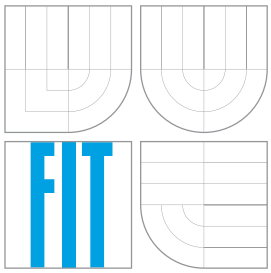
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ PRO GENERÁTOR PAKETŮ

GRAPHICAL USER INTERFACE FOR PACKET GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL CHROMČÁK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ MATOUŠEK
SUPERVISOR

BRNO 2013

Abstrakt

Vzhledem ke stále se zvyšujícím požadavkům na rychlost veškerých hardwarových a softwarových komponent se nalézají řešení, která principiálně mohou dosahovat přijatelnějších parametrů, než řešení běžně známá. Jedním z nich je na poli generování umělého síťového provozu použití softwaru s hardwarovou akcelerací. Pomocí tohoto přístupu byl implementován generátor paketů, ve verzi bez grafického uživatelského rozhraní. Aby se tento systém mohl více rozšířit do kruhu cílových uživatelů, vznikla potřeba grafické uživatelské rozhraní vytvořit. Tato bakalářská práce popisuje návrh rozhraní, jeho implementaci v jazyku JavaFX, testování na skutečných uživatelích a zároveň obsahuje tutoriál sloužící k demonstraci práce s rozhraním.

Abstract

According to increasing requirements on speed of different software and hardware components, there are solutions, which can, by principle, reach better parameters, then solutions commonly known. One of them is to use software with hardware acceleration on the field of generating synthetic network traffic. Exactly this way a packet generator was implemented, in current version without graphical user interface. But to let this system spread into the target group of users, there is need to implement also this interface. This bachelor's thesis describes proposal of graphical interface, its implementation in JavaFX programming language, testing on real users and tutorial demonstrating how to use this interface.

Klíčová slova

grafické uživatelské rozhraní, generátor paketů, GUI, NetCOPE, COMBOv2, JavaFX, NetBeans, JavaFX Scene Builder

Keywords

graphical user interface, packet generator, GUI, NetCOPE, COMBOv2, JavaFX, NetBeans, JavaFX Scene Builder

Citace

Michal Chromčák: Grafické uživatelské rozhraní pro generátor paketů, bakalářská práce, Brno, FIT VUT v Brně, 2013

Grafické uživatelské rozhraní pro generátor paketů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Matouška

.....
Michal Chromčák
14. května 2013

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Matouškovi za veškeré poskytnuté informace, rady, připomínky a v neposlední řadě za vstřícný přístup, jakým vedl tuto bakalářskou práci.

Obsah

Úvod

V dnešní době hraje jakýkoliv ztracený čas velkou roli. Tomuto trendu se přizpůsobuje celý svět, oboru informatiky nevyjímaje. Vzniká tlak na jednoduchost ovládání věcí při zachování jejich funkčnosti. Z hlediska programu se jedná o uživatelské rozhraní – prostředek pro komunikaci člověka s aplikací. Uživatelských rozhraní existuje více druhů, přičemž ve většině případu je nejideálnější volbou grafické uživatelské rozhraní (GUI - Graphical User Interface) díky jeho vysoké informační propustnosti informací.

Tato práce se zabývá tvorbou GUI pro generátor paketů, který je implementovaný na akcelerační PCIe (Peripheral Component Interconnect Express, PCI Express) kartě s FPGA (Field Programmable Gate Array) čipem [3]. Z pohledu vlastností samotné aplikace je zajímavé, že tento generátor poskytuje vyšší rychlost, než čistě softwarové řešení s náklady nižšími, než řešení čistě hardwarové. To pro aplikaci implikuje nadějný výhled do budoucna. Druhým důležitým bodem, tentokrát z pohledu samotné tvorby GUI, jsou stávající rozhraní, která aplikace má.

Před psaním této práce neexistovalo grafické rozhraní pro tento generátor paketů. Implementovány byly dva způsoby komunikace s aplikací. Prvním z nich je rozhraní založené na použití souboru ve formátu XML. Soubor je načten, informace jsou převedeny do interní reprezentace programu, který je načítá, a následně zapsány do paměťového prostoru karty. Zápis se děje pomocí funkcí knihovny `libcombo` [2]. Druhým způsobem je použití knihovny `libcombo` bez souboru XML, a to přímým voláním příslušných funkcí zajišťujících zápis hodnot do paměti karty.

Obě tato rozhraní jsou ve své kategorii výborná. Jsou přehledná, intuitivní, avšak postrádají výhody GUI. Uživateli chybí možnost komunikovat s aplikací bez znalosti jazyka C v případě přímého volání funkcí knihovny `libcombo`, respektive formátu XML v případě použití konfiguračního souboru. Obě rozhraní postrádají kontrolu vstupních hodnot při jejich zadávání, celková orientace v samotném nastavení módu aplikace je náročnější, než může nabídnout GUI a především tato rozhraní nemohou dosahovat takové informační propustnosti jakou má GUI. Na základě těchto nedostatků vznikl požadavek na vytvoření grafického uživatelského rozhraní pro generátor paketů, čemuž se věnuje tato bakalářská práce.

Jako základ pro tvorbu GUI bylo vybráno rozhraní obsahující konfigurační XML soubor, především díky jednoduššímu způsobu testování, snazšímu vývoji rozhraní a svobodné volbě programovacího jazyka. GUI tedy vygeneruje XML soubor a spustí program pro zápis hodnot do paměti karty, čímž zaktivuje generování paketů.

Požadavkům, které by měla uživatelská rozhraní splňovat, rozboru nejpoužívanějších rozhraní a detailnějšímu popisu GUI se věnuje kapitola 1. Ve 2. kapitole je detailněji popsána aplikace, ke které se rozhraní bude vytvářet spolu se stávajícími rozhraními. 3. kapitola se věnuje návrhu GUI a jeho implementace následuje v kapitole 4. Kapitola číslo 5 obsahuje testování rozhraní. Na závěr je vyhodnocena celá práce s ohledem na budoucí rozvoj.

Kapitola 1

Uživatelská rozhraní

1.1 Požadavky na uživatelská rozhraní

Jelikož technika je dnes již nesrovnatelně rychlejší než člověk, nastává otázka, zdali se zaměřit na zrychlování programů a hardwarových součástek v řádu milisekund, či na uživatelská rozhraní těchto aplikací, kde je potenciál zrychlení mnohonásobně vyšší. Rozvoj je samozřejmě potřebný v obou směrech, avšak z hlediska možného zrychlení vyplývá, jakou důležitost hraje rozhraní programu.

Uživatelské rozhraní je pomalé místo na cestě komunikace mezi uživatelem a programem. Zároveň je i kritické z pohledu volby zákazníka mezi různými konkurenčními produkty. Jeho vzhled musí být pro uživatele atraktivní, ovládání intuitivní a přehledné. Musí se přitom ale stále zachovávat požadovaná funkcionalita. Požadavků, které musí rozhraní splňovat, je tedy několik.

1.1.1 Obecné požadavky na rozhraní

Na základě rozdělení požadavků na rozhraní dle [11] jsou v následujících odstavcích rozebrány požadavky, mezi něž patří:

- Konzistence
- Orientace na více skupin uživatelů
- Zpětná vazba
- Navigace
- Předcházení chybám
- Možnost zpět
- Intuitivní ovládání
- Přehlednost

Konzistence

Dodržuje-li rozhraní zavedené stereotypy z oblasti klávesových zkratek (například `Ctrl+C` zkopíruje označený text), vzhledu ikon (křížek pro zrušení akce), menu a dalších ovládacích prvků, získá program pro zákazníka dojem, že když jej spustí, bude se při jeho ovládání cítit jistě, laicky řečeno – bude vědět, kam kliknout.

Pracovat s programem způsobem, kdy se každá věc provádí jinak, je velmi neefektivní a může být uživateli na obtíž.

Orientace na více skupin uživatelů

Je více než vhodné navrhnout UI dle skupin uživatelů, kteří jej budou používat. Počítačově zdatný programátor bude mít jiný přístup k rozhraní, bude jej ovládat jinými způsoby a vyžadovat jiné věci, než žák základní školy, či uživatel staršího věku se zrakovými problémy.

V klíčových momentech návrhu rozhraní je tedy potřeba zaměřit se na nejpravděpodobnější skupinu uživatelů a té rozhraní přizpůsobit. Je však rozumné do návrhu přidat i další způsoby ovládání rozhraní, které zohledňují ostatní potenciální uživatele a vytvořit tak aplikaci, která je použitelná pro širší spektrum uživatelů.

Zpětná vazba

V rámci komunikace uživatele a programu je důležitá zpětná vazba. Konkrétně se může jednat o potvrzování provedení jednotlivých akcí, informace o úspěšném, či neúspěšném provedení nebo kontrola zadaných vstupních hodnot například v textovém poli.

V ideálním případě je vyvážen poměr silné a slabé zpětné vazby, jenž zajistí efektivitu zároveň s informovaností uživatele. Silnou zpětnou vazbou rozumějme situaci, kdy uživatel musí na zpětnou vazbu reagovat, přičemž u slabé zpětné vazby nikoliv.

Navigace

Splněním tohoto požadavku vede výsledná aplikace uživatelskou akci krok po kroku, tedy rozdělí ji na části, čímž ji zpřehlední a zjednoduší. Zároveň je umožněn pohyb mezi jednotlivými kroky a uživatel vždy ví, v jaké části se nachází. Příkladem z praxe může být nákup letenek on-line, kde si uživatel nejdříve vybere destinaci, poté datum a čas, následuje formulář pro pojištění a doprovodné služby a v posledním kroku zaplacení letenky.

Předcházení chybám

Především z hlediska efektivity provádění jednotlivých akcí je vhodné snažit se předcházet chybám. Může se jednat o informovanost uživatele a způsob jejího jazykového vyjádření. Dalším příkladem je zakazování některých položek v dané kombinaci oproti zpětné chybové hlášce, že akci v této kombinaci provést nelze.

Možnost zpět

Pro pohodlí uživatelů, ale zároveň také pro efektivitu jejich práce, obsahuje dobrý návrh rozhraní možnost vrátit provedenou akci, zopakovat ji, či ji přerušit. Jedním ze způsobů, jak výše zmíněné provést, je ukládání a načítání aktuální konfigurace známé před vykonáním akce, stornovací tlačítko při vykonávání akce a tlačítko zpět pro navrácení aplikace do předchozího stavu.

Přehlednost

Dalším bodem, který musí rozhraní splňovat, je přehlednost. Programátor může implementovat bezvadnou aplikaci se spoustou funkcí, aplikaci rychlou, graficky pěknou, avšak ztratí-li rozhraní přehlednost, ztratí se i zájem zákazníků. Ať už třeba jen kvůli potřebným školením pro tento program.

S přehledným rozhraním tyto starosti odpadají, uživatel cítí, že má aplikaci pod kontrolou, zná ji a bez nějakých obtíží si umí vyhledat pomoc v případě, že některé věci nefungují dle jeho přání. Zároveň si nemusí pamatovat, jak rozhraní ovládat.

Intuitivní ovládání

Splnění tohoto bodu, tedy předvídatelného rozhraní, je důležité hlavně z toho hlediska, že se neplýtvá časem uživatele na učení se práce s programem. Čas je naopak produktivně využit k samotné práci.

1.1.2 Vlastní požadavky na rozhraní

K výše zmíněným požadavkům ještě přidám požadavky vlastní, jimiž jsou:

- Rychlost
- Jednoduchost
- Funkcionalita
- Aplikace, ke které je rozhraní tvořeno

Rychlost

Rychlostí UI je myšlena jak odezva při ovládacích akcích, tak množství potřebných úkonů pro vytvoření jedné akce.

Dozajisté by nebyli uživatelé spokojeni, pokud by se program dlouze načítal, při každém kliknutí by museli čekat a zvlášť při časově stresových situacích by je program mohl zradit a oni by díky němu měli problémy.

Velké množství dialogových oken pro nastavení všemožných atributů (například atributy vkládané fotografie), která zdržují samotné vložení, také není ideálním přístupem. Tento problém je řešitelný mnohem elegantněji, a to třeba tak, že vložení fotografie je podmíněno velikostí, pozicí a názvem zdrojového souboru. Další atributy jsou pak nastavitelné z menu, či nástrojových panelů.

Jednoduchost

Existuje spousta úspěšných programů, které nemají jednoduché rozhraní. Jejich úspěch tkví v dalších bodech a také v tom, že se jejich rozhraní a funkce postupně vyvíjely a uživatelé si na daný přístup mohli zvyknout.

Při tvorbě nové aplikace, či jejího prvního rozhraní, je však důležité, aby bylo UI jednoduché a nejlépe si svou jednoduchost na úkor přidávání nové funkcionality zachovalo. První pohled zákazníka na program je nesmírně vlivný při volbě mezi konkurencí, a pokud uživatel uvidí tisíce nic neříkajících ikon, řádků a panelů, automaticky si vytvoří představu o nutnosti studia rozhraní, která ho může velmi jednoduše odradit.

Funkcionalita

U tohoto pohledu neplatí pravidlo - čím více funkcí program nabízí, tím je lepší. Jelikož by se mohlo zdát toto tvrzení zvláštní, uveďme si pár důvodů.

Prvním z nich je nadbytečná funkcionalita. Tím je míněno, že například textový editor nemusí zvládat pokročilé úpravy obrázků tak, jak to umí profesionální grafický program.

Dalším může být omezení funkcí na úkor množství tlačítek. Ke každé funkci je totiž potřeba se nějakým způsobem dostat a většinou to s sebou nese nároky na ovládání. Ať už je to tak, že hlavní menu bude mít až přehnaně velké množství prvků, nebo tak, že hardwarové zařízení musí být zvětšeno, aby se na něj vešla všechna potřebná tlačítka.

Aplikace, pro kterou je rozhraní tvořeno

V tomto případě je vazba oboustranná. Aplikace ovlivňuje UI nejen funkcemi, které poskytuje, ale i způsobem reprezentace dat, použitého programovacího jazyka, knihoven, rychlostí provádění výpočtů a dalších. Rozhraní naopak prodává aplikaci, dělá ji přístupnou veřejnosti a z pohledu aplikace dokáže jednoduše reprezentovat složité věci.

1.2 Typy uživatelských rozhraní

Existuje více druhů uživatelských rozhraní, každé s jinými vlastnostmi a vhodností použití pro různé typy programů. Rozbor bude brát v úvahu historický vývoj (nejpoužívanější rozhraní), komunikační kanály (lidské smysly) a dělení na základě směru komunikace (dle [11]).

1.2.1 Komunikace od uživatele k programu

Obraz

Tento způsob komunikace na cestě od uživatele k programu se používá především u kamerových systémů, které snímají například anomálie chování lidí pro prevenci teroristických útoků, nebo pro hraní her, které využívají rozpoznávání pohybu. Zajímavým využitím může být také snímání obrazu termovizní kamerou například pro identifikaci nakažených lidí se zvýšenou teplotou na letištích.

Zvuk

Historicky druhé nejpoužívanější rozhraní, které však skýtá jednu důležitou nevýhodu, a to zpracování vstupních informací. Lze je zpracovávat člověkem, který přeloží lidskou řeč na specifické ovládání programu, či strojově pomocí rozpoznávání řeči.

První varianta vyžaduje zaměstnat lidskou sílu, což se dnes už jeví jako nepoužitelné, respektive neschopné konkurence.

Na poli zadávání vstupních informací je největším konkurentem pro hmatová zařízení strojové rozpoznávání řeči. Zatím není na tak vysoké úrovni, aby se mohlo bezproblémově používat, avšak výhled do budoucna je v tomto směru pozitivní. I se stoprocentní úspěšností rozpoznání slov však není zaručeno, že tato cesta bude ve všech případech tou nejrychlejší pro zadání konkrétního příkazu, nicméně bude pravděpodobně pohodlnější.

Hmat

Z historického hlediska doposud nejpoužívanější způsob zadávání příkazů pro řízení běhu programu. Mezi první používaná zařízení patří klávesnice, převratnou novinkou se stala myš a časem její alternativy (trackbally, touchpady a trackpointy). S rozvojem mobilních zařízení se do hry dostávají chytré telefony s dotykovou obrazovkou.

Výhodou je, při dobře navrženém rozhraní, rychlost zadávání příkazů (pár kliknutí myší, či použití klávesových zkratek). Nevýhoda je nutnost použití zařízení, které ne vždy člověk může použít (myš na malém prostoru, dotyková obrazovka v rukavicích, a tak dále).

Čich, chuť

Z hlediska náročnosti digitalizace a nároků na periferní zařízení jsou tyto komunikační kanály nekonkurenceschopné.

1.2.2 Komunikace od programu k uživateli

Obraz

Bezpochyby nejpoužívanější rozhraní. Existuje spousta podtypů tohoto přístupu, přičemž začátky jsou u příkazové řádky (CLI – Command Line Interface), kdy uživatel ovládá program na základě klíčových slov v textové podobě. Dalším milníkem pak bylo textové rozhraní (TUI – Text User Interface), které je vykresleno do různých tvarů (okna, menu) pomocí znakové sady, například z ASCII tabulky pro kódování znaků ve výpočetní technice.

Nejdůležitějším a zároveň nejčastějším rozhraním je dnes grafické uživatelské rozhraní (GUI – Graphical User Interface), jehož rozboru je věnována celá kapitola 1.3.

Zvuk

Je používaný především v kombinaci s GUI jako podpůrný prostředek (například navigace do aut). Čistě zvuková prezentace se objevuje především v případech, kde obrazová informace přenášena být nemůže (provozně náročné na přenos dat), nebo nemusí (například v telefonii). V programech, které běží na zařízeních s možností obrazového výstupu se v hojné míře nepoužívá.

Hmat

Z hlediska hmatu se v současnosti používá tzv. Braillský řádek. Ten překládá informace do Braillova písma, které je jedním z důležitých komunikačních kanálů pro nevidomé. Tato periferie je však cenově náročná (v řádech statisíců korun).

Čich, chuť

Tyto dva typy jsou spíše z oblasti budoucnosti - pokud se vůbec někdy používat budou. Jejich vlastnosti je těžké digitalizovat, a tak se komplikuje i přenos informace. Nasazení v praxi lze nalézt například u 4D filmů, které jako pomyslný čtvrtý rozměr přidávají právě smyslové vjemy.

1.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní hraje velkou roli při výběru aplikace v konkurenčním boji. Je to první dojem, který je v zákazníkovi vzbuzen, ať už v dobrém, či špatném slova smyslu. Kvalitním GUI lze nejenže nalákat zákazníky, ale také si je udržet. Způsobů, jak jej vytvořit, je několik, i když mají stejný cíl – uspokojit co nejvíce zákazníků.

1.3.1 Kvalita

Kvalitu lze ovlivnit spoustou způsobů, ať už splněním požadavků na UI zmíněných v podkapitole 1.1 nebo volbou barevných motivů, ikon a dalších věcí popsaných v následujících sekcích. Tyto aspekty se mohou stát velkou výhodou oproti konkurenčním UI, avšak pouze při jejich správném použití.

Volba barev

Silnou informační hodnotu s sebou nesou barvy. Ať už se jedná o zelenou, která člověku podvědomě stimuluje souhlas, tak červenou, reprezentující nesouhlas. Je vhodné těchto znalostí využít a vzhled chybových hlášek, či celých dialogových oken jim přizpůsobit.

Dalším důležitým krokem je celkový barevný vzhled aplikace. Obecně úspěšné doporučení, jaké barvy používat, neexistuje. Co však existuje, jsou rady pro vzhled zaměřující se na kontrast písma a jeho pozadí, či velikost písma tak, aby bylo možno text pohodlně vidět a přečíst. Zbytek je již na grafickém citu člověka, který se stará o vzhled aplikace.

Ikony

Zvláště u tlačítek, která za sebou skrývají určitou funkci, je vhodné použít ikony vysvětlující jejich význam. Ty mohou být na tlačítku s textem, ale i bez něj. U samotného textu je problém s jazykem, tedy, že ne každý uživatel by mohl pochopit, či znát, konkrétní slovo. Pokud se však použije například obrázek noty na tlačítko, které má co do činění s hudbou, lze bez jazykových bariér jednoduše reprezentovat jeho význam.

Menu

Položky menu by měly zpřístupňovat veškerou funkcionalitu programu tak, aby uživatel neznalý prostředí mohl jednoduše zadat příkaz, který chce vykonat. Je také výhodné a často používané označovat v menu jednotlivé položky zároveň s jejich klávesovými zkratkami a je-li to možné, přiřazovat každé položce menu vysvětlující ikonu.

Kapitola 2

Generátor paketů

Aplikací, pro kterou bude vytvořeno grafické uživatelské rozhraní, je generátor paketů. Je implementovaný na akcelerační PCIe kartě s FPGA čipem (více na [1]), čímž získává výhody softwarových i hardwarových generátorů síťového provozu. Je cenově dostupný a zároveň rychlý.

Program je sestaven z modulů a má čtyři operační módy:

- Generování umělého síťového provozu
- Standardní operace síťového rozhraní
- Načítání dat do dynamické paměti
- Vysílání dat z dynamické paměti

První mód umožňuje generovat umělý síťový provoz s podporou verze protokolu IPv4 i IPv6 a umí využít plnou přenosovou rychlost média.

V druhém módu je většina modulů aplikace vypnuta nebo využita jen minimálně a aplikace nezasahuje do standardního běhu síťového rozhraní. Používá se pro ukládání a přehrávání síťového provozu pomocí externích programů (například `tcpdump` [9] a `tcpreplay` [10]).

Funkčnost posledních dvou módů prozatím nebyla v aplikaci implementována, avšak možnost tyto módy zvolit je v aplikaci ponechána. Je-li zvolen jeden z těchto dvou módů, program se chová jako v módu standardních operací síťového rozhraní.

2.1 Množina registrů

Všechny potřebné kontrolní a stavové informace jsou uloženy v tomto modulu, v [3] nazývaném Register Control and Status File. Velikost každého registru je 32 b a vždy spadá do jedné z následujících skupin:

- Hlavní množina registrů
- Registry pro generování polí IPv4 hlavičky
- Registry pro generování polí IPv6 hlavičky

Pokud není 32 b prostor plně využit, jsou zbylé bity nastaveny na 0. Způsob vkládání hodnot do registrů je zmíněn v podkapitole 2.2.

2.1.1 Hlavní množina registrů

Níže uvedené registry ovlivňují proces generování jako takový. Patří sem nastavení omezení provozu, počet generovaných paketů, MAC adres, přenosové rychlosti a volba operačního módu.

Kontrolní registr

Tento registr obsahuje na šesti nejméně významných bitech informace ohledně řízení generování. Ostatní bity nejsou použity a jsou nastaveny na 0. Obsah a význam bitů tohoto registru je znázorněn v tabulce 2.1 převzaté z [3]. Ve chvíli, kdy je zapsána konfigurace do tohoto registru, spustí se samotná aplikace.

