se na provoz uživatelský, tj. datové jednotky generované spuštěnou aplikací nebo uživatelem samotným.

2.3.1 Záznam provozu

Můžeme nalézt více důvodů, proč je vhodné a někdy i nezbytné zaznamenávat datový provoz na počítačové síti. Zaznamenaná data mohou poskytnout cenné informace a statistiky o využití dostupné konektivity, aktivitách uživatelů, o jednotlivých aplikacích a jejich nárocích na připojení. K záznamu datového provozu slouží několik nástrojů s rozdílným portfoliem poskytovaných funkcí. Pro účely mé práce jsem vybral Wireshark[4], v současnosti nejpoužívanější nástroj pro záznam a analýzu datového provozu na počítačové stanici.

Wireshark je spustitelný na pracovní stanici a je schopen zachytávat veškerý síťový provoz na daném rozhraní. Pro větší přehlednost lze zachycená data filtrovat dle protokolů, adres či čísel portů. Jednotlivé pakety lze pak samostatně prohlížet a analyzovat po jednotlivých vrstvách. Zaznamenaný provoz lze uložit pro pozdější potřeby, replikaci či analýzu do souboru s příponou CAP, jehož formát je specifikován na odkazu [10].

[4] dostupný pod GNU GPL licencí na www.wireshark.org

2.4 Návrh sítě a její testování

Síťová topologie[5] nebývá vždy triviální, neboť ne vždy se skládá jen z několika přepínačů a směrovače. Počet navzájem propojených zařízení může jít až do desítek či stovek. Návrh takové sítě je komplikovaný, neboť na jejím správném zapojení a nastavení závisí výsledný výkon a kapacita. Přímo úměrně se složitostí navržené topologie se zvyšuje i složitost testování a ověřování. Existují tři hlavní oblasti, které lze jen těžko sofistikovaně otestovat bez uživatelských dat kolujících po síti - a jelikož v době návrhu a implementace sítě žádní uživatelé na síti nejsou, bude generování provozu klíčové právě zde.

2.4.1 Vysoká dostupnost a rozdělení zátěže

V moderních firemních sítích se po zvážení poměru ceny a přidané hodnoty často implementují techniky a postupy k zajištění vysoké dostupnosti. Pod tímto termínem si můžeme představit soubor doporučení, nástrojů a protokolů, při jejichž dodržení minimalizujeme dopad na funkčnost firmy a produktivitu jejích pracovníků v případě výpadku. V rámci těchto doporučení často zahrnujeme do návrhu sítě tzv. redundantní zařízení a linky. Toto technické vybavení a kabelové spoje nejsou nezbytně nutné pro funkci sítě, nýbrž nám poskytují záložní cesty v případě výpadku či poruše některého zařízení. V kombinaci se správně nastavenými a implementovanými protokoly taková topologie nabízí vyšší odolnost vůči chybám a nulový nebo minimální dopad na produktivitu firmy v případě selhání síťového zařízení.

Mezi protokoly, které se svou funkcí podílí na vytváření vysoce dostupné sítě řadíme:

- **STP protokol** (RSTP, PVRST+, MST) - zajišťuje bezsmyčkovou topologii na druhé vrstvě OSI modelu. Pokud máme více redundantních linek na 2. vrstvě, právě STP protokol blokuje či povoluje záložní linky v případě výpadku.
- **Směrovací protokoly** (EIGRP, OSPF) - vyměňují informace o dostupnosti jednotlivých sítí mezi směrovači. Pokud existují záložní linky na třetí vrstvě OSI modelu, spoléháme na směrovací protokoly při hledání alternativní cesty v případě havárie.
- **Sdílení výchozí brány** (HSRP, VRRP, GLBP) - výchozí brána (angl. Default gateway) je označení pro prvek, který spojuje lokální síť s ostatními. Adresa výchozí brány je důležitá pro všechny stanice, které chtějí přistupovat k prostředkům za hranicemi lokální sítě. Například pokud uživatel zadá adresu webové stránky, obdrží IP adresu serveru, na kterém je daná stránka dostupná. Tento server se pravděpodobně v místní síti nenachází, proto stanice požadavek pošle na svou výchozí bránu, která jí umožní vzdálený server kontaktovat a stránku zobrazit. Tyto tzv. „first-hop redundancy“ protokoly pomáhají překonat problém, kdy máme více směrovačů umožňujících přístup k jiným sítím (primární a záložní brána), ale stanice může mít výchozí bránu jenom jednu. Protokol umožní těmto dvěma branám sdílet virtuální IP adresu, kterou uživatelé využívají, aniž by v případě výpadku primárního směrovače cokoliv zaznamenali.

[5] Topologie sítě - zachycení reálné či logické struktury sítě a vzájemného zapojení aktivních zařízení.

Tyto tři kategorie protokolů se velkou měrou podílí na vysoké dostupnosti sítě a minimalizaci dopadu na koncového uživatele v případě výpadku. Existují sice možnosti, jak uměle vyvolat podmínky jako v situaci výpadku, ale zatížení sítě velmi ovlivňuje rychlost reakce těchto protokolů. Zde opět najde využití generátor síťového provozu, který linky zatíží, čímž zpomalí přenos kontrolního provozu protokolů a poskytne tak ideální příležitost na měření rychlosti reakce a obnovení plné konvergence sítě v reálném prostředí.

2.4.2 Zabezpečení a ochrana sítě

Bezpečnost počítačových sítí je široké téma, o kterém lze hovořit z několika pohledů. Z hlediska návrhu počítačové sítě využíváme zejména seznamy řízení přístupu (tzv. ACL, z anglického Access Control List). ACL seznamy umožňují správci počítačové sítě omezit, zakázat či filtrovat přístup k zařízením či prostředkům dané sítě na základě údajů ze třetí a čtvrté vrstvy OSI modelu. Seznam řízení přístupu je nakonfigurován jako list pravidel, které jsou pro každý paket sekvenčně procházeny a kontrolují, zda-li se dané pravidlo shoduje s informacemi obsaženými v paketu. Pokud je nalezena shoda paketu s pravidlem, je paket propuštěn nebo zahozen (dle akce asociované s pravidlem) a nepokračuje se dále. Pokud však shoda nalezena není a sekvence pravidel dojde k poslednímu, provede se implicitně nastavená akce pro všechny pakety, které se neshodly s žádným explicitním pravidlem dříve. Touto akcí je opět zahození nebo propuštění paketu. Na stejném principu fungují i jednoduché bezstavové firewally, které se integrují do počítačové sítě za účelem zvýšení bezpečnosti uživatelů.

Stejně jako u předchozího tématu, i zde lze částečně nasimulovat vstupní podmínky a otestovat funkčnost daného seznamu řízení přístupu či firewallu. Problém nastává při pravidlech využívajících informace ze čtvrté vrstvy OSI modelu, tj. pravidla rozdělující pakety podle použitého transportního protokolu či zdrojového, resp. cílového portu. V případě testování správnosti a funkčnosti těchto bezpečnostních řešení nám tedy nejde o množství (o zatížení linky), ale o různorodý uživatelský provoz, který lehce ověří funkci jednotlivých pravidel a seznamů. Při vytváření takovýchto datových jednotek může být opět nápomocen generátor síťového provozu, který umožňuje v krátkém čase generovat provoz simulující různorodé aplikace přistupující k prostředkům ve vzdálených sítích.

2.4.3 Quality of Service

Třetí a nejdůležitější oblastí, jejíž implementaci nelze bez pomoci generátoru síťového provozu otestovat, je nastavení pravidel pro řízení kvality služeb (z anglického Quality of Service, dále jen QoS).

QoS je souborem standardů a pravidel umožňující klasifikaci, řazení a prioritizaci jednotlivých datových proudů při síťovém spojení. Moderní počítačové sítě nabízejí několikrát větší kapacity a přenosové rychlosti, než tomu bylo v historii, tudíž je nyní možné a výhodné přenášet uživatelská data, video i zvuk zároveň po stejných datových trasách. Jak už ale bylo naznačeno při popisu protokolů čtvrté vrstvy OSI modelu, ne každý typ provozu na datové síti má stejné nároky. Zatímco datový přenos probíhá spíše dávkově

po částech, vyžaduje velkou šířku pásma a celková rychlost přenosu nehraje velkou roli, přenos zvuku je nenáročný na kapacitu linky, ale nesmírně citlivý na zpoždění. Přenos videa je z těchto třech nejnáročnější, neboť vyžaduje jak trvale dostupnou nezanedbatelnou kapacitu linky, tak i neměnnou rychlost spojení, aby se video přehrávalo příjemci plynule.

