VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÝ NÁSTROJ PRO GENEROVÁNÍ PAKETŮ

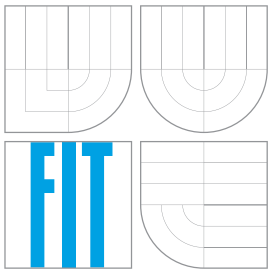
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MAREŠ

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GRAFICKÝ NÁSTROJ PRO GENEROVÁNÍ PAKETŮ

GRAPHICAL TOOL FOR PACKET GENERATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN MAREŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VIKTOR PUŠ
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá problematikou generátorů síťových paketů. První část je věnována teorii komunikace v počítačových sítích z pohledu architektury TCP/IP. Druhá část je věnována výsledkům rešerše aktuálně dostupných nástrojů. Třetí část je věnována návrhu a implementaci vlastního nástroje s grafickým uživatelským rozhraním pro generování paketů. Nástroj je navržen objektovým způsobem s důrazem na snadnou možnost budoucího rozšíření. Koncept nástroje zahrnuje, kromě možnosti generovat jednotlivé pakety, také prostředky pro generování jejich dynamicky se měnících sérií za účelem simulace síťových toků. Navržený nástroj je implementován pomocí programovacího jazyka C++ s využitím frameworku Qt. Poslední část práce obsahuje zhodnocení dosažených výsledků a porovnání vytvořené aplikace s již dostupnými řešeními.

Abstract

This bachelor's thesis is about network packet generators. The first part is devoted to communication theory in computer networks from the perspective of TCP/IP architecture. The second part describes the results of search for currently available tools. The third part is devoted to design and implementation of own tool with graphical user interface for packets generation. The tool is designed object-oriented with an emphasis on ease of future expansion. The application concept includes also tools for generating dynamically changing series of packets to simulate network flows. The designed tool is implemented using the programming language C++ and the Qt framework. The last part contains an evaluation of results and comparison with the already available solutions.

Klíčová slova

Paket, Generátor, PCAP, PCAP-ng, TCP/IP, Protokoly, C++, Qt

Keywords

Packet, Generator, PCAP, PCAP-ng, TCP/IP, Protocols, C++, Qt

Citace

Martin Mareš: Grafický nástroj pro generování paketů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Grafický nástroj pro generování paketů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana inženýra Viktora Puše,

.....
Martin Mareš
7. května 2012

Poděkování

Rád bych poděkoval svému vedoucímu, Ing. Viktoru Pušovi, za vedení práce a poskytnuté věcné připomínky, které mi pomohly při jejím zpracování.

Obsah

Kapitola 1

Úvod

Tato práce se zaměřuje na problematiku softwarových nástrojů pro generování paketů.

V dnešení době si již počítač bez možnosti komunikace s okolím téměř nedokážeme představit. Drtivá většina služeb, které uživatelé na počítačích používají, vyžaduje výměnu dat s okolním světem. Tato výměna probíhá pomocí tzv. počítačových sítí. Nejrozšířenějším typem počítačových sítí jsou dnes bezesporu sítě založené na přepínání paketů. Veškerá informace v těchoto sítích je přenášena po částech, jako obsah tzv. síťových paketů. Komunikace na úrovni paketů by ale byla pro aplikace příliš složitá a příliš závislá na cílové platformě. Z tohoto důvodu poskytují operační systémy nástroje pro síťovou komunikace na abstraktnější úrovni. Komunikující aplikace poté ve většině případů pouze zapisují a přijmají data jako souvislý datový tok. Jádro operačního systému poté zcela zodpovídá za rozdělení odesílané informace, její vložení do vytvářených paketů správného typu a jejich odeslání do sítě, při přijetí naopak za její zkompletování a předání správné aplikaci. Za standartních podmínek je díky tomuto síťová komunikace vcelku jednoduchou, zvládnutou a spolehlivou technologií. I zde se ale ve specifických nebo náročných podmínkách vyskytují problémy, které je třeba efektivně řešit.

Při odstraňování chyb, vývoji nových nebo vylepšování stávajících síťových technologií je třeba mít k dispozici sadu kvalitních nástrojů a pomůcek. Základní pomůckou pro tyto účely jsou tzv. analyzátory síťového provozu. Tyto nástroje umožňují zachytit síťový provoz na zadaném rozhraní a přehledně zobrazit obsah obsažených paketů. Ve specifických případech je ale třeba také mít k dispozici nástroj umožňující vytvoření manuálně definovaného (například i nesprávného nebo netypického) paketu a jeho odeslání do sítě. Tímto způsobem lze například opakově simulovat určitou situaci, při které odstraňovaný problém vzniká, nebo sledovat odezvu systému na řízeně měnící se podmínky. Zatímco kvalitních síťových analyzátorů existuje velké množství, u generátorů síťových paketů je situace horší.

Cílem této práce bylo provést rešerši dostupných knihoven, již existujících nástrojů a poté navrhnout a implementovat systém pro generování síťových paketů. Základními požadavky pro výsledný systém byly : grafické uživatelské rozhraní, důraz na co nejširší podporu protokolů, možnost ukládat pakety do souboru formátu *PCAP*. Následně bylo třeba porovnat dosažené výsledky s již existujícími řešeními a dosažené výsledky zhodnotit.

1.1 Přehled kapitol

Druhá kapitola se věnuje obecné problematice TCP/IP sítí a jejich architektuře vzhledem k referenčnímu ISO/OSI modelu. Popisuje její vrstvy a základní funkce, které musí

plnit. Třetí kapitola obsahuje popis základních protokolů TCP/IP a jejich použití v kontextu síťové komunikace. Čtvrtá kapitola uvadí výsledky rešerše již existujících knihoven a nástrojů pro generování síťových paketů. Pátá kapitola se věnuje návrhu nástroje v návaznosti na stanovené požadavky. Je zde popsán základní koncept aplikace a použitý objektově orientovaný návrh. Šestá kapitola je věnována implementaci nástroje. Popisuje principy implementace, základní aplikační třídy a nejdůležitější procesy aplikační logiky. Kapitola sedm popisuje dosažané výsledky a jejich srovnání s možnostmi již existujících nástrojů. Kapitola osm obsahuje závěr práce.

Kapitola 2

Architektura TCP/IP

Komunikace v počítačových sítích je velmi složitá a při jejím řešení je nutno uplatnit metodu dekompozice na podproblémy. Vzhledem k velké variabilitě požadavků na komunikaci se zde uplatňuje hiearchická dekompozice na tzv. vrstvy. Síťová komunikace jako celek je tedy popsána pomocí tzv. *vrstvového modelu*. Každá vrstva v dané síťové architektuře zajišťuje určité funkce komunikace. Tyto funkce (úkoly) jsou realizovány různými protokoly dané vrstvy. Při zajišťování těchto funkcí využívá služeb nižších vrstev a zároveň poskytuje abstrakci směrem k vyšším vrstvám. Sousední vrstvy spolu komunikují pomocí rozhraní, odpovídající vrstvy spolu komunikují pomocí protokolů. Každá z vrstev používá pojem *základní přenosová jednotka - PDU*. *PDU* označuje jednotku, která je přenášena při komunikaci odpovídajících vrstev. Dekompozice komunikace pomocí vrstvového modelu umožňuje také lepší využití metod softwarového inženýrství a znovupoužitelnost již hotových řešení.

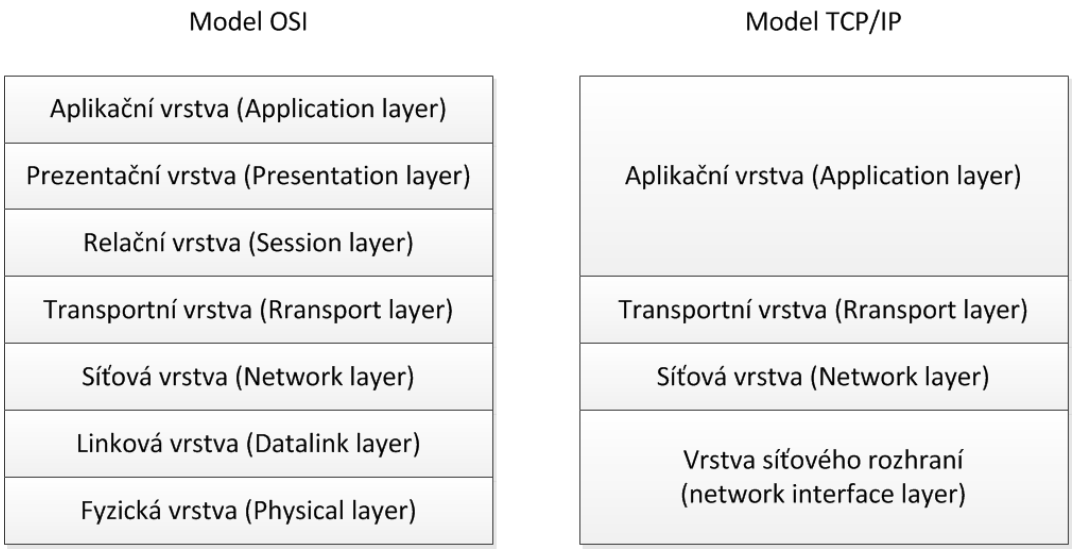
Pro jednotnou a obecnou definici funkcí, které jsou zapotřebí při datové komunikaci po síti mezi různými zařízeními, byl v roce 1987 vytvořen oraganizací ISO model *Open Systems Interconnection - OSI*. Tento model definuje sedm vrstev - viz obr. 2.1. Jeho implementace je ale natolik složitá, že v praxi nebyl nikdy plně nasazen. Naopak dnes nejrozšířenější se stala implementace modelu TCP/IP, která je využívána pro komunikaci v síti *Internet*.

Hlavní výhodnou modelu TCP/IP je jeho výrazná jednoduchost oproti modelu ISO/OSI. Porovnání vrstev obou modelů je na obrázku 2.1. Model TCP/IP spojuje funkce *fyzické* a *linkové* vrstvy do jedné *vrstvy síťového rozhraní* a funkce *aplikační*, *prezentační* a *relační* vrstvy do jediné *aplikační vrstvy*. Také koncepce obou modelů vykazuje určité odlišnosti. TCP/IP například využívá strategie *Best Effor delivery*, která klade při zajištění spolehlivého doručení zodpovědnost na koncové stanice, což umožňuje jednodušší a rychlejší fyzickou implementaci sítí.

Nyní následuje popis jednotlivých vrstev modelu TCP/IP.

2.1 Vrstva síťového rozhraní (Network Interface Layer)

Tato vrstva zajišťuje přístup k fyzickému médiu a zajišťuje přenos informace (jednotlivých bitů) po něm. Může se jednat například o metalický kabel, optické vlákno nebo vzduch v případě rádiového spoje. Standardy pro tuto vrstvu popisují fyzické vlastnosti linky - konektory, napěťové charakteristiky, kódování binární informace na určitou úroveň napětí, časový průběh jednotlivých signálů atd. Vrstva síťového rozhraní poskytuje jednotné rozhraní pro přenos datagramů síťové vsrtvy nezávislé na daném médiu. *PDU* této vrstvy se názývá *rámec*. Dnes nejpoužívanější technologií pro budování lokálních sítí na fyzické a linkové



Obrázek 2.1: Porovnání modelů OSI a TCP/IP

úrovni je *Ethernet*, dalšími mohou být např. *Token ring*, *FDDI*, nebo *X.25*.

2.2 Síťová vrstva (Internet Layer)

PDU na této vrstvě se nazývá datagram. Síťová vrstva zajišťuje jejich směrování (doručení) i přes různé lokální sítě - vytváří logické spojení uzlů. Každý z datagramů putuje sítí nezávisle na ostatních, tzn. není vytvářeno stálé spojení, které by předem určovalo jejich cestu. Jednotlivé datagramy patřící ke stejné komunikaci mohou sítí cestovat od zdroje k cíli různými cestami. Datagram obsahuje vlastní uživatelská data (anglicky *payload*), ale také metainformace (zejména adresy) potřebné pro jejich doručení do místa určení. Adresování na této vrstvě používá 32 bitové IP verze 4 adresy, nebo nověji 128 bitové IP verze 6 adresy - více o těchto adresách v kapitole 3.2.3 IPv4 a 3.2.4. IPv6.

2.3 Transportní vrstva (Transport layer)

Transportní vrstva zajišťuje logické spojení mezi dvěma koncovými komunikujícími aplikačními procesy. Jednotlivé aplikační procesy na stanici se stejnou IP adresou jsou na této vrstvě jednoznačně rozlišovány (adesovány) pomocí 16 bitového čísla portu. Transportní vrstva umožňuje dva základní typy komunikace: spolehlivý - spojově orientovaný přenos a nespolehlivý - přenost bez vytváření spojení. Spolehlivý přenos narozdíl od nespolehlivého využívá mechanismu *Sliding window*[36], který umožňuje automatickou regulaci velikosti toku dat, doručení ve správném pořadí a znovupřenesení dat v případě výpadku nebo jejich poruchy. Typ přenosu je při návrhu aplikace volen na základě různých specifických požadavků. Obecně lze ale říci, že nespojované služby transportní vrstvy využívají aplikace, které přenášejí menší množství dat v paketu, ale kladou důraz na jejich včasné doručení, jako jsou například multimediální aplikace nebo aplikace pro sledování a správu sítě. Spojově orientované služby jsou využívány většinou ostatních aplikací. Více je o protokolech zajišťujících oba zmíněné typy přenosu uvedeno v kapitole 3.3 Protokoly transportní vrstvy.

2.4 Aplikační vrstva (Application layer)

Aplikační vrstva je v architektuře TCP/IP tvořena konkrétními aplikačními procesy komunikujícími prostřednictvím počítačové sítě. Tato vrstva zajišťuje reprezetanci dat včetně jejich kódování. Aplikační vrstva data odesílá a přijímá jako souvislý datový tok. Aplikační protokoly jsou buď uživatelské, nebo systémové.

Kapitola 3

Protokoly TCP/IP

3.1 Protokoly vrstvy síťového rozhraní

3.1.1 Ethernet

Ethernet realizuje v architektuře TCP/IP kromě komunikačního protkolu pro vrstvu síťového rozhraní také standardy pro přenosová média, konektory, kódování bitů na elektrické signály, mechanismy pro detekci a zotavení z kolizí a další. Ethernet se stal oblíbeným hlavně pro svou jednoduchost a nízkou cenu oproti konkurenčním technologiím. Původně využíval Ethernet koaxiální kabel se sběrnicovou topologií, v současnosti je nejběžnější kroucená dvoulinka nebo optický kabel, využívající aktivní rozbočovače (*hub*) nebo přepínače (*switch*) - hvězdicová topologie [24]. Adresování v lokálních sítích je zajištěno pomocí MAC adresy.