Název	Bity	Hodnoty	Význam	Výchozí
Operační mód	0–1	00 01 10 11	standardní operace síťového rozhraní načítání dat do paměti vysílání dat z paměti generování síťového provozu	✓
Mód omezující generování paketů	2–3	00 01 10 11	bez limitování omezení na konkrétní přenosovou rychlost omezení založené na 64 b časových známkách rezervováno pro budoucí užití	✓
IPv4/IPv6	4	0 1	generování IPv4 hlaviček generování IPv6 hlaviček	✓
DMA/NET	5	0 1	načítání dat z počítače načítání dat ze sítě	✓

Tabulka 2.1: Kontrolní registr [3]

Počet generovaných paketů

Ve výchozím nastavení nabývá tento registr hodnot 0 na všech svých bitech. Tato speciální hodnota má význam neomezeného počtu generovaných paketů. V jiném případě určuje hodnota registru počet generovaných paketů, nejvíce pak $2^{32} - 1$.

MAC adresy

Díky 48 b délce MAC adresy musí být jak zdrojová, tak cílová adresa uloženy ve dvou 32 b registrech. Jedná se o následující registry:

1. Spodní část zdrojové MAC adresy
2. Vrchní část zdrojové MAC adresy
3. Spodní část cílové MAC adresy
4. Vrchní část cílové MAC adresy

Registry pro spodní část adresy (1. a 3.) jsou využity plně. Registry pro vrchní část adresy (2. a 4.) jsou využity jen z poloviny, konkrétně na 16 spodních bitech.

Vzor datové části paketu

Ve výchozím nastavení nabývá všech 8 bitů, které se v tomto registru používají, hodnoty 0. Hodnota na těchto 8 nejméně důležitých bitech určuje pro pole Payload vzor generování. Ten je využit v modulu pro sestavování paketů na naplnění paketu daty. Více informací o způsobu sestavování paketů je k nalezení v [3].

Přenosová rychlost

Tento registr může nabývat hodnot v rozsahu 1–10 000, které určují na jakou přenosovou rychlost v Mb/s má být nastaveno vysílání do sítě. Hodnotu tohoto pole využívá modul limitující generování paketů pro generování na zadanou přenosovou rychlost. Všechny hodnoty mimo výše zmíněný rozsah jsou interpretovány jako maximální možná hodnota.

Způsob generování polí hlavičky

Způsoby, kterými jsou pole hlaviček generovány, jsou tři:

- Konstanta
- Další hodnota z posloupnosti čísel
- Náhodná hodnota

Pro každé pole hlavičky paketu je určena 2 b hodnota, která rozlišuje výše uvedené způsoby. Pro konstantu je to hodnota `00`, pro další hodnotu z posloupnosti čísel je to `01` a pro náhodnou hodnotu `10`.

Tato data jsou uložena v registru pro způsob generování polí hlavičky na pozicích dle následující tabulky 2.2 převzaté z [3]:

Název	**Pozice (bity)**
IPv4 Typ služby (ToS)	0–1
IPv4 Celková délka datagramu	2–3
IPv4 Příznaky	4–5
IPv4 TTL	6–7
IPv4 Číslo protokolu	8–9
IPv4 Adresa zdroje	10–11
IPv4 Adresa cíle	12–13
IPv6 Třída provozu	14–15
IPv6 Značka toku	16–17
IPv6 Délka dat	18–19
IPv6 Maximum skoků (Hop limit)	20–21
IPv6 Adresa zdroje	22–23
IPv6 Adresa cíle	24–25
Rezervováno pro budoucí použití	26–31

Tabulka 2.2: Rozložení hodnot pro způsob generování hlavičky [3]

2.1.2 Registry pro generování polí IPv4 a IPv6 hlavičky

V těchto dvou množinách registrů jsou uložena veškerá data potřebná pro generování hlaviček paketů verze IPv4, respektive IPv6. Seznam potřebných skupin registrů pro IPv4 i IPv6 hlavičky byl již uveden v tabulce 2.2.

V každé skupině registrů je potřeba ukládat hodnoty *od*, *do* a *velikost kroku* pro dostatečné informace k jejich generování. Proto se každá skupina těchto registrů skládá ve skutečnosti z 3 registrů určujících tyto hodnoty. Výjimku tvoří registry pro zdrojovou a cílovou IPv6 adresu, protože mají délku 128b. Výsledné řešení tohoto problému dedikuje 12 registrů pro IPv6 adresy - 4 registry pro hodnotu *od*, 4 registry pro hodnotu *do* a další 4 pro hodnotu *velikost kroku*. Podrobnější popis rozložení jednotlivých bitů v registrech je detailně popsán v sekci 4.3.5 v práci [3].

V následujících tabulkách 2.3 a 2.4 převzatých z [3] je přehled možných hodnot jednotlivých polí IPv4 a IPv6 hlaviček.

Název	Velikost (v bitech)	Pozice	Typ	Hodnota
Verze	4	0–3	konstanta	4
IHL	4	0–3	konstanta	5
Typ služby	8	0–5 6–7	generováno konstanta	$0–2^6-1$ 0
Celková délka	16	0–15	generováno	21–1 500
Identifikace	16	0–15	konstanta	0
Příznaky	3	0 1 2	konstanta generováno konstanta	0 0–1 0
Fragmentovací offset	13	0–12	konstanta	0
TTL	8	0–7	generováno	$1–2^8-1$
Protokol	8	0–7	generováno	$0–2^8-1$
Kontrolní součet	16	0–15	konstanta	0
Zdrojová adresa	32	0–32	generováno	$0–2^{32}-1$
Cílová adresa	32	0–32	generováno	$0–2^{32}-1$

Tabulka 2.3: Možné hodnoty polí hlaviček paketů verze IPv4 [3]

Název	Velikost (v bitech)	Typ	Hodnota
Verze	4	konstanta	6
Třída provozu	8	generováno	$0–2^8-1$
Značka toku	20	generováno	$0–2^{20}-1$
Délka dat	16	generováno	1–1 460
Další hlavička	8	konstanta	0
Maximum skoků	8	generováno	$1–2^8-1$
Zdrojová adresa	128	generováno	$0–2^{128}-1$
Cílová adresa	128	generováno	$0–2^{128}-1$

Tabulka 2.4: Možné hodnoty polí hlaviček paketů verze IPv6 [3]

Protože některé části polí Typ služby a Příznaky mohou být generovány a některé ne, je v tabulce 2.3 sloupec Pozice pro specifikování bitového rozsahu pole. Další zvláštností může být pohled na hodnotu Kontrolní součet jako konstantu. Hodnota tohoto pole je počítána v submodulu Packet Completer a algoritmus pro jeho počítání vyžaduje počáteční hodnotu tohoto pole nastavenou na nulu.

Také rozsah hodnot pro pole Celková délka je potřeba objasnit. Hodnota předpokládá data paketu o délce nejméně 1 B. Jelikož hlavička IPv4 zabírá 20 B, pak minimální hodnota pole Celková délka se rovná 21 B. Maximální délka paketu je omezena velikostí ethernetového rámce na 1 500 B.

Hodnota TTL v IPv4 (obdobně Maximum skoků u IPv6) značí dobu životnosti paketu. V praxi každý směrovač na cestě paketu toto číslo dekrementuje o 1. Přímé spojení dvou počítačů tedy vyžaduje hodnotu TTL minimálně 1, jinak je paket zničen. To z hlediska logiky tohoto pole implikuje tuto minimální hodnotu pole TTL právě na číslo 1.

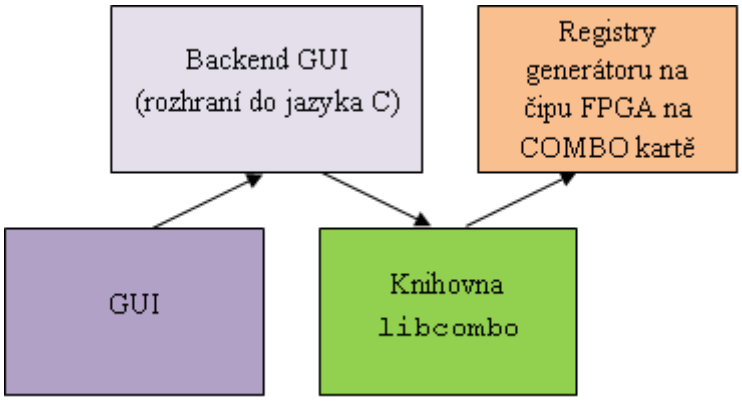
Rozsah hodnot pole IPv6 hlavičky s názvem Délka dat uvažuje velikost IPv6 hlavičky 40 B a data o délce nejméně 1 B. Jelikož toto pole určuje pouze velikost dat bez hlavičky (rozdíl oproti poli Celková délka u IPv4), pak minimální hodnota je 1 B a maximální hodnota je 1 500 B−40 B= 1460 B, kde 1 500 B je maximální velikost ethernetového rámce a 40 B je velikost IPv6 hlavičky.

2.2 Stávající rozhraní aplikace

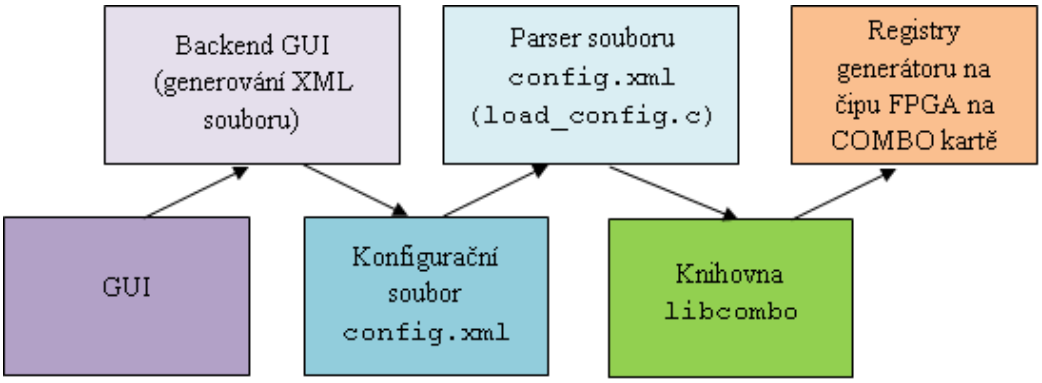
Cílem této práce je vytvořit grafické uživatelské rozhraní pro generátor paketů. Jelikož GUI musí umět s aplikací nějakým způsobem komunikovat, je potřeba znát rozhraní aplikace. V současné době existují dva přístupy, jak tuto komunikaci realizovat.

Nejpřímější cesta je pomocí volání funkcí knihovny `libcombo`, které generování zajišťují nastavením hodnot do příslušných registrů. Tato cesta zahrnuje detailní znalost těchto funkcí, jejich kombinaci a přinejmenším základy programování v jazyce C. Diagram komunikace entit tohoto systému je zobrazen na obrázku 2.1.

Druhou variantou je rozhraní používající konfigurační soubor ve formátu XML. Ten je zpracován programem `load_config` a na základě hodnot XML elementů je nastavena interní reprezentace jednotlivých registrů aplikace. Zápis této reprezentace do skutečných registrů je realizován opět funkcemi knihovny `libcombo`. Obrázek 2.2 ilustruje komunikaci jednotlivých prvků tohoto systému.



Obrázek 2.1: Rozhraní generátoru paketů bez použití XML souborů



Obrázek 2.2: Rozhraní generátoru paketů používající XML soubory

2.2.1 Soubory ve formátu XML

Společně s rozhraním používajícím XML soubory jsou k dispozici dva soubory tohoto typu. Jedním z nich je `config.xml`, který je vzorovým konfiguračním souborem. Druhým je `config_spec.xml`, který obsahuje definici hodnot atributů. Ukázkové obsahy obou souborů jsou uvedeny v přílohách B.1 a B.2.

Rozbor souboru `config.xml`

`config.xml` obsahuje konkrétní hodnoty příslušných atributů jednotlivých registrů. Z hlediska struktury dokumentu je element `<configuration>` kořenovým elementem, který obsahuje tři množiny registrů reprezentovanými elementy `<general_registers>`, `<ipv4_registers>` a `<ipv6_registers>`. Ty v sobě obsahují další registry (například `<control_register>`), jejichž atributy (`<operation_mode>`) mohou nabývat různých hodnot (`"nic"`).

Rozbor souboru `config_spec.xml`

Jak již bylo zmíněno, soubor obsahuje definici hodnot atributů. Obecně se zde objevují dva druhy této definice – výčet a číslo.

Jak je znázorněno v B.2, hodnota atributu `generated_ip` kontrolního registru je specifikována právě výčtem. Konkrétně tento atribut může nabývat hodnot `"ipv4"` a `"ipv6"`.

Hodnota atributu `value` u registru zdrojové MAC adresy je naopak definována jako číslo.

Tento soubor navíc obsahuje vyčerpávající popis registrů a jejich povolených hodnot, zvláštnosti registrů a v některých případech vzorová vstupní data.

2.2.2 Funkce knihovny libcombo

Pro potřeby aplikace, tedy zápis hodnot do daných registrů (na určité místo v paměti), je zapotřebí funkcí knihovny `libcombo` (více na [2]), které jsou volány v obou typech rozhraní. U souboru XML, jsou pouze odstíněny od uživatele. Níže následuje jejich podrobnější popis.

```
int cs_attach (cs_device_t **dev, const char *dev_path)
```

Funkce slouží k připojení karty k danému procesu. Implementuje výlučný přístup a vyžaduje dva parametry.

První parametr slouží k uložení informací o připojeném zařízení. Druhý se používá pro identifikaci cesty k zařízení. Zároveň nastavuje tuto cestu do struktury zadané předchozím parametrem a ostatní funkce již nadále komunikují přes strukturu zadanou parametrem prvním.

```
int cs_space_map (cs_device_t *dev, cs_space_t **space, cs_target_t target,
cs_size_t mem_size, cs_addr_t mem_base, u_int32_t attrib)
```

Úkolem této funkce je alokovat buffer pro vykonávání zápisových a čtecích operací na kartu.

První parametr typu je struktura z předchozí funkce sloužící k přístupu k základním informacím o zařízení. V pořadí druhý obsahuje informace o namapovaném prostoru. Třetí parametr určuje typ čipu, v našem případě vždy FPGA. Čtvrtý zadává velikost paměťového prostoru pro namapování v bajtech. Pátý parametr identifikuje bázovou adresu bloku paměti v paměťovém prostoru určeném pro dané zařízení. Šestý, a zároveň poslední, parametr slouží pro definici dodatečného atributu, ale v našem případě využit není.

```
void cs_space_write_4 (cs_device_t *dev, cs_space_t *space, u_int32_t address,
u_int32_t data)
```

Poslední použitá funkce zapisuje data o velikosti 32 bitů na základě hodnot z předchozích dvou funkcí.

První dva parametry jsou odkazy na struktury z předchozích funkcí, obsahujících informace o zařízení a namapovaném prostoru. Třetí je adresa, kam se data zapíší a čtvrtý jsou konkrétní data.

2.2.3 Volba mezi stávajícími rozhraními

Výhodou prvního přístupu je rychlost a vynechání jednoho kroku (XML souboru) na cestě od příkazu uživatele k samotné akci. Nevýhodou je omezení z hlediska použitého programovacího jazyka, který lze zvolit pro tvorbu GUI. Některé programovací jazyky implementují rozhraní do jiných jazyků, avšak nese to s sebou určitá rizika.

Výhodou použití druhého přístupu je čitelný textový formát vstupu. Díky své jednoduchosti pak toto rozhraní umožňuje rychlejší implementaci nástavbových modulů, či aplikací. V neposlední řadě pak uživatel nemusí znát jazyk C. Nevýhodou je nutnost zřetězení jednotlivých procesů (akce UI zapsat do souboru XML a na základě jeho změny volat program obsahující funkce pro zápis hodnot atributů objektů z XML souboru na kartu).

Pro tvorbu GUI byl zvolen druhý přístup, především z důvodu snazšího testování rozhraní a svobodné volby programovacího jazyka. Zřetězení procesů bude probíhat pomocí volání funkce, která se postará o spouštění externích programů z příkazového řádku.

Kapitola 3

Návrh

Jednou z nejdůležitějších součástí životního cyklu uživatelského rozhraní je jeho návrh. Odvíjí se od něj celková interakce s uživatelem, styl, jakým budou informace prezentovány, závisí na něm rychlost zpracování jednotlivých akcí a v neposlední řadě řeší design. Požadavky, které UI musí splňovat, byly uvedeny v podkapitole 1.1 a zde budou aplikovány na konkrétní problém, tedy GUI pro generátor paketů. Zároveň bude v sekci 3.4 naznačeno grafické rozmístění jednotlivých prvků uživatelského rozhraní tak, jak by mělo být naimplementováno.

3.1 Cílová skupina uživatelů

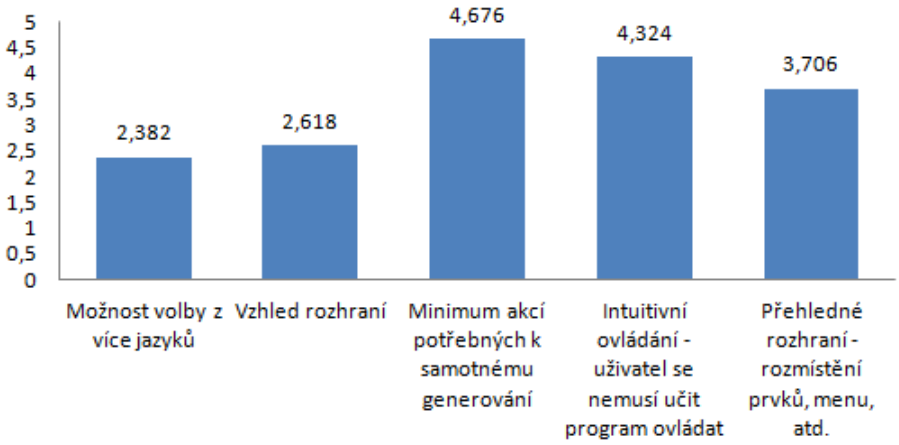
Prvním důležitým faktem, který je třeba brát v úvahu, je skupina uživatelů, pro kterou bude rozhraní k aplikaci tvořeno. Z obecného hlediska platí, že jiné rozhraní bude vyvinuto pro programy, které mezi sebou komunikují, pro lidi, kteří mají smyslové indispozice, či pro děti.

Každá ze zmíněných skupin reprezentuje jiný pohled uživatele na rozhraní a upřednostňuje různé vlastnosti. Cílovou skupinou pro GUI generátoru paketů jsou uživatelé znalí počítačového prostředí, správci sítí, programátoři a další lidé na podobné úrovni práce s počítačem. V následujících sekcích jsou zmíněny požadavky, které tato skupina nejčastěji reprezentuje.

3.1.1 Požadavky na rozhraní ovlivněné cílovou skupinou uživatelů

Na obrázku 3.1 je graf znázorňující výsledky průzkumu ohledně požadavků na GUI ke generátoru paketů ze strany potenciálních uživatelů. Tento dotazník byl vyplněn 34 lidmi z oblasti IT, mezi něž patří studenti Fakulty informačních technologií Vysokého učení technického v Brně, Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně, programátoři zaměstnáním a správce sítě. Korespondenti v dotazníku hodnotili každý požadavek na stupnici od 1 do 5, kdy číslo 5 reprezentovalo nejvyšší prioritu požadavku a číslo 1 nejnižší. Uživatelé mohli udělit i více stejných známek různým požadavkům.

Nejdůležitějšími faktory se ukázaly být minimum akcí nutných pro samotné generování, intuitivní ovládání, které nenutí uživatele učit se pracovat s rozhraním a přehledné rozestavení prvků. Ostatní požadavky plynoucí z obecných doporučení uvedených v kapitole 1 doplňují tyto specifické. S vědomím těchto informací a požadavků je tedy potřeba rozhraní navrhnout.



Obrázek 3.1: Průzkum požadavků na GUI pro generátor paketů

3.1.2 Zapracování specifických požadavků do návrhu rozhraní

Minimum akcí nutných pro samotné generování je v návrhu zohledněno ve více případech. Jedním z nich jsou pouhá 3 okna reprezentující úrovně zanoření z pohledu generování. První oknem je *volba aplikačního módu*. Druhým oknem je *nastavení obecných hodnot*, kam patří způsob generování hodnot registrů (konstanta, sekvence čísel, náhodné číslo), nastavení MAC adres, počtu generovaných paketů, verze protokolu IP a dalších. Třetí okno se věnuje *nastavení konkrétních hodnot pro hlavičky IP datagramů*.

Aby se ušetřilo akcí uživatele (kliknutí myši, vyplnění textového pole, ...), budou jednotlivé hodnoty při spuštění aplikace již přednastaveny ve správném formátu. K rozlišení, zda byly zadány uživatelem, či aplikací, poslouží barvy. Uživatel tedy uvidí, které hodnoty změnil a které ponechal. Tento přístup také napoví, v jakém formátu mají být očekávaná data zadána a omezí jejich nechtěnou desinterpretaci.

Intuitivní ovládání je v návrhu reprezentováno používáním stereotypů. Ať už se jedná o klávesové zkratky (Ctrl+C pro kopírování, atd.) a používání elementárních prvků rozhraní, jako jsou přepínače, zaškrtávací políčka a textová pole. Rozhraní uživatele vede v jednotlivých krocích (tlačítko další) a snaží se ve většině případů přidat vysvětlující text, který doplňuje jednotlivé ovládací prvky.

Přehledné rozestavení prvků je důležité pro orientaci uživatele v jednotlivých oknech. Návrh dodržuje geometrické předpoklady, kdy prvky logicky náležící do jedné skupiny, jsou zarovnány stejně a tuto skupinu tím graficky reprezentují.

3.2 Poskytované funkce rozhraní

Další nedílnou součástí rozhraní jsou jeho funkce, které lze implementovat na základě funkcí samotné aplikace. Ty v některých případech a některých kombinacích poskytují zajímavé, a uživatelsky často vítané, možnosti.

V našem případě se jedná o mapování hodnot jednotlivých registrů na příjemnější způsob zadávání. Tuto problematiku řeší textová pole, přepínače a další kontrolní prvky rozhraní, které jsou k tomu přímo určené. Formát vstupních hodnot je zvolen dle zažitých stereotypů, což usnadní celkovou práci s aplikací (IPv4 adresa v desítkové soustavě oddělena tečkami, apod.).

Přídavnými funkcemi (oproti základnímu mapování hodnot prvků GUI na hodnoty registrů) jsou možnost uložení a načtení konfigurace, jazykové mutace, různé vzhledy rozhraní, historie akcí a kontrola správnosti zadaných hodnot a jejich kombinací.