Tyto různorodé požadavky vyústily v potřebu datový provoz klasifikovat, rezervovat mu příslušnou šířku pásma a v neposlední řadě také prioritizovat odesílání důležitějších, časově citlivých paketů, před ostatními daty. V dnešní době je k dosažení tohoto cíle využívaná architektura **Differentiated Services** (DiffServ), která poskytuje rozšiřitelnou a relativně jednoduchou implementaci QoS pravidel napříč širokým spektrem síťových zařízení. DiffServ využívá pro klasifikaci jednotlivých paketů 6-ti bitové pole DSCP (Differentiated services code point), které obsahuje číselnou hodnotu zajišťující paketu přiřazení do odpovídající prioritní třídy a odpovídající zacházení. DSCP pole se nachází v IP hlavičce paketu (3. vrstva OSI modelu) a může nabývat hodnot 0–64, přičemž nejvyšší hodnoty (64–48) se v DiffServ nepoužívají. Zároveň také zachovává zpětnou kompatibilitu s dříve používaným systémem klasifikace - IP Precedence, který rozlišoval jen 8 prioritních tříd. Přehled všech prioritních tříd najdete v tabulce 1.

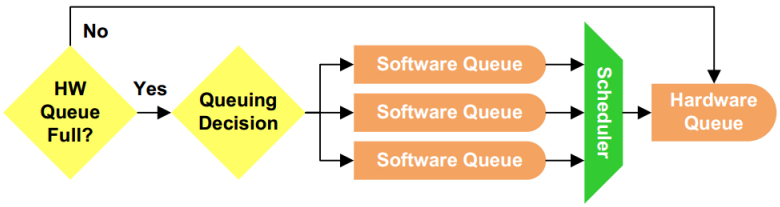
Název třídy	**Označení**	**DSCP**	**Použití**
Expedite Forwarding	EF	46	třída pro bezztrátový provoz s nejvyšší prioritou
Assured Forwarding	AF11-43	14-34	čtyři prioritní třídy zaručující přednostní zacházení za nastavených podmínek
Class Selectors	CS1-4	8,16,24,32	třídní selektory zajišťující zpětnou kompatibilitu s IP Precedence 1-4
Best Effort	BE	0	třída bez zvýšené priority pro běžný provoz

Tabulka 1: Prioritní třídy architektury DiffServ využívající DSCP

Ze všech zmíněných oblastí je právě nastavení QoS nejobtížněji otestovatelné, neboť pravidla pro klasifikaci síťového provozu se aktivují až ve chvíli, kdy je „fyzická“ odesílací fronta síťového zařízení plná a dochází ke stavu zahlcení.

V dané situaci je postup následující:

1. Kontrola, zda došlo k zahlcení odesílací fronty. Pokud ne, paket je odeslán normálně.
2. Pokud ano, je paket dle nastavené klasifikace zařazen do odpovídající prioritní fronty.
3. Odesílání z front probíhá dle priorit. Nejvyšší třída (EF) má vždy přednost.
4. Pokud neustále dochází k zahlcování, jsou zahazovány pakety z nejnižších tříd.



Obrázek 4: Proces prioritizace a řazení odeslaných datových jednotek při použití QoS (zdroj www.packetlife.net)

Celý postup je graficky znázorněn na obrázku č. 4. Architektura DiffServ je implementována jako PHB (z anglického Per-Hop Behavior)[9], což znamená, že rozdělení do tříd a jejich zpracovávání s danou prioritou je plně konfigurovatelné na každém aktivním zařízení, kterým paket prochází. Není tedy zaručeno, že budou síťové prostředky rezervovány předem skrze celou cestu sítí, jak tomu bylo u předchozí QoS architektury, ale na druhou stranu, DiffServ poskytuje mnohem větší flexibilitu a možnosti konfigurace.

Z uvedených informací vyplývá, že jedinou možností, jak otestovat funkčnost a správnost implementace pravidel QoS, je zahlcení linky. Dovolím si tvrdit, že bez fyzického připojení uživatelů a uvedení sítě do plného provozu nebo bez použití generátoru síťového provozu toto není možné. Jednoduchými diagnostickými nástroji a pomůckami nejsme totiž nikdy schopni vytvořit dostatečné množství dat pro zahlcení linky a aktivování prioritních tříd. Vzhledem k neustálému nárustu integrace přenosu zvuku a videa do datových sítí se stává implementace QoS nutností a stejnou měrou narůstá i potřeba jejího testování a ověřování, která se bez generátoru neobejde.

3 Analýza stávajících řešení

V této sekci popíši některé volně dostupné programy pro generování síťového provozu. Kritériem pro výběr byla podpora operačního systému Linux a ovládání pomocí grafického uživatelského rozhraní.

3.1 Ostinato

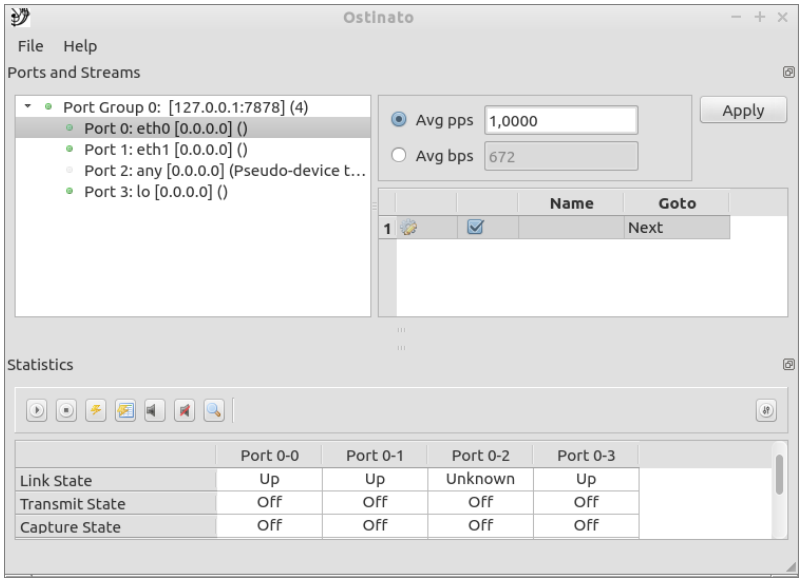
Ostinato je open-source[6], multiplatformní generátor síťového provozu.

Grafické uživatelské rozhraní je přehledné, ale také docela složité a nedá se zde hovořit o intuitivním ovládání. Program se klasifikuje jako „Wireshark naruby" s nabídkou načítání PCAP souborů a ovládání jednotlivých datových proudů. Ty nabízí velmi podrobnou nabídku a postupné plnění všech vrstev paketu, ale veškeré jejich nastavení je nutné a v mnoha případech pro obyčejného uživatele i zbytečné. Program sice nabízí načítání PCAP souborů (uložených záznamů z programu Wireshark), ale načte z nich všechny pakety jako jednotlivé proudy a nedokáže ani oddělit adresaci od obsahu. Veškeré následné úpravy jsou proto komplikované. Např. rychlost a počet generovaných paketů či adresu příjemce nelze ovlivnit globálně, ale musíme je upravit v každém jednotlivém proudu. Pokud tedy chceme replikovat uložený provoz bez úprav či generovat jen několik datových proudů, ať už jakkoliv specifických, může být Ostinato k tomuto účelu ideální.

Shrnutí:

- **Licence** - Open Source GNU GPL v3
- **Dostupnost** - volně ke stažení na *https://code.google.com/p/ostinato/*
- **Výhody**
  - široké možnosti nastavení jednotlivých proudů
  - podpora dlouhé řady protokolů na všech vrstvách
  - vhodné pro odesílání jakkoliv specifických paketů, nikoliv však náhodných
- **Nevýhody**
  - chybí možnosti globálně ovlivnit parametry odesílaných dat
  - při načítání PCAP souboru neexistuje možnost oddělit adresaci od obsahu
  - nastavit a odesílat provoz podobný reálnému je nesmírně komplikované
  - chybí intuitivní nastavení adresace, tj. načtení lokální IP adresy či náhodné generování adres příjemců

[6] Open-source je označení pro otevřený software, tj. s volně technicky i legálně dostupným zdrojovým kódem



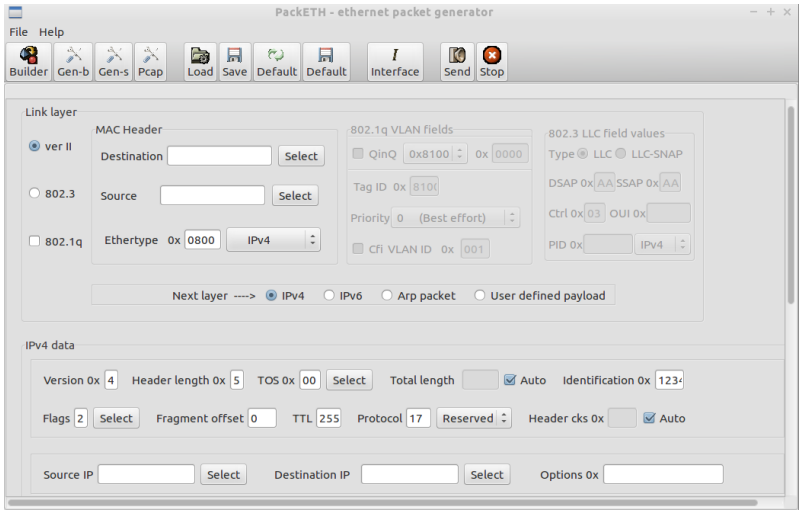
Obrázek 5: Grafické uživatelské rozhraní programu Ostinato

3.2 packETH

Další z testovaných programů je packETH, otevřený software, volně dostupný ke stažení a fungující pod operačním systémem Windows, MAC či Linux. Umožňuje generování Ethernetových paketů skrze grafické uživatelské rozhraní nebo přes příkazovou řádku. Druhá možnost ovšem otestována nebyla, protože kritériem pro zařazení a otestování programu bylo ovládání pomocí GUI.