Rámec se skládá z *preambule*, *adresy příjemce*, *adresy odesilatele*, *typu*, *vlastních dat* a *kontroloního součtu (CRC)* - obr 2.2. Ethenet je normalizovnován pomocí normy IEEE 802.3 [35].

MAC adresy

MAC (fyzické) adresy mají délku 48 bitů. Prvních 24 bitů je specifických pro výrobce síťových zařízení [27], zbylých 24 bitů je pak rozdílných pro různá zařízení stejného výrobce. Adresa se zapisuje ve tvrau osmi hexadecimálních číslic oddělených dvojtečkou. Speciální adresou určenou k doručení rámce všem zařízením v síti je adresa obsahující samé jedničky (FF:FF:FF:FF:FF:FF), označuje se jako *linkový broadcast*.

Preambule (Preamble) 8 bytu	Cíl (Destination) 6 bytu	Zdroj (Source) 6 bytu	Typ (Type) 2 byty	Data (Payload) 46-1500 bytu	Kontrolní součet (CRC) 4 byty

Obrázek 3.1: Ethernet - Rámec

3.2 Protokoly síťové vrstvy

Protkoly IP (IPv4 a IPv6), ARP, RARP, ICMP (ICMP a ICMPv6) a IGMP tvoří základní pětici protokolů síťové vrstvy nutnou pro komunikaci prostřednictvím TCP/IP sítě. Tyto protokoly musí být implementovány v operačním systému a jsou součástí tzv. sady protokolů internetu (*Internet Protocol Suite*) [37].

3.2.1 Protokol ARP

Tento protokol slouží k získání fyzické (MAC) adresy daného zařízení v lokální síti na základě znalosti jeho IP adresy (viz Protokol IPv4). Tato situace nastává např., pokud chceme odeslat IP datagram pomocí Ethernetu v lokální síti. K sestavení rámce totiž odesilatel kromě své IP a MAC adresy potřebuje znát také tyto údaje o příjemci. Pro zjištění MAC adresy hledané stanice odešle nejdříve odesilatel tzv. *ARP request*, ve kterém uvede svou MAC a IP adresu a IP adresu hledané stanice. Tento dotaz je zasílán na linkovou broadcastovou adresu. Stanice, která rozpozná svoji IP adresu v dotazu, odešle odpověď přímo odesilateli, který tak zjistí její MAC adresu. Kvůli úspoře přenosového pásma kapacity lokální sítě si uchovávají stanice dočasnou (cache) tabulku mapující IP adresy na MAC adresy. Protokol ARP je v systému IPv6 nahrazen zprávami protokolu ICMPv6 - více níže. Detailní specifikaci protokolu lze nalézt v RFC 826 [38].

3.2.2 Protokol RARP

Tento protokol má přesně opačné chování oproti protokolu ARP. K zadané MAC adrese se snaží získat IP adresu k ní přidruženou. Dříve se protokol RARP využíval pro automatickou konfiguraci IP adres stanic v lokálních sítích. Později byl ale nahrazen aplikačními protokoly *BOOTP* a *DHCP*, které kromě přidělovaní adres v rozsáhlejších sítích umožňují také přenos dalších informací, jako je síťová maska, adresa výchozí brány, adresa *DNS* serveru a podobně. Detailní specifikaci protkolu lze nalézt v RFC 903 [30].

3.2.3 Protokol IPv4

Internet protokol verze 4 (IPv4) je základní protokol síťové vrstvy využívaný k přenosu uživatelských dat. Jeho použití zajišťuje základní požadavky na tuto vrstvu architektury TCP/IP - směrování přenosu datové informace přes různé lokální sítě do místa určení specifikovaného pomocí IPv4 adresy. Protkol IPv4 nezjišťuje doručení dat, stejně jako jejich doručení ve správném pořadí (strategie *Best effor delivery*). Hlavička protokolu IPv4 je uvedena na obr. 3.2.

Bajty	0		1		2	3
0-3	Verze	Délka H.	DSCP	ECN	Celková délka	
4-7	Identifikace				Příznaky	Offset fragmentu
8-11	TTL		Protokol		Kontrolní součet hlavičky	
12-15	IPv4 adresa odesilatele					
16-19	IPv4 adresa příjemce					
20-23	(Volby)					
20/24+	Data					

Obrázek 3.2: Hlavička protokolu IPv4

Hlavička obsahuje kromě verze a její délky také :

- Zdrojovou a cílovou IPv4 adresu (2x32 bitů)
- Délku (16 bitů) - maximální délka IP datagramu je tedy 65535 bytů

- TTL - Time To Live (8 bitů) - zajišťuje ochranu proti směrovacím smyčkám - datagram je směrován pouze, pokud je tato hodnota větší než nula, přičemž na každém směrovači je tato hodnota snížena pávě o 1
- Protokol (8 bitů) - určuje, kterému protkolu vyšší vyrstvy má být datagram po přijetí předán ke zpracování - viz RFC 3232 [43]
- Kontrolní součet hlavičky (CRC) - tento součet je nutno při změne TTL přepočítat
- DSCP (Differentiated Services Code Point) - příznak pro zajištění *QoS*
- Identifikace (16 bitů), Příznaky (3 bity), Offset fragmentu (13 bitů) - slouží k řízení fragmentace datagramů (dnes málo využívané)

Detailní specifikace IPv4 lze naléz v dokumentu RFC 791 [41].

Adresy IPv4

Protokol IPv4 využívá 32bitové adresy. Adresy se zapisují ve tvaru čtveřice (4 x 8 bitů) dekadických čísel oddělených tečkou. Společně s adresou je třeba specifikovat také tzv. *síťovou masku*. Síťová maska určuje, kolik bitů z adresy má být použito k adresování sítě, zbytek adresy slouží k adresování stanice v dané síti. Pokud jsou bity adresující stanici nastaveny na samé 1, jedná se o sepeciální *broadcast adresu* - adresa pro všechny stanice v dané síti. Kromě běžných adres je rezervován prostor pro další typy adresy, např. pro multicast nebo pro experimentální účely. Prostor IPv4 adres spravuje společnost IANA[1], která jeho části deleguje na lokání admistrátory [31]. V současné době byly IPv4 adresy vyčerpány[33].

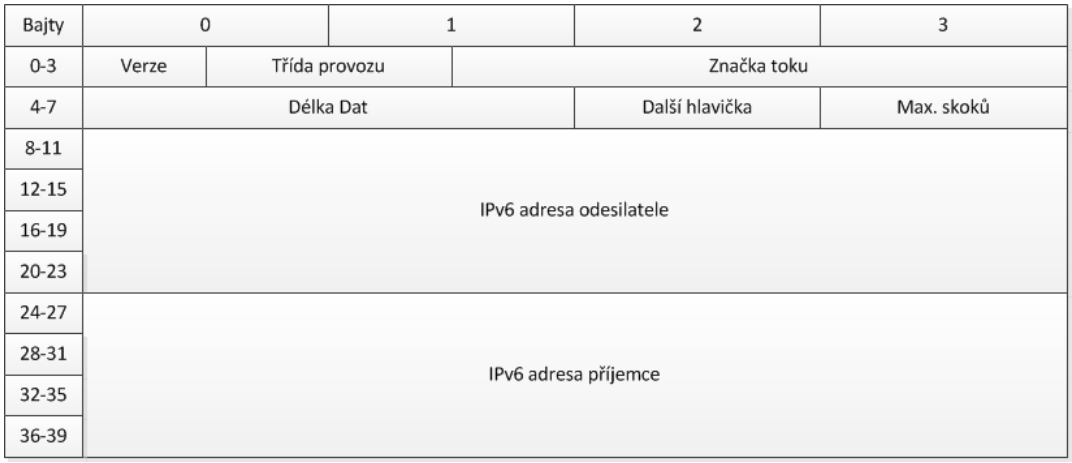
3.2.4 Protokol IPv6

Internet protokol verze 6 je nástupcem dosluhujcího IPv4. Hlavními důvody pro jeho zavedení byly zejména již dnes nedostačující velikost prostoru adres IPv4 a možnost zvýšení výkonu směrovacích prvků. Struktura IPv6 hlavičky vychází z IPv4 a je uvedena na obrázku 3.3.

Oproti hlavičce IPv4 neobsahuje hlavička IPv6 její kontrolní součet, čímž odpadá nutnost jeho přepočítání na každém směrovači při snížení TTL, které velmi zatěžovalo směrovače IPv4. Maximum skoků (*Hop Limit*) nahrazuje pole TTL u IPv4. Třída provozu (*Traffic Class*) nahrazuje pole DSCP a ECN u IPv4 pro zajištění QoS. Protokol IPv6 díky pevné velikosti hlavičky, která také představuje velkou výhodu při zvyšování výkonu směrovacích prvků, nepotřebuje pole *délka hlavičky*. Pole *další hlavička (Next Header)* obsahuje 8bitový kód určující sémantiku následujících dat. Mezi další výhody technologie IPv6, vycházející z filozofie jejího směrování, patří například automatická bezstavová konfigurace adres, implicitní podpora multicastu (multicast také nahrazuje broadcast komunikaci) nebo tzv. *Jumbogramy*[22] (snížení režie díky větší maximální velikosti přenosové jednotky).

Protokol IPv6 se stále vyvíjí, i když jeho základní principy jsou již běžně implmentovány v operačních systémech stanic nebo směrovačů. Specifikaci IPv6 lze nalézt v dokumentu RFC 2460 [26].

[1] http://www.iana.org/



Obrázek 3.3: Hlavička protokolu IPv6

Adresy IPv6

Protokol IPv6 využívá 128bitové adresy. Stejně jako u adres IPv4 i zde je nutno kromě adresy uvést *síťovou masku*. U IPv6 adres existují 3 základní druhy adres : *unicast*, *multicast* a *anycast*. Broadcastové adresy jsou nahrazeny adresami typu multicast. Adresy unicast adresují dané síťové stanice (rozhraní) a dále se dělí podle rozsahu platnosti. IPv6 implicitně umožňuje jejich automatickou konfiguraci. Adresy typu multicast umožňují adresovat v síti více rozhraní, data jsou doručována všem. Anycast adresy jsou také přiřazeny více síťovým rozhraním, například v geograficky odlišných oblastech, data jsou zde ale doručována pouze jednomu "nejbližšímu"zařízení. Tento princip je výhodný pro rozdělení zátěže (tzv. *load balancing*) nebo jako jeden z mechanismů obrany proti *DoS* útokům[28]. Více o adresování v protokolu IPv6 se lze dozvědět v dokumentu RFC 4291 [32].

3.2.5 Protokol ICMP

Protokol Internet Control Message Protocol (ICMP) je z hlediska komunikace pomocí TCP/IP sítě stejně důležitý jako protokoly IP. Pomocí tohoto protokolu si aktivní prvky v síti pracující na IP vrstvě (např. operační systém stanice a operační systém směrovače) vyměňují stavové informace a chybová hlášení. Bez ICMP by se například vysílací stanice nemohla dozvědět o skutečnosti, že určitý směrovač nemůže přeposlat data směrem k cíli (neexistující cesta, filtrovací pravidla atd.). Pomocí ICMP lze také ověřit záladní konektivitu mezi danými body sítě na IP vrstvě (zprávy echo, echo reply). Hlavička ICMP zprávy obsahuje typ, upřesňující kód, kontrolní součet a data (specifická pro daný typ a kód). Základními typy ICMP zpráv jsou :

- Echo a Echo Reply - tento typ využívají programy typu *ping*
- Destination Unreachable - cílová síť nebo stanice je nedostupná (obsahuje dalších 13 upřesňujících podtypů)
- Time Exceeded - Daný směrovač obdržel paket s TTL = 1 a zahodil ho (využíváno například programy typu *traceroute*)

Základní specifikaci protokolu ICMP obsahuje dokument RFC 792 [40].

3.2.6 Protokol ICMPv6

Internet Control Message Protocol verze 6 (ICMPv6) je obdobou ICMP pro protokol IPv6. Protokol, hlavička i jeho zprávy vycházejí z protokolu ICMP. ICMPv6 dále přebírá funkce protokolu ARP - protokol Neighbor Discovery Protocol (NDP) a další. Kromě základních typů zpráv z ICMP pro IPv4 (ovšem s jinými číselnými kódy) ale obsahuje další pro něj typické. Zprávy, jejichž typ je označen kódem do 127, jsou označovány jako chybové *ICMPv6 Error Messages*, zprávy s kódem větším než 127 jako informační *ICMPv6 Informational Messages*. Příklady ICMPv6 zpráv :

- Packet Too Big - slouží například pro zjištění MTU trasy.
- Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement - zprávy pro funkce protokolu NDP.

Detailní specifikaci protokolu ICMPv6 lze nalézt v dokumentu RFC 4443 [25].

3.2.7 Protokol IGMP

Internet Group Management Protocol (IGMP) slouží k dynamickému přihlašování a odhlašování z tzv. multicastové skupiny u IPv4 sítí. Funkce IGMP je u IPv6 sítí nahrazena protokolem *Multicast Listener Discovery* (MLD) - viz. RFC 4605[29]. Komunikace pomocí IGMP probíhá mezi klientskou stanicí a lokálním multicasovým směrovačem. Směrovače mezi sebou ale využívají protokol *Protocol Independent Multicast* (PIM). IGMP nyní existuje ve třech verzích - IGMPv1, IGMpv2 a IGMPv3. Všechny obsahují dvě základní zprávy - stanice žádá o členství v multicastové skupině pomocí zprávy *Join group* a odhlášení ze skupiny pomocí zpávy *Leave group*. Specifikaci IGMPv3 lze nalézt v dokumentu RFC 3376 [23].

3.3 Protokoly transportní vrstvy

Základní protokoly transportní vrstvy jsou protokoly TCP a UDP. Tyto protokoly implementují nejpodstatnější funkci transportní vrstvy - logické spojení mezi procesy. Procesy jsou rozlišeny pomocí 16 bitového čísla portu. Čísla portů menší než 1024 jsou označovány jako dobře známé porty *Well known ports* - daným portů je přiřazena služba (toto je spravováno společnosti IANA)[34]. Uživatelé si tak nemusí pamatovat čísla portů, ale pouze názvy protokolů - například portu 80 je přiřazen protkol HTTP. Porty od 1024 do 49151 jsou označovány jako registrované porty - použití by mělo být registrováno u společnosti ICANN. Zbytek prostoru (porty 49152 až 65535) se označuje jako dynamické, nebo soukromé porty - nejsou pevně přiděleny žádné aplikaci. TCP i UDP jsou součástí tzv. sady protokolů internetu (*Internet Protocol Suite*).