Rozhraní je navrženo tak, aby se v prvním kroku nastavily všechny potřebné hodnoty prvků rozhraní a v druhém je uživatel stiskem tlačítka zapsal do registrů. Jak již bylo zmíněno v kapitole 2, zápis do registrů způsobí samotné generování.

3.3 Vzhled aplikace

Barvy hrají velkou roli na psychiku uživatele, a tak jejich výběr není náhodný. Se zapojením vlastní zkušenosti vznikne více barevných kombinací, mezi kterými si uživatel bude moci vybrat. Půjde o nestylovanou původní šedou verzi, dále modro-zelenou a černo-červenou.

Během návrhu vznikly spekulace ohledně možnosti uživatele zvolit si například tři barvy, které rozhraní dominují a na jejichž základě se celý vzhled aplikace upraví. Tato možnost je nadstandardní a samotnému generování je vzdálená. Pokud však zbylé části rozhraní již budou implementovány, tímto směrem se bude ubírat další vývoj.

3.4 Prvky rozhraní

3.4.1 Menu

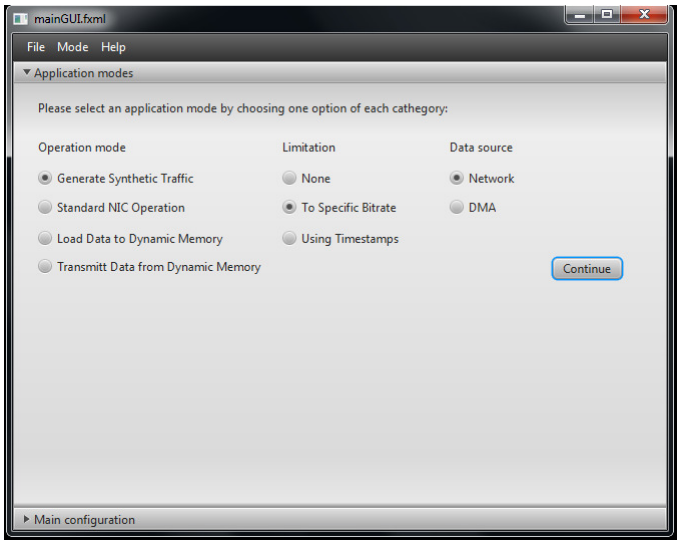
Menu je jedním z nejpoužívanějších stereotypů většiny aplikací. Vzhledem k zažité práci s ním, jej bude obsahovat i GUI pro generátor paketů. Jednotlivé položky budou odpovídat akcím pro uložení a načtení konfigurace, pohyb v historii, práci s nápovědou, změnou vzhledu rozhraní, jazyka a dalších věcí, které vyplynou při implementaci.

3.4.2 Dialogové okno pro volbu jazyka

První kontakt s aplikací nabídne dialogové okno pro výběr jazyka. Volba se zapíše do konfiguračního souboru a při dalším spuštění bude aplikace v požadované lokalizaci. Pozdější změnu bude možno provést přímo z menu.

3.4.3 Okno pro volbu aplikačního módu

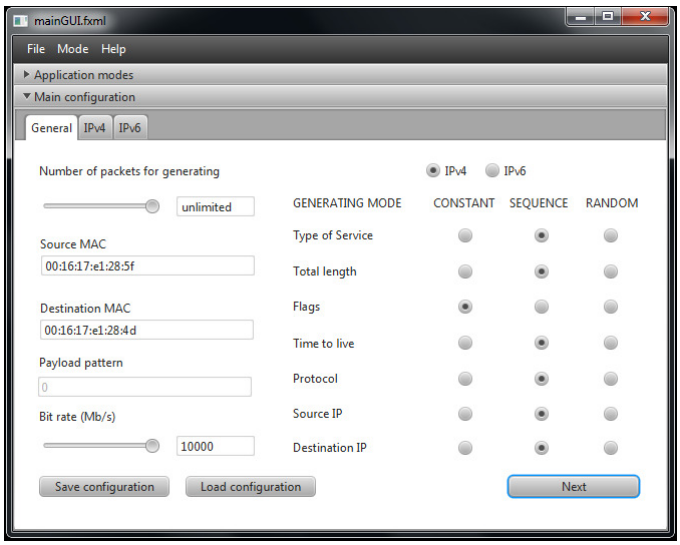
Po volbě jazyka se zobrazí výběr módu aplikace, jak je znázorněno na obrázku 3.2.



Obrázek 3.2: Okno pro volbu aplikačního módu

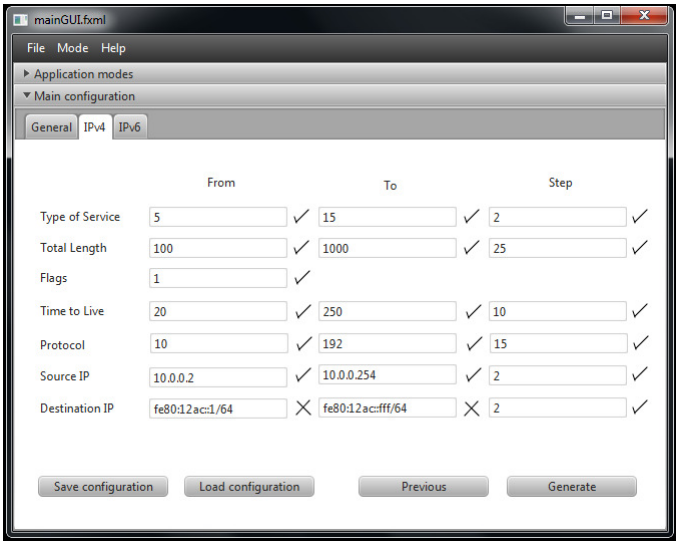
3.4.4 Okno hlavní konfigurace

Pomocí rozbalovacího seznamu se uživatel může pohybovat mezi oknem hlavní konfigurace a volbou aplikačního módu, jak je znázorněno na obrázku 3.3.

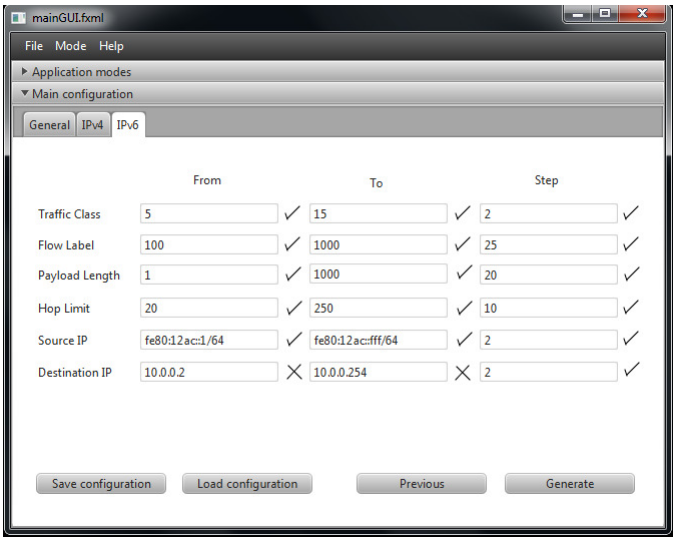


Obrázek 3.3: Okno pro hlavní konfiguraci

Okno pro hlavní konfiguraci zároveň obsahuje ještě další dvě karty, které dle zvolené verze protokolu IP budou viditelné, či nikoliv. Návrh těchto oken je na obrázcích 3.4 a 3.5.



Obrázek 3.4: Okno pro konfiguraci IPv4 hlavičky



Obrázek 3.5: Okno pro konfiguraci IPv6 hlavičky

3.5 Varianty vzhledu rozhraní

Během tvorby návrhu rozhraní vznikalo více možností, jak problém uchopit. Jedním z nich bylo jedno velké okno s veškerou konfigurací. Vzhledem k množství vyplňovacích polí a možných kombinací však vyhrála varianta znázorněná výše díky své názornosti, splněnému požadavku o navigaci a také šetřením místa.

Dalším přístupem byl postranní panel s předpokládanými nejpoužívanějšími prvky menu. Tento přístup se neujal především proto, že panel zabíral místo a ve skutečnosti neobsahoval tak důležité funkce, které by nebyly přístupné odjinud, a tak tyto prvky zůstaly v menu umístěném v horní části aplikace.

Výsledné zvolené rozmístění je tedy znázorněno na obrázcích 3.2, 3.3, 3.4 a 3.5.

Kapitola 4

Implementace

Předešlé fáze životního cyklu uživatelského rozhraní byly především teoretické. Teorie bez praxe však v případě tvorby rozhraní nepostačí. Nejpraktičtější částí vývoje je právě implementace.

Ideální stav představuje perfektní návrh, který je do posledního detailu zároveň naprogramován. Nežijeme však v ideálním světě, a tak některé body návrhu se během implementace ukáží nahraditelné lepším řešením a ne všechny body návrhu bývají nakonec implementovány. Velkou roli přitom hraje prostředí, které ovlivňuje možnosti programátora. Ať už pozitivně, či negativně.

Následující sekce se věnují popisu implementační části více dopodrobna se zaměřením na výslednou aplikaci.

4.1 Vývojové prostředí

Jak už bylo výše zmíněno, prostředí ovlivňuje možnosti programátora, a to nejen nástroji, které poskytuje program, v němž se kód píše, ale zároveň vlastnostmi programovacího jazyka. Program byl implementován pro prostředí Windows, je však připraven i pro nasazení v systémech Linux. Níže jsou popsány pojmy z oblasti vývojového prostředí, které byly při programování uživatelského rozhraní pro generátor paketů použity.

Základním kamenem vývojového prostředí byla distribuce JDK (Java Development Kit) verze 1.7, která v sobě obsahuje Javu SE (Second Edition) 7 spolu s JavaFX 2.2.21 a NetBeans 7.2.1.

Java FX je softwarová platforma založená na Javě. Díky tomuto faktu může programátor kódovat jak v jazyce Java, tak v jazyce JavaFX Script, který je syntakticky podobný Javasciptu. Při vývoji aplikace byla zvolena varianta první, především díky lepší znalosti tohoto jazyka.

Mezi největší výhody JavaFX patří (dle [6])

- Široká přenositelnost
- Podpora vývojářských nástrojů, které používá Java
- Široké spektrum existujících knihoven
- Použití FXML jazyka, jenž umožňuje přehledně a jednoduše vývoj komplexního uživatelského rozhraní
- Podpora běhu ve webových prohlížečích
- Podpora médií v běžných formátech
- Optimalizované vykreslování
- Podpora většiny prvků uživatelského rozhraní a změny jejich vzhledu pomocí CSS

Obzvláště poslední bod, tedy podpora CSS, byl využit téměř u všech prvků výsledného rozhraní a dodal celkový design aplikace.

Pro tvorbu FXML souboru definujícího mimojiné rozmístění prvků rozhraní byl použit program JavaFX Scene Builder [7] a na úpravu instalačního souboru program Inno Setup Compiler [8].

4.2 Vazby mezi módy

Důležitým bodem implementační části je zanést jednotlivé logické závislosti. A to konkrétně mezi prvky rozhraní, které korespondují s hodnotami registrů zmíněných v kapitole 2. Pro každou možnost tak existují různá omezení, která jsou dále v práci rozdělena dle množin registrů.

4.2.1 Omezení pro hlavní množinu registrů

Jelikož počet všech možností nastavení této množiny registrů je velký, data byla zhuštěna do tabulky 4.1. Názvy módů jsou vypsány ve zkratce, přičemž jejich význam je následující:

- **NIC** - Standardní operace síťové karty
- **Generování** - Generování umělého provozu
- **Načítání** - Načítání dat do dynamické paměti
- **Vysílání** - Vysílání dat z dynamické paměti

V případě tabulek 4.1 a 4.2 významu křížku (x) rozumějte tak, že dané textové pole je ve zvolené konfiguraci zakázáno (uživatel do něj nemůže zapisovat). Je-li v tabulce uvedeno zatržítko (✓), textové pole je povoleno.

Módy	Limitace	Počet paketů ke generovaní	Přenosová rychlost	Vzor dat	Zdrojová MAC	Cílová MAC	Verze IP	Zdroj dat
NIC	žádná	x	x	x	x	x	x	x
	přenosovou rychlostí	x	✓	x	x	x	x	x
	časovými známkami	x	x	x	x	x	x	x
Generování	žádná	✓	x	✓	✓	✓	✓	x
	přenosovou rychlostí	✓	✓	✓	✓	✓	✓	x
Načítání	žádná	x	x	x	x	x	x	✓
Vysílání	žádná	x	x	x	x	x	x	x
	přenosovou rychlostí	x	✓	x	x	x	x	x

Tabulka 4.1: Logické vazby u hlavní množiny registrů

Způsob generování	Pole		
	From	To	Step
Konstanta	✓	x	x
Sekvence čísel	✓	✓	✓
Náhodné čísla z rozsahu	✓	✓	x

Tabulka 4.2: Logické vazby u množiny registrů pro pole IPv4 a IPv6 hlaviček

4.2.2 Množina registrů pro pole IPv4 a IPv6 hlaviček

Tyto registry jsou v aplikaci implementovány jako textová pole, jenž každé obsahuje položky `From`, `To` a `Step`. Jedinou výjimku zde tvoří pole `Flags` v IPv4 hlavičce, kde postačuje pole `From`.

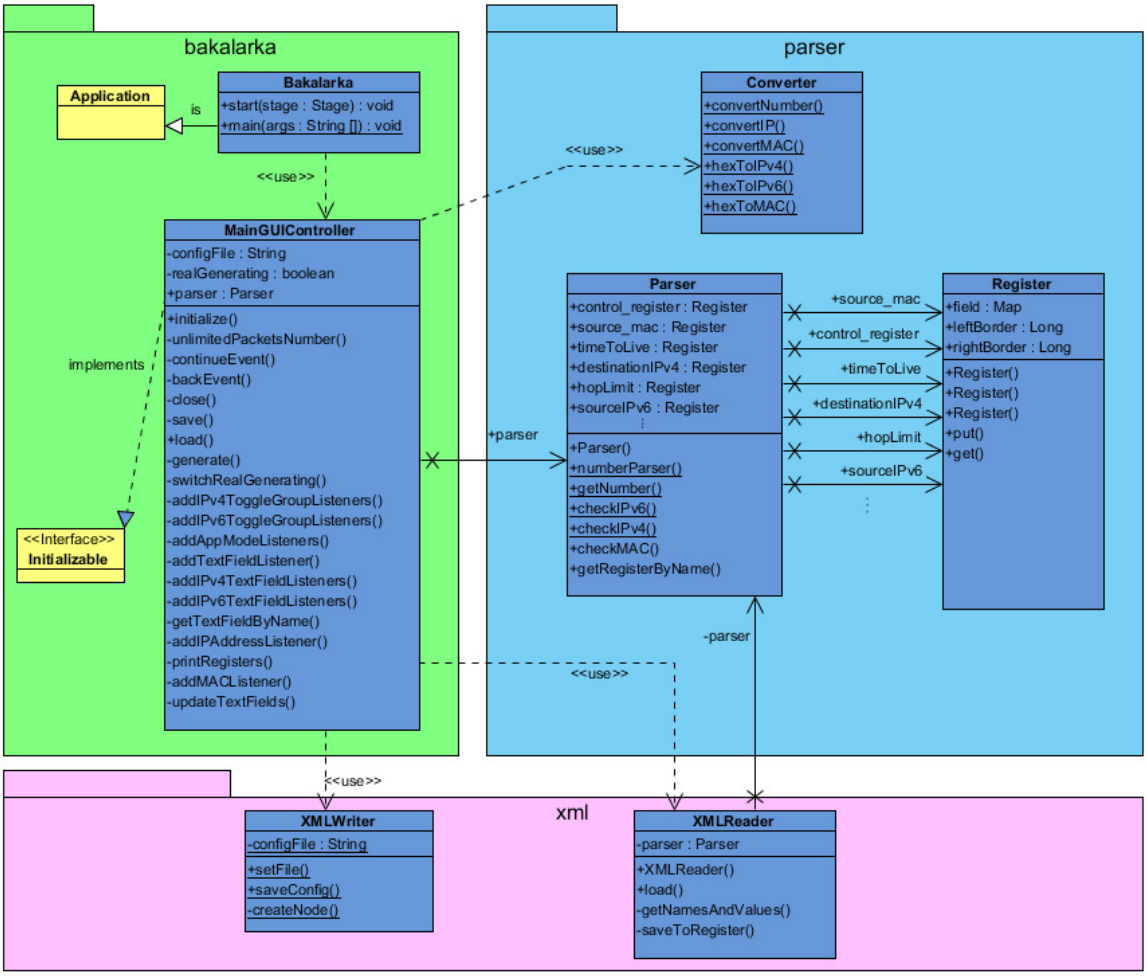
Na základě hodnot v registru pro způsob generování polí hlavičky (jeho struktura je k nalezení v tabulce 2.2) jsou jednotlivá textová pole povolena, či zakázána, jak lze vidět v tabulce 4.2.

4.3 Hierarchie tříd

Java, jakožto objektově orientovaný jazyk, poskytuje přehledné členění tříd a objektů ve zdrojových souborech aplikace, čehož je využito i v případě GUI pro generátor paketů.

Samotnou aplikaci spouští metody `main` a `start` umístěné ve soboru `Bakalarka.java`. Metoda `start` pouze nastaví výchozí jazykovou sadu a vytvoří scénu pomocí souboru `MainGUI.fxml`, který slouží pro popis prvků rozhraní. Funkčnost těchto prvků je implementována ve třídě `MainGUIController`, která je ve své podstatě jádrem celé aplikace. O definici vzhledu se stará soubor `maingui.css`.

Na zjednodušeném UML diagramu tříd 4.1 (více o diagramech je k nalezení například v [4]) lze vidět, jak jsou jednotlivé třídy aplikace mezi sebou logicky spojeny, jaké obsahují atributy a metody. Zjednodušení tkví ve vynechání prvků GUI ve třídě `MainGUIController`, u metod nejsou zobrazeny parametry a návratové hodnoty a v třídě `Parser` nejsou znázorněny všechny registry. Důvodem těchto zjednodušení je především úspora místa na obrázku.



Obrázek 4.1: Struktura závislosti jednotlivých tříd

4.3.1 MainGUIController

V souboru `MainGUIController.java` je popsána třída sloužící ke zpracování událostí při akcích prvků uživatelského rozhraní, povolování a zakazování prvků na základě aktuální konfigurace nebo například k podbarvování pozadí při zadávání hodnot do textových polí.

4.3.2 Register

Třída popsaná v `Register.java` obsahuje asocitativní kontejner se stromovou strukturou, kam se ukládají hodnoty registrů. Děje se tak ve tvaru dvou logicky spojených dvojic `(0,from)` a `(from,192.168.1.1)`. První dvojice slouží ke kontrole řazení při generování XML souboru, druhá uchovává samotná data.

Dále slouží k uchování hranic povoleného rozsahu, přičemž je uchovává v proměnných `leftBorder` a `rightBorder`. Tyto proměnné jsou zároveň použity jako výchozí hodnoty pro pole `From` a `To` příslušného registru, čímž reprezentují rozsah povolených hodnot v těchto polích.

4.3.3 Converter

V `Converter.java` nalezneme metody sloužící k různým převodům mezi formáty vnitřní a vnější reprezentace. Konkrétně jde o převod čísla do textového tvaru, či převod adres do šestnáctkové soustavy vzhledem k vnitřní reprezentaci. Pro potřeby vnější reprezentace jde o převod do tvaru s tečkovou notací pro adresy protokolu IPv4, respektive dvojtečkovou notací pro MAC adresy a IPv6 adresy.

4.3.4 Parser

Soubor `Parser.java` slouží pro inicilizaci registrů psolu s jejich mezními hodnotami, kontrolu správnosti adres, čísla v daném rozsahu, či pro mapování textového řetězce na odpovídající registr.

U kontroly správnosti IP adres byly použity funkce z knihovny `IPAddressUtil` [5]. Konkrétně `isIPv6LiteralAddress(String IPv6)` a `isIPv6LiteralAddress(String IPv4)`. Při překladu aplikace způsobují varování, jelikož se jedná o proprietární API firmy Sun a tyto funkce mohou být v budoucnu smazány. Jedním z možných řešení je použití funkce `InetAddress.getByName(String IP)`, která však ve svém těle obsahuje funkce z knihovny IPAddressUtil, čímž se uzavírá pomyslný kruh závislosti obou knihoven.

Výhodou použití druhé varianty je, že se již nejedná o proprietární API, a tak pokud bude v budoucnu knihovna IPAddressUtil smazána, bude zajištěno její nahrazení ze strany programátorů knihovny InetAddress. Zároveň překladač tedy nehlásí varování.

Nevýhodou použití funkce `getByName()` je, že při kontrole správnosti se funkce snaží zároveň dostat odpověď z této adresy. To je nepřijatelné ve chvíli, kdy jsou textová pole kontrolována při zadání každého znaku, protože takto vzniká velká časová náročnost obzvláště při čekání na odpověď z adresy, která neexistuje.

4.3.5 XMLReader

Třida implementovaná v souboru `XMLReader.java` je určena k načtení hodnot z XML souboru do vnitřní reprezentace. Zároveň je zde implementována problematika dialogu pro výběr XML souboru.

4.3.6 XMLWriter

Soubor `XMLWriter.java` popisuje třídu sloužící pro zápis dat do konfiguračního souboru XML, a to v čitelném výstupu formátovaném pomocí bílých znaků.

4.4 Kontrola zadávaných hodnot

Jak návrh předpokládal, byla implementována okamžitá kontrola zadávaných hodnot do textových polí. Zjednodušeně se jedná o funkci, kdy s každým napsaným znakem probíhá kontrola správnosti celého řetězce vzhledem k povoleným hodnotám daného pole. O úspěchu či neúspěchu je uživatel informován zeleným, respektive červeným podbarvením textového pole. Při kontrole hodnot vznikají specifické situace, konkrétně kontrola prázdného pole a generování/ukládání konfigurace s nesprávnými hodnotami.

4.4.1 Kontrola prázdného pole

Pokud uživatel do pole nezadá žádnou hodnotu a spustí generování, respektive uloží aktuální konfiguraci, vzniká první problém, a to kontrola prázdného pole. Implicitním chováním aplikace je nahrazení prázdného řetězce výchozí hodnotou. Tu má každé pole uloženu v příslušné datové struktuře. Načte-li posléze uživatel uloženou konfiguraci, pak původně prázdná pole budou nyní vyplněna.