Pokud jsem zmiňoval nedostatek intiutivního ovládání u předchozího programu, zde tento faktor chybí úplně. Nedokáži si představit uvedení tohoto programu do provozu bez přečtení uživatelské příručky. Ovládací lišta je nepřehledná, popisky tlačítek jsou většinou nejasné. První karta umožňuje definovat paket, vrstvu po vrstvě. PackETH se zaměřuje především na vytváření a odesílání Ethernetových paketů, ale to nelze vnímat jako nevýhodu, vzhledem k rozšířenosti Ethernetových sítí. V kartě „Builder“ tedy můžeme vybrat či nastavit jednotlivé parametry, začínající od MAC adres na druhé vrstvě až po náplň TCP, UDP či jiných protokolů transportní vrstvy. Nastavení nejsou nijak zjednodušeny či popsány, proto sestavení paketu opět není práce pro neinformovaný personál. Jakmile ale specifikaci dokončíme, můžeme přejít na druhou kartu, výmluvně pojmenovanou „Gen-b“, která nám umožní paket z karty předchozí replikovat a opakovaně odesílat. Najdeme zde sice základní oddělení obsahové části od adresace, ovšem možnosti modifikace jsou velmi omezené. Můžeme přepsat zdrojovou MAC nebo IP adresu náhodnými hodnotami nebo modifikovat data každého dalšího paketu na nízké úrovni (modifikovat jednotlivé Byty). Nedokáži si představit praktické využití takových funkcí.

Další dvě záložky, „Gen-s“ a „PCAP“, už nabízejí práci s PCAP souborz. **Gen-s** umožňuje uživateli zvolit dříve uložené PCAP soubory a definovat sekvenci, v jaké se budou číst a odesílat. V této chvíli bez jakékoliv možnosti editace. Karta **PCAP** na druhou



Obrázek 6: Grafické rozhraní programu packETH

stranu zobrazuje uživatelem načtený PCAP soubor, ze kterého lze zvolit jediný paket a kopírovat jeho specifika do příslušných polí karty **Builder**.

Ovládání programu je tedy velmi specifické a nedostatečně intuitivní. Nastavení a následné odesílání provozu, který by simuloval běžně zatíženou síť, je nepředstavitelné. Stejně jako u programu předchozího si dovolím konstatovat, že reálné využití najde při odesílání specifických jednotvárných proudů, při zatěžování sítě „hrubou silou“ nebo při replikování provozu, který byl zachycen na stejné pracovní stanici.

Shrnutí:

- **Autor** - Miha Jemec
- **Licence** - Open Source GNU GPL v2
- **Dostupnost** - volně ke stažení na *http://packeth.sourceforge.net*
- **Výhody**
  - použití i skrze příkazovou řádku
  - dle webových stránek vysoká efektivita odesílání
- **Nevýhody**
  - nevyhovující, nepřehledné uživatelské rozhraní
  - načítání PCAP souboru opět není efektivní, neboť jej můžeme pouze celý odeslat nebo vybrat jediný paket k úpravám
  - umožňuje oddělení adresace od obsahu a její náhodné nahrazování, ovšem bez dalších možností je tato funkce v praxi nepoužitelná

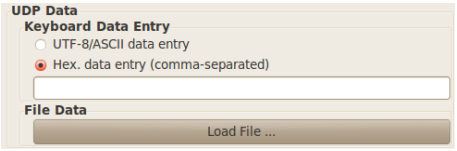
- práce s programem je obtížná, neprovází uživatele celým procesem
- zobrazuje nesrozumitelné chybové hlášení, které běžnému uživateli příliš nenapoví

3.3 Simple Packet Sender

Posledním z testovaných generátorů síťového provozu je program **Simple Packet Sender** (dále jen SPS). Hned při prvním pohledu je ovšem jasné, že slovo „jednoduchý“ (z anglického slova Simple v názvu) není pro popis uživatelského rozhraní programu zrovna výstižné. Jednotlivé karty skrývají nastavení paketu pro nejpoužívanější protokoly transportní (čtvrté) vrstvy, tj. TCP, UDP a ICMP. Pro všechny tyto protokoly je zde také jejich IPv6 varianta.

Až na kartě zvoleného protokolu uvidíme sekce pro nastavení nižších vrstev (druhé a třetí, tj. IP a MAC adresy). Jak lze vidět na obrázku 8, program SPS umožňuje nastavení snad každého možného pole hlaviček třetí a čtvrté vrstvy, poskytuje i základní nastavení druhé vrstvy, ale až zarážejícím způsobem omezuje datový obsah vyšších vrstev:

Uživatel může manuálně zadat obsah přenášených dat ve formě alfanumerických znaků nebo hexadecimálních hodnot, popř. nahrát libovolný soubor, jehož datová stopa vytvoří datovou náplň paketu - viz obrázek 7.



Obrázek 7: Možnosti naplnění datového obsahu UDP paketu v programu SPS

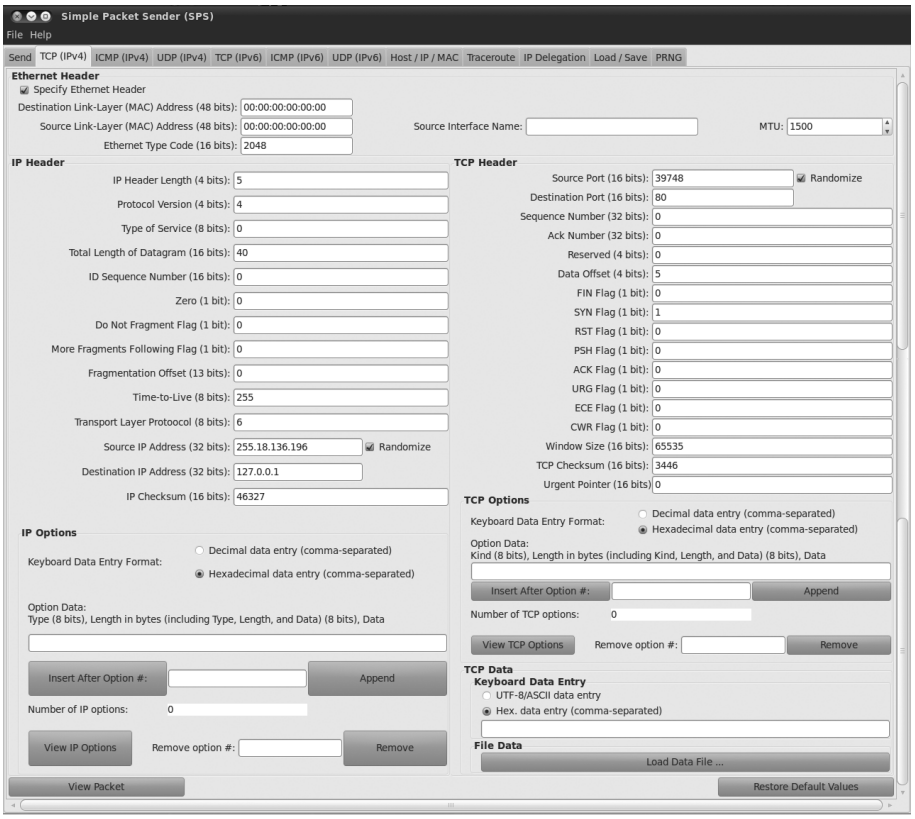
Zvolené řešení hodnotím jako nevhodné, neboť jakýkoliv běžný soubor je větší než maximální povolená velikost obyčejného paketu a navíc taková data budou postrádat smysl. Chybí zde jakákoliv možnost nahrání PCAP souboru nebo jiné smysluplné naplnění paketů daty, proto nejsme schopni pomocí tohoto programu simulovat náhodný a různorodý provoz.

Dalším výrazným negativem je pak nutnost manuálně specifikovat zvolené síťové rozhraní, jeho MAC i IP adresu. Tyhle informace jsou dosažitelné v systému a zdá se mi kontraproduktivní nutit uživatele tyto informace vyhledávat a kopírovat je do programu k tomu určenému.

Shrnutí:

- **Autor** - Hohlraum (h0h1r4um@yahoo.com)
- **Licence** - Open Source GNU GPL v2
- **Dostupnost** - volně ke stažení na *https://sites.google.com/site/simplepacketsender/*

- **Výhody**
  - přehledné uživatelské rozhraní, každý protokol na samostatné kartě
  - možnost nastavit i ty nejjemnější detaily jednotlivých paketů
- **Nevýhody**
  - složité ovládání
  - chybí načítání PCAP souborů
  - nutnost zadávat adresy a název rozhraní ručně
  - chybí možnost generování různorodého provozu
  - chybí možnost naplnění paketu smysluplnými (reálnými) daty



Obrázek 8: Definice TCP paketu v programu Simple Packet Sender

4 Návrh vlastního řešení

Cílem této práce je po předchozí analýze dostupných stávajících řešení navrhnout a implementovat nový generátor síťového provozu, který bude řešit nedostatky programů zmíněných výše a přinášet novou funkcionalitu a nové způsoby využití.