3.3.1 Protokol TCP

Transmission Control Protocol (TCP) je spojově orientovaný protokol transportní vrstvy modelu TCP/IP. TCP zajišťuje spolehlivé doručení, doručení ve správném pořadí, tzv. řízení zahlecení a adresování jednotlivých aplikačních procesů koncových komunikujících stanic. Plnění těchto funkcí vyžaduje znovuposílání IP datagramů při jejich nedoručení a také jejich případné přeuspořádání. Pro řízení toku jednotlivých IP datagramů je využíván

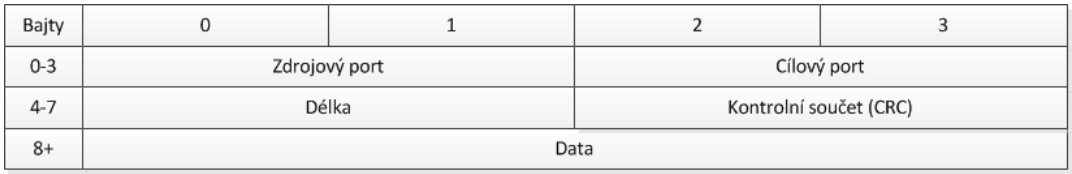
mechanismus *Sliding Window*[36]. Před vlastním přenosem dat je třeba navázat spojení. Hlavička TCP paketu je uvedena na obrázku 3.4.

Obrázek 3.4: Hlavička protokolu TCP

Pole *zdrojový port* a *cílový port* zajišťují adresování aplikačních procesů. Sekvenční a potvrzovací číslo slouží k spolehlivému doručení a doručení ve správném pořadí. Bity v poli *příznaky* jsou využívány při navazování a ukončování spojení. *Velikost okna* určuje velikost okna na straně příjemce (maximální počet nepotvrzených paketů, které je schopen příjemce při zřetězeném přenosu uchovat). Protokol TCP využívá většina běžně používaných aplikačních protokolů, jako je například HTTP, POP3, SSH apod. Detailní specifikaci protkolu TCP lze nalézt v dokumentu RFC 793 [42].

3.3.2 Protokol UDP

User Datagram Protocol (UDP) je nespojově orientovaný protokol transportní vrstvy modelu TCP/IP. UDP neposkytuje ani spolehlivé doručení, ani doručení v požadovaném pořadí. Před vlastní komunikací není potřeba navazovat spojení. Hlavička protokolu UDP je uvedena na obrázku 3.5.



Obrázek 3.5: Hlavička protokolu UDP

Hlavička obsahuje pouze zdrojový a cílový port (zajištění adresování aplikačních procesů), délku hlavičky a dat a kontrolní součet. Protokol UDP je využíván tam, kde je vyžadována jednodušší implementace (například hardwarové řešení), nebo tam, kde je vyžadováno více než spolehlivé včasné doručení - multimediální aplikace jako je *VoIP*. Protokol UDP využívají například aplikační protokoly RTP nebo DNS. Detailní specifikaci protokolu UDP lze nalézt v dokumentu RFC 768 [39].

3.4 Protokoly aplikační vrstvy

Protokolů aplikační vrstvy existuje oproti nižším vrstvám řádově větší množství. Některé jsou otevřené a standardizované, jiné propritální nebo čistě uživatelské. Protokoly této

vrstvy jsou narozdíl od protokolů nižších vrstev běžně v režii aplikačního programátra - nižší protokoly jsou pro něj transparentní. Z těchto důvodů je zde uveden pouze přehled nejznámějších aplikačních protokolů [37] :

- DNS (Domain Name System) - protokol pro překlad doménových jmen na IP adresy a další
- DHCP (Dynamic Host Configuration Protocol) - protokol pro automatickou konfiguraci síťových adres počítačů
- Telnet (Telecommunication Network) - základní protokol pro připojení ke vzdálenému počítači
- SSH (Secure Shell) - zabezpečená obdoba Telnetu, poskytuje další rozšiřující služby jako například zabezpečený přenos souborů
- HTTP (Hypertext Transfer Protocol) - protokol pro výměnu hypertextových dokumentů ve formátu HTML
- HTTPS (Hypertext Transfer Protocol Secure) - rozšíření protokolu HTTP umožňující zabezpečený přenos
- POP3 (Post Office Protocol verze 3) - protokol pro stahování elektronické pošty ze vzdáleného serveru
- SMTP (Simple Mail Transfer Protocol) - protokol pro přenos elektronické pošty mezi servery
- FTP (File Transfer Protocol) - protokol pro síťový přenos souborů mezi počítači

Kapitola 4

Dostupné knihovny a nástroje

Tato kapitola uvádí přehled dostupných knihoven a nástrojů pro zachytávání, manipulaci a generování síťových paketů.

4.1 Dostupné knihovny

Většina nástrojů pro analýzu, vytváření nebo modifikaci síťových paketů je založena na použití některé z následujících knihoven. Použití specializované knihovny většinou nabízí kromě pohodlnější práce také vyšší výkon - knihovny jsou optimalizovány na komunikaci se systémovým jádrem, které za normálních okolností zodpovídá za zpracování síťových paketů.

4.1.1 Knihovna Libpcap

Knihovna Libpcap je součástí open-source projektu Tcpdump & Libpcap. Jedná se o přenosnou knihovnu pro jazyk C/C++ zaměřenou zejména na zachytávání síťového provozu. Knihovna obsahuje základní rozhraní pro přístup k síťovým rozhraním - přehled dostupných rozhraní, jim příslušející adresy (MAC i IP) a další. Dále umožňuje otevřít zadané rozhraní v normálním, nebo tzv. promiskuitním módu a podle zadaných filtrovacích pravidel zachytávat příchozí pakety. Po zachycení paketu je volána programátorem definovaná *callback* funkce, ve které je možné obsah přijatého paketu zpracovat. Knihovna také obsahuje rozhraní pro vysílání paketů na úrovni vrstvy síťového rozhraní do sítě. Pro knihovnu je nativní formát souborů *PCAP*. Knihovna ve verzi pro operační systém Windows nese označení WinPcap. Více o knihovně se lze dozvědět na webových stránkách projektu[3].

4.1.2 Knihovna Libdnet

Knihovna Libdnet je open-source projekt distribuovaný pod licencí BSD, který si klade za cíl poskytnout pohodlné rozhraní pro přístup k některým procedurám nízkoúrovňového síťového programování. Libdnet lze podle autorů použít jako doplněk funkcionality knihovny Libpcap. Knihovna nabízí několik typů rutin: operace s adresami (MAC, IPv4 a IPv6): porovnání, převod z/do síťového formátu a další, rutiny pro přístup a modifikaci ARP tabulky (ARP cache) jádra, přístup k síťovým rozhraním - získání a nastavení různých parametrů, odesílání Ethernetových rámců nebo IP paketů a další. Více o knihovně se lze dozvědět na webových stránkách projektu[1].

4.1.3 Knihovna Libnet

Knihovna Libnet - Packet Construction Library je součástí projektu Packetfactory Network Security Project. Knihovna je open-source a distribuována pod licencí BSD. Libnet poskytuje vysokoúrovňové rozhraní pro konstrukci a modifikaci síťových paketů. Knihovna odstiňuje programátora od problémů spojených s alokací bufferů, problémů s převody dat z/do síťového formátu a dalších. Nabízí přímý přístup k položkám hlaviček různých protokolů a odesílání připravených paketů přes zadané rozhraní do sítě. Více o knihovně se lze dozvědět na webových stránkách projektu[2].

4.2 Dostupné nástroje

V této kapitole následuje přehled aktuálně dostupných nástrojů pro generování síťových paketů.

4.2.1 Nástroj Scapy

Nástroj Scapy je open-source program pro příkazovou řádku Pythonu umožňující zachytávání, vytváření a odesílání síťových paketů. Ke svému běhu vyžaduje Python, Pcapy (Libpcap pro Python) a Dnet (Libdnet pro Python). Scapy lze díky Pythonu využít na mnoha operačních systémech. Pro vytváření nebo manipulaci s pakety používá objektově orientovaný přístup. Výhodou je snadné použití a podpora formátů PCAP. Scapy je volně dostupný na webových stránkách projektu[11].

4.2.2 Nástroj AnetTest

Nástroj AnetTest je open-source program určený pro generování paketů. Program lze použít v prostředí příkazové řádky operačních systémů Unix a Windows. AnetTest ke svému běhu vyžaduje knihovnu Libpcap. Program umí generovat a odesílat síťové pakety podle zadaného textového předpisu. Dále nabízí možnost porovnání paketů odeslaných na určité rozhraní a paketů přijatých na jiném rozhraní pro jednoduché testy firewallu. Program lze stáhnout z webových stránek projektu[4].

4.2.3 Nástroj Nemesis

Nástroj Nemesis je open-source program pro příkazovou řádku Unix-like systémů a pro Windows. Program umožňuje podle zadaných argumentů vytvořit pakety předdefinované struktury (Ethernet, IP, TCP, UDP a další). Program vytvořený paket odešle do sítě prostřednictvím zadaného rozhraní. Program lze stáhnout z webových stránek projektu[8].

4.2.4 Nástroj Bit-Twist

Nástroj Bit-Twist je program pro příkazovou řádku Unix-like systémů a Windows. Bit-Twist pracuje se soubory formátu PCAP. Umožňuje modifikaci specifikovaných polí hlavičky, znovuzkonstruování paketu a jeho odeslání do sítě. Autoři doporučují program používat v kombinaci s programem Tcpdump. Program lze stáhnout z webových stránkách projektu[5].

4.2.5 Nástroj Pierf

Program Pierf je program pro generování a analýzu síťových paketů pro příkazovou řádku. Pierf by měl být nazávislý na operačním systému, v současné době lze ale získat pouze binární obraz pro systém Windows. Program generuje paket podle zadaného předpisu ve formátu XML. Vygenerovaný paket je odeslán na rozhraní specifikované společně se strukturou paketu. Dokumentaci k XML fomátu lze stáhnout společně s programem v podobě binárního spustitelného souboru z jeho webových stránek[10].

4.2.6 Nástroj Cat Karat Packet Builder

Nástroj Cat Karat Packet Builder je program s grafickým uživatelským rozhraním pro operační systém Windows. Program lze zdarma využívat pro osobní účely, pro komerční využití je nutné zakoupit licenci. Program nabízí uživateli volbu z několika protokolů každé vrstvy, nastavení položek jednotlivých hlaviček a náhled na celkový binární obraz paketu. Vytvořený paket lze odeslat do sítě, nebo uložit do souboru PCAP. Program lze stáhnout z webových stránek projektu[6].

4.2.7 Nástroj Colasoft Packet Builder

Program Colasoft Packet Builder je freeware pro operační systém Windows. Colasoft Packet Builder nabízí grafické uživatelské rozhraní. Program umožňuje načtení a uložení množiny paketů z/do souboru. Do načtené posloupnosti lze vkládat několik typů nových paketů podle předdefinovaných šablon (ARP, IP, TCP a UDP paket) . Některé položky některých hlaviček paketů je možno editovat a vytvořenou posloupnost odeslat přes zadané zařízení do sítě. Program lze stáhnout z webových stránek projektu[7].

4.2.8 Nástroj PackETH

Program PackETH je open-source freeware distribuovaný pod licení GNU GPL. PackETH má grafické uživatelské rozhraní a je primárně určený pro systém Linux (existují ale i verze pro Windows a MAC OS). Program umožňuje generovat pakety, jejichž struktura a jednotlivé položky hlaviček jsou definovány na různých vrstvách modelu TCP/IP, nebo podle zadaného binárního (hexadecimálního zápisu) obrazu. Vytvořený paket lze uložit do formátu PCAP, jehož obsah umí program poté odeslat přes zadané rozhraní do sítě. Zdrojové kódy programu jsou dosupné na webových stránkách programu[9].

4.2.9 Nástroj Ostinato

Program Ostinato je open-source nástroj s grafickým uživatelským rozhraním pro Unix-like systémy, Windows a MAC OS. Program umožňuje vytvářet tzv. streamy, které poté podle zadaných parametrů odesílá do sítě. Stream se skládá z paketu definovaného podle architektury TCP/IP. Položky jednotlivých hlaviček lze nadefinovat a paket případně i uložit. Program má příjemné uživatelské rozhraní. Zdrojové kódy nebo binární obrazy jsou dostupné na webových stránkách projektu[15].

4.3 Další typy nástrojů pro generování paketů

Softwarová řešení (jako jsou knihovny nebo aplikace) jsou oblíbené pro svou jednoduchost použití a také pořizovací cenu. Pro náročnější použití jsou ale omezeny především max-

imální velikostí výstupního datového toku.

4.3.1 Komerční zařízení

Síťové emulátory Spirent

Společnost Spirent[19] nabízí řadu nástrojů pro testování a vývoj síťových technologií. Mezi jinými jsou to právě také nástroje pro generování síťového provozu. Základem systému je tzv. SPIRENT TESTCENTER jednotka. Do této jednotky nabízí firma velké množství modulů v podobě zásuvných karet pro různě síťové technologie (Ethernet, Optické sítě ...). Jednotky po nastavení generují zvolený síťový provoz za účelem testů. Vícero jednotek je při testech mezi sebou velmi přesně synchronizováno, což umožňuje provádět i testy sítí s vysokou propustností. Více o této proprietální technologii se lze dozvědět na webových stránkách společnosti [19].

Název	**GUI**	**OS**	**Licence**
Scapy	NE	Windows, Unix, MAC OS - (Python)	GPLv2
AnetTest	NE	Unix, Windows	GPL
Nemesis	NE	Unix, Windows	?
Bit-Twist	NE	Unix, Windows	GPL
Pierf	NE	Windows	viz web
Cat Karat Packet Builder	ANO	Windows	viz web
Colasoft Packet Builder	ANO	Windows	Freeware
PackETH	ANO	Linux (Windows, MAC OS)	GPL
Ostinato	ANO	Unix, Windows, MAC OS	GPL v3

Obrázek 4.1: Přehled nástrojů pro generování paketů

Kapitola 5

Návrh nástroje

5.1 Požadavky

Tato kapitola uvádí požadavky, které byly kladeny na výslednou podobu aplikace.

5.1.1 Základní požadavky

Základní požadavky vyplývají ze zadaní práce a jsou klíčové pro její návrh.

- běh aplikace na platformě GNU/Linux
- implementace v programovacím jazyce Python, C, C++, nebo Java
- grafické uživatelské rozhraní
- podpora souborů formátu PCAP
- co nejširší podpora protokolů

5.1.2 Rozšiřující požadavky

Tyto požadavky byly doplněny během návrhu aplikace po konzultaci s vedoucím práce. Jedná se o požadavky výrazně zvyšující použitelnost a rozšiřitelnost výsledné aplikace.