4.4.2 Generování a ukládání konfigurace s nesprávnými hodnotami

Tato situace může vzniknout ve více případech. Může se jednat například o zadání nepovoleného znaku, zadání čísla mimo rozsah povolených hodnot, nebo neúplného zadání MAC adresy. Ve všech případech aplikace použije poslední správnou hodnotu pole, přičemž první korektní vstup je prázdné pole.

Pro lepší názornost demonstrace chování aplikace jsou v tabulce 4.3 uvedeny příklady různých situací.

Typ vstupu	**Vstup od uživatele**	**Reprezentace**
0–255 výchozí hodnota `0`	5	5
	bílý znak	0
	2005	200
	abc	0
	1a2b3c	1
	2 b následně smazány oba znaky	2
MAC Adresa výchozí hodnota `00:00:00:00:00:00`	11:22:33:44:55:66	11:22:33:44:55:66
	aa-bb-cc-dd-ee-ff	aa:bb:cc:dd:ee:ff
	bílý znak	00:00:00:00:00:00
	11:22:33:44:55-66	00:00:00:00:00:00
	11:22:33:44:55:66:77	11:22:33:44:55:66
	11:22:33:44:55-66:77	00:00:00:00:00:00
	11:22:33:44:55	00:00:00:00:00:00
IPv4 Adresa výchozí hodnota `0.0.0.0`	100.200.300.400	100.200.300.400
	100.200.300.400.500	100.200.300.400
	100.200.300	0.0.0.0
	0xAABBCCDD	0.0.0.0
	AABBCCDD	0.0.0.0
	bitová podoba	0.0.0.0

Tabulka 4.3: Příklady vstupních hodnot

4.5 Změny oproti návrhu

Jak již bylo zmíněno na začátku této kapitoly, pro některé body návrhu se během implementace našla lepší řešení, či nebyly implementovány vůbec. Těmto rozdílům se věnují následující odstavce.

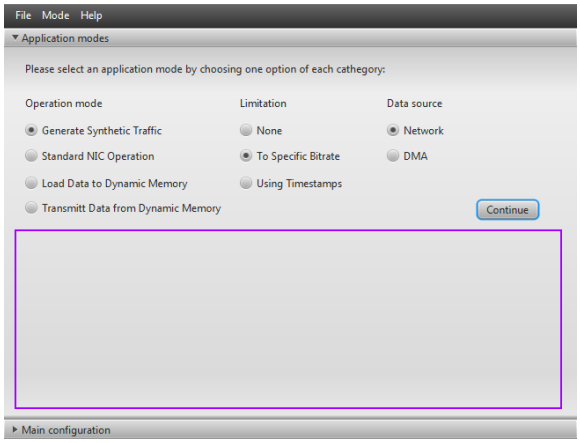
4.5.1 Grafická předloha

Změny provedené v grafickém návrhu se odvíjí od snížení minimální doby potřebné pro konfiguraci a zvýšení orientace uživatele v celé aplikaci. Původní návrh od finální verze lze rozeznat na základě barvy pozadí, přičemž finální verze má barvu zelenou.

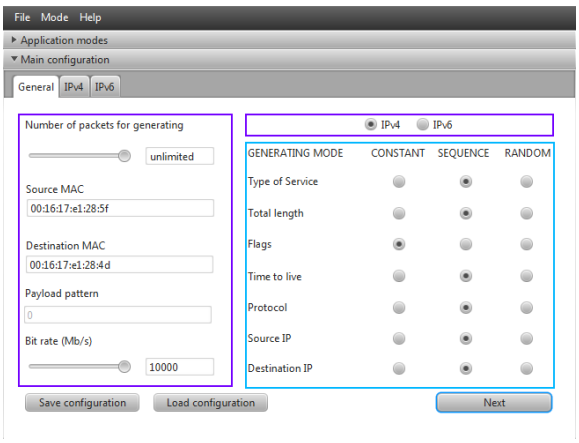
Jak lze vidět na následujících obrázích, došlo hned k několika změnám. První z nich je zrušení celé karty s názvem `General`. Další pak změna velikosti a rozložení textových polí v kartě `IPv6`, kontrola textových polí je znázorněna zeleným a červeným podbarvením namísto speciálních znaků a zbytek změn se už týká spíše rozmístění prvků.

Přesun konfigurace pro generování

Okno s hlavní konfigurací bylo dříve zaplněné z poloviny, jak znázorňuje fialová barva na obrázku 4.2. Jako výhodnější řešení z pohledu zaplnění místa a rychlosti samotného rozhraní, se ukázalo vyplnit tuto oblast prvky, které blíže konfigurují jednotlivé módy (fialová barva na obrázku 4.3).

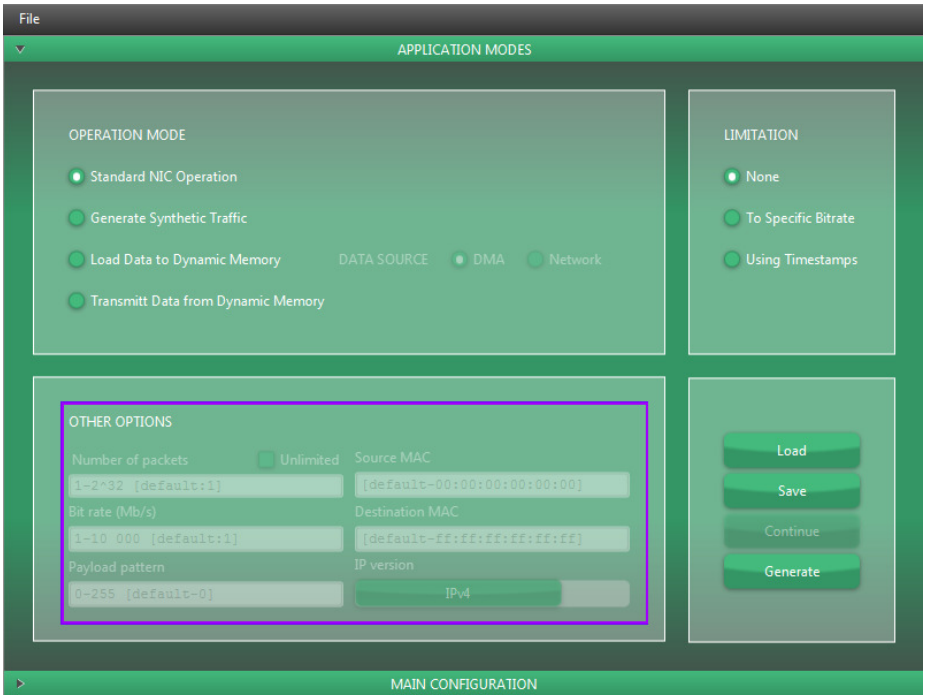


Obrázek 4.2: Původní okno hlavní konfigurace



Obrázek 4.3: Původní karta General

Aktuální rozmístění zmíněných prvků reprezentuje obdélník s fialovou barvou v obrázku 4.4. Zároveň došlo odstranění posuvníků, které byly nahrazeny textovým poli. Důvodem byla příliš velká citlovost posuvníků díky velkému povolenému rozsahu hodnot u polí s názvem Bitrate a Number of Packets. Aby výběr verze protokolu IP zapadl do současného designu rohraní, bylo místo skupiny dvou přepínačů zvoleno přepínací tlačítko.

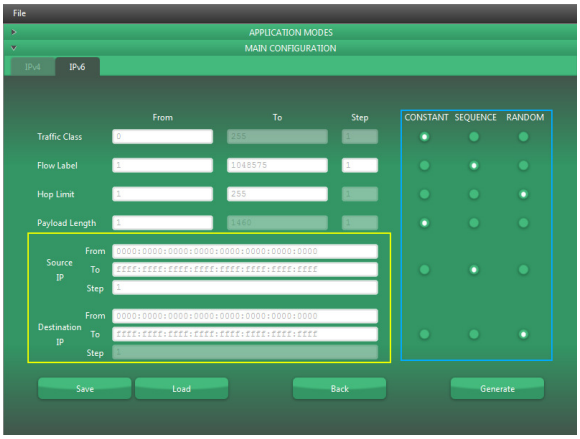


Obrázek 4.4: Aktuální okno hlavní konfigurace

Přesun přepínačů

Především díky názornosti vztahů mezi přepínači a povolenými textovými poli souvisejícími s kartami IPv4 a IPv6 byly skupiny přepínačů umístěny do těchto karet. Na obrázcích 4.3, 4.5 a 4.6 je tato změna znázorněna světle modrou barvou.

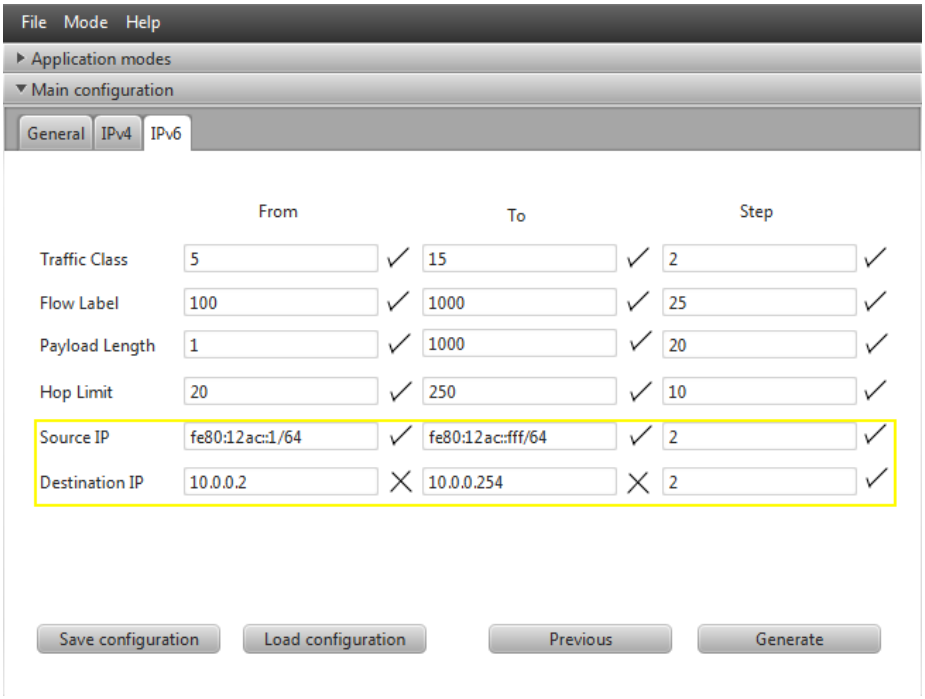
Obrázek 4.5: Aktuální vzhled karty IPv4



Obrázek 4.6: Aktuální vzhled karty IPv6

4.5.2 Zvětšení polí pro IPv6 adresy

Na základě primárního testování (více v kapitole 5) byl upraven návrh karty pro konfiguraci polí IPv6 hlavičky, a to na základě požadavku na zvětšení textových polí tak, aby odpovídaly velikosti IPv6 adresy v maximální délce.



Obrázek 4.7: Původní návrh karty IPv6

4.5.3 Funkční stránka

Aplikace oproti návrhu postrádá některé funkce. Jedná se však o méně důležité vlastnosti rozhraní, které mohou být doimplementovány v budoucích verzích.

Jazykové mutace

Jediným jazykem, kterým rozhraní nyní komunikuje, je angličtina. Vzhledem k cílové skupině uživatelů je ale implementace této funkce spíše třešničkou na dortu.

Více vzhledů

Možnost volby mezi více vzhledy nebyla implementována na úkor kvality výchozího designu. Aplikace sází na zelenou barvu v kombinaci s bílým a černým textem. Jelikož je barevný vzhled řešen pomocí CSS, nemělo by být složité vytvořit více předloh pouhou změnou barev ve zdrojovém souboru a jeho naklonováním.

Kapitola 5

Testování

Hnacím motorem pokroku je zpětná vazba. Tu programátor při vývoji nejlépe získá testováním aplikace na skupině potenciálních uživatelů. Nejinak tomu bylo u tvorby uživatelského rozhraní pro generátor paketů.

Testování probíhalo nejprve na úzké skupině jedinců a po odladění základních nedostatků i na širším spektru. Pro účel druhé vlny testování byl vytvořen dotazník, jehož originální podoba je dostupná na http://www.survio.com/survey/d/F2J7A8H7S5N2U7P4V a přibližná podoba zformátována do vzhledu této práce je k nalezení v příloze C.

5.1 Primární testování

5.1.1 Instalace Javy

Jedním z prvních požadavků vycházejícím z testování na úzké skupině jedinců bylo odstranění nutnosti instalace Javy pro běh samotné aplikace. Tento problém byl vyřešen vytvořením instalačního souboru, který do zvolené destinace rozbalí aplikaci spolu s prostředím nutným k jejímu běhu. Instalační soubor tímto nabývá na velikosti přibližně 30 MB.

V poměru rychlosti internetu a velikosti instalačního souboru se toto řešení ukázalo jako vhodné. Během druhé vlny testování totiž nebyla zaznamenána žádná negativní zpětná vazba zmiňující velikost instalačního souboru.

5.1.2 Ukládání a načítání konfigurace

Druhým problémem, který byl odhalen již na začátku testování, bylo ukládání a načítání konfigurace. Konkrétně se jednalo o nepřítomnost koncovky `.xml` u ukládaných souborů. Tato situace vznikla, pokud ji uživatel nezadal za požadovaný název souboru. A jelikož dialog pro načítání konfigurace obsahuje filtr na soubory s příponou `.xml`, nebyl ukládaný soubor viditelný.

Zmíněný problém byl vyřešen ještě před druhou vlnou testování. Pokud přípona zadána není, program ji k souboru přidá automaticky.

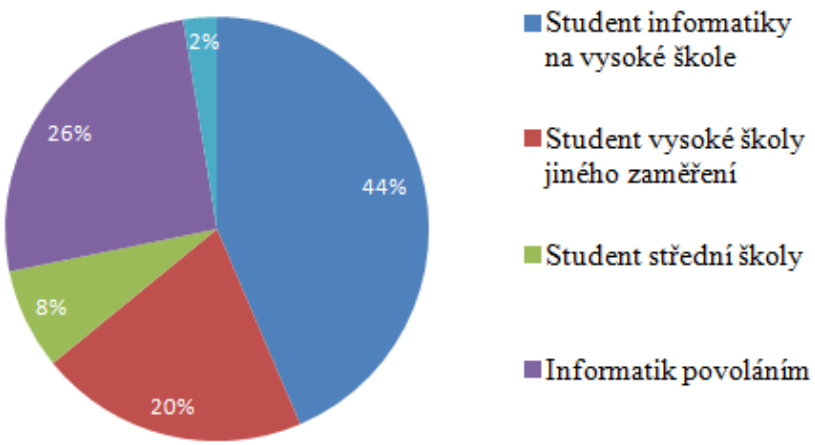
5.1.3 Velikost textových polí u IPv6 adres

Posledním požadavkem, který byl do aplikace zapracován na základě prvního testování, byla změna velikosti textových polí u adres verze IPv6. Tato změna byla již popsána v podsekci 4.5.2.

5.2 Sekundární testování

Po vyřešení problémů zmíněných výše, následovala tvorba otázek pro dotazník sloužící k druhé vlně testování. V konečné fázi dotazník obsahoval celkem 9 otázek různých typů a zároveň odkaz ke stažení instalačního souboru aplikace pro operační systém Windows 7 64-bit.

Po instalaci a krátkém seznámení s programem odpovídalo na otázky 39 uživatelů. Téměř polovinu korespondentů tvořili studenti informatiky na vysoké škole, čtvrtinu informatici z povolání téměř čtvrtinu studenti jiné vysoké školy. Celkovou skladbu znázorňuje graf 5.1.



Obrázek 5.1: Rozdělení skupiny korespondentů

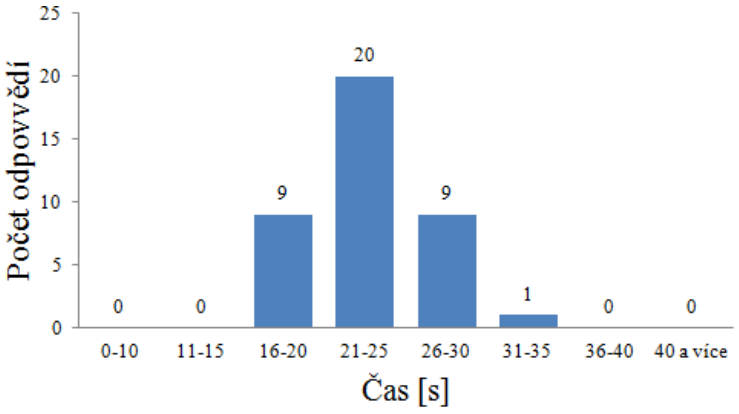
5.2.1 Hodnocení designu

K tomuto účelu sloužila jedna z otázek, kde na pětihvězdičkové škále uživatelé hodnotili jejich názor na vzhled. Vzhledem k důležitosti designu, jak již bylo zmíněno v sekci 1.1, je výsledek $4,54$ z možných 5 skvělým úspěchem.

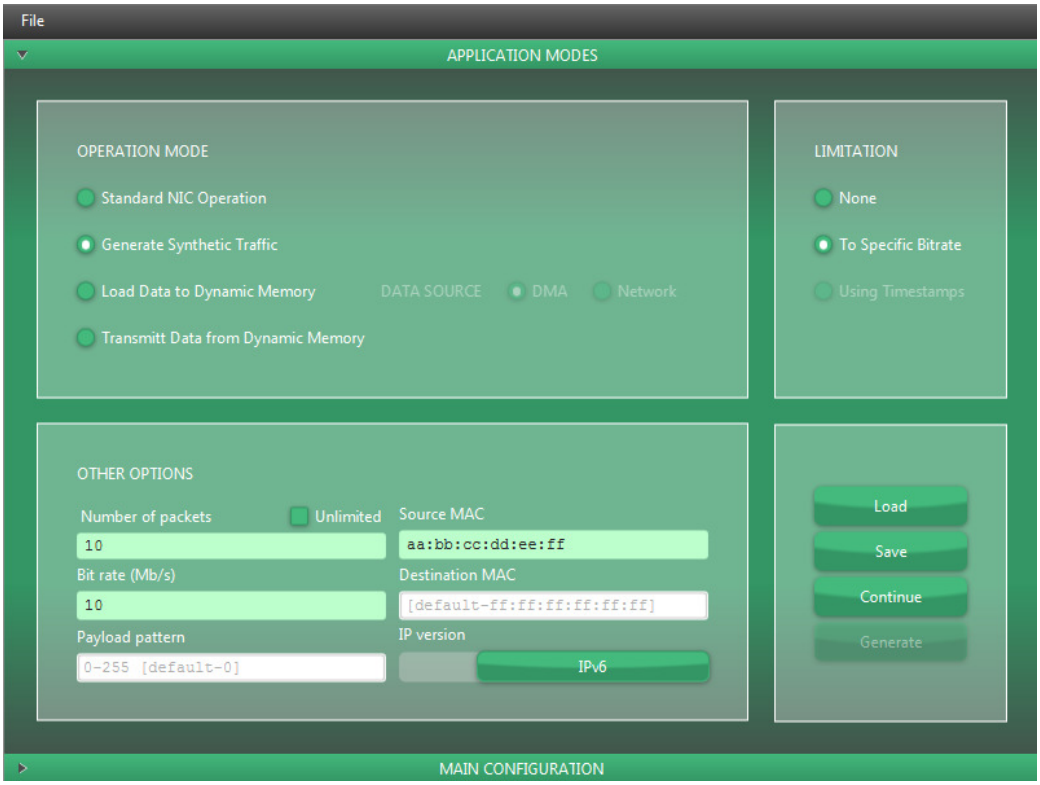
5.2.2 Otázky měřicí rychlost rozhraní

Z celkového počtu devíti otázek zabírá tento typ dvě místa. Cílem zpracování jejich odpovědí je vyhodnocení rychlosti rozhraní. Korespondenti začínali i končili své pokusy vždy v levém horním rohu aplikace.

První otázka byla zaměřena na konfiguraci na hlavním okně aplikace (její podoba je na obrázku 5.3). V grafu 5.2 jsou zachyceny výsledky tohoto měření.

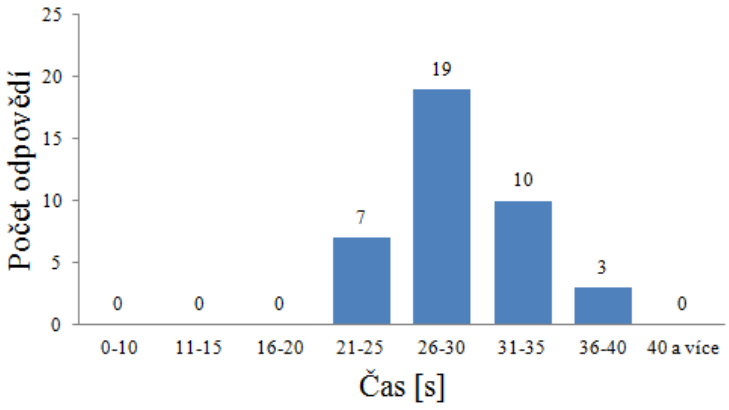


Obrázek 5.2: Výsledky měření doby potřebné pro nastavení konfigurace v hlavním okně



Obrázek 5.3: Požadované nastavení konfigurace v hlavním okně

Výsledky druhé otázky zaměřené na konfiguraci polí IPv6 hlavičky (celková podoba na 5.5) znázorňuje graf 5.4.

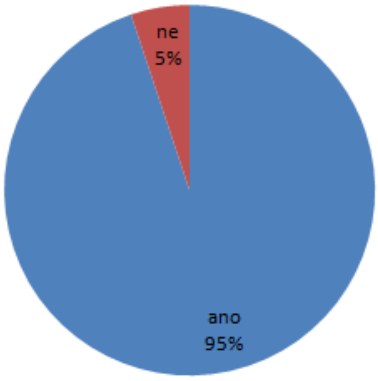


Obrázek 5.4: Výsledky měření doby potřebné pro nastavení konfigurace v kartě IPv6

Obrázek 5.5: Požadované nastavení konfigurace na kartě IPv6

5.2.3 Ostatní otázky

Stejné rozložení odpovědí, zobrazených grafem 5.6, měly otázky zabývající se přehledností a okamžitou kontrolou správnosti zadávaných vstupů (celé znění na Ca C).



Obrázek 5.6: Výsledky otázek o přehlednosti a kontrole vstupních dat

Další dvě otázky kontrolovaly ukládání a načítání konfigurace a problémy s instalací programu. Obě dopadly shodně - všichni odpověděli, že se nenaskytly problémy.

Poslední otázka dávala odpovídajícím prostor pro další připomínky. Zde se objevily pouze kladné odpovědi typu „Jen tak dál“ a „Super!“.