Mezi nejpalčivější nedostatky momentálně dostupných programů patří:

- přílišná složitost ovládání
- nemožnost globálně nastavit adresaci a parametry datového proudu
- chybějící smysluplné využití PCAP souborů
- nedostatečné možnosti pro generování různorodého provozu

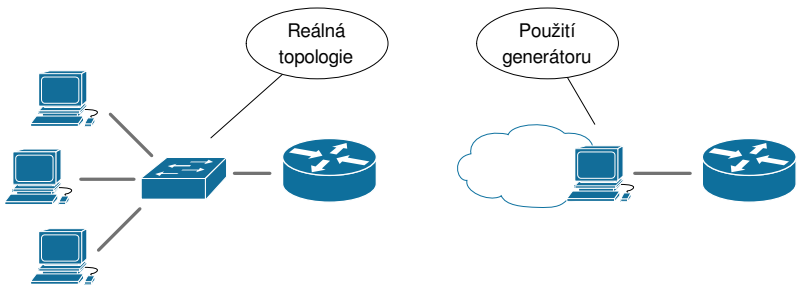
Cílový program tedy musí mít **jednoduché ovládání**. Technici pracující nebo vzdělávající se v oblasti počítačových sítí nemusí vždy vyžadovat exaktní nastavení každého paketu. Mnohokrát si všechna jednotlivá nastavení na každé vrstvě ani nepamatují a nejsou pro ně důležité. Zůstává jenom potřeba zaplnit linky datovými proudy, které se budou podobat reálnému provozu, který by nastal při připojení více stanic a uživatelů. Proto musí být výsledný program jednoduše a rychle použitelný při zachování možnosti nastavení všech důležitých parametrů.

Dalším bodem je **globální nastavení adresace** a s tím související umístění generátoru v síti. Program je navržen tak, aby běžel na pracovní stanici s operačním systémem Linux, která se nachází uvnitř jedné LAN sítě a má nastaveny všechny IP parametry pro danou síť. Generátor se pak stará o simulaci provozu a napodobení situace, kdy je připojeno ke směrovači či přepínači více stanic.

Jak je patrné z obrázku 9, generátor může svou činností zastupovat celou lokální síť a z pohledu ostatních aktivních prvků na síti vystupovat jako přepínač neboli koncentrátor. Zde vyvstává potřeba náhodně generovaných adres, ovšem za dodržení určitých pravidel. Ideálně by měl generátor umožňovat uživateli náhodně generovat adresy z přidělené místní sítě a stejným způsobem by měl volit i adresy příjemce. Samozřejmě musí program zachovávat i možnosti specifikace exaktní adresy v případě potřeby.

Jedním z největších nedostatků stávájících generátorů jsou právě **PCAP soubory**. Ty sice umožňují uložení záznamu datového provozu, ale jejich znovu-vytvoření je většinou neužitečné. Existuje jenom jediný případ, kdy bychom potřebovali znovu vytvořit již zaznamenanou komunikaci, a to v případě, pokud záznam proběhl na té samé stanici. Pokud ale PCAP soubor získáme jakoukoliv jinou cestou, je nám zaznamenaný provoz téměř k ničemu, protože pomocí stávajících řešení jej nejsme schopni rekonstruovat.

Řešení navrhované v této práci nabízí možnost načtení **PCAP souborů**, ale **využití** jenom jejich **datové části**. Separací adresace od datového obsahu dosáhneme jedinečné možnosti rekonstruovat samotný provoz a to v jakékoliv síti a směrovat jej taktéž do libovolné sítě.



Obrázek 9: Srovnání reálné a simulované topologie

S tím také souvisí poslední bod. Z analýzy jakéhokoliv dlouhodobého záznamu provozu na počítačové síti je patrné, že protokolů a aplikací je mnohem více, než může pokrýt jediný generátor, ale jisté trendy převažují nad ostatními. Navrhované řešení musí tedy nejen poskytovat možnosti generování různorodého datového provozu, podporovat více protokolů a aplikací, ale také umožnit napodobení reálného rozdělení datového provozu **procentuálním podílem** mezi vybrané datové proudy.

Posledním nedostatkem, kterým trpěla většina testovaných programů, byla nedostatečná modularita a flexibilita. Jakkoliv sofistikovaný algoritmus použijeme pro „náhodný“ výběr paketů, adres a dat, vždy se budeme jenom blížit reálnému provozu. Navrhovaný program by tedy měl umožnit uživateli díky své modulární architektuře jistou měrou **zasahovat do algoritmů**. Pomocí externího modulu bude možné předložit programu připravená data, získaná jakoukoliv jinou, např. nedeterministickou metodou.

Obecným předpokladem je samozřejmě **možnost uložení momentálního profilu** generovaného provozu, aby mohl být posléze do programu jen načten a znovu použit.

5 Implementace

V této kapitole popíši samotný vývoj navrhovaného řešení, použité vývojové prostředí, algoritmy a knihovny třetích stran.

5.1 Vývojové prostředí

Vývoj probíhal pod OS Linux, konkrétně v distribuci Lubuntu[7] virtualizované v prostředí Oracle VM VirtualBox[8].

Celá implementace je napsána v programovacím jazyce **C++** za použití open-source frameworku **Qt** ve verzi 5.2.1. Tato multiplatformní knihovna je jednou z nejpoužívanějších nástrojů při tvorbě programů s grafickým rozhraním a je vyvíjena již od roku 1999. Její výhody a přednosti jsou podrobněji popsány na oficiálních webových stránkách[9].

K implementaci bylo zvoleno podpůrné vývojové prostředí (dále jen IDE, z anglického „integrated development environment“) pro tento framework - **Qt Creator**, společně s integrovaným programem **Qt Designer** pro návrh grafického uživatelského rozhraní.

5.2 Použité knihovny a algoritmy

Implementace programu využívala standardní knihovny jazyka C++ a Qt frameworku. Mnoho standardizovaných datových typů bylo frameworkem nahrazeno vlastní alternativou (např. namísto `std::string` nabízí Qt datový typ `QString`). Tyto upravené implementace poskytují rozšířenou integraci užitečných funkcí, čímž značně zjednodušují práci na vývoji a zpřehledňují celý kód.

5.2.1 Knihovna libcrafter

Libcrafter je vysoceúrovňová knihovna napsaná v C++, zjednodušující vytváření, upravování a odesílání datových jednotek po síti. Na rozdíl od běžných knihoven, které paket chápou jako strukturu Bytů uložených v paměti, **libcrafter** volí sofistikovanější přístup k problému. Zprostředkovává vývojáři funkce pro manipulaci s jednotlivými vrstvami tak, jak se v paketu opravdu nachází a jak je popisuje referenční OSI model. Paket je zde definován jako souhrn vrstev, které jsou umístěny jedna nad druhou v definovaném pořadí.

Každou vrstvu může vývojář definovat jako instanci třídy `Ethernet`, `IP`, `ARP`, `ICMP`, `TCP` nebo `UDP`, pokud je její obsah a účel předem znám. Pokud rezignuji na definici pravidel pro pole v hlavičce každého z těchto protokolů a vystačím si jako vývojář pouze s datovou, resp. textovou reprezentací vrstvy, využívám instance třídy `RawLayer` (v překladu „syrová“ vrstvá). Jednotlivé vrstvy (objekty daných tříd) pak postupně umisťuji na zásobník reprezentující paket. Knihovna sice nehlídá faktickou správnost dat a pořadí

[7] http://lubuntu.net
[8] https://www.virtualbox.org
[9] http://qt-project.org/

vkládaných vrstev, ale zato kontroluje jednotlivá pole, doplňuje chybějící informace do povinných segmentů a i bez pozornosti vývojáře sama provádí výpočty a vyplňování kontrolních součtů v jednotlivých hlavičkách.

Souhrnné informace o knihovně:

- **Autor:** Esteban Pellegrino (pellegre.esteban@gmail.com)
- **Licence:** Open Source New BSD License
- **Rok vzniku:** 2012
- **Dostupná na:** *https://github.com/pellegre/libcrafter*

5.2.2 Generování náhodných čísel

Další problém, kterému čelí implementace navrhovaného generátoru, je generování náhodných čísel, například při vytváření seznamu IP adres z daného vzoru.

Zde je důležité zmínit, že „náhodná" čísla v tomto kontextu jsou pouze **pseudo-náhodná**, jelikož počítač (jakožto deterministický stroj) nic náhodně generovat nemůže. Za náhodná čísla se tedy vydávají výsledky algoritmů, které závisí na několika nestálých proměnných (např. aktuální čas) a dávají tak iluzi náhodné volby.