- běh aplikace na dalších platformách - MS Windows, Mac OS
- možnost generování posloupnosti paketů simulující určitý typ síťového provozu :
  - variabilní zpoždění mezi pakety
  - možnost změny obsahu paketů v dané posloupnosti
  - možnost variabilně zadat celkou velikost paketu
- rozlišení času v posloupnosti paketů v řádu ns a podpora formátu PCAP-ng
- ukládání a načítání nastavení určujících podobu výsledného paketu
- ukládání a načítání nastavení určujících podobu výsledné posloupnosti
- autonomní chování jednotlivých protokolů zajišťující jejich rozšiřitelnost

5.2 Koncept

Z požadavků byl odvozen následující koncept fungování aplikace.

Hlavní funkcí aplikace bude generování síťových paketů. Uživatel bude moci nastavovat veškeré potřebné údaje přes grafické uživatelské rozhraní. Kromě struktury dané pořadím obsažených protokolů bude moci uživatel také specifikovat obsah hlaviček protokolů v paketu. Mimo generování jednotlivých paketů bude program umožňovat také generovaní jejich sekvencí za účelem modelování síťového provozu. V sekvenci paketů se kromě zpoždění mezi pakety budou moci zadaným způsobem měnit také obsahy hlaviček jednotlivých protokolů. Vlastnosti sekvence paketů bude možno specifikovat deterministicky nebo stochasisticky. Na výstupu bude možno sloučit do jediné posloupnosti výstup více generátorů. Nastavení struktury paketu, obsahu protokolů a sekvence bude možno uložit nebo načíst z/do souboru. Výstupem programu bude soubor uživatelem zvoleného formátu obsahující výsledný záznam obrazu síťové komunikace. Tento záznam bude moci být později odvysílán do sítě. Aplikace nebude umožňovat přímé vyslání paketů do síťě z důvodu časové náročnosti sestavování jednotlivých paketů.

5.3 Objektový návrh

Tato kapitola uvádí základní objektově orientovaný návrh fungovaní výsledné aplikace.

Aplikace je tvořena základními komponentami (řadič generátorů, paket ...) řízenými z grafického uživatelského rozhraní. Tyto základní komponenty mezi sebou komunikují a udržují v sobě stav aplikace definovaný uživatelem. Stav je kromě standardních vlastností definován také existencí dalších zapouzdřených objektů (generátor, enkapsulace ...). Aplikační logika je poté založena na delegaci událostí směrem od zapoudřujících objeků k zapouzdřeným přes abstraktní rozhraní. Jednotlivé objekty, jako je například určitý protokol, se díky tomu chovají autonomně, se zapouzdřujícím objektem komunikují na vyžádání pomocí pevně stanovených bodů, což umožňuje snadnou rozšiřitelnost.

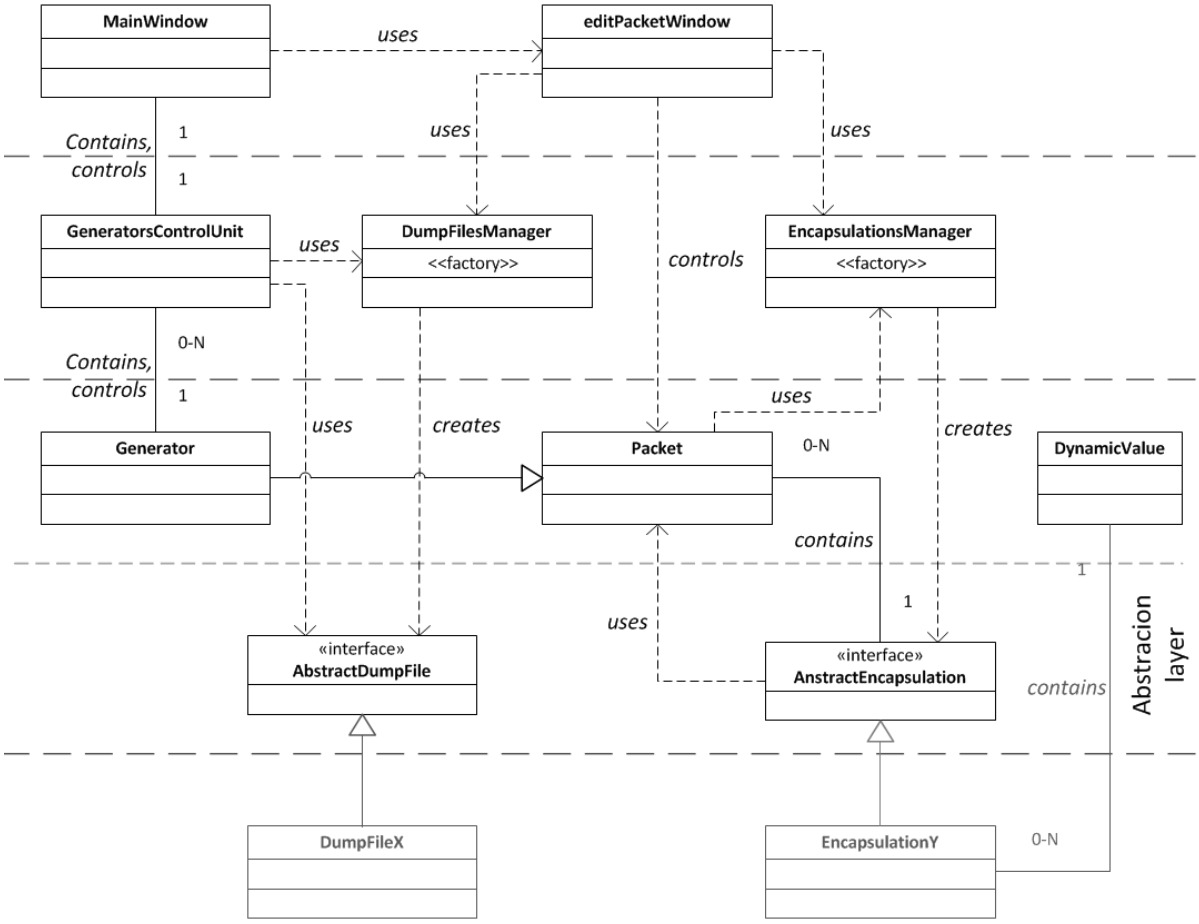
Struktura aplikace je přehledově znázorněna na obrázku 5.1 (Některé části jsou z důvodu přehlednosti vypuštěny - viz níže). Následuje popis hlavních funkcí navržených objektů (komponent).

5.3.1 Základní komponenty

Tyto komponenty tvoří jádro programu a zajišťují jeho hlavní funkcionalitu.

Řadič generátorů *(GeneratorsControlUnit)*

Hlavním úkolem řadiče generátorů je serializovat (řadit) posloupnost paketů z pseudoparalelně běžících generátorů. Tento řadič také zajišťujě persistenci jednotlivých generátorů - jejich vytváření, odstraňovaní a přístup k nim. Generátor umožňuje nastavení jako je výstupní formát nebo celkové maximum vygenerovaných paketů (počet nebo velikost). Řadič generátorů na požádání provede simulaci běhu generátorů a jejich serializovaný výstup uloží do souboru specifikovaného formátu. Práce s výstupními soubory probíhá za pomoci správce souborů síťové komunikace přes abstraktní rozhraní.



Obrázek 5.1: Přehled základní struktury aplikace

Generátor *(Generator)*

Generátor je základní jednotka reprezentující určitou posloupnost (sekvenci) paketů. Generátor obsahuje její nastavení, jako je název, maximum vygenerovaných paketů v sekvenci (počet, celková velikost ...) nebo nastavení zpoždění mezi pakety. Generátor pracuje na principu návrhového vzoru *Iterátor* - inicializace, současný obsah a vypočtená prodleva, další obsah. Typ paketu, který je generátorem generován je dán objektem Paket, od kterého Generátor dědí a který také řídí.

Paket *(Packet)*

Objekt paket reprezentuje síťový paket určitého typu (dané struktury). Struktura paketu je tvořena orientovaným vektorem jednotlivých Enkapsulací (protokolů). Enkapsulace je možno přidávat, odebírat, nebo měnit jejich pořadí ve struktuře. Instance enkapsulací vytváří na vyžádání správce enkapsulací - viz níže. Struktura paketu se v sekvenci řízené generátorem, případně jinou komponetnou nemění, obsah ale v závislosti na nastavení ano. Paket stejně jako generátor pracuje na principu návrhového vzoru *Iterátor*. Paket deleguje události spojené s průchodem sekvence na jednotlivé enkapsulace, které tak mohou měnit svůj obsah. Paket na požádání sestaví binární obraz (se zadanou endianitou) odpovídající

aktuální pozici v sekvenci.

Dynamická hodnota *(DynamicValue)*

Dynamická hodnota je objekt usnadňující implementaci dynamicky se měnicích polí hlaviček protokolů. Umožňuje nastavit rozsah hodnot a způsob výběru z tohoto rozsahu (fixní, inkrementální nebo náhodný s daným rozložením). Dynamická hodnota umožňuje také načíst soubor hodnot ze zadaného textového souboru. Přístup k těmto hodnotám je obdobný jako přístup k hodnotám zadaným rozsahem. Dynamická hodnota sestaví binární obraz aktuální hodnoty se zadanou endianitou.

5.3.2 Komponenty protokolů

Správce protokolů *(EncapsulationsManager)*

Správce protokolů pracuje na principu návrhového vzoru *Abstraktní továrna*. Obsahuje katalog aktuálně dostupných protokolů (enkapsulací), dostupný v podobě seznamu jejich názvů. Na požádání vytvoří instanci enkapsulace podle zadaného typu nebo názvu a předá její referenci k dalšímu použití.

Abstraktní enkapsulace *(AbstractEncapsulation)*

Abstraktní enkapsulace je rozhraní, přes které může paket přistupovat a řídit jednotlivé konkrétní enkapsulace. Rozhraní umožňuje inicializaci, získání binárního obrazu odpovídajícího aktuální pozici v sekvenci, posun v sekvenci a další. Ve speciálních případech lze přes toho rozhraní získat také specifikaci typu konkrétní enkapsulace.

5.3.3 Komponenty pro práci se vstupními a výstupními soubory

Správce souborů pro záznam síťové komunikace (*DumpFilesManager)*

Správce souborů pro záznam síťové komunikace pracuje na principu návrhového vzoru *Abstraktní továrna*. Obsahuje katalog dostupných typů souborů v podobě názvů. Na požádání vyvtoří instanci souboru daného typem nebo názvem typu. Tento správce také umožňuje rozpoznání typu souboru.

Abstraktní soubor pro záznam síťové komunikace (*AbstractDumpFile)*

Toto rozhraní umožňuje jednotný přístup k více typům podporovaných souborových formátů. Rozhraní umožňuje inicializaci souboru, vložení paketů do souboru a jeho uložení. Rozhraní také umožňuje načtení již existujícího souboru a přístup k paketům v něm obsaženým.

5.3.4 Komponenty grafického uživatelského rozhraní

Hlavní okno (*MainWindow)*

Hlavní okno obsahuje prvky pro ovládání a nastavení základních kompoment jako je řadič generátorů, generátor a další. Existence hlavního okna také zajišťuje persistenci všech objektů aplikace v paměti.

Okno editace paketu (*EditPacketWindow)*

Toto okno obsahuje ovládací prvky pro nastavení struktury paketu a obsahu hlaviček protokolů. Binární obraz paketu je možno pomocí tohoto okna zobrazit a editovat díky vestavěnému hexadecimálního editoru.

Kapitola 6

Implementace

Tato kapitola popisuje aspekty implementace nástroje včetně použitých technologií.

Vzhledem k výše uvedným požadavkům a objektovému návrhu byl pro implementaci zvolen programovací jazyk C++ společně s frameworkem Qt[18], který zajišťuje přenositelné grafické uživatelské rozhraní mezi nejpoužívanějšími operačními systémy. Třídy byly z důvodu přehlednosti rozděleny do několika skupin (projektů) odpovídajících návrhu aplikace. Uvedený popis je pouze přehledový, detailní informace o implementaci všech programových tříd lze získat z dokumentace generované ze zdrojového kódu programem Doxygen[17].

6.1 Aplikační třídy

Jazyk C++ neobsahuje přímou podporu rozhraní (*Interface*) z teorie objektů. Rozhraní je tedy v aplikaci nahrazeno použitím dědičnosti. Třídy, které měly implementovat určité rozhraní, dědí od společné abstraktní třídy, která obsahuje čistě virtuální metody (*Pure virtual methods*), které nepřímo vynucují svou implementaci - výslednou třídu bez implementace těchto virtuálních metod nelze zkompilovat. Aplikační třídy mohou díky implicitnímu přetypování pracovat se všemi třídami toho typu jako s typem jejich rodičovské abstrakní třídy bez nutnosti znalosti konkrétního typu.

6.1.1 Hlavní aplikační třídy

Tyto třídy odpovídají objektovému návrhu a zajišťují hlavní funkcionalitu aplikace.

GeneratorsControlUnit

Třída implementuje objekt řadič generátorů. *GeneratorsControlUnit* obsahuje vektor referencí na objekty typu *Generator*. Tento vektor obsahuje odkazy na všechny generátory, které existují v daný časový okamžik. Třída obsahuje metody pro vytvoření generátoru (*newGenerator()*), odstranění (*removeGenerator()*), přesun v posloupnosti nahoru (*moveGeneratorUp()*) nebo dolů (*moveGeneratorDown()*) . Seznam aktuálně obsažených generátorů lze získat v podobě vektoru řetězců názvů voláním *generatorsNames()*. Referenci na instanci generátoru daného jména lze získat voláním *generator()*. Referenci na packet obsažený v generátoru lze získat voláním *generatorPacket()*. Generování výstupu lze zahájit voláním *exportToFile()* - viz. 6.2.4 Generování výstupního souboru.

Generator

Třída implementuje objekt generátor. *Generator* rozšiřuje implementaci třídy *Packet*. Třída obsahuje rozsáhlou sadu tzv. getteru a setteru, které umožňují nastavit parametry generování prodlevy mezi jednotlivými pakety. Z ostatních metod jsou nejdůležitější následující: metoda *setName()* - nastaví jméno generátoru v podobě textového řetězce. Metoda *initGenerator()* - inicializuje generátor před jeho použitím. Incializace probíhá nezávisle na módu ve kterém bude generátor použit. Metoda *isNext()* - indikuje, zdali generátor může vygenerovat další paket (vyčerpání nastavených limitů, chybějící odkaz v sekvenčím módu, etc.). Metoda *currentPacket()* exportuje aktuální binární obraz obsaženého paketu se zadanou endianitou. Metoda *nextPacket()* zajistí posun v sekvenci paketů. Metody *simNextIn()* a *seqNextIn()* vrací vypočtenou prodlevu mezi aktuálním paketem a dalším paketem v sekvenci. Metody *exportGenerator()* a *importGenerator()* slouží k načtení a uložení všech nastavení.