Závěr

Cílem této práce bylo seznámit se s hardwarově akcelerovaným generátorem paketů, navrhnout pro něj snadně ovladatelné uživatelské rozhraní, toto rozhraní implementovat, otestovat a vytvořit tutoriál k jeho použití.

Nejdříve ze všeho bylo zapotřebí porozumět, jak samotný generátor paketů pracuje. Tomuto účelu sloužily konzultace s autorem aplikace Ing. Jiřím Matouškem a zároveň jeho diplomová práce [3] zaměřená na toto téma.

Navrhnout a vytvořit grafické uživatelské rozhraní vyžadovalo perfektní pochopení již existujících rozhraní. K jejich studiu byly poskytnuty soubory, mezi než patří zdrojový kód programu `load_config` sloužící k načtení konfigurace z XML souborů, vzorový konfigurační soubor `config.xml` a `config_spec.xml` zmiňující možné varianty hodnot jednotlivých registrů.

Ve chvíli, kdy bylo jasno, jak generátor paketů pracuje, následoval rozbor požadavků na jeho rozhraní. Nejprve šlo o teoretický základ zabývající se rozhraními z pohledu komunikačních kanálů, následně pak praktický průzkum na esenciální vlastnosti, které by mělo grafické uživatelské rozhraní pro generátor paketů splňovat. Cílovou skupinou zde přitom byli budoucí potenciální uživatelé aplikace, tedy lidé pohybující se v oblasti informačních technologií.

Na základě průzkumu zaměřeného na požadavky na rozhraní vznikl první návrh, který respektoval rozmístění prvků rozhraní, jeho rychlost, jednoduchost, ale také například jazykové mutace. Návrh byl zároveň tvořen v programu JavaFX Scene Builder, takže byl později použitelný pro potřeby implementace.

Poté, co byly nabyty teoretické znalosti, a byla zpracována předloha vzhledu aplikace, následovala implementační část. Na platformě JavaFX v prostředí NetBeans tak vzniklo grafické uživatelské rozhraní, u něhož byl zároveň dotvořen barevný design založený na zelené barvě v kombinaci s bílým a černým textem.

Vzhledem k požadované funkčnosti a přehodnocení důležitosti některých částí návrhu nebyly implementovány jazykové mutace a volba z více možných vzhledů. Naopak bylo změněno rozmístění prvků, které dovoluje redukci potřebných mezikroků pro generování. Oproti návrhu přibyla možnost volby ostrého generování, což v praxi znamená, že si může uživatel vybrat, zda program pouze vygeneruje XML soubor, nebo jestli po jeho vytvoření navíc spustí program pro načtení této konfigurace na kartu. Tím je dosaženo reálného generování.

Naprogramované rozhraní bylo otestováno ve dvou vlnách. Nejprve na menší skupině uživatelů, na které se odladily chyby, a následně na širším počtu korespondentů, kteří odpovídali na otázky týkající se splnění požadavků na rozhraní. Během testování se odhalilo několik chyb, přičemž většina byla doimplementována. Testování dopadlo velmi nadprůměrným výsledkem u hodnocení designu, výsledky z ostatních otázek dopadly také pozitivně.

Během testování byl zároveň vytvořen tutoriál k použití grafického rozhraní. Soustředí se na objasnění možných kombinací pro konfiguraci, způsob implementace validace vstupních hodnot v textových polích, ukládání, načítání a generování konfigurace. Jeho podoba je k nalezení v příloze A.

Rozhraní v současné době neposkytuje jazykové mutace, podporu běhu na více operačních systémech, volbu z více vzhledů, implementaci vlastní kontroly správnosti IP adres, či tutoriál s vyhledáváním zapracovaný přímo do programu. Toto jsou směry, kterými by se měl udávat další vývoj rozhraní. Implementováním výše zmíněných funkcí bude rozhraní uživatelsky příjemnější, ale již teď si dozajisté zaslouží své místo v kontextu již existujících řešení.

Literatura

[1] CESNET: Liberouter - technologies. [online], [cit. 2012-11-09].
URL https://www.liberouter.org/technologies/

[2] MARTÍNEK, T.: Handbook, nepublikováno.

[3] MATOUŠEK, J.: *Simulace a generování síťového provozu*. Diplomová práce, FIT VUT v Brně, 2011.

[4] NĚMEC, M.: UML:Diagramy tříd. [online], [cit. 2013-05-01].
URL http://www.milosnemec.cz/clanek.php?id=199

[5] ORACLE: IPAddressUtil. [online], [cit. 2013-04-09].
URL http://javasourcecode.org/html/open-source/jdk/jdk-6u23/sun/net/util/IPAddressUtil.java.html

[6] ORACLE: JavaFX - The Rich Client Platform. [online], [cit. 2013-04-30].
URL http://www.oracle.com/technetwork/java/javafx/overview/index.html

[7] ORACLE: JavaFX Scene Builder. [online], [cit. 2013-04-09].
URL http://www.oracle.com/technetwork/java/javafx/tools/index.html

[8] RUSSEL, J.: Inno Setup. [online], [cit. 2013-04-09].
URL http://www.jrsoftware.org/isinfo.php

[9] TCPDUMP: Tcpdump & Libpcap. [online], [cit. 2013-04-09].
URL http://www.tcpdump.org/

[10] TURNER, A.; SIAM, Y.: Tcpreplay. [online], [cit. 2013-04-09].
URL http://tcpreplay.synfin.net/

[11] ZEMČÍK, P.: *Tvorba uživatelských rozhraní*. Studijní opora, UPGM FIT VUT v Brně, 2006.

Příloha A

Tutoriál

A.1 Varianty možného nastavení

Na následujících stranách jsou umístěny vzorové kombinace různých konfigurací. Jsou rozděleny dle aplikačního módu a způsobu limitace vysílání paketů do sítě.

A.1.1 Standardní operace síťové karty

V tomto nastavení aplikace nezasahuje do normálního provozu na síťové kartě. Tento mód se používá pro odchytávání a následné přehrání provozu zpět do sítě. Výše zmíněné úlohy mohou být zpracovány programy jako je `tcpdump`, `tcpreplay`, či nástroji platformy NetCOPE – `sze2read`, `sze2write`.

Bez limitace

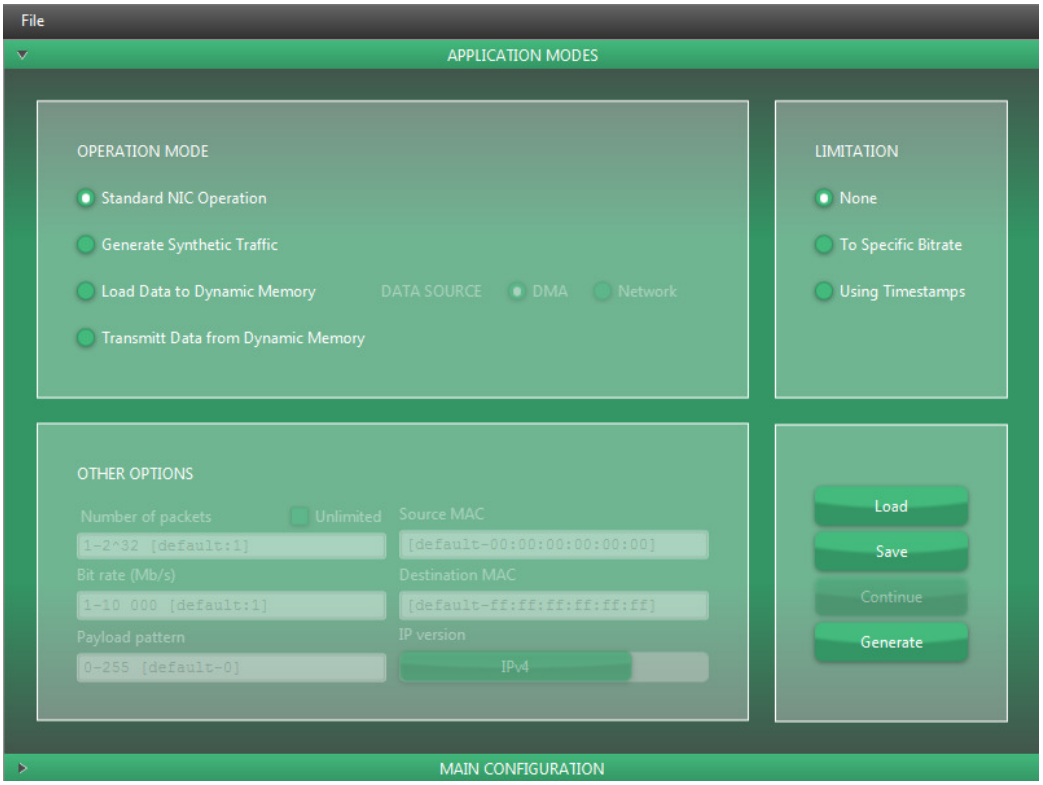
V této konfiguraci program žádným způsobem neomezuje rychlost, s jakou se pakety vysílají zpět do sítě. Zároveň se jedná o výchozí konfiguraci pro generátor paketů (nastavení této konfigurace znázorňuje obrázek A.1).

Limitace na přenosovou rychlost

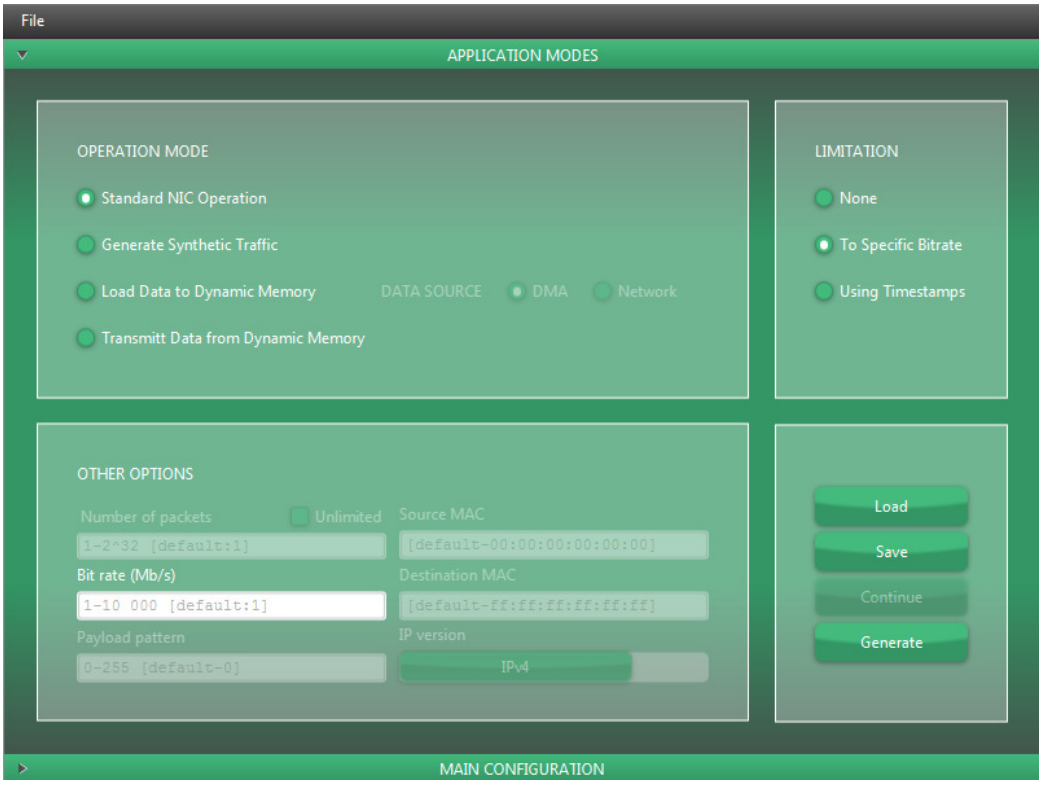
Hodnota jediného aktivního textového pole s názvem Bitrate udává v Mb/s jakou rychlostí se budou zachycená data vysílat. Použití tohoto nastavení odpovídá předchozí zmíněné konfiguraci. Grafickou podobu této konfigurace naleznete na obrázku A.2.

Limitace založená na použití časových známek

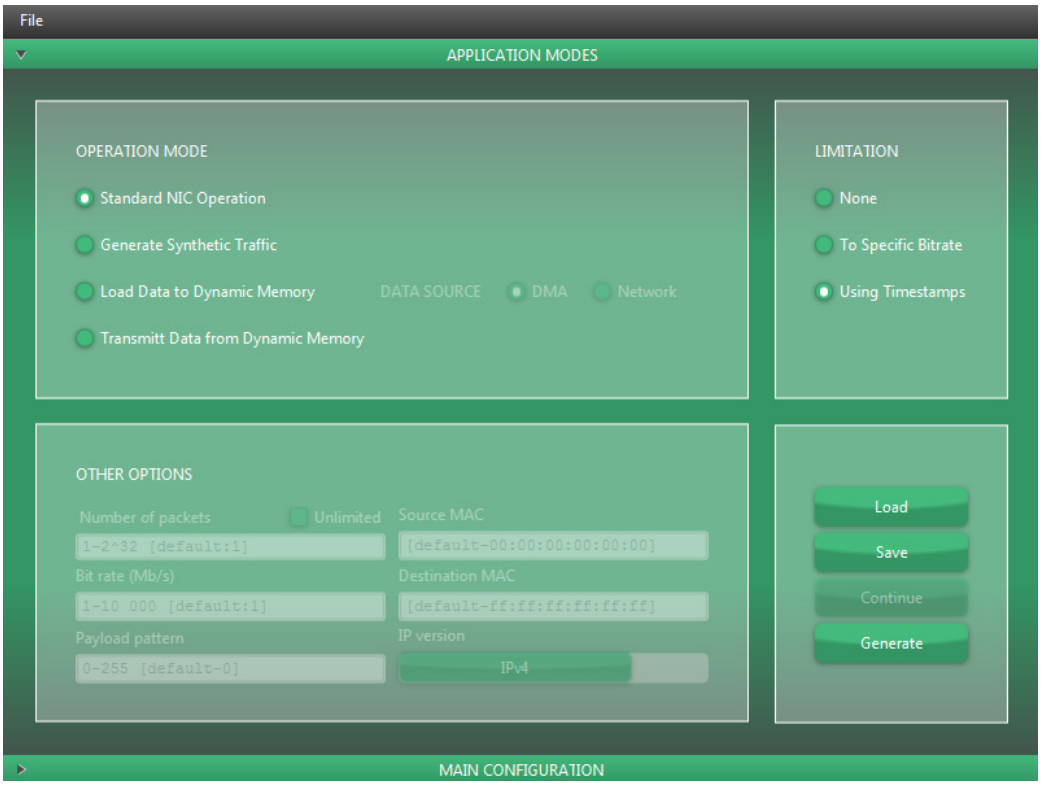
Toto nastavení se používá v případě využití programů `tcpreplay`, či `sze2write`. Ty vysílají zachycený provoz, jenž obsahoval časové známky, podle nichž je vysílání řízeno. Typicky se k originální časové známce při příchodu přičítá konstanta. Výsledkem je nová časová známka, posunutá v čase vpřed. Takto je program schopen zkopírovat předchozí provoz s důrazem na zachování mezer mezi příchody paketů. Toto nastavení popisuje obrázek A.3.



Obrázek A.1: Standardní operace síťové karty - bez limitace



Obrázek A.2: Standardní operace síťové karty - limitace přenosovou rychlostí



Obrázek A.3: Standardní operace síťové karty - limitace časovými známkami

A.1.2 Generování umělého síťového provozu

Předpokládá se, že tento mód bude jedním z nejpoužívanějších, jelikož se jedná o stěžejní funkci apilkace - tedy umělé generování. K dokončení celého nastavení je zapotřebí vyplnit navíc textová pole reprezentující pole hlaviček paketů IPv4, či IPv6. Karty k tomuto nastavení jsou popsány v A.1.5

Bez limitace - IPv4

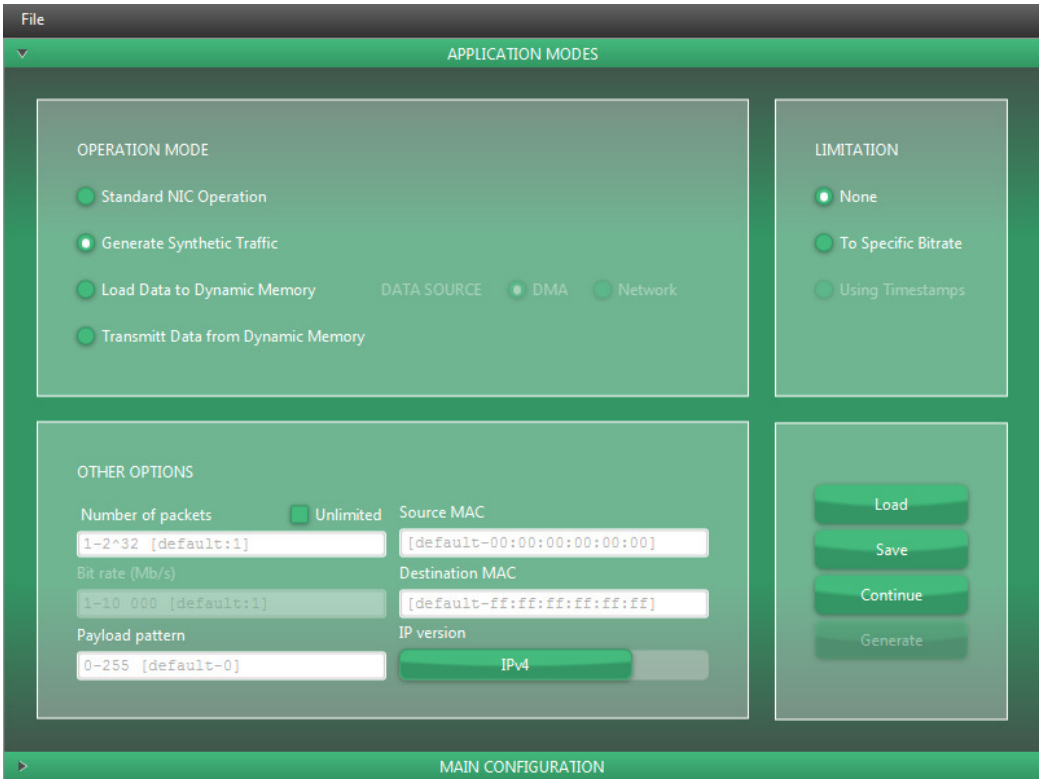
Jakmile jsou nastaveny příslušné hodnoty hlaviček paketů, jsou tyto pakety generovány a vysílány do sítě bez jakéhokoliv omezení rychlosti ze strany aplikace. Nastavení těchto parametrů ukazuje obrázek A.4.

Bez limitace - IPv6

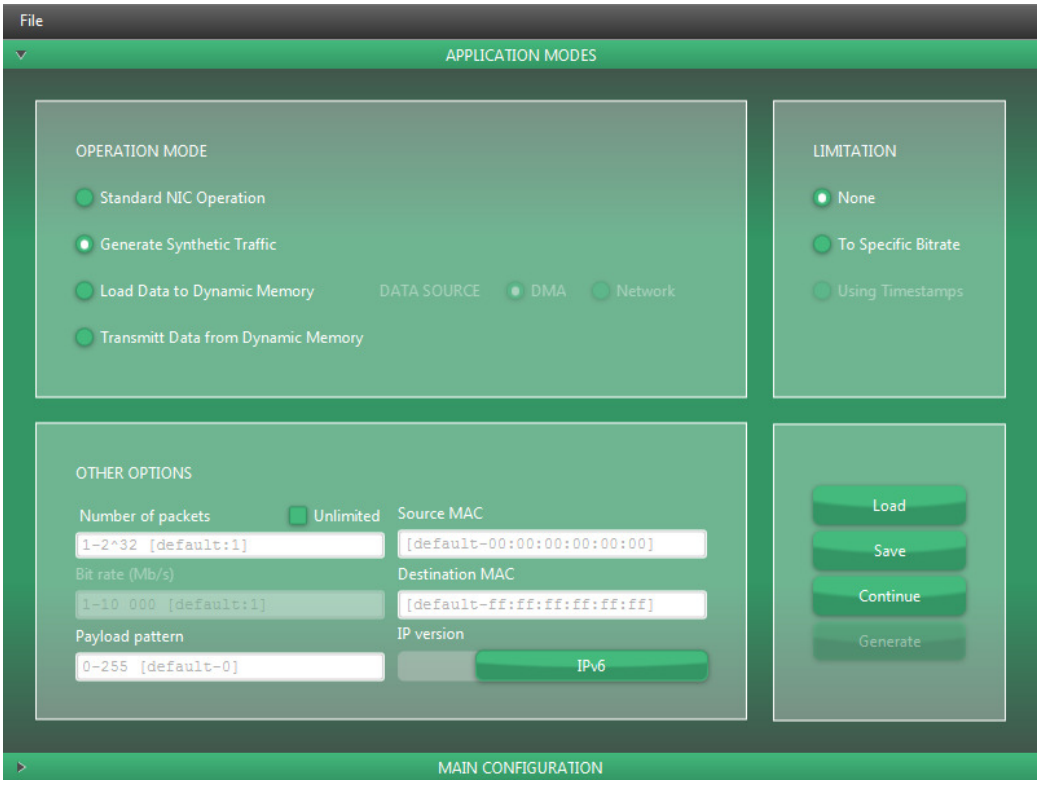
Obdoba předchozího nastavení pro protokol IP verze 6 (konfigurace na obrázku A.5)

Limitace na přenosovou rychlost

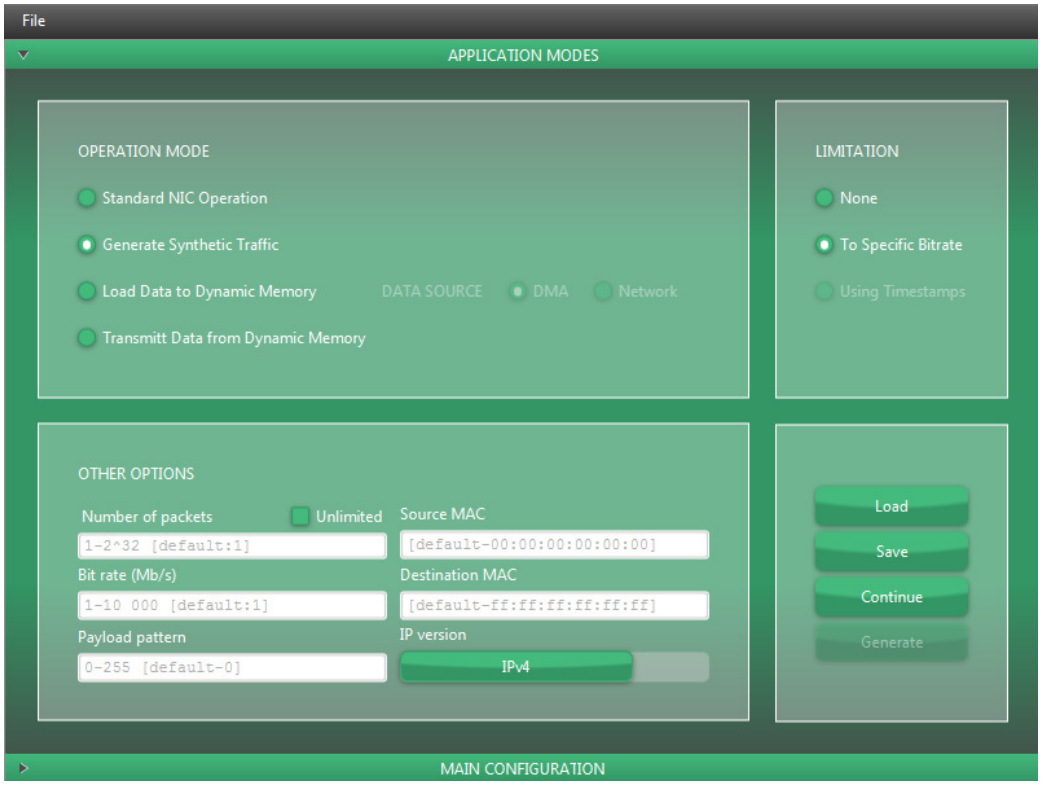
Pod tímto nastavením se oproti předchozímu skrývá ještě omezení vysílání generovaných paketů do počítačové sítě. A to konkrétně z pohledu přenosové rychlosti, která je určena z textového pole s názvem Bitrate. Grafická podoba tohoto nastavení je zobrazena na A.6.