Klasickým, zavedeným způsobem pro generování náhodných čísel byla C++ funkce *rand()*[10], jenže ta není pro potřeby generátoru úplně přesná. Funkce `rand()` sice vrací pseudo-náhodné číslo v jistém intervalu, přičemž každé číslo z intervalu může být vráceno se stejnou pravděpodobností, ale v kombinaci s omezením tohoto intervalu pomocí funkce *modulo* (zbytek po celočíselném dělení) zjistíme, že některé výsledky jsou vraceny častěji, než jiné.

Tento problém byl eliminován implementací knihovny ⟨ *random* ⟩ ze standardu **C++11**. Jako stroj pro generování pseudo-náhodných čísel byl zvolen Mersenne Twister[11] a generované výsledky byly rozřazeny dle rovnoměrného rozdělení pravděpodobnosti (tzv. uniformní distribuce). Kombinace těchto nástrojů umožňuje mnohem pokročilejší generování náhodných čísel, kdy je opravdu každému prvku z intervalu zaručena stejná pravděpodobnost.

5.2.3 Použíté funkce

V této sekci jsou popsány dvě metody implementované v programu, které řeší zajímavé programátorské problémy. Zbytek metod je podrobně popsán v dokumentaci programu, která je přiložena na CD u této práce.

[10] http://www.cplusplus.com/reference/cstdlib/rand/
[11] http://www.math.sci.hiroshima-u.ac.jp/ m-mat/MT/emt.html

Get Gateway

Tato funkce zjistí IP a MAC adresu výchozí brány, která je nastavená na pracovní stanici pro dané síťové rozhraní. Jako první parametr přijímá název rozhraní, další dva parametry jsou odkazem předané proměnné pro uložení získaných adres.

Na metodě *Get Gateway* lze názorně vidět použití knihovny **libcrafter**. Ve funkci program vytvoří IP hlavičku (3. vrstva), zdrojovou IP adresu přiřadí ze zvoleného síťového rozhraní a jako cílovou adresu vloží nesmyslnou IP (1.2.3.4). Vzhledem k faktu, že se tato nesmyslná IP adresa téměř jistě nachází mimo rozsah lokální sítě, spoléháme na hostovský operační systém, že tento paket přepošle na adresu výchozí brány. Řešení tohoto problému spočívá v jeho jednoduchosti. Nesmyslná cílová adresa zajistí přeposlání na správný směrovač a nízká hodnota pole TTL[12] zajistí vypršení platnosti paketu právě v momentě, kdy dorazí na adresu výchozí brány. Ta má za povinnost informovat odesílatele o tomto faktu za pomocí ICMP protokolu. Jednoduše tedy v metodě zachytíme ICMP odpověď, extrahujeme hlavičky druhé a třetí vrstvy a uložíme MAC a IP adresu výchozí brány do proměnných poskytnutých druhým a třetím argumentem funkce. Návratovou hodnotou funkce je 1 v případě úspěšného nalezení adres výchozí brány a 0 v případě neúspěchu.

```
int PacketGenerator::GetGateway(QString interface, QString &IPadd, QString &MACadd) {
    string iface = interface.toStdString();

      /* nastaveni IP hlavicky */
    IP ip_header;
    ip_header.SetSourceIP(GetMyIP(iface));
    ip_header.SetDestinationIP("1.2.3.4");
    ip_header. SetIdentification (RNG16());
    ip_header.SetTTL(1);

      /* libovolna datova napln paketu */
    RawLayer raw_header;
    raw_header.SetPayload("Packetpayload\n");
      /* vytvoreni paketu pomoci skladani vrstev */
    Packet packet = ip_header / raw_header;
      /* odeslani danym rozhranim a ocekavani ICMP odpovedi */
    Packet* rcv = packet.SendRecv(iface,0.2,1,"icmp");
      /* pokud jsme obdrzeli odpoved, extrahujeme potrebne informace */
    if (rcv) {
            IP* ip_layer = rcv->GetLayer<IP>();
            IPadd = QString::fromStdString(ip_layer->GetSourceIP());
            Ethernet* eth_layer = rcv->GetLayer<Ethernet>();
            MACadd = QString::fromStdString(eth_layer->GetSourceMAC());
            delete rcv;
            return 1;
    } else
        return 0;
}
```

Výpis 1: Kód funkce GetGateway

[12]Time To Live - pole v IP hlavičce, určující životnost paketu.

Get IP addresses

Objekt třídy `Network` má k dispozici metodu `GetIPaddresses`, jejímž návratovým typem je seznam všech použitelných IP adres v daném rozsahu.

Uživatel zadá z důvodů jednoduchosti a přívětivosti jakoukoliv IP adresu a masku sítě ve zkráceném formátu CIDR[13] - tedy počet použitých bitů v rozsahu 1-32, kde 32 označuje IP adresu jednoho zařízení. Úkolem této metody je tedy správně určit rozsah IP adres ze zadané hodnoty, vyloučit nepoužitelné IP adresy (identifikační adresu sítě a adresu pro všesměrové vysílání) a vrátit programu seznam použitelných IP adres ve formě standardních řetězců.

Uvádím zde pro přehlednost jen část této funkce, která je zodpovědná za rozdělení IP adresy dle teček na jednotlivé části, převod číselné CIDR masky na decimální hodnotu a určení počáteční IP adresy.

```
  ...
  /* rozdelime IP adresu na jednotlive casti */
  parsed = network.split(".");

  /* cast IP adresy, ktera se nemenni, ulozime do vysledneho retezce */
while (mask >= 8)
{
    base += parsed.takeFirst()+".";
    mask -= 8;
}

  /* oktet, ve kterem zacina dana podsit, prevedeme na cele cislo */
int startIPpart = parsed.takeFirst().toInt();

  /* maska je nyni v rozsahu 0-7, ziskame tedy doplnek do osmi */
int maskfull = 8 - mask;
  /* .. prevedeme na decimalni podobu */
maskfull = 1 << maskfull;
  /* .. a ziskame masku site v decimalnim zapisu */
maskfull = 256 - maskfull;

  /* nyni staci nad zadanym cislem a maskou site provest bitovou operaci AND */
startIPpart = startIPpart & maskfull;
  ...
```

Výpis 2: Kód funkce Get IP addressses

5.3 Architektura programu

Program je navržen jako modulární, využívá objektově orientované přístupy a díky tomu umožňuje uživateli ukládat a znovu načítat celé konfigurace, jednotlivé protokoly či celý generovaný provoz.

[13] CIDR - Classless Inter-Domain Routing - systém přidělování masky sítě pro každou podsíť, který nahradil zastaralé třídní rozdělení.

Pro nastavení celého programu je při spuštění programu vytvořena instance třídy `Pagen`, která udržuje veškeré parametry provozu v jediném objektu. Přístup k těmto vlastnostem je možný skrze veřejné metody `Get` a `Set` pro každou proměnnou.

Další použité třídy jsou `Network` a `ProtocolSpec`, jejichž instance slouží pro definici jedné IP sítě, resp. jednoho použitého profilu provozu.

Poslední tři třídy - `PacketGenerator`, `PercentDialog` a `ProtocolDialog` jsou stavebními kameny pro grafické uživatelské rozhraní. Dialogová okna a logika spojená s jejich prováděním jsou uloženy v příslušných třídách. `PacketGenerator` drží veškerý kód hlavního programu, spojuje jednotlivé třídy a zajišťuje veškerou obsluhu programu.

Jakmile jsou uloženy či nahrány všechny informace potřebné k samotnému generování, vytváří program kontejner s nastavitelnou velikostí, který slouží jako úložiště paketů připravených na odeslání. Před jeho naplněním jsou vytvořeny seznamy o stejné délce, držící v každém řádku informace pro vytvoření jednoho paketu.

Postup přípravy datového kontejneru je následující:

1. Získáme list použitelných zdrojových IP adres. Z listu je vybíráno náhodně. Pokud je jich méně, vybíráme stále ze stejného seznamu a můžou se opakovat.
2. Získáme list použitelných cílových IP adres. Opět náhodným výběrem plníme seznam řádek po řádku. Je zde zajištěna uniformní distribuce z celého rozsahu.
3. Seznam objektů čtvrté vrstvy se omezuje na TCP nebo UDP pakety. Jestliže načítáme z PCAP souboru, jsou z jednotlivých paketů extrahovány objekty čtvrté vrstvy společně s příslušnými daty a uloženy do seznamu. Pokud generujeme protokol definovaný ručně, vytvoří se objekt čtvrté vrstvy „manuálně“, nastaví se požadované hodnoty a připojí se náhodná data o zadané velikosti.
4. Podle zvolené metody nastavení QoS plníme seznam pro QoS při výběru zdrojových adres, cílových adres nebo protokolu.
5. Po naplnění všech těchto seznamů jsou tvořeny objekty kontejneru - instance třídy `Packet` z knihovny **libcrafter**
    - Je vytvořen objekt druhé vrstvy dle příslušné konfigurace, společný pro všechny pakety.
    - Vzhledem k faktu, že IP adresy již jsou v náhodném pořadí, procházíme seznamy systematicky a vytváříme pro každý řádek příslušnou IP hlavičku. Ze seznamu DSCP hodnot je přiřiřazeno odpovídající nastavení QoS.
    - Objekty protokolů (čtvrtá a pátá vrstva) však náhodně generovány nebyly, proto pro každou IP hlavičku je náhodně vybrán objekt protokolu a následně odebrán ze seznamu.
    - Všechny vrstvy jsou umístěny do paketu a ten následně umístěn na konec kontejneru.