Packet

Třída implementuje objekt paket. Packet v sobě obsahuje vektor referencí na třídu *AbstractEncapsulation*, který během existence objektu reprezentuje aktuální strukturu paketu. Následující metody umožňují práci s touto stukturou : *appendEncapsulation()* - přidání enkapsulace specifikované jejím názvem, *removeEncapsulation()*, *moveUpEncapsulation()*, *moveDownEncapsulation()* odebrání, přesun nahoru, přesun dolů enkapsulace specifikované pozicí (indexem) v aktuální struktuře. Metody pro přístup k obsahu paketu a řízení sekvence jsou následující : *initContent()* - inicializace sekvence, *currentContent()* získání aktuálního binárního obrazu se zadanou endianitou, *nextContent()* - posun v sérii paketů. Metoda *findEncapsulation()* umožňuje vyhledání enkapsulace v aktuální struktuře podle zadaných pravidel : typ (konkrétní, nebo jakýkoliv), pozice (pozice, od které má vyhledání začít - začátek, konec, daná pozice), směr (nahoru, dolů). Tuto metodu mohou využít obsažené enkapsulace pro zjištění informací nutných pro správné vygenerování obsahu - např. kontrolní součet v hlavičce protokolu UDP.

DynamicValue

Třída implementuje objekt dynamická hodnota. Při její implementaci byl kladen důraz na univerzálnost a možnosti jejího použití. Pro vnitřní reprezentaci byl proto zvolen datový typ z knihovny GMP[16], který umožňuje uložit i velké číselné hodnoty jako je například 128bitová IPv6 adresa. Před použitím je třeba nastavit velikost výsledného binárního obrazu v bytech pomocí metody *resize()*. Třída umožňuje nastavit rozsah hodnot pomocí metod *setFrom()* a *setTo()*. Metodou *setFileName()* lze nastavit zdrojový textový soubor s hodnotami. Aktuální mód a přístup k hodnotám nastavují metody *changeType()* a *changeAccessMethod()*. Při použití náhodného vyběru lze specifikovat rozložení hustoty pravděpodobnosti, se kterou bude prováděn (lineární, normální nebo exponenciální). Toto rozložení lze také parametrizovat. Řízení sekvence zajišťují metody *initValue()* a *nextValue()*. Klíčová je metoda *exportValue()*, která z vnitřní reprezentace vytvoří pole bytu dané endianity obsahující odpovídající hodnotu. Třída obsahuje další metody, které usnadňují její nastavení a použití - viz generovaná dokumentace.

EncapsulationsManager

Tato třída implementuje objekt správce protokolů. *EncapsulationsManager* obsahuje enumerátor *EncapsulationType*, který v celé aplikaci jednoznačně identifikuje všechny podporované enkapsulace (protokoly). Tato třída také obsahuje metody pro mapování názvů enkapsulací na tyto identifikátory a naopak. Voláním metody *availableEncapsulations()* lze od instance této třídy získat vektor názvů všech dostupných enkapsulací. *EncapsulationsManager* na základě volání metody *creator()* se zadaným typem nebo názvem vytvoří instanci požadované enkapsulace a předá její referenci k dalšímu použití.

Třídy odvozené od AbstractEncapsulation

Tyto třídy implementují jednotlivé protokoly. Odvození od *AbstractEncapsulation* vynucuje implementaci následujících metod zajišťujících použití v třídě *Packet*. Metoda *name()* - vrací textový řetězec identifikující enkapsulaci (protokol), *initContent()* - inicializace enkapsulace, *nextContent()* - posun na další obsah protokolu v sekvenci paketů, *cPrefix()* - aktuální binární obraz pro připojení před aktuální obsah, *cSuffix()* - aktuální binární obraz pro připojení za aktuální obsah[1], *editByte()* - upraví byte v počátečním obsahu.

DumpFilesManager

Tato třída implementuje objekt správce souborů pro záznam síťové komunikace. Princip implementace je obdobný třídě *EncapsulationsManager*. Kromě tohoto implmentuje třída také metody *identifyType()* - identifikace typu souboru, *fileExt()* - název standardní přípony souboru daného typu a další.

Třídy odvozené od AbstractDumpFile

Tyto třídy zajišťují práci se soubory pro záznam síťové komunikace tzv. *dump files*. Rozhraní vynucuje mimo jiných implementaci následujících metod : *initFile()* - inicializace souboru (pro vytváření nového), *loadFromFile()* - inicializace a načtení paketů ze zadaného souboru, *writeToFile()* - zápis aktuálního obsahu do specifikovaného souboru, *addPacket()* - přidání paketu, *removePacket()* - odstranění paketu, *getPacketContent()* získání binárního obrazu zvoleného paketu.

6.1.2 Pomocné aplikační třídy

Tyto třídy byly do aplikace doplněny v průběhu implementace a nemají přímý vliv na její základní fungování. Zajišťují například zobrazení nápovědy nebo správu šablon.

LogAdmin

Tato třída zajišťuje sběr informací ze všech hlavních aplikačních tříd. Voláním *logEvent()* mohou jejich instance informovat třemi základními typy zpráv - zpráva, varování a chyba o průběhu provádění dané operace. LogAdmin poté podle zadaných nastavení tyto informace zobrazuje v grafickém uživatelském rozhraní, vypisuje na standardní a chybový výstup, nebo do zadaného souboru.

[1] Teoretická možnost, žádná z aktuáně implementovaných enkapsulací nevyužívá.

EncapsulationsHelp

Tato třída umožňuje zobrazit stručnou nápovědu k jednotlivým enkapsulacím v okně editace paketu. Třída ze zadaného souboru načte obsah nápovědy, na požádání poté obsah identifikovaný názvem enkapsualce vysází do formátu html a umožní tak jeho zobrazení v grafickém uživatelském rozhraní.

PacketsTemplates

Tato třída zajišťuje správu šablon struktur paketů. Třída umožňuje jejich načtení z textového XML souboru. Voláním *availableTemplates()* vrátí vektor řetězců jejich názvů. Na základě volání *templateEncapsulations()* s požadovaným názvem poté vrací strukturu šablony.

ApplicationSettings

Tato třída zajišťuje práci s globálními aplikačními nastaveními. Základem je jednoduchá databáze dvojic klíč - hodnota. Volání *loadSettings()* a *writeSettings()* slouží k uložení nebo načtení databáze z/do souboru. Voláním *writeValue()* a *readValue()* lze zapsat nebo získat hodnotu z databáze. Pokud při vkládání dabáze hodnotu již obsahuje, je tato přepsána. Pokud databáze hodnotu neobsahuje, je vrácen prázdný řetězec. Odlišení neexistující hodnoty od prázdné lze provést voláním metody *hasValue()*.

6.1.3 Třídy grafického uživatelského rozhraní

MainWindow

Třída MainWindow implementuje grafické uživatelské rozhraní hlavního okna aplikace. MainWindow je událostně řízeno uživatelem. Na základě těchto událostí volá příslušné metody objektů, které tvoří jádro aplikace. Dobou existence této třídy jsou také limitovány všechny ostatní objekty v aplikaci - třída obsahuje instance všech základních komponent.

EditPacketWindow

Třída EditPacketWindow implementuje grafické uživatelské rozhraní okna pro editaci paketu. Tato třída ve svém konstruktoru vyžaduje referenci na objekt typu packet, který během své existence upravuje a řídí.

RandomDistributionDialog

Tato třída implementuje grafické uživatelské rozhraní pro nastavovaní parametrů rozdělení pravděpodobnosti, reprezentovaného instancí třídy *RandomDistribution*. Zobrazované okno je koncipováno jako modální dialog, uživatel tedy může změny potvrdit nebo zahodit.

SelectPcapPacketDialog

Třída *SelectPcapPacketDialog* implementuje grafické uživatelské rozhraní pro výběr paketu ze seznamu (typicky obsah souboru). Okno této třídy je zobrazováno jako modální dialog.

6.2 Základní procesy aplikační logiky

6.2.1 Tvorba struktury paketu

Struktura paketu může být vytvořena dvěma základními způsoby: načtením ze šablony, nebo postupným skládáním enkapsulací. Z hlediska aplikační logiky se jedná o stejný způsob, pouze při načtení ze šablony jsou akce spojené s odstraněním stávající struktury a s přidáním nových protokolů prováděny automaticky.

Vložení enkapsulace do struktury paketu probíhá následujícím způsobem: uživatelem zvolená enkapsulace je identifikována řetězcem obsahujícím její jméno. Toto jméno je předáno instanci třídy *EncapsulationsManager*, která zajistí vytvoření instance třídy odpovídající zvolené enkapsulaci. Reference na tuto třídu je uložena do vektoru v třídě *Packet* reprezentujícího strukturu paketu. Po aktualizaci struktury je také aktualizováno grafické uživatelské rozhraní tak, aby zajistilo přístup k nastavením dané enkapsulace.

6.2.2 Import paketu

Aplikace umožňuje v současné implementaci načítat jednotlivé pakety ze souboru typu PCAP nebo PCAP-ng.

Uživatelem vybraný soubor k importu je po otevření předán třídě *DumpFilesManager*, která pomocí metody *identifyType()* provede pokus o identifikaci jeho typu. Třída *DumpFilesManager* projde katalog známých typů souborů a porovná uložené sekvence bajtů s tzv. magickým číslem (*Magic number*)[20][21] obsaženým v souboru. Pokud je nalezena odpovídající sekvence, třída vrací enumerátor odpovídajícího typu identifikující formát souboru. Na základě tohoto typu může *DumpFilesManager* vytvořit instanci třídy odpovídající danému souboru. Vytvořená třída poté načte pakety obsažené v souboru. Uživatel zvolí paket v zobrazeném dialogu. Vybraný paket je poté načten do předem zvolené struktury paketu.

6.2.3 Export binární podoby paketu

Export binární podoby paketu z instance třídy *Packet* probíhá na základě volání metody *currentContent()*. Před tímto voláním musí být volána také metoda *initContent()*, která zajistí správnou inicializaci série paketů. Vlastní export probíhá postupným voláním metod na získání aktuálního binárního obrazu jednotlivých obsažených enkapsulací. Při těchto voláních je předávána také reference na již vytvořený obsah, například pro potřeby výpočtu kontrolního součtu apod. Po volání je získaný obsah připojen k vytvářenému obrazu. Endianitu exportovaného obrazu lze parametrizovat.

Získaný binární obraz paketu je možno např. vložit do souboru nebo zobrazit v hexadecimalní podobě.

6.2.4 Generování výstupního souboru

Výstupem aplikace je v současné implementaci soubor typu PCAP nebo PCAP-ng obsahující jeden, nebo sérii vygenerovaných paketů.

Po zadání příkazu ke generování aplikace požádá třídu *DumpFilesManager* o vytvoření instance třídy odpovídající uživatelem zvolenému výstupnímu formátu. Reference na rozhraní této třídy společně s módem a dalšími parametry je předáno třídě *GeneratorsControlUnit*. Tato třída provede simulaci běhu generátorů.

GeneratorsControlUnit inicializuje všechny obsažené generátory voláním *initGenerator()*. Jednotlivé Generátory delegují toto volání na jejich obsaženou rodičovskou implmentaci třídy *Packet* volání *initContent()*. *Packet* toto volání poté přes rozhraní deleguje voláním *initContent()* na jednotlivé zapouzdřené enkapsulace. Pokud probíhá inicializace v paralelním módu, je také vygenerován tzv. vektor zpoždění generátorů, který obsahuje čas aktivace generátoru. Po úspěšné inicializaci probíhá simulace běhu generátorů v závislosti na módu, ve kterém byla spuštěna:

Je vybrán první generátor - v sekvenčím módu první v pořadí, v paralelením generátor s nejmenším časem ve vektoru zpoždění. Z vybraného generátoru je exportován binární obraz aktuálního paketu voláním *currentPacket()* (popis ve výše uvedené sekci). Po získání obsahu je volána metoda *next()*, která principem obdobným inicializaci zajistí posun v sekvenci paketů generovaných daným generátorem. Získaný obsah je vložen do třídy výstupního souboru pomocí rozhraní voláním *addPacket()*. Po exportu je opět vybrán generátor k aktivaci - v sekvenčím módu na základě odkazu, v paralelním generátor s nejmenším aktivačním časem po odečtení aktuálního simulačního času. Simulace je ukončena v případě, že generátor neobsahuje odkaz na další, nebo byl vyčerpán celkový limit pro generování (velikost nebo počet).

Po ukončení simulace je na třídě reprezentující výstupní soubor volána metoda *writeToFile()*, která zajistí zápis jejího obsahu do souboru.

6.2.5 Ukládání a načítání nastavení

Ukládání nebo načítání nastavení lze rozdělit do několika skupin :

Ukládání globálních aplikačních nastavení

Aplikace ukládá svá nastavení, jako je například poloha okna, volby spojené se zobrazením nápovědy nebo editací struktury do souboru formátu XML. Načtení souboru, uložení stejně jako zápis či přístup k hodnotnám zajišťuje instance třídy *ApplicationSettings*.

Ukládání nastavení paketu

Při ukládání uživatelem vytvořeného paketu si nelze vystačit pouze s vytvořeným binárním obrazem. Takto vytvořený paket neobsahuje například nastavení sekvence a další metainformace. Veškerá nastavení týkající se struktury a obsahu paketu se z tohoto důvodu ukládají do textového souboru formátu XML.

Nastavení lze uložit nebo načíst z řetězce typu *QString*[18] nebo přímo pomocí z tzv. *QXmlStream*[18]. Proces ukládání je zahájen voláním *exportPacket()*. Toto volání vytvoří kořenový element "packet". *Packet* poté postupně deleguje ukládání na jednotlivé enkapsulace, které do něj vkládájí svoje aktuální nastavení obalené elmetem typu "encapsulation". Načítání probíhá opačným způsobem oproti ukládání. Po nalezení elementu "encapsulation"je vytvořena instance typu odpovídající uvedenému názvu. Instanci této třídy je poté předána reference na *XMLStream*, ta může provést načtení svých nastavení.

