VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 David Hudec

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

GENEROVÁNÍ ZÁPLAVOVÝCH ÚTOKŮ
GENERATING OF FLOOD ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **David Hudec**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. David Smékal**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: David Hudec ***ID:*** 165289
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Generování záplavových útoků

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a podrobně popište princip záplavových útoků a jejich typy. Seznamte se s možností simulace pomocí softwaru (Riverbed Modeler, NS3). Pokuste se realizovat záplavový útok pomocí vysokorychlostní síťové karty na platformě FPGA. Změřte parametry generovaných útoků. Následně vyhodnoťte příčiny a důsledky útoku, možnosti zabezpečení.

DOPORUČENÁ LITERATURA:

[1] PROWELL, Stacy J, Rob KRAUS, Mike BORKIN. Seven deadliest network attacks. Boston: Syngress, 2010, 142 s. ISBN 15-974-9549-2.

[2] STALLINGS, William. Cryptography and network security: principles and practice. 6. vyd. Upper Saddle River: Prentice Hall, 2013, 752 s. ISBN 01-333-5469-5.

[3] WOLF, Wayne. FPGA based system design. New Jersey: Prentice-Hall, 2004, 530 s. ISBN 0-13-142461-0.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. David Smékal
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Obsahem teoretické části práce je rozbor záplavových útoků, jejich možností, používaných taktik a metod, a popis simulace vyrobené za tímto účelem. Prostředí takto vytvořené ukazuje chování sítě napadené zkoumaným typem útoku a demonstruje potenciální postup a prostředky útočníka. Navazující praktická část poté popisuje tvorbu záplavových dat dvěma převzatými hardwarovými řešeními a jedním softwarovým, představovaným vlastní C# aplikací. Srovnání těchto přístupů, vyhodnocení výsledků útoků a návrh obrany proti nim jsou rovněž uvedeny.

KLÍČOVÁ SLOVA

FPGA, VHDL, DoS, DDoS, NetCOPE, útok, síť, síťový tester, síťový útok, generování útoku, generování packetu, ns-3, netanim

ABSTRACT

The assessment comprises of two parts, describing theory and generating of flood attacks respectively. The first part covers flood attacks' analysis, deals with their available techniques and practices, known in the area, and a computer simulation program, revealing the behavior of a contested network as well as the attacker's procedure. In the following part, data generating solutions itself are described. These are represented by two hardware programs, adapted from existing solutions, and one C# application, created by the author. The comparison of these two approaches is included, as well as are the generation results and mitigation proposal.

KEYWORDS

FPGA, VHDL, DoS, DDoS, NetCOPE, attack, network, network tester, network attack, attack generation, packet generation, ns-3, netanim

HUDEC, David *Generování záplavových útoků*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 54 s. Vedoucí práce byl Ing. David Smékal,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Generování záplavových útoků“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Davidu Smékalovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

1 ÚVOD

Otázka bezpečnosti počítačových sítí se spolu s rozvojem této oblasti nese ruku v ruce od samotných počátků programovatelných strojů. Avšak mnohem více, než tehdy, zabírají počítače v dnešní společnosti neodmyslitelnou pozici, spoléháme na ně a mnoho věcí by bez jejich pomoci nebylo vůbec možných. Jsou všude kolem nás, používáme je, svěřujeme jim svoje data, ale často je nacházíme i tam, kde by je neznalý člověk nečekal. Aby mohly tyto jednotky fungovat, být řízeny, nastavovány a spouštěny, připojují se většinou jedna k druhé a vytváří tak sítě; největší takovou sítí je *internet*, jenž reprezentuje možnost globální výměny dat mezi počítači, které k němu mají zajištěné připojení – tedy jsou on-line. Komunikaci po informačních kanálech, spojujících jednotlivé stroje, řídí servery – jiné specializované počítače, uložené na bezpečných místech po celém světě. Ty jsou navázány mezi sebou a mezi jednotkami nižších úrovní logické struktury a mají tak zajištěné spojení na vše, co potřebují.