Takto připravený kontejner paketů už lze jen iterátorem procházet a pakety odesílat bez dalšího zdržení.

5.3.1 Použití XML

Jak již bylo uvedeno výše, program využívá XML soubory pro ukládání a načítání nastavení jednotlivých objektů. Prvním případem jsou definice jednotlivých profilů provozu (protokolů). Při zvolení sekce `Protocols` program načte všechny XML soubory obsahující definice protokolů ze složky `protocols/` v adresáři programu. Pokud XML soubor obsahuje validní definici protokolu odpovídající zavedené struktuře, je zobrazen uživateli k výběru na kartě s protokoly. Pokud si uživatel zvolí možnost přidání nového protokolu na téže kartě, je výsledná specifikace exportována do XML pomocí metody `SaveToXML` ze třídy `ProtocolSpec` a uložena ve společném adresáři. V případě dalšího spuštění programu je již tato nová specifikace načtena a zobrazena uživateli k výběru automaticky.

Grafické prostředí programu je sice intuitivní a přehledné, přesto ale specifikace generovaného provozu vyžaduje jistý čas. Opět za účelem zvýšení uživatelské přívětivosti a celkové použitelnosti programu můžeme nastavení exportovat do XML souboru a při příštím spuštění programu jej znovu použít. Stejně jako třídy protokolu má i instance třídy `Pagen` k dispozici metodu `saveToXML` pro uložení veškerého nezbytného nastavení do souboru.

```
<?xml version="1.0" encoding="UTF-8"?>
<pagen_profile>
    <interfaceName>eth0</interfaceName>
    <interfaceIP>10.0.2.15</interfaceIP>
    <interfaceMask>255.255.255.0</interfaceMask>
    <interfaceMAC>08:00:27:ac:40:b6</interfaceMAC>
    <gatewayIP>10.0.2.2</gatewayIP>
    <gatewayMAC>52:54:00:12:35:02</gatewayMAC>
    <destMAC>arp</destMAC>
    <sourceMAC>local</sourceMAC>
    <qosMode>1</qosMode>
    <destIPs count="1">
        <destNet>
            <net>189.12.12.19</net>
            <mask>22</mask>
            <dscp>0</dscp>
        </destNet>
    </destIPs>
    <sourceIPs count="1">
        <sourceNet>
            <net>10.0.2.15</net>
            <mask>32</mask>
            <dscp>28</dscp>
        </sourceNet>
    </sourceIPs>
    <protoMix count="3">
        <protocol>
            <name>TCPtest</name>
            <dscp>0</dscp>
            <priority>56</priority>
        </protocol>
        <protocol>
            <name>UDPtest</name>
```

```
            <dscp>0</dscp>
            <priority>44</priority>
        </protocol>
    </protoMix>
</pagen_profile>
```

Výpis 3: Konfigurace programu exportovaná do XML souboru

5.3.2 Modularita

Slovo „modularita“ slovník cizích slov vysvětluje jako „stavebnicovost“ či „skladebnost“. U navrhovaného řešení programu pod tímto pojmem chápeme nabízené možnosti, jak modifikovat činnost programu či generovaná data. Jedním z cílů této práce bylo právě dosažení jistého stupně modularity, neboť nabízená a implementovaná řešení nemusí být dostatečná pro určitou skupinu uživatelů.

Existují dvě možnosti, jak ovlivnit způsob fungování implementovaného programu.

První způsob byl již zmíněn v sekci 4 - načítání PCAP souborů. Uživatel není nucen spoléhat na generovací schopnosti programu, nýbrž může při vytváření specifikace nového protokolu přiřadit PCAP soubor a využít data z jednou již zaznamenaného provozu. Následně lze samozřejmě protokoly navzájem kombinovat a určovat jejich procentuální podíl na výsledném provozu.

Druhý způsob nabízí uživateli kompletní kontrolu nad generovanými jednotkami dat. Mezi zdrojovými kódy programu najdeme i hlavičkový soubor `extmodule.h`, který definuje třídu, jejíž instanci bude program očekávat po spuštění dynamické knihovny s příponou **.so**.

```
#include <list>
#include <string>

using namespace std;

class ExtModule
{
public:
 ExtModule() {}

  list <string> L2MACSource;
  list <string> L2MACDest;
 enum L3Type_s { IP, ARP, ICMP};
  list <L3Type_s> L3Type;
  list <string> L3IPSource;
  list <string> L3IPDest;
  list <int> L3IPDSCP;
  list <int> L3IPTTL;
  list <int> LayerCount;

 struct Layer_s {
      list <int> payloadSize;
```

```
        list <unsigned char*> payloadData;
    } ;

    list <Layer_s> Layers;
};
```

Výpis 4: Hlavičkový soubor pro externí moduly.

Třída `ExtModule` definuje vlastnosti objektu, který program očekává jako návratovou hodnotu funkce `getCont()`, kterou externí modul musí implementovat. Jsou zde využity standardní C++ třídy `list` pro seznamy a `string` pro jednotlivé řetězce - k implementaci modulu tedy nejsou zapotřebí žádné knihovny třetích stran.

Celá signatura hlavní funkce je následující:

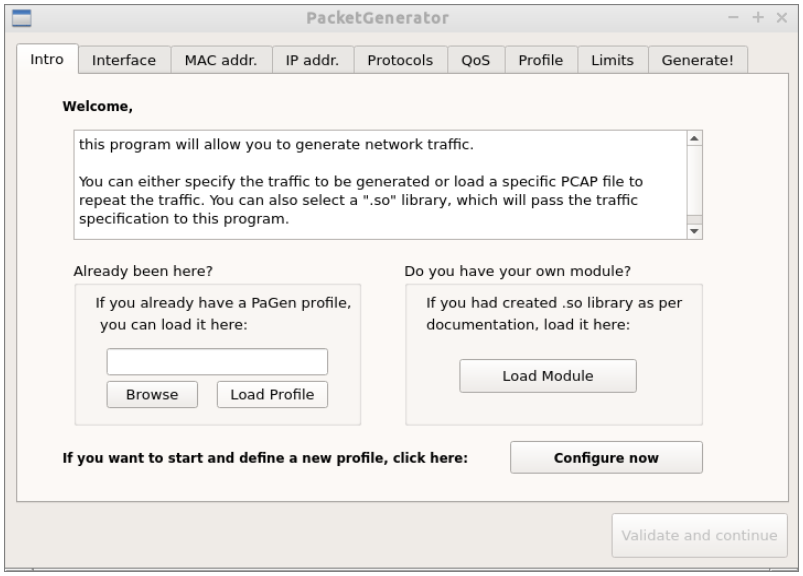
```
ExtModule * getCont (int count);
```

Jediným argumentem metody je celočíselná hodnota definující velikost požadovaného zásobníku, tj. kolik záznamů musí mít každý seznam v předávaném objektu. Ten je posléze sekvenčně procházen, z dat každého „řádku“ všech seznamů je vytvořen paket a umístěn do kontejneru k odeslání.

Každý záznam musí obsahovat:

- **L2MACSource** - zdrojová MAC adresa 2. vrstva
- **L2MACDest** - cílová MAC adresa 2. vrstva
- **L3Type** - záznam jednoho prvku z výčtového typu `L3Types`, určující protokol třetí vrstvy
  - **IP** - klasická IP hlavička, za níž může následovat datový záznam
  - **ARP** - pakety protokolu ARP pro zjištění MAC adresy příjemce (využívá pole L3IPSource a L3IPDest)
  - **ICMP** - paket protokolu ICMP, tj. klasický příkaz `ping` (využívá pole L3IPSource a L3IPDest)
- **L3IPSource** a **L3IPDest** - zdrojová a cílová IP adresa, využívaná všemi třemi dostupnými možnostmi
- **L3IPDSCP** a **L3IPTTL** - nastavení hodnot polí DSCP a TTL v rámci IP hlavičky
- **LayerCount** - počet datových vrstev za IP hlavičkou
- **Layers** - seznam struktur, kde každá obsahuje určitý (definovaný vlastností LayerCount) počet datových vrstev, definovaných polem Bytů a velikostí tohoto pole

Požadavky k implementaci externího modulu jsou tedy jenom dva - načtení hlavičkového souboru `extmodule.h` a korektní implementace metody `getCont`, která naplní objekt předaným počtem záznamů. Žádné další restrikce neexistují. Externí modul může využívat jakýkoliv algoritmus a generovat jakýkoliv provoz. Po načtení externího modulu a naplnění kontejneru můžeme ovlivnit jenom rychlost a limit odesílání.