Ukládání nastavení "Session"

Pojem Session v aplikaci označuje vytvořené generátory, jejich nastavení včetně nastavení obsažených paketů. Uložení nebo export lze provést voláním *exportSession()* z *GeneratorsControlUnit*. Vlastní ukládání provádějí samotné instance generátorů. Tyto generátory

nejdříve do *XMLStream* vloží svoje nastavení a poté nechají uložit nastavení paketu obsaženou implementací třídy *Packet*. Načtení probíhá opačným způsobem.

6.2.6 Simulace protokolů aplikační vrstvy - Payload

Payload je speciální typ enkapsulace umožňující simulovat obsah protokolů aplikační vrstvy. Obsah lze simulovat třemi působy - vzorkem (hexadecimální zápis, daná velikost a počet opakování), obsahem binárního souboru, nebo náhodně generovanými daty. U náhodně generovaných dat lze také varibilně specifikovat jejich velikost - velikost fixní, nebo náhodně vybraná z rozsahu. U náhodného výběru lze specifikovat také rozložení hustoty pravděpodobnosti pro tento výběr - lineární, normální, exponenciální, nebo obecně jiné. Jiný typ rozložení uživatel specifikuje textovým souborem obsahujícím dvojice význačných bodů kumulativní distribuční funkce (CDF) daného rozložení. Aplikace po načtení těchto bodů provede lineární aproximaci této funkce. Výběr ze zadaného rozsahu hodnot poté probíhá na základě metody inverzní transformace. U náhodně generovaného obsahu lze také specifikovat volbu, která zajistí odečtení velikosti obsažených protokolů od vybrané velikosti - toto umožňuje uživateli pohodlně specifkovat celkovou velikost paketu bez nutnosti počítat aktuální velikost ostatních dat.

6.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní aplikace se skládá ze dvou hlavních oken - hlavní okno, okno editace paketu a několika pomocných. Při jeho návrhu byl kladen důraz na přehlednost a intuitivnost během ovládání aplikace. Prvky, které umožňují nastavovat vlastnosti objektů, jsou zobrazeny přímo na ploše daného okna. Změny zde provedené jsou okamžitě automaticky uloženy do datové vrstvy aplikace. Tlačítka spouštějící akce v aplikaci (generování výstupu, uložení nebo načtení session ...) jsou umístěna v tzv. hlavním menu. Popis ovládání lze nalézt v příloze B - Manuál, snímky toho rozhraní lze nalézt v příloze A - Snímky uživatelského rozhraní.

6.4 Rozhraní pro příkazovou řádku

Aplikaci lze z příkazové řádky spustit bez argumentů, nebo volitelně s některými z následujících v libovolném pořadí.

- `-h` - zobrazení nápovědy
- `-f [filename]` - soubor s uloženou session (viz. 6.2.5) - po inicializaci aplikace bude tento soubor automaticky načten
- `-w [level]` - zapne výpis logu aplikace na standartní výstup, uroveň výpisu je specifikována číslem "level"(0 - všechny zprávy, 1 - pouze varování a chyby, 2 - pouze chyby)
- `-a [filename]` - nastavení cesty k XML souboru s nápovědou - pokud není nastaveno, je použita cesta `./help.phxml`
- `-t [filename]` - nastavení cesty k XML souboru se šablonami struktur paketu - pokud není nastaveno, je použita cesta `./templates.ptxml`

6.5 Podporované protokoly

Přehled aktuálně podporovaných protokolů, ze kterých lze vytvářet strukturu a obsah paketu :

- 802.1Q (VLAN)
- IEEE 802.3 (RAW) / Ethernet II
- ICMP
- IPv4
- IPv6
- MPLS
- TCP
- UDP
- Payload - simulace protokolů aplikační vrstvy

Kromě výše uvedených protokolů lze také strukturu a obsah specifikovat binárním obrazem paketu načteným z některého z podporovaných souborů.

6.6 Podporované soubory

Přehled aktuálně podporovaných souborů pro záznam síťové komunikace (*Dump files*) :

- PCAP
- PCAP-ng

6.7 Metriky zdrojového kódu

Počet souborů	:	85 (včetně hlavičkových)
Počet řádků	:	26614 (včetně komentářů)
Počet základních aplikačních tříd	:	38
Velikost spustitelného souboru	:	1 014 455 b (bez ladících informací, Linux 64bit)
Velikost statických dat	:	4 192 b

Kapitola 7

Dosažené výsledky

Aplikace byla implementována způsobem popsaným v kapitole 6. Po odladění byly zhodnoceny dosažené výsledky.

Nástroj je možné přeložit a spustit na platformách GNU/Linux, MS Windows a MAC OS podporující minimálně 32bitové celočíselné datové typy. Po spuštění umožňuje aplikace přidávat odebírat a modifikovat tzv. generátory, které reprezentují tok paketů dané struktury.

U každého generátoru uživatel specifikuje jeho jednoznačně identifikující jméno a tzv. limity sekvence - neomezeno, omezeno počtem nebo omezeno celkovou vygenerovanou velikostí (v bajtech). Generátory mohou pracovat v módu sekvenčním nebo paralelním (simultánním). V sekvenčním módu uživatel specifikuje následující parametry: následující aktivovaný generátor, velikost shluku paketů (burst size) - počet vygenerovaných paketů bez prodlevy před přechodem na další generátor, prodleva před přechodem na další generátor (v nanosekundách) - fixní čas nebo náhodně vybraný čas ze zadaného rozsahu (výběr na základě zvoleného parametrizovatelného rozložení hustoty pravděpodobnosti). V paralením módu uživatel specifikuje pouze prodlevu mezi aktivacemi daného generátoru. Tuto prodlevu lze nastavit jako zpoždění (v nanosekundách), nebo jako počet vygenerovaných paketů za sekundu - v obou případech fixně nebo z rozsahu obdobně jako v sekvenčním módu. Všechna nastavení generátorů, včetně nastavení obsažených paketů, lze uložit jako tzv. session do textového XML souboru.

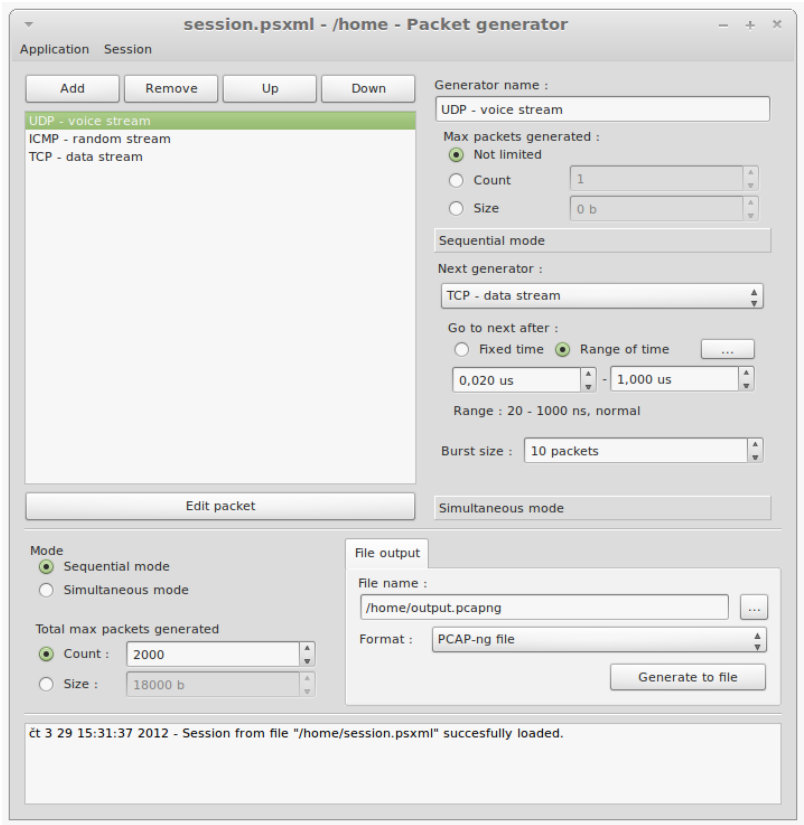
Nastavení paketu, který je generátorem v sekvenci generován, se provádí v tzv. okně editace paketu. Zde uživatel nejdříve specifikuje strukturu paketu. Strukturu může vytvořit postupným skládáním jednotlivých protokolů (enkapsulací) nebo výběrem a použitím šablony. Strukturu paketu je možno libovolně měnit i po následujících nastaveních, bez ztráty nastavení. Po vytvoření struktury může uživatel nastavit obsah hlaviček jednotlivých protokolů. Toto provádí zvlášť pro každý použitý protokol, na jeho vlastní záložce. Jednotlivá nastavení jsou specifická pro daný protokol. Většina nastavovaných hodnot je koncipována jako tzv. dynamické hodnoty. Uživatel díky tomu může danou hodnotu nastavit jako fixní, z rozsahu nebo z výčtu hodnot načtených z textového souboru. Výběr hodnoty z rozsahu nebo výčtu je možno nastavit jako inkrementální, nebo náhodný s daným parametrizovatelným rozložením hustoty pravděpodobnosti. V okně editace paketu si může uživatel také v hexadecimálním editoru zobrazit výslednou binární podobu paketu. V tomto editoru si lze také prohlížet, jak se bude paket měnit v průběhu sekvence. Pokud uživatel změní určitý bajt v tomto editoru, aplikace se pokusí přenést zadanou změnu do nastavení předpisu pro generování paketu (u některých nastavení, jako je například náhodný obsah, to ovšem nemá smysl a takováto změna je ignorována). Osamocený první paket ze sekvence, nebo ak-

tuálně zobrazený paket lze také exportovat do jakéhokoliv z podporovaných formátů (např. PCAP). Aplikace podporuje také částečný import vybraného paketu. Částečný proto, že před tímto importem je nutné ručně nastavit odpovídající strukturu paketu. Veškerá nastavení paketu lze uložit nebo načíst do/z vlastního XML souboru. Okno editace paketu také zpřístupňuje pomocnou funkci paketu, kterou je automatické vyplnění tzv. známých políček. Tato funkce požádá jednotlivé protokoly, aby na základě aktuální struktury a dalších dostupných informací automaticky vyplnily zjistitelné hodnoty, jako je například číslo následujícího protokolu a podobně. Lze také nastavit automatické provádění této funkce při změně struktury. Další z pomocných funkcí je zobrazení nápovědy. Při použití této volby se uživateli v pravé části okna při editaci daného protokolu zobrazuje stručná nápověda k sématice jednotlivých polí hlavičky daného protokolu.

V hlavním okně lze provést generování výstupu. Výstup je generován do souboru uživatelem specifikovaného typu. Po zadání příkazu ke generování aplikace vygeneruje série paketů dané nastavením a sloučený (serializovaný) výstup uloží do zadaného souboru. Generování je ukončeno po dosažení uživatelem specifikovaných globálních limitů (počet nebo celková velikost), nebo předčasně v situaci, kdy jednotlivé generátory již nemohou poskytnout další výstup (dosažení lokálních limitů).

Aplikace informuje uživatele o provedených akcích a případných chybách ve spodní části hlavního okna v podobě textového logu. Tento log lze také nechat vypisovat na standardní výstup, nebo do zadaného souboru.

Detailní popis ovládání lze nalézt příloze B - Manuál.



Obrázek 7.1: Hlavní okno aplikace, OS GNU/Linux - Mint

7.1 Srovnání s již existujícími nástroji

Srovnání je provedeno jak na základě objektivních měřítek, tak na základě čistě subjektivních názorů uživatelů.

Aplikace má grafické uživatelské rozhraní a lze spustit na nejpoužívanějších operačních systémech, tuto možnost nabízí pouze projekt Ostinato[15] - viz. 4.2.9. Oproti ostatním dostupným nástrojům (s i bez GUI) umožňuje aplikace větší škálovatelnost a to díky plné uživatelské kontrole nad strukturou paketu (nejrozšířenější přístup je výběr z předdefinované struktury). Nástroj umožňuje generovat série paketů různých typů - tuto možnost nabízí Colasoft Packet Builder[7] a projekt Ostinato. Kromě deterministicky definovaných sérií nabízí ale jako jediný také možnost stochasisticky modelovat síťový provoz. Pakety mohou také měnit svůj obsah - tuto možnost nabízí v omezné míře pouze nástroj Ostinato. Aplikace v současném stavu implementace nabízí podporu formátů PCAP a PCAP-ng. Dostupné nástroje nabízí většinou podporu pouze formátu PCAP (Vyjimku tvoří nástroj Colasoft Packet Builder s podporou deseti různých formátů ovšem mimo PCAP-ng). Pro ukládání nastavení používají nástroje stejně jako vytvořená aplikace čistě proprietální souborové formáty. Aplikace neumožňuje z výše popsaných důvodů narozdíl od většiny ostatních přímé vysílání výtvořených paketů do počítačové sítě. Vytvořená aplikace v současném stavu implementace podporuje osm základních protokolů - viz. 6.5. Tento počet i jejich typ je průměrem mezi dostupnými nástroji. Díky kvalitnímu objektovému návrhu a implementovaným podpůrným prostředkům je přidávání dalších protokolů velmi snadné. Při implementaci podpory protokolu je práce omezena pouze na vlastní protokol a není nutno zasahovat do zbytku aplikace.

Ovládání aplikace bylo hodnoceno na základě zkušeností testovacích uživatelů s vytvořenou aplikací a jinými. Uživatelé hodnotili ovládání aplikace jako přehledné a intuitivní stejně jako u ostatních nástrojů s grafickým uživatelským rozhraním. U testovacích uživatelů ale také převládál názor, že jeho použití oproti ostatním vyžaduje více znalostí z oboru síťových technologií.

7.2 Testování

Aplikace byla po odladění testována a kontrolována následujícími způsoby :

Inspekce obsahu vygenerovaných paketů

Aplikací byly vygnerovány a uloženy pakety dané struktury a obsahu. Výstupní soubor byl poté načten pomocí nástroje pro analýzu paketů (Wireshark[14] nebo TCPdump[12]). Po načtení byl porovnán očekávaný obsah s obsahem zobrazeným (analyzovaným).

Tento test byl také modifikován a realizován následujícím způsobem : Obsah výstupního souboru byl odeslán do jednoduché počítačové sítě pomocí nástroje TCPreplay[13]. Síť se skládala ze dvou počítačů navzájem propojených pomocí technologie Ethernet. Na straně příjmacího počítače byla spuštěna aplikace pro zachycení síťového provozu TCPdump[12]. Po dokončení odesílání bylo zachytávání zastaveno a zachycený obsah byl analyzován způsobem obdobným první variantě.