S celou touto infrastrukturou je pochopitelné, že se najdou bezpečnostní díry zneužitelné k útoku na síť, stejně jako lidé schopní a ochotní tento útok provést. Takovýchto napadení se děje nespočetné množství každý den, je však potřeba rozlišit *napadení dat*, kdy jsou cílem soukromá data uživatele a jejich zneužití nebo zpeněžení, a *napadení služby* s úmyslem vyřadit ji z provozu a odepřít k ní přístup požadavkům legitimních uživatelů, což je případ Denial of Service (DoS) útoků. Zatímco ale poslední dobou oblíbenost těchto úderů roste kvůli jejich snižující se náročnosti, přiměřená obrana zůstává pořád stejně složitá. To je způsobeno otevřeností celého internetového světa, jeho podstatou a vrozenou naivitou. U DoS a z nich odvozených útoků se tedy častěji řeší následky, než příčiny, neboť ty buď nelze odhalit a identifikovat vůbec, nebo ne v čase, kdy je to potřeba.

Jako příklad poslouží přihlašovací webová stránka internetového obchodu. Pokud server, sloužící v pozadí jako obsluha tohoto webu, dokáže zpracovat pouze sto požadavků o přihlášení za sekundu, protože s větší zátěží nebyl při návrhu důvod počítat, stačí útočníkovi vygenerovat alespoň sto falešných žádostí za vteřinu, a pokud se server pod tímto náporem úplně nezhroutí, bude alespoň po dobu tohoto zákeřného provozu zamezeno legitimním uživatelům v jejich přihlášení. To může u on-line obchodníků nebo třeba poskytovatelů Software as a Service (SaaS) znamenat velký ušlý zisk.

1.1 Cíle bakalářské práce

Cílem této práce je popsat používané techniky útoků DoS a DDoS, jejich klíčové vlastnosti a trendy. V teoretické polovině zjistit možnosti a postupy tvorby takovýchto útoků dostupnými nástroji. Na základech těchto poznatků pak vytvořit počítačovou simulaci napadené sítě s vyhodnocením dopadů útoku. Tato simulace bude probíhat v prostředí Network simulator 3 (ns-3), spolupracujícím s programy NetAnim a Wireshark v OS Ubuntu a pomocí Riverbed Modeleru v OS Windows.

V následující praktické části práce je pak cílem uplatnit poznatky získané z předešlého a provést pokus o uskutečnění záplavového DoS útoku v reálné uzavřené síti na FEKT VUT. V tomto testovacím prostředí budou data generována pomocí vývojové desky technologie FPGA. Takovýto útok následně změřit, analyzovat a zjistit možnosti vytváření skutečného útoku pomocí navrženého hardwaru s vlastním softwarem, spolu s vyhodnocením případných možností obrany. Srovnat schopnost generování dat softwarovou a hardwarovou cestou.

2 TEORIE O ÚTOCÍCH

V kapitole jsou shrnuty podstatné vlastnosti škodlivých internetových záplav.

2.1 Zevrubný popis útoků

Síťové útoky tedy tvoří podstatnou část internetového povědomí, jsou obavou každé firmy a nikdo, kdo používá internet, před nimi nejspíš není nikdy plně chráněn. Je nepravděpodobné, že se je podaří jakýmkoli způsobem v budoucnu vymýtit, a tak je třeba spoléhat na obranu proti nim. Ta, aby byla co nejúčinnější, musí sledovat, dohánět, nebo dokonce předbíhat vývoj v oblasti, ve které působí. Abychom tedy mohli porozumět, jak síťové DoS a DDoS útoky ničit, nebo naopak vytvářet a generovat, musíme vědět, jak fungují. To je detailněji popsáno v této kapitole.

Podle US-CERT[1] jsou symptomy napadení ve smyslu DoS tyto:

- neobvyklé zpomalení služby,
- celková nedostupnost části nebo celých stránek,
- nemožnost se ke stránkám připojit,
- extrémní nárůst obdrženého spamu.

Splnění tohoto popisu však nemusí značit útok, může jít o problém nebo výpadek na straně techniky. Potvrdit to naopak může prozkoumání vývoje provozu na síti, kde bude probíhající invaze poznat z mimořádného vytížení některých uzlů nebo linek.

2.1.1 DoS – Denial of service

Ačkoli může DoS znít složitě, jde o jednoduchý princip. Definice podle Technopedie[2] v překladu říká:

DoS je jakýkoli typ útoku, kdy se útočníci (hackeři) pokusí zabránit legitimním uživatelům v přístupu k službě.