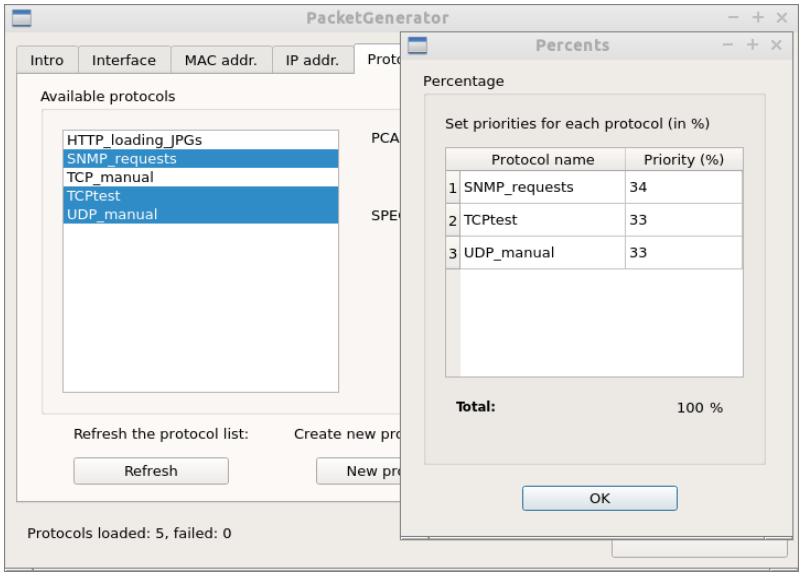
Obrázek 10: Úvodní obrazovka programu

5.4 Ovládání programu

Programem uživatele provází intuitivní grafické prostředí vytvořené za pomocí programu **Qt Designer**. Celý program je implementován v angličtině za účelem eliminace jazykových bariér při reálném nasazení programu.

Hlavní okno se skládá ze tří částí, jak můžete vidět na obrázku 10. Nahoře se nachází karty reprezentující jednotlivé kroky v průběhu nastavení programu.

1. Intro - úvodní stránka
2. Interface - výběr síťového rozhraní (1. vrstva OSI modelu)
3. MAC addr. - nastavení MAC adresace (2. vrstva)
4. IP addr. - výběr zdrojových a cílových sítí (3. vrstva)
5. Protocols - zobrazení, výběr či definice nové specifikace protokolu
6. QoS - nastavení QoS, tj. volba pravidel a příslušné DSCP hodnoty
7. Profile - kontrola zadaných údajů, možnost exportu nastavení do XML
8. Limits - jakým způsobem se mají data odesílat a kdy se má program zastavit
9. Generate - zde probíhá samotné generování dat a zobrazování statistik



Obrázek 11: Výběr více protokolů a jejich procentuální podíl

Pod kartami je umístěno hlavní okno pro veškerou interakci s uživatelem a ve spodní části najdeme textový popisek zobrazující aktuální stav a validační tlačítko, které zkontroluje na každé kartě zadaná data a umožní uživateli přejít dál. Toto tlačítko je také jediný hlavní ovládací prvek programu, proto je jeho použití velmi intuitivní a jednoduché.

Proces nastavení generovaného provozu začíná na úvodní straně, kde jsou uživateli nabídnuty tři možnosti. Pokud již má **uložený profil** a chce generování zopakovat, může jej vybrat a načíst, čímž program přeskočí všechna nastavení a pro kontrolu zobrazí kartu s načtenými údaji, odkud může uživatel pokračovat.

Jestliže má uživatel k dispozici **externí modul** (dynamickou knihovnu .so), je mu k dispozici druhá z nabízených možností. Tlačítko `Load Module` knihovnu načte, provede uložené algoritmy k připravení kontejneru paketů a umožní uživateli nastavit jenom limity a styl generování těchto dat.

Třetí a poslední možností je **ruční nastavení** krok po kroku, které začíná zvolením tlačítka `Configure now`. To aktivuje validační prvek a zpřístupní první kartu s nastavením síťového rozhraní. Výběr rozhraní a následné nastavení MAC adres je naprosto intuitivní.

Záložka s IP adresací zobrazuje dvě části - jednu pro zdrojové IP adresy, druhou pro cílové IP adresy. V případě, že chceme generovat provoz imitující aktivitu uživatelů na celé lokální síti, nemusíme zadávat adresy ručně, stačí zvolit příslušnou volbu. Sítě se zadávají v obvyklém formátu, tj. čtyři oktety s hodnotou 0-255 a maska sítě v rozsahu 1-32 (maska 32 značí jedinou IP adresu).

Na kartě **Protocols** je nutné specifikovat, které protokoly mají být zahrnuty do generovaného provozu. Více položek lze vybrat při držení klávesy `Ctrl`. Při vícenásobném

výběru se zobrazí okno, kde je nutné explicitně definovat procentuální podíl jednotlivých protokolů na obsahu kontejneru, jak lze vidět na obrázku 11.

Další v řadě je nastavení pravidel QoS. Pokud chceme pravidla QoS otestovat, jsou zde k dispozici tři způsoby přepisu DSCP hodnot:

1. podle zdrojové sítě
2. podle cílové sítě
3. podle protokolu

Podle zvoleného pravidla se uživateli aktivuje příslušné pole a může nastavit DSCP hodnotu pro každou položku zvlášť.

Poslední tři záložky jsou **Review, Limits** a **Generate**, jejichž účel je zcela jednoznačný a ovládání intuitivní.

5.5 Možnosti nasazení

Nasazení programu v praxi je reálné, neboť žádné ze stávajících řešení neposkytuje takovou funkcionalitu při zachování jednoduchosti a intuitivního ovládání. Velké společnosti, zaměřené na správu sítě, mají často síťové laboratoře, kde probíhá výuka a testování síťových zařízení či protokolů. Taktéž se tam konají cvičení připravující zájemce na složení certifikačních zkoušek. Podobné laboratoře existují i na školách k výuce odborných předmětů a právě zde by navrhované řešení mohlo najít své uplatnění.

Možnost připojení jediné pracovní stanice a „oživení“ topologie reálným datovým proudem by jistě ocenilo mnoho síťových techniků a studentů. Nasazením tohoto programu by bylo konečně možné otestovat chování protokolů zmíněných v kapitole 2.4 na opravdové síti. Potenciálním místem pro nasazení a využití tohoto programu jsou tedy školní a firemní síťové laboratoře, popř. najde uplatnění jako jednoduchá pomůcka síťového technika při testování propustnosti.

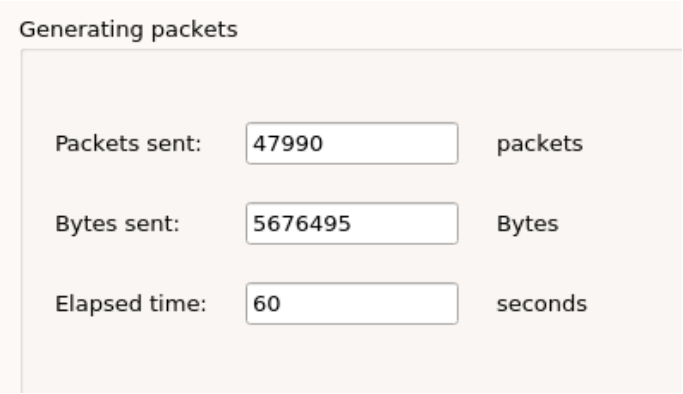
6 Testování

Testování možností a funkčnosti implementovaného generátoru probíhalo na přenosném počítači s procesorem **Intel Core i5-3317U**, osazeným **6 GB** operační paměti. Program byl spuštěn ve virtualizovaném prostředí programu **Oracle VM VirtualBox** s nainstalovaným operačním systémem Lubuntu v 32-bitové verzi. Zprostředkování síťového rozhraní pro virtualizovaný operační systém spravuje taktéž program VirtualBox. Tomuto virtuálnímu stroji bylo přiřazeno 1024 MB operační paměti. Několika pokusy byla otestována maximální rychlost, tj. maximální počet paketů odeslaných za jednotku času, dále také největší dosažitelné zatížení sítě a stabilita programu při kombinaci několika faktorů.

6.1 Rychlost

Maximální počet paketů odeslaných za jednu minutu (resp. vteřinu), kterého je program schopen dosáhnout, byl otestován následujícím nastavením:

- zdrojová síť: IP adresa lokálního rozhraní
- cílová síť: 10.10.10.0 /24 (náhodně zvolená)
- protokol: SNMP requests (z PCAP souboru)
- profil odesílání: proudové, 999999 paketů za vteřinu (žádné umělé pauzy mezi pakety)
- limit: časový, 60 sekund



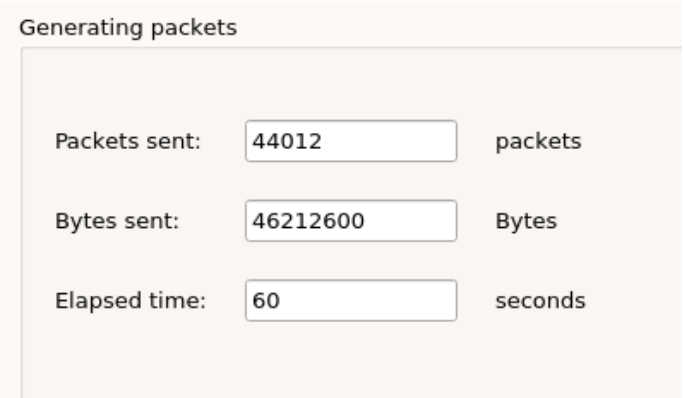
Obrázek 12: Výsledek testování maximální rychlosti odesílání

Výsledek testování maximální rychlosti odesílání lze vidět na obrázku 12. Při odesílání relativně malých SNMP paketů bylo odesláno **47 990 paketů** za minutu, což odpovídá rychlosti přibližně **800 paketů** za vteřinu.