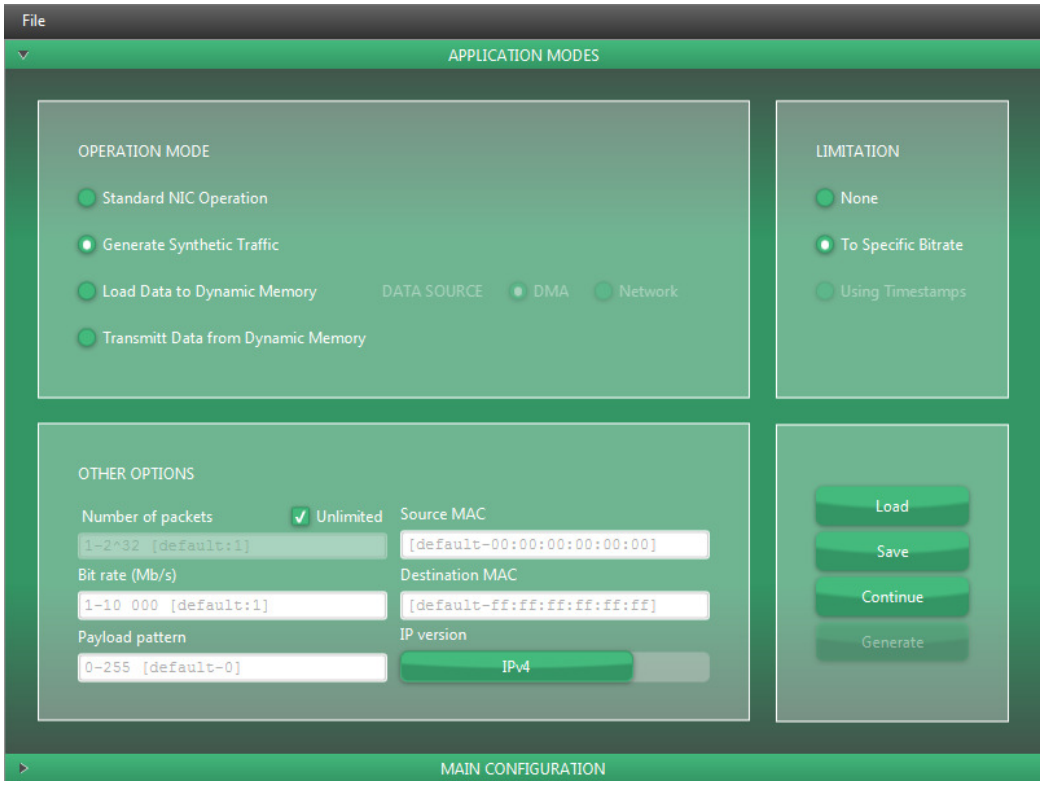
Obrázek A.4: Generování umělého síťového provozu - bez limitace - IPv4



Obrázek A.5: Generování umělého síťového provozu - bez limitace - IPv6



Obrázek A.6: Generování umělého síťového provozu - limitace přenosovou rychlostí



Obrázek A.7: Generování umělého síťového provozu - neomezený počet paketů

Neomezený počet paketů

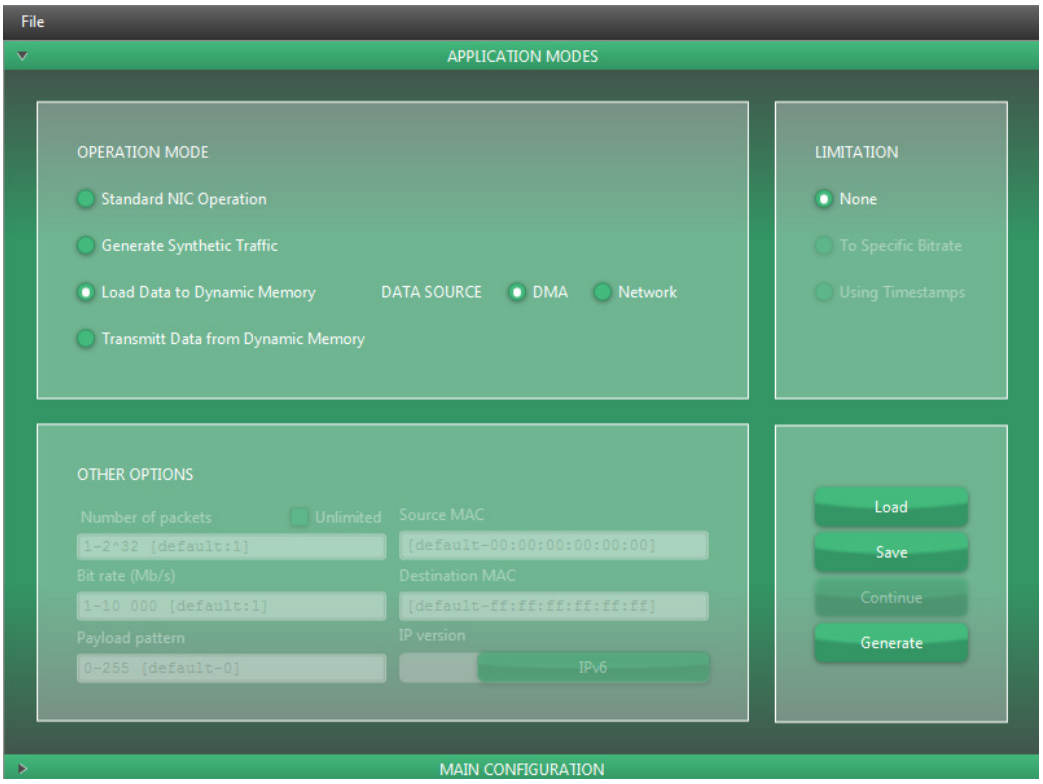
Ve srovnání s nastavením z předešlých zmíněných možností, je v této konfiguraci zvolen neomezený počet generovaných paketů. Tato volba implikuje v nepřetržité vysílání paketů do sítě. V předchozích případech je počet generovaných paketů určen hodnotou v poli s názvem Number of Packets. Toto nastavení znázorňuje obrázek A.7.

A.1.3 Načítání dat do dynamické paměti

Pro možnost pozdějšího opětovného přehrání dříve zachyceného provozu, musí být data nejprve načtena do externí dynamické paměti karty.

Zdroj DMA

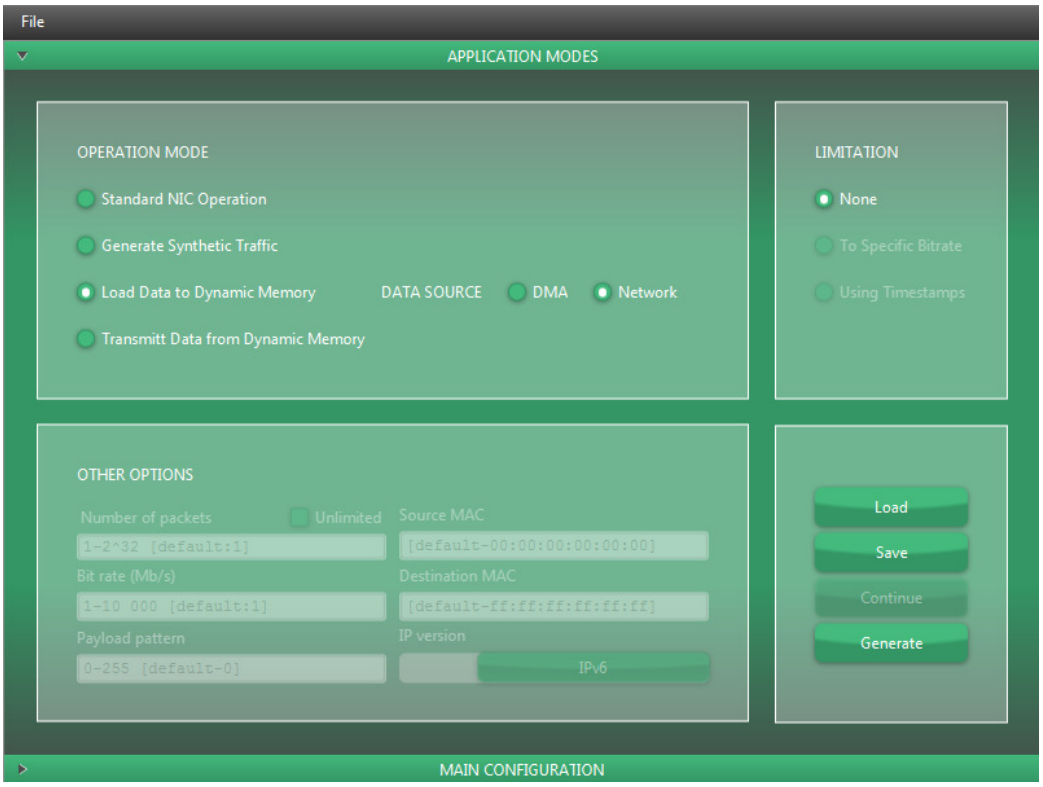
V tomto nastavení je zdrojem dat PCAP soubor uložený na hostitelském počítači. Tento mód neobsahuje žádné další nastavení v textových polích, ale data uložená v PCAP souboru mohou obsahovat časové známky použitelné pro pozdější vysílání. Zmíněnou konfiguraci popisuje obrázek A.8.



Obrázek A.8: Načítání dat do dynamické paměti - zdroj DMA

Zdroj síťové rozhraní

Tentokrát je zdrojem dat přímo síťové rozhraní, odkud jsou data získávána a ukládána. V tomto případě data obsahují téměř vždy časové známky, které se dají použít při opětovném vysílání paketů zpět do sítě. Toto nastavení znázorňuje obrázek A.9.



Obrázek A.9: Načítání dat do dynamické paměti - zdroj síťové rozhraní

A.1.4 Vysílání dat z dynamické paměti

Mód umožňující zpožděné vysílání dat, která byla v minulosti zachycena. Data jsou uložena v dynamické paměti karty a jsou vysílána zpět do sítě.

Bez limitace

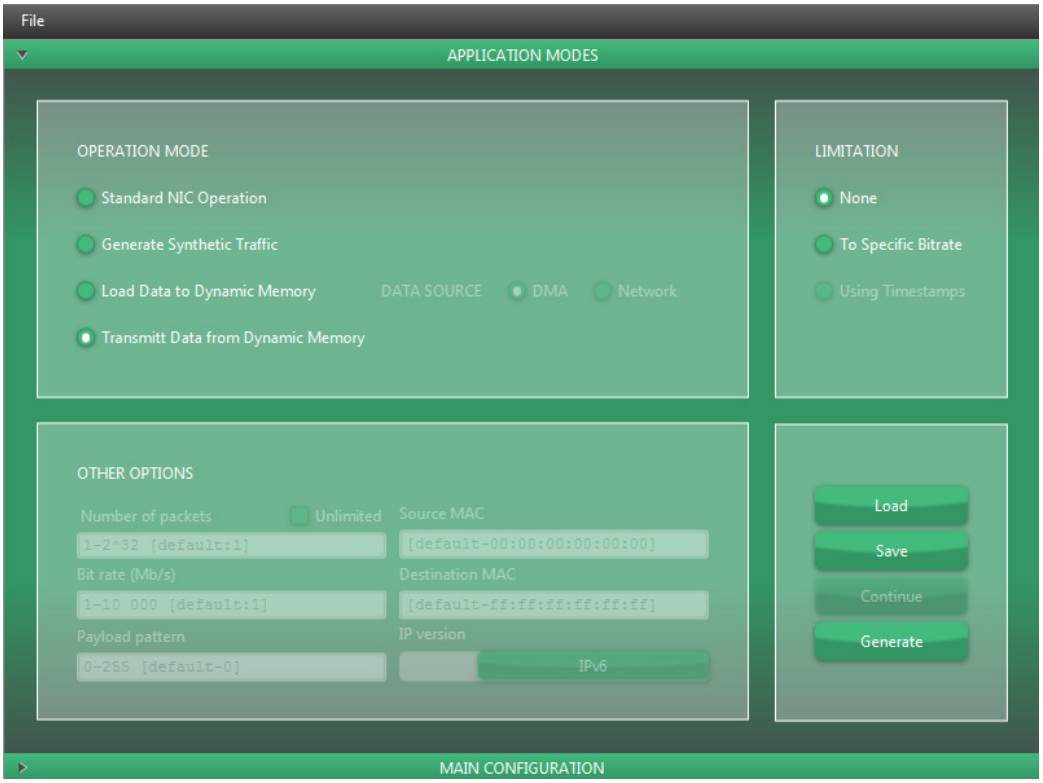
Aplikace nezasahuje do řízení vysílací rychlosti a využívá maximální dostupnou kapacitu přenosové rychlosti. Této konfiguraci odpovídá obrázek A.10.

Limitace přenosovou rychlostí

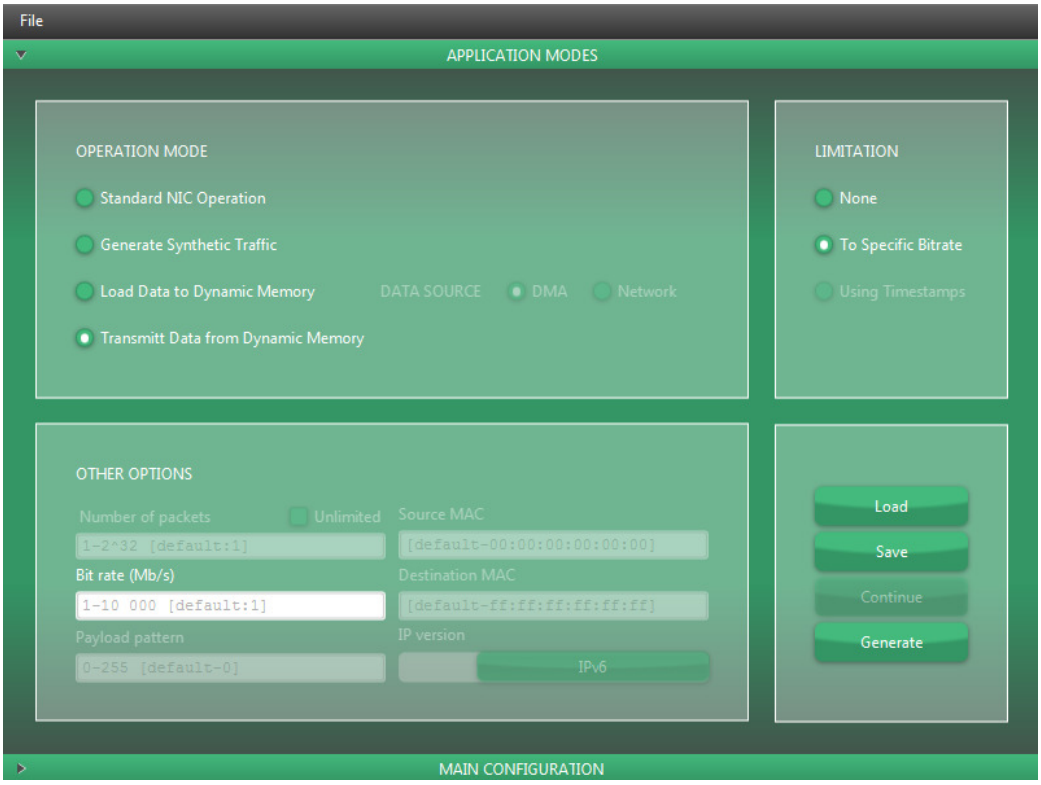
Pole s názvem Bitrate určuje přenosovou rychlost, se kterou se mají data vysílat zpět do sítě. Miminum je 1 Mb/s, maximum 10 Gb/s. (Konfiguraci zachycuje obrázek A.11).

Limitace na základě časových známek

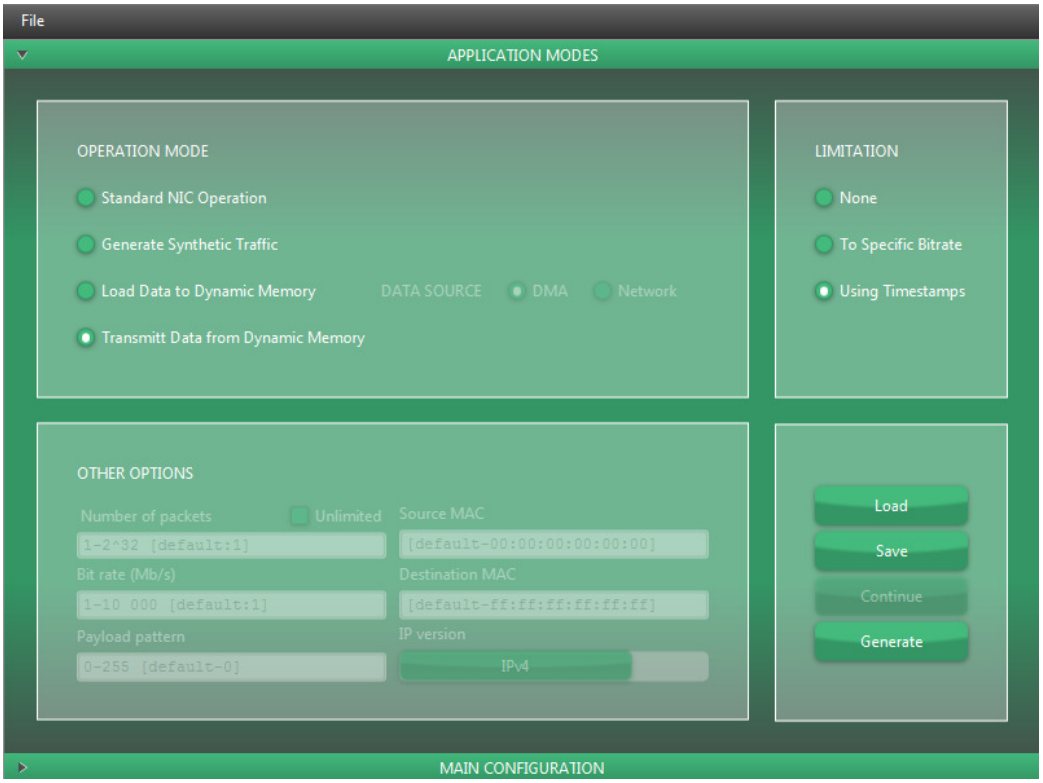
Tato varianta slouží k demonstraci původní zachycené komunikace, tentokrát takové, která byla dříve uložena do dynamické paměti karty. Nastevní odpovídá obrázek A.12.