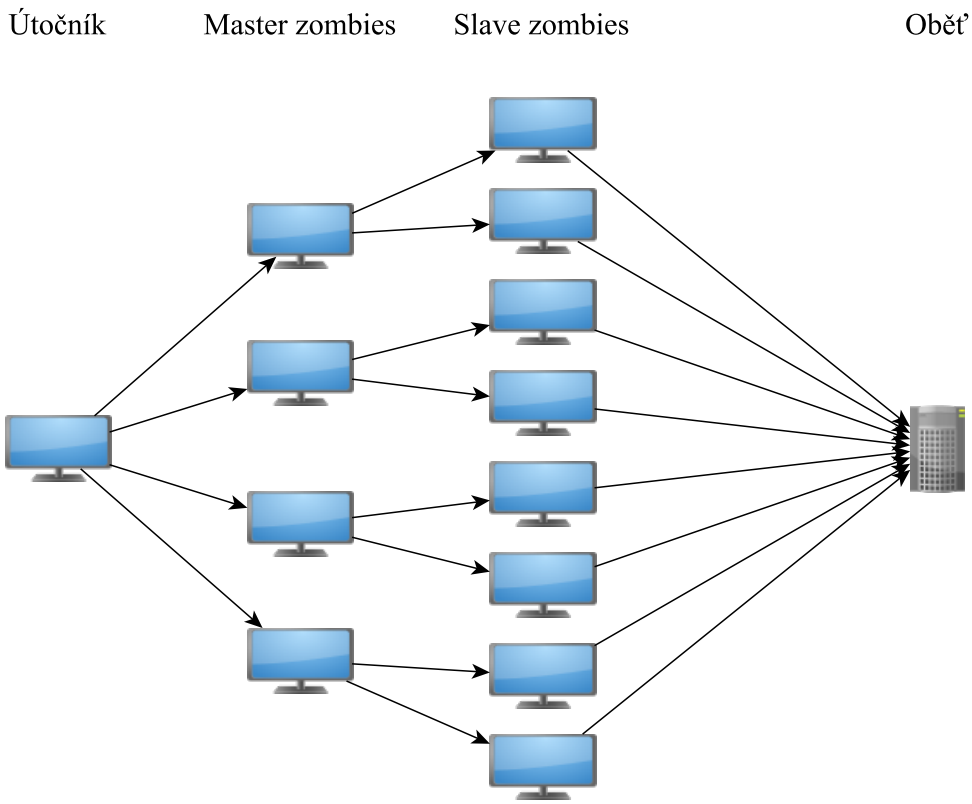
Obr. 2.1: Schéma DoS

Při takovémto počínání útočník posílá zprávy nadměrné velikosti nebo počtu s požadavky s často podvrhnutými zdrojovými adresami. Server je zatížen buď samotným

objemem dat, nebo množstvím požadavků. Navíc je možné žádosti poslat tak, aby server čekal s vyhrazenými prostředky pro další zprávy, které ovšem nepřijdou. V případě DoS je takový čin veden z jediného počítače, vykonávajícího veškeré činnosti (obr. 2.1).