Test rozložení sledované hodnoty

Aplikací byly vygenerovány pakety dané struktury s vybranou hodnotou zadanou pomocí rozsahu s daným rozložením. Rozložení sledované hodnoty bylo poté analyzováno pomocí

aplikace Wireshark[14] a porovnáno s předpokládaným. Dále byly analyzovány prodlevy mezi pakety ve výstupním souboru. Četnost délek prodlevy byla zobrazena v podobě histogramu a porovnána s předpokládaným rozložením.

Kapitola 8

Závěr

Tato práce se zabývá problematikou generátorů síťových paketů. V její první části byla provedena rešerše dostupných nástrojů a knihoven pro generování paketů. Nástroje byly hodnoceny z hlediska několika kritérií. Diskutována byla použitelnost na různých operačních systémech, typ rozhraní, přes které je možno nástroj ovládat, množina podporovovaných protokolů, typ lincence i celkové možnosti použití.

Druhá část práce byla věnována tvorbě vlastního nástroje pro generování síťových paketů. Po shromáždění a doplnění požadavků na výslednou aplikaci byl sestaven koncept jejího fungování. Na základě tohoto konceptu byl vytvořen objektově orientovaný návrh. V tomto návrhu byl kladen důraz na univerzálnost použití a snadnou rozšiřitelnost zejména u podporovaných protokolů a souborových formátů. Návrh byl poté postupně implementován v programovacím jazyce C++. Po odladění a otestování aplikace byly zhodnoceny dosažené výsledky.

Vytvořená aplikace splňuje všechny požadavky odpovídající zadání. Kromě tohoto obsahuje aplikace také další přidanou funkčnost výrazně zvyšující možnosti jejího použití. Dostupné generátory umožňují pouze generování deterministicky určených jednotlivých paketů. Aplikace vytvořená v rámci této práce kromě tohoto umožňuje také generování sérií nebo skupin sérií paketů stochasisticky modelujících určitý síťový tok.

Nástroj umožňuje parametrizovat rozložení paketů v těchto sériích i obsah hlaviček obsažených protokolů. Uživatel určí množinu přípustných hodnot (rozsah nebo soubor) a způsob výběru z této množiny. Způsob výběru je možno nastavit jako inkrementální, nebo náhodný. U náhodného výběru lze poté nastavit typ rozložení hustoty pravděpodobnosti a parametrizovat ho. U simulace protokolů aplikační vrstvy, tzv. payload, lze celkovou délku specifikovat obdobným způsobem. Kromě tohoto zde lze ale rozložení pravděpodobnosti zadat také pomocí význačných bodů libovolné distribuční funkce. Takto lze například simulovat bimodální rozložení velikosti paketů v jejich toku[44].

Vytvořená aplikace také nabízí vnitřní rozlišení času v řádu nanosekund. Toto umožňuje přesněji definovat/simulovat požadovaný síťový tok a podporu souborových formátů nové generace, jako je například PCAP-ng. Standardním časovým rozlišením jsou mikrosekundy. Rozestupy v tomto řádu se ale v moderních síťových technologiích považují za nedostačující.

Aplikace je připravena pro budoucí rozšíření. Jednotlivé podporované protokoly lze díky jejich autonomnímu chování snadno implementovat a za použití standardního rozhraní nezávisle začlenit do aplikace. Obdobným způsobem lze doplnit i podporované souborové formáty.

Výstupem aplikace je soubor obsahující vygenerované pakety. Aplikace neumožňuje jejich přímé vysílání do počítačové sítě, zejména z toho důvodu, že nelze zajistit dostačně

rychlé sestavování jednotlivých paketů. Pro vysílání paketů do sítě exituje celé řada optimalizovaných nástrojů.

Použití aplikace je velice variabilní. Pomocí jí generovaného výstupu je možno například testovat odezvu aktivních síťových prvků na určitý síťový tok, jako jsou routery nebo firewaly a podobně.

Literatura

[1] Internetové stránky knihovny Libdnet [online]. 2012 [cit. 2011-12-13].
URL http://libdnet.sourceforge.net/

[2] Internetové stránky knihovny Libnet [online]. 2012 [cit. 2011-12-13].
URL http://packetfactory.openwall.net/projects/libnet/

[3] Internetové stránky knihovny Libpcap [online]. 2012 [cit. 2011-12-13].
URL http://www.tcpdump.org/

[4] Internetové stránky programu AnetTest [online]. 2012 [cit. 2011-12-13].
URL http://anettest.sourceforge.net/

[5] Internetové stránky programu Bit-Twist [online]. 2012 [cit. 2011-12-13].
URL http://bittwist.sourceforge.net/

[6] Internetové stránky programu Cat Karat Packet Builder [online]. 2012 [cit. 2011-12-13].
URL https://sites.google.com/site/catkaratpacketbuilder/

[7] Internetové stránky programu Colasoft Packet Builder [online]. 2012 [cit. 2011-12-13].
URL http://www.colasoft.com/packet_builder/

[8] Internetové stránky programu Nemesis [online]. 2012 [cit. 2011-12-13].
URL http://nemesis.sourceforge.net/

[9] Internetové stránky programu PackETH [online]. 2012 [cit. 2011-12-13].
URL http://packeth.sourceforge.net/

[10] Internetové stránky programu Pierf [online]. 2012 [cit. 2011-12-13].
URL http://pierf.sourceforge.net/

[11] Internetové stránky programu Scapy [online]. 2012 [cit. 2011-12-13].
URL http://www.secdev.org/projects/scapy/

[12] Internetové stránky programu TCPdump [online]. 2012 [cit. 2011-12-13].
URL http://www.tcpdump.org/

[13] Internetové stránky programu TCPreplay [online]. 2012 [cit. 2011-12-13].
URL http://tcpreplay.synfin.net/

[14] Internetové stránky programu Wireshark [online]. 2012 [cit. 2011-12-13].
URL http://www.wireshark.org/

[15] Internetové stránky projektu Ostinato [online]. 2012 [cit. 2011-12-13].
URL http://code.google.com/p/ostinato/

[16] Internetové stránky knihovny GMP (The GNU Multiple Precision Arithmetic Library) [online]. 2012 [cit. 2012-3-18].

[17] Internetová stránka programu Doxygen [online]. 2012 [cit. 2012-4-1].
URL http://www.stack.nl/~dimitri/doxygen/

[18] Internetové stránky frameworku Qt (Qt - A cross-platform application and UI framework) [online]. 2012 [cit. 2012-4-10].
URL http://qt.nokia.com/

[19] Internetové stránky společnosti Spirent [online]. 2012 [cit. 2012-4-10].
URL http://www.spirent.com/

[20] Internetové stránky věnované formátu Libpcap (Libpcap File Format) [online]. 2012 [cit. 2012-4-5].
URL http://wiki.wireshark.org/Development/LibpcapFileFormat

[21] Internetové stránky věnované formátu PCAP-ng (PCAP Next Generation Dump File Format) [online]. 2012 [cit. 2012-4-5].
URL http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.%html

[22] Borman, D.; Deering, S.; Hinden, R.: IPv6 Jumbograms. RFC 2675 (Proposed Standard), Srpen 1999.
URL http://www.ietf.org/RFC/RFC2675.txt

[23] Cain, B.; Deering, S.; Kouvelas, I.; aj.: Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), Říjen 2002, updated by RFC 4604.
URL http://www.ietf.org/RFC/RFC3376.txt

[24] Cisco Systems, Inc.: *Internetworking Technologies Handbook (4th Edition)*. Cisco Press, Čtvrté vydání, 2004, ISBN 978-1-58705-119-7.

[25] Conta, A.; Deering, S.; Gupta, M.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), Březen 2006, updated by RFC 4884.
URL http://www.ietf.org/RFC/RFC4443.txt

[26] Deering, S.; Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Prosinec 1998, updated by RFCs 5095, 5722, 5871.
URL http://www.ietf.org/RFC/RFC2460.txt

[27] Eastlake 3rd, D.: IANA Considerations and IETF Protocol Usage for IEEE 802 Parameters. RFC 5342 (Best Current Practice), Září 2008.
URL http://www.ietf.org/RFC/RFC5342.txt

[28] Farinacci, D.; Cai, Y.: Anycast-RP Using Protocol Independent Multicast (PIM). RFC 4610 (Proposed Standard), Srpen 2006.
URL http://www.ietf.org/RFC/RFC4610.txt

[29] Fenner, B.; He, H.; Haberman, B.; aj.: Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying"). RFC 4605 (Proposed Standard), Srpen 2006.
URL http://www.ietf.org/RFC/RFC4605.txt

[30] Finlayson, R.; Mann, T.; Mogul, J.; aj.: A Reverse Address Resolution Protocol. RFC 903 (Standard), Červen 1984.
URL http://www.ietf.org/RFC/RFC903.txt

[31] Gerich, E.: Guidelines for Management of IP Address Space. RFC 1466 (Informational), Květen 1993, obsoleted by RFC 2050.
URL http://www.ietf.org/RFC/RFC1466.txt

[32] Hinden, R.; Deering, S.: IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Únor 2006, updated by RFCs 5952, 6052.
URL http://www.ietf.org/RFC/RFC4291.txt

[33] IANA: IANA IPv4 Address Space Registry [online]. 2012 [cit. 2012-4-15].
URL http://www.iana.org/assignments/ipv4-address-space/

[34] IANA: Service Name and Transport Protocol Port Number Registry [online]. 2012 [cit. 2012-4-5].
URL http://www.iana.org/assignments/service-names-port-numbers/

[35] IEEE Computer Society: *IEEE Std 802.3-2008*. The Institute of Electrical and Electronics Engineers, Inc, 2008, ISBN 973-07381-5796-2.

[36] Jacobson, V.; Braden, R.; Borman, D.: TCP Extensions for High Performance. RFC 1323 (Proposed Standard), Květen 1992.
URL http://www.ietf.org/RFC/RFC1323.txt

[37] Javvin Technologies: *Network Protocols Handbook*. Javvin Technologies, Incorporated, 2004, ISBN 0-9740945-2-8.

[38] Plummer, D.: Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), Listopad 1982, updated by RFCs 5227, 5494.
URL http://www.ietf.org/RFC/RFC826.txt

[39] Postel, J.: User Datagram Protocol. RFC 768 (Standard), Srpen 1980.
URL http://www.ietf.org/RFC/RFC768.txt

[40] Postel, J.: Internet Control Message Protocol. RFC 792 (Standard), Září 1981, updated by RFCs 950, 4884.
URL http://www.ietf.org/RFC/RFC792.txt

[41] Postel, J.: Internet Protocol. RFC 791 (Standard), Září 1981, updated by RFC 1349.
URL http://www.ietf.org/RFC/RFC791.txt

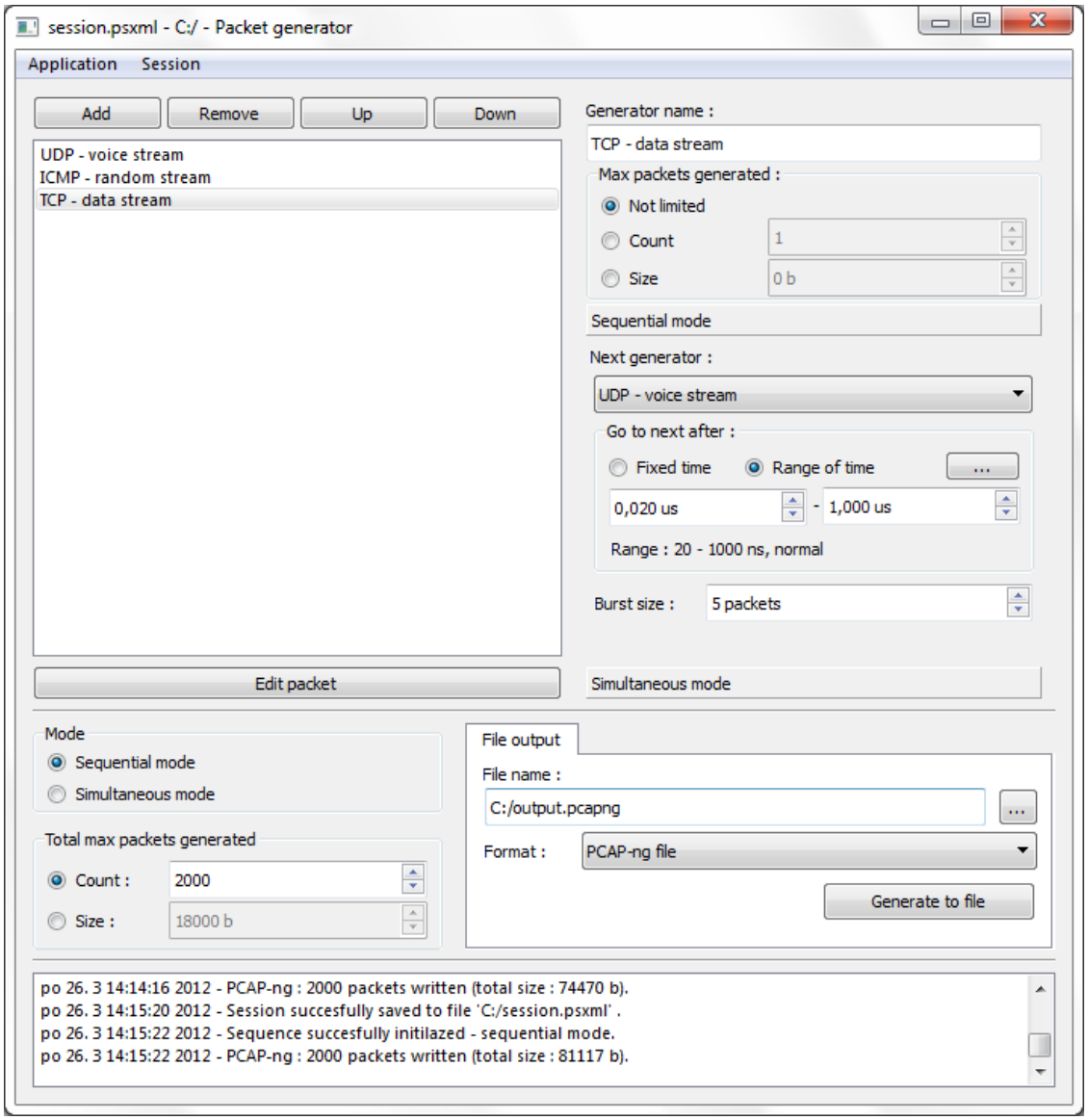
[42] Postel, J.: Transmission Control Protocol. RFC 793 (Standard), Září 1981, updated by RFCs 1122, 3168, 6093.
URL http://www.ietf.org/RFC/RFC793.txt

[43] Reynolds, J.: Assigned Numbers: RFC 1700 is Replaced by an On-line Database. RFC 3232 (Informational), Leden 2002.
URL http://www.ietf.org/RFC/RFC3232.txt