Obrázek A.10: Vysílání dat z dynamické paměti - bez limitace



Obrázek A.11: Vysílání dat z dynamické paměti - limitace přenosovou rychlostí



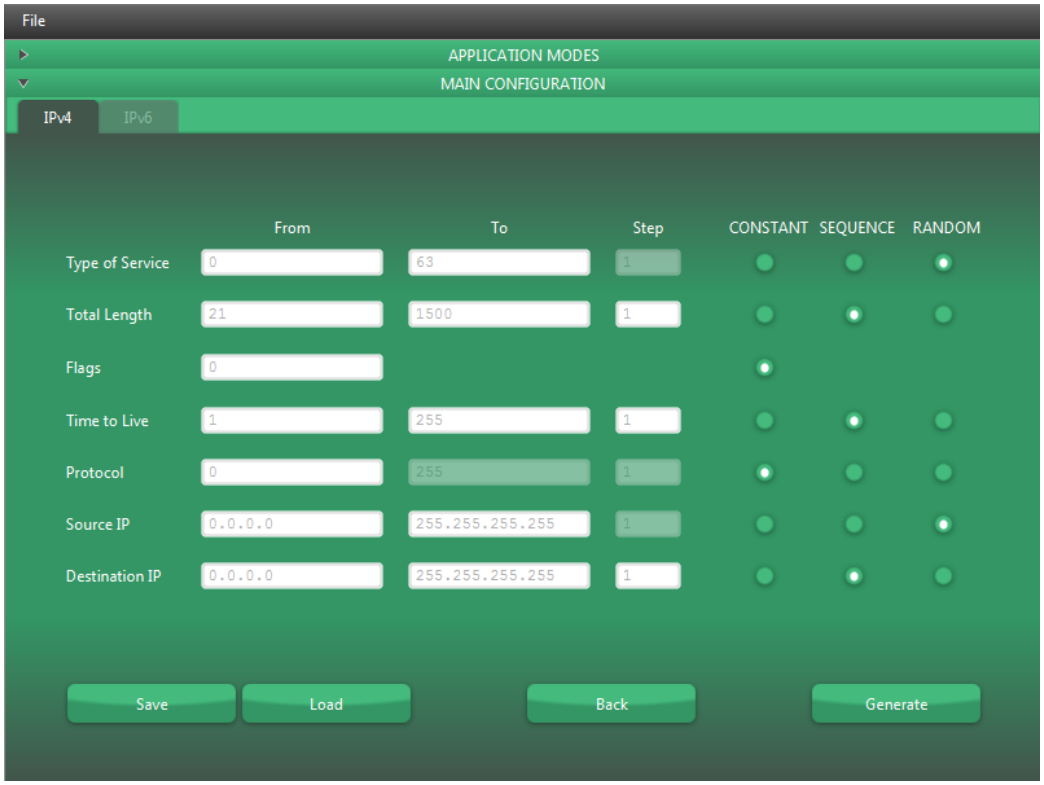
Obrázek A.12: Vysílání dat z dynamické paměti - limitace časovými známkami

A.1.5 Karty pro nastavení polí hlaviček paketů

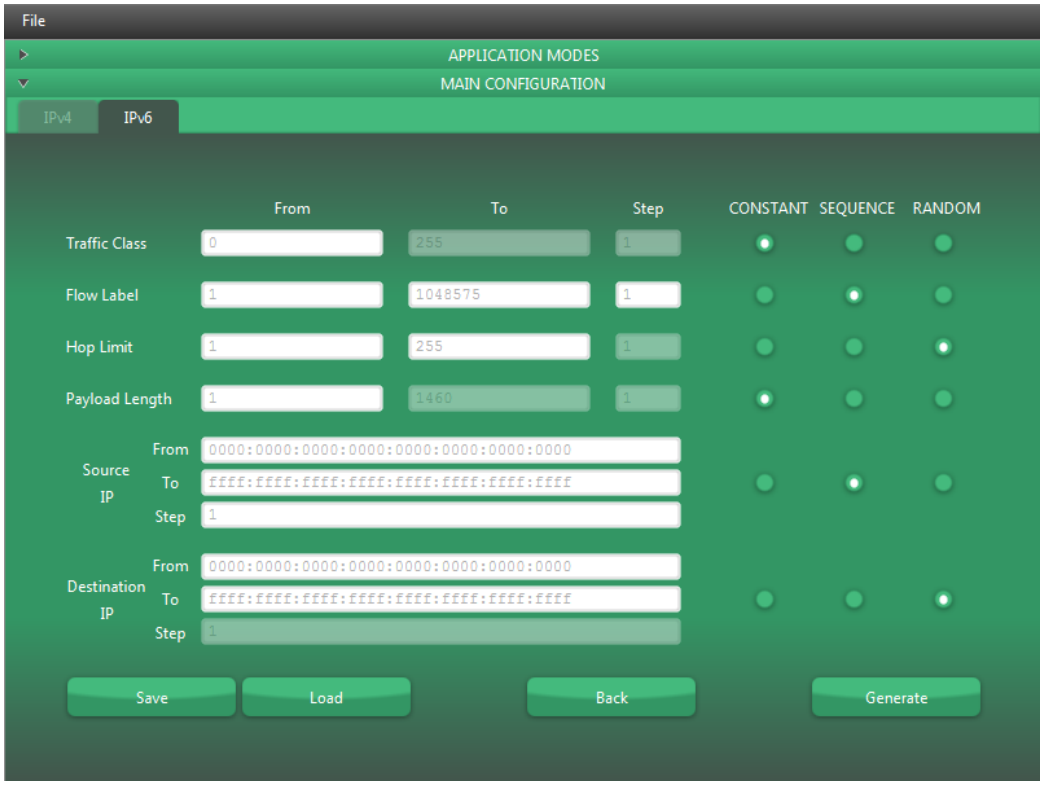
Tato konfigurace je pokračováním nastavení v módu generování umělého síťového provozu. Dle přepínačů se určuje typ generování hodnot zapisovaných do hlaviček paketů. Může se jednat o konstantní hodnotu, či více hodnot z daného rozsahu, a to náhodně, či sekvenčně po určitém kroku.

Pokud zůstanou hodnoty některých textových polí nevyplněné, zapíše se do konfiguračního souboru hodnota výchozí. Ta je v každém poli zobrazena šedou barvou a zároveň se jedná o reprezentaci možného rozsahu, který může uživatel do pole zadat.

Obrázek A.13 zobrazuje kartu pro nastavení polí hlavičky IPv4, obrázek A.14 je obdobou pro protokol IP verze 6.



Obrázek A.13: Karta pro nastavení polí hlaviček paketů - IPv4



Obrázek A.14: Karta pro nastavení polí hlaviček paketů - IPv6

A.2 Validace zadávaných vstupních hodnot

Při každém zapsaném znaku do textového pole je prováděna kontrola správnosti celého řetězce.

A.2.1 Hodnoty

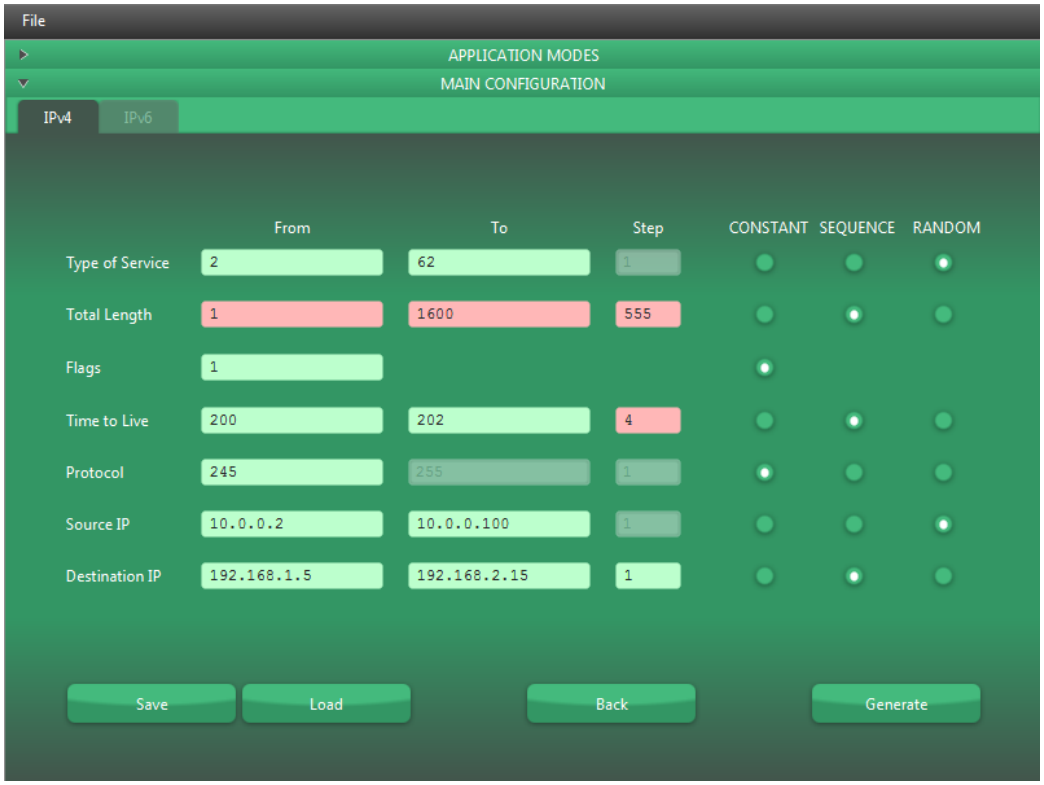
Tabulka A.1 popisuje příklady hodnot, které se mohou objevit v těchto polích a chování aplikace v každém z těchto případů.

Typ vstupu	Vstup od uživatele	Reprezentace
0–255 výchozí 0	5	5
	`[bílý znak]`	0
	2005	200
	abc	0
	1a2b3c	1
	2 b následně smazány oba znaky	2
MAC Adresa výchozí 00:00:00:00:00:00	11:22:33:44:55:66	11:22:33:44:55:66
	aa-bb-cc-dd-ee-ff	aa:bb:cc:dd:ee:ff
	`[bílý znak]`	00:00:00:00:00:00
	11:22:33:44:55-66	00:00:00:00:00:00
	11:22:33:44:55:66:77	11:22:33:44:55:66
	11:22:33:44:55-66:77	00:00:00:00:00:00
	11:22:33:44:55	00:00:00:00:00:00
IPv4 Adresa výchozí 0.0.0.0	100.200.300.400	100.200.300.400
	100.200.300.400.500	100.200.300.400
	100.200.300	0.0.0.0
	0xAABBCCDD	0.0.0.0
	AABBCCDD	0.0.0.0
	`[bitová podoba]`	0.0.0.0

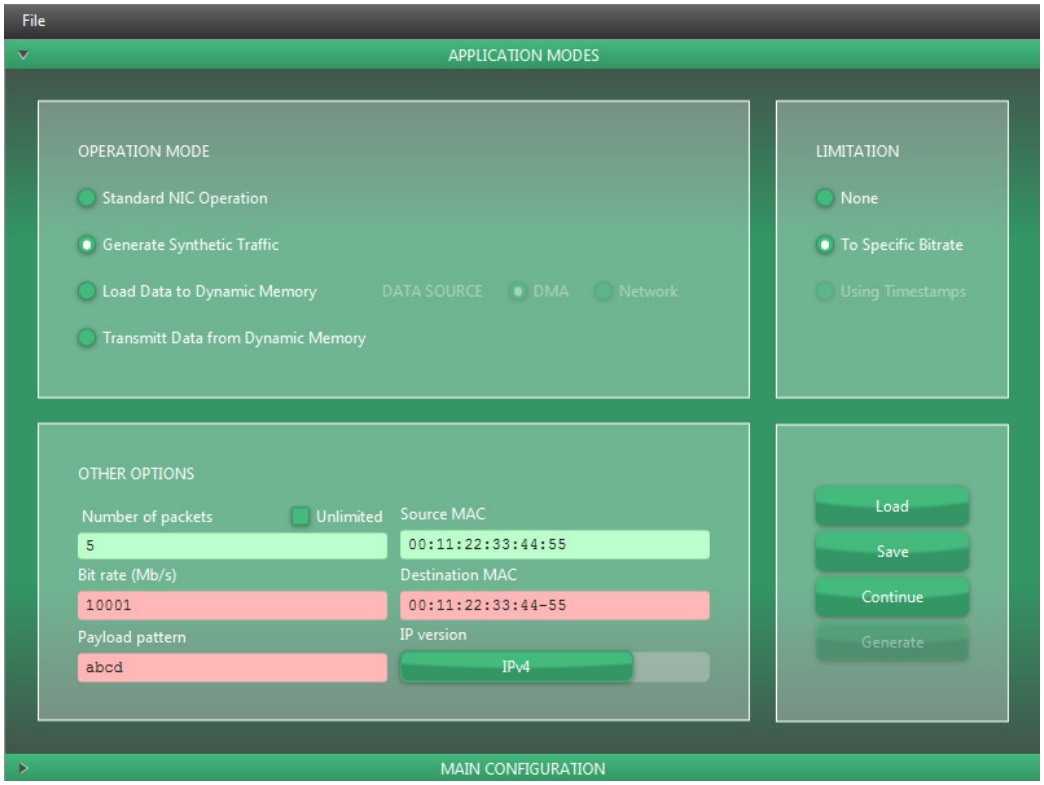
Tabulka A.1: Příklady vstupních hodnot

A.2.2 Grafická podoba

Na obrázcích A.15 a A.16 je zobrazen způsob podání informace uživateli o správnosti zadaných hodnot - tedy červeným a zeleným podbarvením vyplněných hodnot v textových polích na konkrétních situacích.



Obrázek A.15: Demonstrace kontroly vstupních hodnot v kartě IPv4

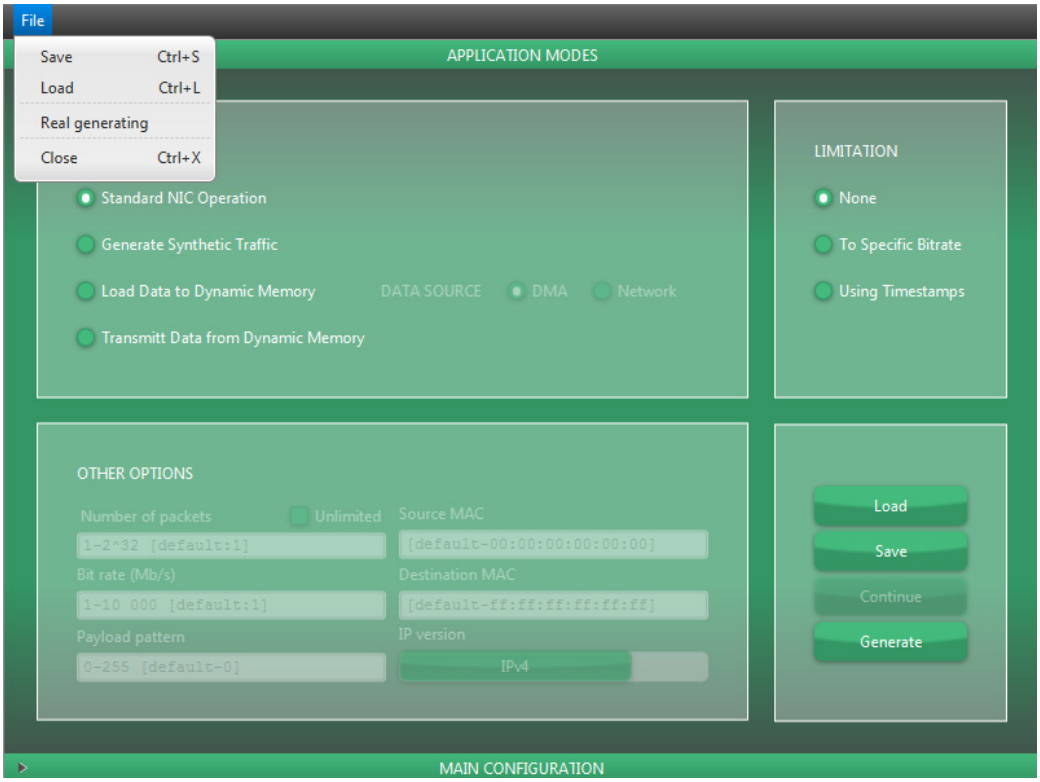


Obrázek A.16: Demonstrace kontroly vstupních hodnot v hlavním okně

A.3 Načítání a ukládání konfigurace

Ve všech oknech programu je možnost uložit si a načíst konfiguraci. Uživatel může tyto akce vyvolat těmito způsoby

- Stiskem tlačítek Load, či Save v daném okně
- Výběrem těchto položek z menu (jak je na obrázku A.17)
- Použitím klávesových zkratek `Ctrl+L` pro načtení a `Ctrl+S` pro uložení konfigurace



Obrázek A.17: Možnosti pro uložení a načtení konfigurace

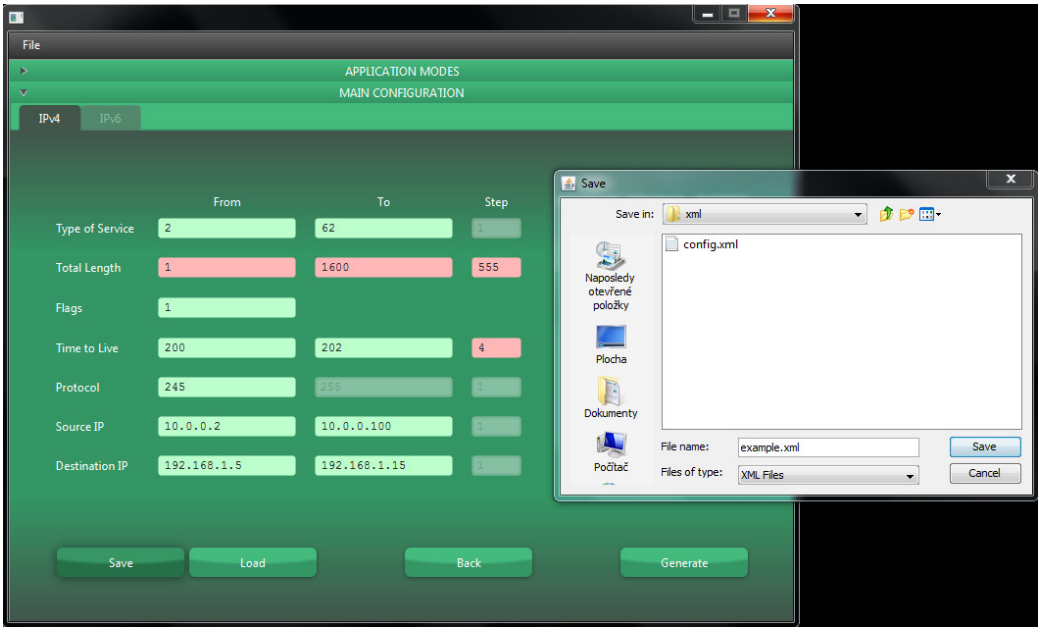
A.3.1 Uložení konfigurace

Obrázek A.18 ukazuje zapisovanou konfiguraci a zároveň dialog pro výběr souboru a cesty, kam se mají data zapsat. Formát souboru je XML, jenž je pro větší přehlednost zároveň filtrován při výběru existujících souborů. Tak může uživatel vidět jen soubory XML, složky a zástupce, namísto zobrazení souborů všech formátů.

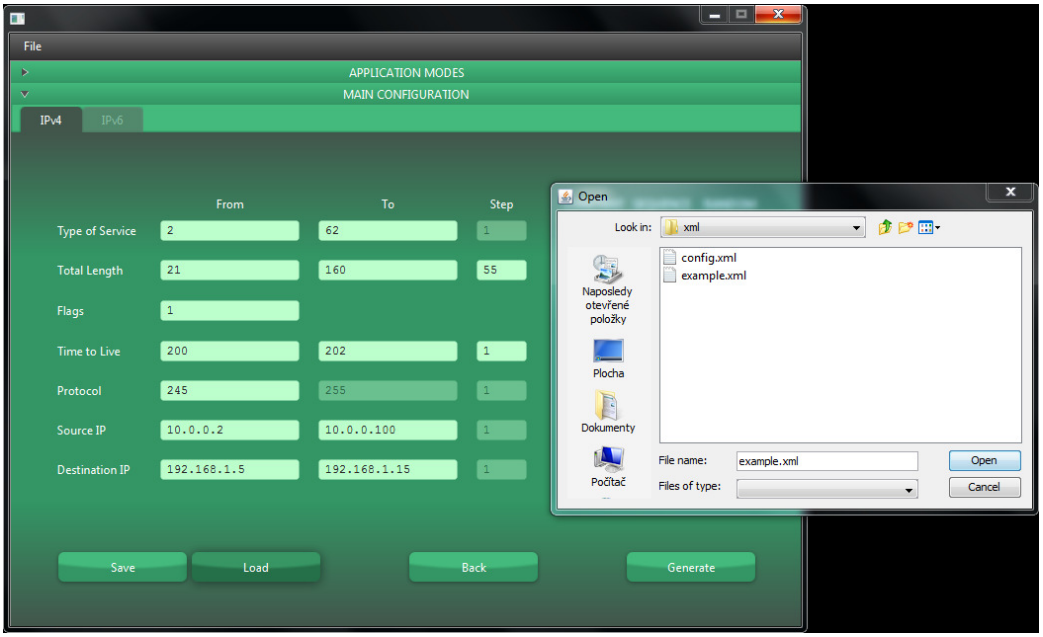
Záměrně byla nastavena konfigurace, která obsahuje chybně zadané údaje. I tak se ale konfigurace uloží s daty, která jsou validní, a uživatel dle tabulky 4.3 může zjistit, jakým způsobem se aplikace v těchto případech zachová.

A.3.2 Načtení konfigurace

Na obrázku A.19 je znázorněn dialog pro výběr XML souboru obsahujícího uloženou konfiguraci a zároveň data, která byla uložena dle obrázku A.18.



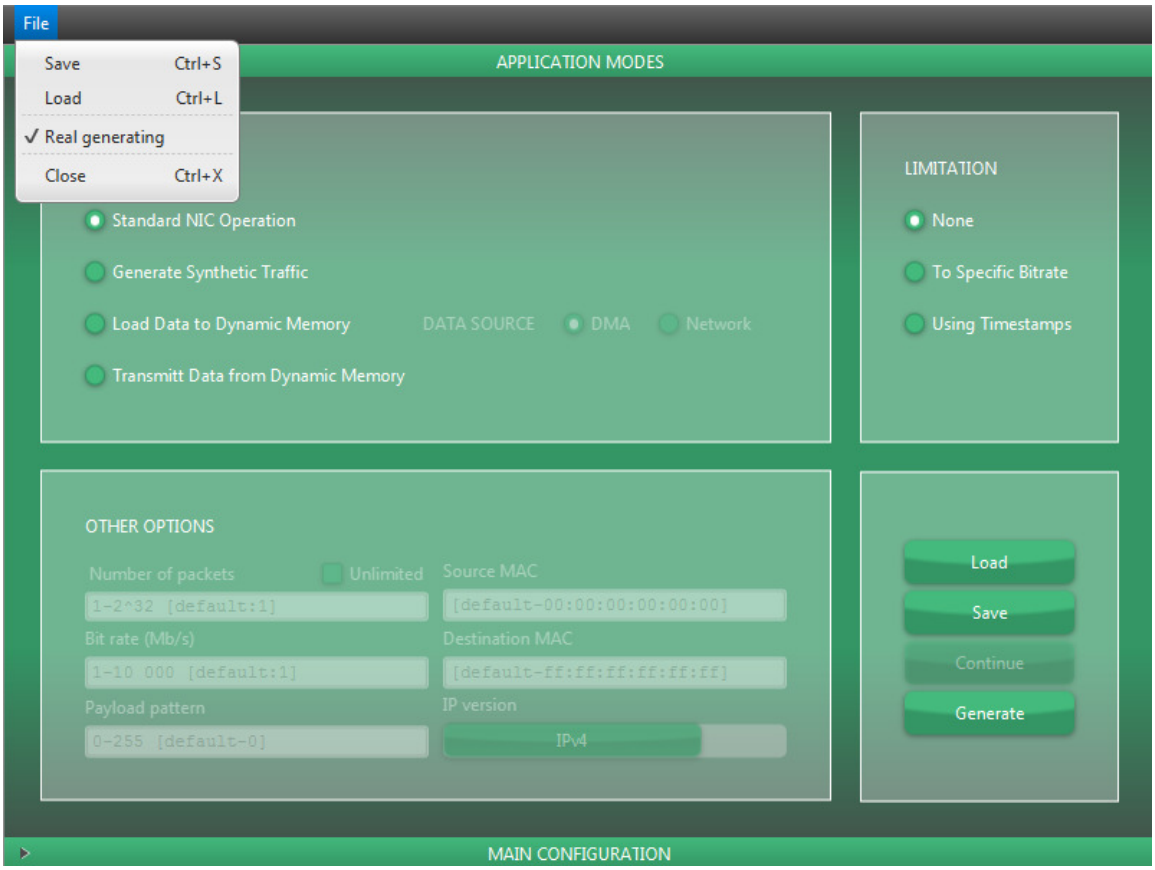
Obrázek A.18: Demonstrace ukládání konfigurace



Obrázek A.19: Demonstrace načítání konfigurace

A.4 Generování

V případě stisku tlačítka s názvem Generate se spustí zápis do souboru s názvem `config.xml`. Je-li aktivní položka `Real Generating` v menu, pak následuje otevření dialogu pro výběr aplikace, které se tento konfigurační soubor předá jako argument a aplikace je spuštěna. Očekávanou aplikací je v tomto případě `load_config`.