2.1.2 DDoS – Distributed denial of service

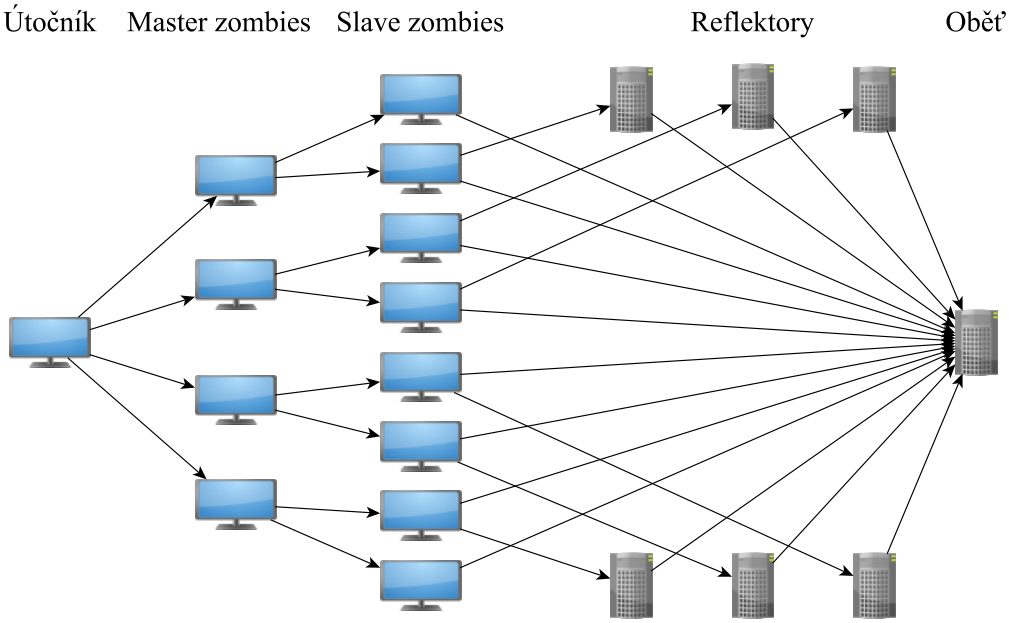
U DDoS se zasílání škodlivých zpráv účastní velké množství počítačů, tzv. *slave zombies*, otroků, které jsou řízeny *master zombies*. Útočník tedy seskupil více stanic, buď jejich infikováním, nebo dobrovolným náborem, a zadáním příkazu k útoku začnou všechny tyto master zombies spouštět zahlcování cíle útoku nebezpečným provozem pomocí slave zombies. Tím je zajištěno lepší utajení samotného strůjce a zároveň mnohonásobně větší poškození oběti, jak je vidět na obr. 2.2. Největší takový útok byl zaznamenán[3] v únoru roku 2014, kdy datový tok nepravých zpráv dosahoval 400 Gb/s a šlo o NTP DDoS (vysvětleno v sekci 2.2.7), nicméně v budoucnosti jsou předpovídány DRDoS (oddíl 2.1.3) pokusy s ještě větším provozem – kolem 1 Tb/s.



Obr. 2.2: Schéma DDoS

2.1.3 DRDoS – Distributed reflected denial of service

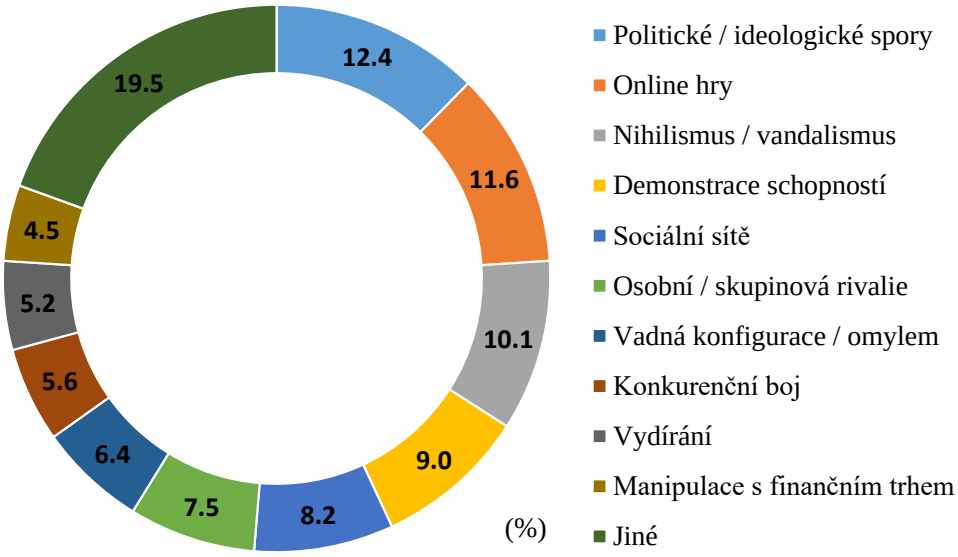
Distribuované odrážené odepření služby má s běžným DDoS společnou první část. Iniciované master zombies spouští slave zombies a ty zahlcují oběť nadměrným zákeřným provozem. Kromě toho ale také slave zombies kontaktují stanice třetích stran s veřejnou zneužitelnou IP adresou, jako jsou servery, například DNS (Domain name system), a v této žádosti zamění zdrojovou (svou) adresu za adresu oběti. Tyto cizí neinfikované jednotky potom odpovídají oběti ve snaze vyřešit její dotaz a přispívají tak jejímu přetížení, znázorněno na obr. 2.3.