6.2 Maximální velikost

Pro otestování maximální „propustnosti“ byla vytvořena nová specifikace protokolu (viz. níže). Opět bylo testování prováděno několikrát po dobu jedné minuty.

- zdrojová síť: IP adresa lokálního rozhraní
- cílová síť: 20.20.20.0 /24 (náhodně zvolená)
- protokol: UDP, cílový port 555, zdrojový port náhodný, 1000 Bytů v každém paketu
- profil odesílání: proudové, 999999 paketů za vteřinu (žádné umělé pauzy mezi pakety)
- limit: časový, 60 sekund



Obrázek 13: Výsledek testování maximální velikosti generovaného provozu

Jak lze vidět z výsledku tohoto testování na obrázku 13, maximální množství dat, přenesených za minutu, bylo **46 212 600 Bytů**. Tato velikost odpovídá 44 MB, což znamená, že program dokáže bez problému zatížit připojení se **šířkou pásma 6 Mbps**.

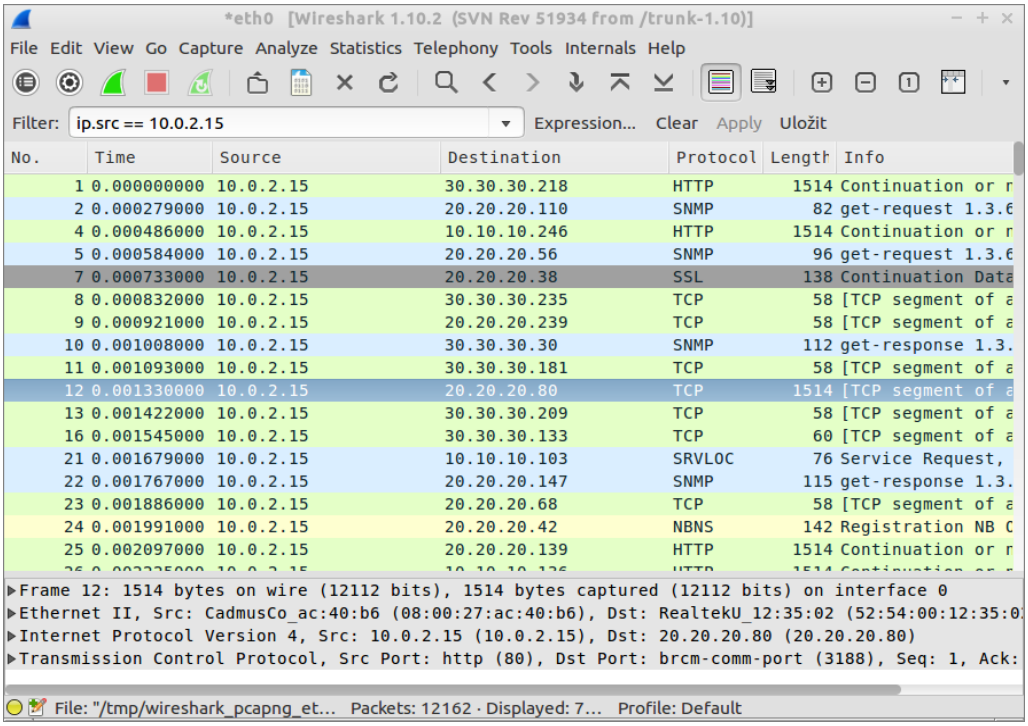
6.3 Stabilita

Testy stability zahrnovaly rozsáhlejší konfiguraci - více cílových sítí, více protokolů, přepisování QoS pravidel a nastavení rozdílného způsobu odesílání. Taktéž byla výsledná konfigurace exportována do XML a opětovně do programu nahrána.

- zdrojová síť: IP adresa lokálního rozhraní
- cílové sítě:
  - 10.10.10.0 /24

  - 20.20.20.0 /24
  - 30.30.30.0 /24

- protokoly:
  - SNMP (z PCAP souboru)
  - HTTP načítání JPG souboru (z PCAP souboru)
  - TCP, zdrojový port náhodný, cílový port 443 (specifikovaný manuálně)
- profil odesílání: po dávkách, 100 paketů v každé dávce, 500ms pauza mezi dávkami
- limit: 10 000 odeslaných paketů



Obrázek 14: Výsledek testování stability

Část záznamu výsledného generovaného provozu je zachycen na obrázku 14. Program byl **stabilní** v průběhu celého testování, odeslané pakety **odpovídaly nastaveným pravidlům** a taktéž **způsob a časování odesílání** paketů **korespondoval se specifikací** v programu.

7 Závěr

Ve své práci jsem čtenáře seznámil se základy a pravidly komunikace na počítačové síti a následně definoval důvody a situace, které vyžadují použití generátoru síťového provozu. Byla položena teoretická východiska spjatá s problematikou datového provozu na počítačové síti a jeho generováním. Dále jsem z uživatelského pohledu analyzoval tři stávající řešení a poukázal na jejich nedostatky, jenž komplikují jejich použití.

S ohledem na tyto nedostatky bylo navrženo a implementováno řešení, které umožňuje generovat různorodé datové proudy a simulovat tak reálný provoz. Program implementuje jednoduché, uživatelsky příjemné a intuitivní grafické rozhraní umožňující promptní ovládání programu bez studia uživatelské příručky. Uživateli je tímto programem dán k dispozici efektivní nástroj pro testování počítačové sítě, který umožňuje generovat provoz jinak, než jeho předchůdci. Velkou výhodou navrhovaného řešení je možnost oddělení adresace od obsahu načteného souboru a následná replikace zaznamenaného provozu do jiné sítě.

Implementované řešení je v neposlední řadě také velmi modulární - umožňuje uživateli nahrání vlastní knihovny a její využití k naplnění zásobníku pro odesílání. Celý algoritmus přípravy paketů je tak modifikovatelný a je tedy čistě na tvůrci knihovny, jaká data bude program odesílat.

Aplikace tedy s nemalou mírou úspěšnosti kompenzuje nedostatky stávajících generátorů a stává se tak možným kandidátem pro reálné využití, a to především při výuce nebo při testování počítačových sítí.

Ondřej Jaroš

8 Reference

[1] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS.* 5., aktualiz. vyd. Brno: Computer Press, 2008, 488s. ISBN 978-80-251-2236-5.

[2] KERNIGHAN, Brian W a Dennis M RITCHIE. *The C programming language*. 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, c1988, xii, 272 s. ISBN 01-311-0362-8.

[3] PUŽMANOVÁ, Rita. *Moderní komunikační sítě od A do Z*. 2. aktualiz. vyd. Brno: Computer Press, 2006, 430 s. ISBN 80-251-1278-0.

[4] VIRIUS, Miroslav. *Jazyky C a C: Komplexní průvodce.* 2. aktualizované vydání. Praha: Grada, 2011, 367 s. Knihovna programátora. ISBN 978-80-247-3917-5.

[5] QT PROJECT HOSTING. *Qt Project Documentation.* [online]. [cit. 2014-05-06]. Dostupné z: http://qt-project.org/doc/

[6] PELLEGRINO, Esteban. *Libcrafter Quick Start Guide:* Introduction to libcrafter. [online]. [cit. 2014-05-06]. Dostupné z: https://code.google.com/p/libcrafter/wiki/QuickStartGuide

[7] CPPREFERENCE.COM. *Pseudo-random number generation.* [online]. [cit. 2014-05-06]. Dostupné z: http://en.cppreference.com/w/cpp/numeric/random

[8] IETF. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* [online]. [cit. 2014-05-06]. Dostupné z: http://tools.ietf.org/html/rfc2474

[9] CISCO SYSTEMS, Inc. *DiffServ:* The Scalable End-to-End QoS Model. [online]. [cit. 2014-05-06]. Dostupné z: http://www.cisco.com/en/US/technologies/tk543/tk766/technologies_white_paper09186a00800a3e2f.html

[10] LAMPING, Ulf. *Libpcap File Format.* [online]. [cit. 2014-05-06]. Dostupné z: http://wiki.wireshark.org/Development/LibpcapFileFormat

A Obsah CD

Na přiloženém CD se nachází tyto adresáře:

- **PacketGenerator-documentation** - dokumentace k implementovanému programu
- **PacketGenerator-compiled** - zkompilovaná aplikace
- **PacketGenerator-source** - zdrojové kódy aplikace
- **TeX** - zdrojové kódy textu bakalářské práce
- **PDF** - text bakalářské práce ve formátu PDF