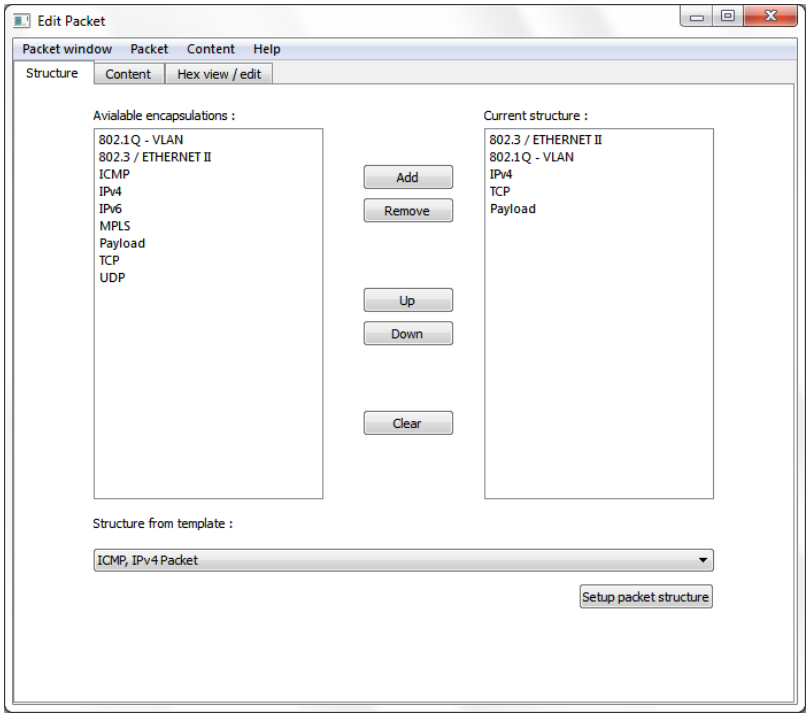
[44] Sinha, R.; Papadopoulos, C.; Heidemann, J.: Internet Packet Size Distributions: Some Observations. Technická Zpráva ISI-TR-2007-643, USC/Information Sciences Institute, May 2007, orignally released October 2005 as web page http://netweb.usc.edu/~rsinha/pkt-sizes/.
URL http://www.isi.edu/~johnh/PAPERS/Sinha07a.html

Příloha A

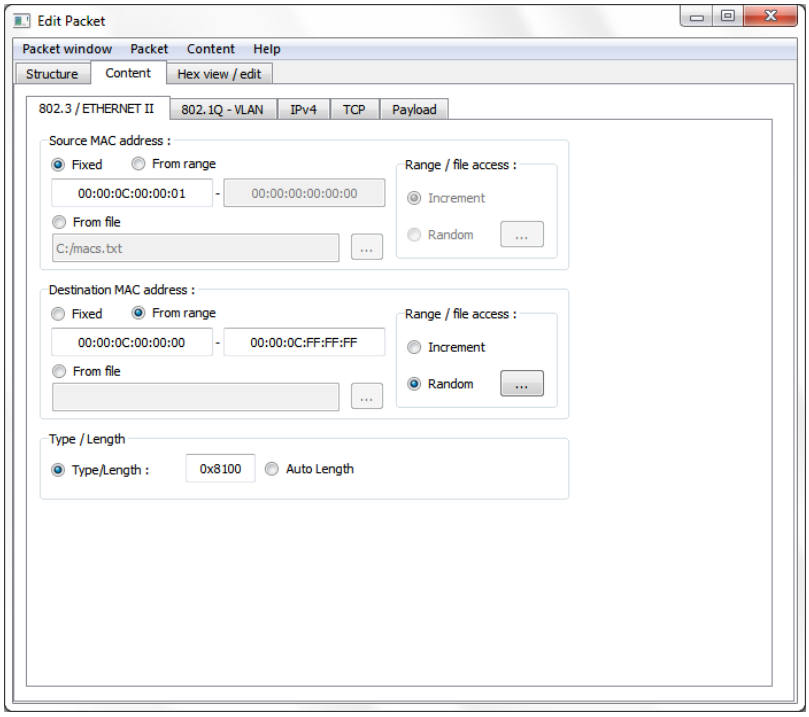
Snímky uživatelského rozhraní



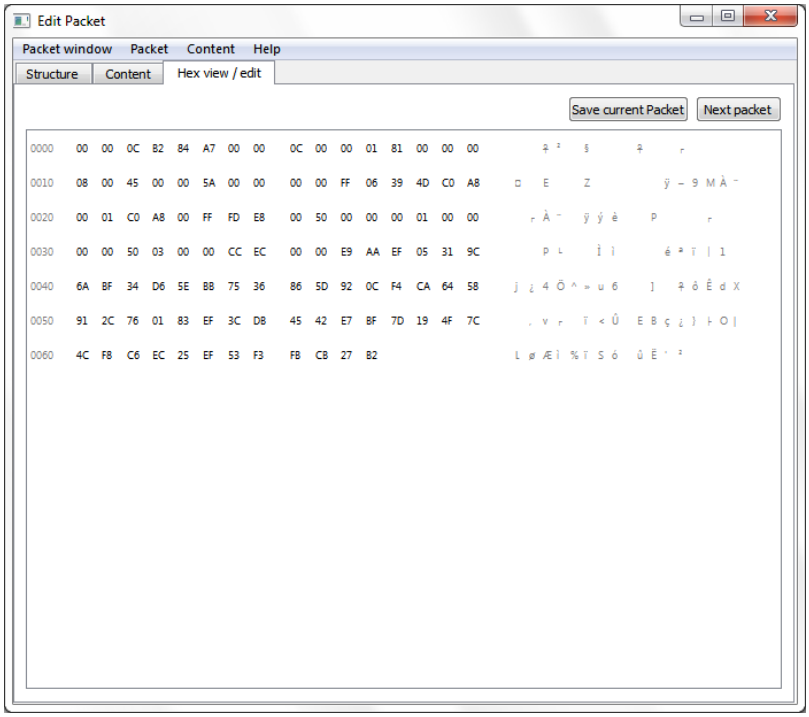
Obrázek A.1: Hlavní okno aplikace, OS MS Windows 7



Obrázek A.2: Okno editace packetu - struktura, OS MS Windows 7



Obrázek A.3: Okno editace packetu - obsah, OS MS Windows 7



Obrázek A.4: Okno editace packetu - hexadecimální editor, OS MS Windows 7

Příloha B

Manuál

B.1 Spuštění aplikace

Aplikaci lze spustit z grafického rozhraní systému, nebo z jeho příkazové řádky za volitelného použití těchto parametrů (argumentů):

- `-h` - zobrazení nápovědy
- `-f [filename]` - soubor s uloženou session (viz. 6.2.5) - po inicializaci aplikace bude tento soubor automaticky načten
- `-w [level]` - zapne výpis logu aplikace na standartní výstup, uroveň výpisu je specifikována číslem "level"(0 - všechny zprávy, 1 - pouze varování a chyby, 2 - pouze chyby)
- `-a [filename]` - nastavení cesty k XML souboru s nápovědou - pokud není nastaveno, je použita cesta `./help.phxml`
- `-t [filename]` - nastavení cesty k XML souboru se šablonami struktur paketu - pokud není nastaveno, je použita cesta `./templates.ptxml`

Po spuštění se zobrazí hlavní okno aplikace.

B.2 Práce s generátory

V hlavním okně aplikace v levé horní části se nachází čtveřice tlačítek pro práci s generátory. Tlačítko "Add"pro přidání generátoru, tlačítko "Remove"pro odebrání vybraného generátoru a tlačítka "Up"a "Down"pro přesun generátoru nahoru nebo dolů. Všechny provedené změny se okamžitě projeví v seznamu pod těmito tlačítky zobrazující aktuální stav a pořadí všech generátorů. Při přidávání generátoru je uživatel vyzván k zadání jeho jména. Jméno může být libovolný textový řetězec jednozačně identifikující přidávaný generátor (jméno musí být unikátní). Jméno generátoru lze později změnit - viz níže.

Po zvolení generátoru (výběrem ze seznamu) jsou napravo od toho seznamu automaticky zpřístupněny ovládací prvky pro jeho editaci. Pole "Generator name"slouží ke změně názvu generátoru. Box "Max packets generated"slouží k volbě limitů generátoru. Limit lze nastavit takto: "Not limited"- limit není nastaven, "Count"- omezení počtem vygenerovaných paketů, "Size"- omezení součtem velikostí vygenerovaných paketů (v bajtech).

V panelu "Sequential mode"se nastavují parametry generátoru pro běh v tzv. sekvenčním módu, viz. B.4. Volba "Next generator"slouží k určení generátoru, který bude aktivovován po vygnerování balíku paketů z právě editovaného generátoru. Box "Go to next after"slouží k nastavení prodlevy mezi koncem generování a aktivací vybraného následujícího generátoru. V tomto boxu lze nastavit fixní čas nebo zadat čas rozsahem. Pokud je čas zadán rozsahem, může uživatel zvolit rozložení pravděpodpobnosti a jeho prametry pro výběr z tohoto rozsahu kliknutím na tlačítko se symbolem tří teček v tomto boxu. Poslední nastavení pro sekvenční mód je tzv. "Burst size". Tato volba specifikuje počet paketů vygenerovaných bez prodlevy před aktivací dalšího generátoru.

V panelu "Simultaneous mode"se nastavují vlastnosti generátoru pro běh v tzv. paralelním módu - viz. B.4. Prodlevu mezi aktivacemi editovaného generátoru lze v tomto panelu nastavit dvěma způsoby - jako prodlevu ("Delay") v ns, nebo jako počet paketů vygenerovaných za sekundu ("Packets per second"). Oba typy se nastavují obdobně jako prodleva v sekvenčním módu - viz. výše.

Aktuální stav a nastavení generátorů včetně obsažených generátorů lze uložit nebo načíst z/do XML souboru za použití tlačítek v menu "Session".

B.3 Editace paketu

Editace paketu generovaného zvoleným generátorem je prováděno ve vlastním okně. Toto okno lze vyvolat po výběru generátoru kliknutím na tlačítko "Edit packet".

Okno editace paketu obsahuje tři záložky. Záložka "Structure"slouží k nastavení struktury paketu. Záložka "Content"slouží k nastavení obsahu paketu a záložka "Hex view / edit"slouží k prohlížení a editaci obsahu paketu v hexadecimálním editoru.

Strukturu paketu lze vytvářet přidáváním dostupných protokolů z listu "Aviabale encapsulations"do aktuální struktury "Current structure". Přidání protokolu - tlačítko "Add", odebrání protokolu - tlačítko "Remove", přesun protokolu ve struktuře nahoru nebo dolů - tlačítka "Up"a "Down". Strukturu lze také vytvořit výběrem ze šablon - rozbalovací nabídka "Structure from template"a kliknutí na "Setup packet structure".

V závislosti na aktuální struktuře paketu obsahuje záložka pro editaci obsahu záložky s názvy jednotlivých použitých protokolů. Na každé takovéto záložce lze provádět nastavení odpovídající obsahu jednotlivých protokolů. Obecně lze většinu hodnot nastavovat jako tzv. dynamické. Dynamická hodnota je specifkována typem: fixní "fixed", z rosahu "from range", nebo ze souboru "from file"a vlastními daty - meze rozsahu, nebo zdrojový soubor s daty (textový soubor s hodnotami oddělenými konci řádků). U hodnoty specifkované rozsahem nebo souborem lze také specifkovat metodu výběru z těchto hodnot - inkrementální "increment"nebo náhodná "random". U náhodné lze poté v dialogu (vyvolání tlačítkem se symbolem tří teček) specifkovat také rozdělení pravděpodobnosti a jeho příslušné parametry.

Na záložce prohlížení a editace si lze prohlédnout binární podobu paketu zobrazenou jako hexadecimální zápis bajtů. Dále je zde možno prohlížet změny při průchodu série paketu - talčítko "Next packet".

Aktuálně editovaný paket lze načíst nebo uložit do XML formátu aplikace, nebo exporotovat a importovat z jekéhokoliv z podporovaných souborvých formátů - menu "Packet".

Po dokončení editace paketu lze okno editace paketu uzavřít - změny jsou uloženy okamžitě po jejich provedení.

B.4 Generování výstupu

Před generováním výstupu je nutno vybrat tzv. mód generování. Výběr lze provést v hlavním okně apliakce v boxu "Mode". Podle vybraného módu bude řízeno generování výstupu.

V sekvenční módu ("Sequential mode") je nejdříve aktitován první generátor v pořadí. Aktivní generátor vygeneruje počet paketů daných nastavením velikostí shluku. Po vypočtené prodlevě, je aktivován generátor specifikovaný v nastavení - viz. Práce s generátory. Pokud aktivovaný generátor neobsahuje odkaz na další, nebo byly vyčerpány všechny lokální nebo globální limity, je generování ukončeno.

V paralelením módu ("Simultaneous mode") program simuluje paralelní běh všech generátorů současně. Pokud má být v daný časový okamžik aktivováno více generátorů, rozhoduje jejich pořadí. Generování v tomto módu končí po vyčerpání všech lokálních limitů, nebo po vyčerpání globálních limitů.

Po výběru požadovaného módu a nastavení všech generátorů je třeba specifkovat globální limity, výstupní formát a cestu k výstupnímu souboru. Globální limity se nastavují v boxu "Total max packets generated". Lze zvolit celkový mazimální počet, nebo celkovou maximální velikost (v bajtech). Formát lze nastavit pomocí rozbalovací nabídky "Format". Cestu k výstupnímu souboru lze zadat ručně do řádku "File name", nebo za použití systémového dialogu po kliknutí na talčítko se symbolem tří teček napravo od toho řádku.

Po kliknutí na tlačítko "Generate to file", nebo výběrem z menu "Session -> Generate to file (Ctrl + G)"je provedeno generování a výstup je uložen do specifikovaného souboru. O průběhu generování je uživatel informován prostřednictvím logu ve spodní části hlavního okna.

Příloha C

Obsah CD

- Adresář `application/src/` - zdrojové kódy aplikace
- Adresář `application/doc/` - vygenerovaná dokumentace ve formátu HTML
- Adresář `application/bin/win_x86` - spustitelná aplikace pro OS MS Windows 32bit
- Adresář `application/bin/win_x64` - spustitelná aplikace pro OS MS Windows 64bit
- Adresář `application/bin/linux_x64` - spustitelná aplikace pro GNU/Linux 64bit
- Adresář `text/TeX` - zdrojové kódy textu práce ve formátu LaTeX
- Adresář `text/pdf` - text práce ve formátu PDF