Obrázek A.20: Generování se spuštěním aplikace

Příloha B

Konfigurační soubory

B.1 Vzorový vstupní soubor `config.xml`

```
<? xml version="1.0" ?>
<!-- ========= Root Element =========  -->
<configuration>
   <!-- ========= General Registers Section  =========  -->
   <general_registers>
      <control_register          operation_mode           = "nic"
                                 limitation_mode          = "no"
                                 generated_ip             = "ipv4"
                                 data_source              = "dma" />

      <generated_packets_number  unlimited                = "yes"
                                 number                   = "0" />

      <source_mac                value                    = "0xAABBCCDDEEFF" />

      <destination_mac           value                    = "0xFFEEDDCCBBAA" />

      <payload_pattern           value                    = "0xAA" />

      <bit_rate                  value                    = "1000" />

      <generating_mode           ipv4_type_of_service     = "sequence"
                                 ipv4_source_address      = "constant"
                                 ipv4_destination_address = "random"
                                 ipv6_destination_address = "constant" />
   </general_registers>
```

```
<!-- ========= IPv4 Registers Section  =========  -->
<ipv4_registers>
   <type_of_service     from = "5"
                        to   = "15"
                        step = "2" />

   <total_length        from = "100"
                        to   = "1000"
                        step = "25"  />
   <flags               from = "1" />

   <time_to_live        from = "20"
                        to   = "250"
                        step = "10" />

   <protocol            from = "10"
                        to   = "192"
                        step = "15" />

   <source_address      from = "0xAABBCCDD"
                        to   = "0xFFEEDDCC"
                        step = "0x2" />

   <destination_address from = "0xAAAAAAAA"
                        to   = "0xCCCCCCCC"
                        step = "0x10" />
</ipv4_registers>
```

```
   <!-- ========= IPv6 Registers Section  =========  -->
   <ipv6_registers>
      <traffic_class       from = "0xF"
                           to   = "0x7F"
                           step = "2" />

      <flow_label          from = "0x55"
                           to   = "0xFFFF"
                           step = "10" />

      <payload_length      from = "10"
                           to   = "1000"
                           step = "15" />

      <hop_limit           from = "100"
                           to   = "200"
                           step = "10"  />

      <source_address      from = "0x00112233445566778899AABBCCDDEEFF"
                           to   = "0xFFEEDDCCBBAA99887766554433221100"
                           step = "0x00000000000000000000000000000010" />

      <destination_address from = "0x00001111222233334444555566667777"
                           to   = "0x88889999AAAABBBBCCCCDDDDEEEEFFFF"
                           step = "0x000000000000000000000000000000FF" />

   </ipv6_registers>
</configuration>
```

B.2 Definiční soubor `config_spec.xml`

```
<!--
  -  config_spec.xml: specification of all elements (together with their
  -                   attributes and possible values) which can be used within
  -                   an XML configuration
  -  Author: Jiri Matousek <xmatou06@stud.fit.vutbr.cz>
  -->
<? xml version="1.0" ?>
<!-- ========= Root Element =========  -->
<configuration>
   <!-- ========= General Registers Section  =========  -->
   <general_registers>
      <!--
        -  Control Register
        -  Possible attributes of this element specify different options which
        -  can be set in the Control Register. Each attribute can have only one
        -  of the listed values. Otherwise (or if an attribute is not
        -  specified), an option is set to the default value (the first value
        -  in the list).
        -->
      <control_register           operation_mode  = "nic|load|transmit|generate"
                                  limitation_mode = "no|bitrate|timestamps"
                                  generated_ip    = "ipv4|ipv6"
                                  data_source     = "dma|network"/>
      <!--
        -  Number of Packets to be Generated
        -  The "unlimited" attribute determines whether the number of generated
        -  packets will be unlimited (which is default behaviour) or only
        -  specified number of packets will be generated. When second
        -  possibility is chosen, the number of generated packets is specified
        -  as a value of the "number" attribute. Allowed values for this
        -  attribute are all numbers which can be binary encoded on 32 bits,
        -  except the value 0. When generating packets should be limited and
        -  the "number" attribute is not specified, the default value (all bits
        -  set to '1') is used. The "number" attribute can be specified as an
        -  octal literal (i.e. with 0 prefix), a decimal literal (i.e. without
        -  any prefix) or a hexadecimal literal (i.e. with 0x or 0X prefix).
        -->
        <generated_packets_number unlimited       = "yes|no"
                                  number          = "number" />
```

```
<!--
  -  Source MAC address
  -  Value of the "value" attribute represents a source MAC address of
  -  generated Ethernet frames. Allowed values for this attribute are all
  -  numbers which can be binary encoded on 48 bits. Default source MAC
  -  address contains 48 bits set to '0' and it is used when the "value"
  -  attribute is not specified. The "value" attribute has to be
  -  specified as a hexadecimal literal (i.e. with 0x or 0X prefix) which
  -  explicitly specifies value of all 48 bits of source MAC address.
  -  Otherwise, the default value is used.
  -->
<source_mac      value = "number" />
<!--
  -  Destination MAC address
  -  Value of the "value" attribute represents a destination MAC address
  -  of generated Ethernet frames. Allowed values for this attribute are
  -  all numbers which can be binary encoded on 48 bits. Default
  -  destination MAC address contains 48 bits set to '0' and it is used
  -  when the "value" attribute is not specified.  The "value" attribute
  -  has to be specified as a hexadecimal literal (i.e. with 0x or 0X
  -  prefix) which explicitly specifies value of all 48 bits of source
  -  MAC address. Otherwise, the default value is used.
  -->
<destination_mac value = "number" />
<!--
  -  Payload Pattern
  -  8-bit value of the "value" attribute is used for filling a payload
  -  of generated IPv4/IPv6 packets. When this attribute is not
  -  specified (or value which does not explicitly specifies exactly
  -  8 bits is used), the register is set to the default value (all 8
  -  bits set to '0'). The "value" attribute has to be specified as
  -  a hexadecimal literal (i.e. with 0x or 0X prefix).
  -->
<payload_pattern value = "number" />
<!--
  -  Bit Rate
  -  The attribute "value" specifies a bit rate for the case when the
  -  "limitation_mode" attribute of "control_register" element is set to
  -  the value "bitrate". Any integer number between 1 and 10000 is
  -  allowed. The value 10000 is also considered to be the default value
  -  used when the attribute "value" is missing. The "value" attribute
  -  can be specified as an octal literal (i.e. with 0 prefix), a decimal
  -  literal (i.e. without any prefix) or a hexadecimal literal (i.e.
  -  with 0x or 0X prefix).
  -->
<bit_rate        value = "number" />
```

```
    <!--
      -  Header Fields Generating Mode Register
      -  Each attribute of this element specifies a way of generating
      -  corresponding IPv4 or IPv6 header field. Each header field can be
      -  generated in one of three ways - as a constant (value "constant"),
      -  as a next number from a sequence specified by two limit values and
      -  increment size (value "sequence") or as a random nuber from a range
      -  given by two limit values (value "random"). Value of each attribute
      -  can be set independently and the default value is "constant".
      -->
    <generating_mode ipv4_type_of_service     = "constant|sequence|random"
                     ipv4_total_length        = "constant|sequence|random"
                     ipv4_flags               = "constant|sequence|random"
                     ipv4_time_to_live        = "constant|sequence|random"
                     ipv4_protocol            = "constant|sequence|random"
                     ipv4_source_address      = "constant|sequence|random"
                     ipv4_destination_address = "constant|sequence|random"
                     ipv6_traffic_class       = "constant|sequence|random"
                     ipv6_flow_label          = "constant|sequence|random"
                     ipv6_payload_length      = "constant|sequence|random"
                     ipv6_hop_limit           = "constant|sequence|random"
                     ipv6_source_address      = "constant|sequence|random"
                     ipv6_destination_address = "constant|sequence|random" />
</general_registers>
```

```
<!-- ========= IPv4 Registers Section  =========  -->
<!--
  -  Each element in this section contains three attributes defining values
  -  important in generating corresponding IPv4 header field. The only
  -  exception is the Flags field, where only one value is enough. Each
  -  attribute corresponds to one 32-bit register and comments preceding
  -  particular example elements precisely specify utilization of these 32
  -  bits. All attributes of elements in this section can be specified as an
  -  octal literal (i.e. with 0 prefix), a decimal literal (i.e. without any
  -  prefix) or a hexadecimal literal (i.e. with 0x or 0X prefix). The only
  -  exception are IPv4 addresses, where attributes have to be specified as
  -  a hexadecimal literal. When attribute is not specified, all utilized
  -  bits of the corresponding register are set to 0 or to a minimal allowed
  -  value if 0 is not allowed.
  -->
<ipv4_registers>
   <!--
     -  IPv4 Type of Service Registers
     -  Value of attributes of the "type_of_service" element are stored on
     -  6 lowest bits of corresponding registers. Thus, allowed value of
     -  these attributes are all numbers which can be binary encoded on 6
     -  bits.
     -->
   <type_of_service from = "number"
                    to   = "number"
                    step = "number" />
   <!--
     -  IPv4 Total Length Registers
     -  Attributes' value are stored on 11 lowest bits of corresponding
     -  registers. Allowed values for the "from" attribute and the "to"
     -  attribute are integer numbers between 21 and 1500. Allowed values
     -  for the "step" attribute are integer numbers between 0 and 1479.
     -->
   <total_length    from = "number"
                    to   = "number"
                    step = "number" />
   <!--
     -  IPv4 Flags Registers
     -  The value of the "from" attribute is stored on only 1 bit of the
     -  corresponding regster, so there are only two allowed values: 0 and
     -  1.
     -->
   <flags           from = "number" />
```

```
    <!--
      -  IPv4 Time to Live Registers
      -  Attributes' value are stored on 8 lowest bits of corresponding
      -  registers. For attributes "from" and "to", allowed values are all
      -  numbers which can be binary encoded on 8 bits, except the value 0.
      -  Allowed values for the attribute "step" are all numbers which can be
      -  binary encoded on 8 bits.
      -->
    <time_to_live        from = "number"
                         to   = "number"
                         step = "number" />
    <!--
      -  IPv4 Protocol Registers
      -  Value of attributes of the "protocol" element are stored on 8 lowest
      -  bits of corresponding registers. Thus, allowed value of these
      -  attributes are all numbers which can be binary encoded on 8 bits.
      -->
    <protocol            from = "number"
                         to   = "number"
                         step = "number" />
    <!--
      -  IPv4 Source Address Registers
      -  Since registers corresponding to attributes of this element are
      -  fully utilized, all values which can be binary encoded on 32 bits
      -  are allowed. Value of the "from" and "to" attributes have to
      -  explicitly specify all 32 bits.
      -->
    <source_address      from = "number"
                         to   = "number"
                         step = "number" />
    <!--
      -  IPv4 Destination Address Registers
      -  Since registers corresponding to attributes of this element are
      -  fully utilized, all values which can be binary encoded on 32 bits
      -  are allowed. Value of the "from" and "to" attributes have to
      -  explicitly specify all 32 bits.
      -->
    <destination_address from = "number"
                         to   = "number"
                         step = "number" />
</ipv4_registers>
```

```
<!-- ========= IPv6 Registers Section  =========  -->
<!--
  -  Each element in this section contains three attributes defining values
  -  important in generating corresponding IPv6 header field. Value of each
  -  attribute is stored in separate 32-bit register and utilization of
  -  particular bits of this register is precisely specified in comment
  -  preceding specific element. The only exception from currently mentioned
  -  rule are value of attributes of the "source_address" and the
  -  "destination_address" elements which are stored in four 32-bit
  -  registers. All attributes of elements in this section, except
  -  attributes of the "source_address" and the "destination_address"
  -  elements, can be specified as an octal literal (i.e. with 0 prefix),
  -  a decimal literal (i.e. without any prefix) or a hexadecimal literal
  -  (i.e. with 0x or 0X prefix). Source and destination IPv6 addresses can
  -  be specified only as a hexadecimal literal. When attribute is not
  -  specified, all utilized bits of the corresponding register are set to
  -  0 or to a minimal allowed value if 0 is not allowed.
  -->
<ipv6_registers>
   <!--
     -  IPv6 Traffic Class Registers
     -  Value of attributes of the "traffic_class" element are stored on
     -  8 lowest bits of corresponding registers. Thus, allowed value of
     -  these attributes are all numbers which can be binary encoded on
     -  8 bits.
     -->
   <traffic_class from = "number"
                  to   = "number"
                  step = "number" />
   <!--
     -  IPv6 Flow Label Registers
     -  Value of attributes of the "flow_label" element are stored on
     -  20 lowest bits of corresponding registers. Thus, allowed value of
     -  these attributes are all numbers which can be binary encoded on
     -  20 bits.
     -->
   <flow_label    from = "number"
                  to   = "number"
                  step = "number" />
```

```
      <!--
        -  IPv6 Payload Length Registers
        -  Attributes' value are stored on 11 lowest bits of corresponding
        -  registers. Allowed values for the "from" attribute and the "to"
        -  attribute are integer numbers between 1 and 1460. Allowed values
        -  for the "step" attribute are integer numbers between 0 and 1459.
        -->
      <payload_length      from = "number"
                           to   = "number"
                           step = "number" />
      <!--
        -  IPv6 Hop Limit Registers
        -  Attributes' value are stored on 8 lowest bits of corresponding
        -  registers. For attributes "from" and "to", allowed values are all
        -  numbers which can be binary encoded on 8 bits, except the value 0.
        -  Allowed values for the attribute "step" are all numbers which can be
        -  binary encoded on 8 bits.
        -->
      <hop_limit           from = "number"
                           to   = "number"
                           step = "number" />
      <!--
        -  IPv6 Source Address Registers
        -  Since all four registers corresponding to attributes of this element
        -  are fully utilized, all values which can be binary encoded on
        -  128 bits are allowed. Provided values of attributes have to
        -  explicitly specify all 128 bits. Othrwise, the default value is
        -  used.
        -->
      <source_address      from = "number"
                           to   = "number"
                           step = "number" />
      <!--
        -  IPv6 Destination Address Registers
        -  Since all four registers corresponding to attributes of this element
        -  are fully utilized, all values which can be binary encoded on
        -  128 bits are allowed. Provided values of attributes have to
        -  explicitly specify all 128 bits. Othrwise, the default value is
        -  used.
        -->
      <destination_address from = "number"
                           to   = "number"
                           step = "number" />
   </ipv6_registers>
</configuration>
```

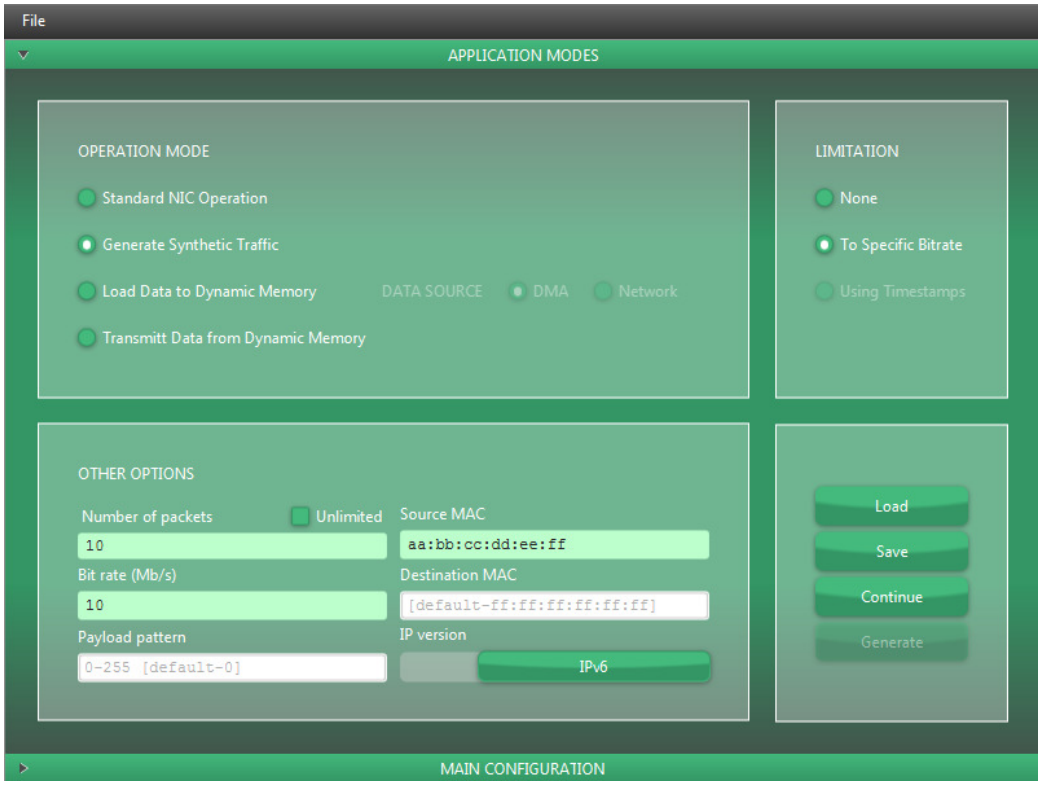
Příloha C

Dotazník pro testování

Vyberte prosím jednu z kategorií, do které spadáte

- Student informatiky na vysoké škole
- Informatik povoláním
- Uživatel v zaměstnání (i OSVČ) mimo obor
- Student vysoké školy jiného zaměření
- Student střední školy

Kolik sekund Vám zabere následující nastavení konfigurace?



- 0-10
- 11-15
- 16-20
- 21-25
- 26-30
- 31-35
- 36-40
- 40 a více

Kolik sekund Vám zabere následující nastavení konfigurace?

- 0-10
- 11-15
- 16-20
- 21-25
- 26-30
- 31-35
- 36-40
- 40 a více

Jak se Vám líbí celkový design aplikace?

- 1 hvězda
- 2 hvězdy
- 3 hvězdy
- 4 hvězdy
- 5 hvězd

Působí na Vás uspořádání prvků rozhraní aplikace přehledně?

- Ano
- Ne

Oceňujete okamžitou kontrolu správnosti zadaných údajů, nebo byste raději napsali celý text a poté jej teprve zkontrolovali?

- Okamžitá kontrola mi vyhovuje
- Závěrečná kontrola mi vyhovuje

Pokuste si uložit a načíst konfiguraci. Proběhlo vše v pořádku?

- Ano
- Ne. Co se stalo?

Měli jste problém s instalací programu?

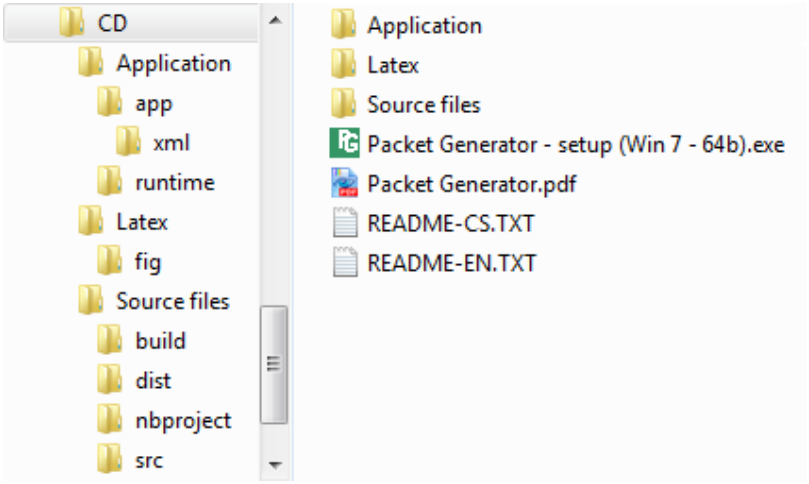
- Ne
- Ano. Jaký?

Máte-li jiné připomínky, tipy, či rady, uveďte je prosím zde.

Příloha D

Obsah CD

V této příloze jsou popsány jednotlivé soubory a složky, které jsou obsaženy na přiloženém CD. Na obrázku D.1 je znázorněna struktura složek a zároveň soubory uložené přímo ve složce CD. Tato práce je zde uložena pod názvem `Packet Generator.pdf`.



Obrázek D.1: Struktura souborů a složek uložených na CD

D.1 Aplikace

Ve složce `Application` jsou uloženy veškeré soubory potřebné pro běh programu, tak, jak jsou nainstalovány pomocí souboru `Packet Generator - setup (Win 7 - 64b)`. Konkrétně se jedná o spustitelný soubor `Packet Generator.exe`, ikonu `Packet Generator.ico` a soubory pro odinstalování aplikace `uninstall.exe` a `uninstall.dat`. Dále jsou zde obsaženy složky `app`, kde lze nalézt spustitelný soubor `Bakalarka.jar`, jeho konfigurační soubor `package.cfg` a složku `xml`. V této složce jsou vzorové konfigurační soubory pro aplikaci označeny jako `example[číslo]` a soubor `config.xml`, do kterého se ukládá konfigurace při generování.

D.2 Instalační soubor

Pro tutu aplikaci byl vytvořen instalační soubor s názvem `Packet Generator - setup (Win 7 - 64b)`. Při samotné instalaci má uživatel na výběr z instalačního jazyka (čestina, angličtina), umístění souboru, zdali chce vytvořit položku v menu start, zdali chce vytvořit zástupce na ploše a na závěr, zda si přeje spustit nainstalovanou aplikaci.

D.3 Zdrojové soubory

Zdrojové soubory aplikace jsou uloženy ve složce `Source files`. Ta mimojiné obsahuje složku obsahující data pro projekt v prostředí NetBeans `nbproject`, různé elementární instalační a spouštěcí balíky (v podsložce `dist`), soubor `build.xml` sloužící k sestavení aplikace, ikonu programu (`Packet Generator.ico`) a skript s názvem `setupScript.iss` sloužící pro program Inno Setup Compiler, pomocí něhož se tvoří pokročilý instalační soubor.

D.4 Soubory pro Latex

Ve složce `Latex` jsou obsaženy všechny potřebné soubory pro tvorbu textu bakalářské práce, ať už se jedná o obrázky, soubory s příponou `.tex`, citační šablonu, šablonu bakalářské práce, Makefile, apod.

D.5 Soubory README

Kořenový adresář zároveň obsahuje soubory určené ke čtení, které obsahují popis, jak aplikaci přeložit, jaký je obsah CD, a další. Jedná se o soubory `README-CS.TXT` v českém jazyce a `README-EN.TXT` v jazyce aglickém.