Obr. 2.3: Schéma DRDoS

2.1.4 Motivace a zaměření útočníka

Nejčastějším důvodem, proč se uchýlit k odpírání cizí služby, je ideologický hacktivismus – vyjadřování politických či sociálních postojů a nesouhlasů pomocí hackingu. Spolu s nihilismem mohou být takové útoky směřovány proti vládním organizacím, politickým stranám nebo jiným ideologickým skupinám. Podle grafu 2.4[4] podstatnou část také zabírají motivace spojené s on-line gamingem, kde může jít o odplatu soupeři nebo celému hernímu serveru. Důležité ovšem je, že již převládajícím důvodem nejsou peníze, jako v případě vydírání nebo mezikonkurenčního poškozování, ale odlišnost názorů, která může v extrémním případě vést až ke kyberterorismu, tedy odstavování systémů důležitých pro fungování státu nebo státní infrastruktury. Pohnutky k DoS shrnuje následující graf.



Obr. 2.4: Důvody vedoucí k DoS útoku[4]

Zaměření a následné dopady na postižené firmy a organizace lze předpokládat zejména v oblastech:

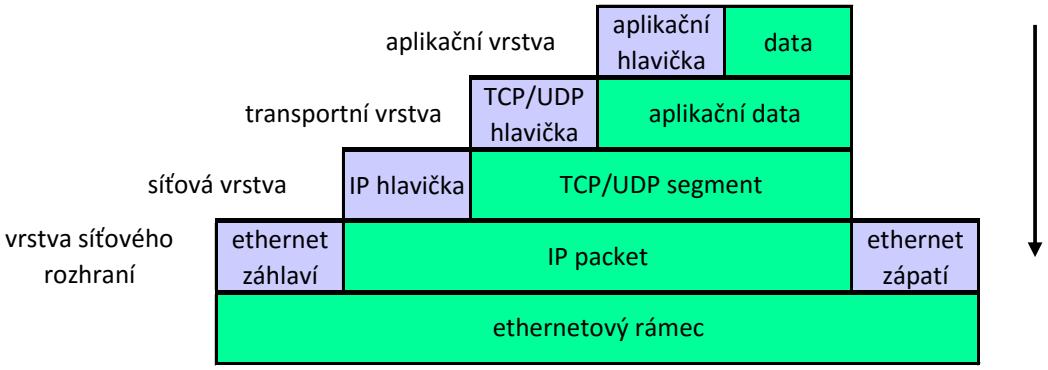
- poškození dobré reputace,
- ztráty příjmů a
- krádeže dat,

kdy jsou první dvě výsledkem nedostupnosti nabízené či prodávané služby, a krádež dat nebo intelektuálního vlastnictví dosažena dalším, přímým útokem, využívajícím DoS pro odlákání pozornosti.

2.1.5 TCP/IP model

Model sítě je sada pravidel, rozdělujících komunikaci mezi stanicemi do logických vrstev. V současné době populární TCP/IP (Transmission Control Protocol / Internet Protocol) model vychází z původního referenčního modelu ISO/OSI a rozděluje spojení do čtyř vrstev. Při odesílání dat se provádí tzv. *zapouzdření* (encapsulation) od nejvyšší vrstvy po nejnižší, podle tabulky 2.1.

Typický příklad vypadá takto: aplikační vrstva – nejvyšší – přijme data a přidá k nim aplikační hlavičku. Tento celek poté předá nižší, transportní vrstvě, ta data rozdělí, přidá TCP (nebo UDP) popis a vznikne *segment*. Další vrstva, síťová, doplní každému segmentu IP hlavičku, vzniknou IP *packety*, ke kterým poslední vrstva



Tab. 2.1: Postup zapouzdření v TCP/IP modelu

síťového rozhraní přidá svoje informace, umístěné na začátek, vypočítá kontrolní frame check sequence (FCS) na konec a vytvoří ethernetové *rámce*. Po přenosu sítí přijme cílové zařízení takovýto rámec a provede *rozbalení* (decapsulation) od nejnižší vrstvy nahoru, až aplikační vrstva získá přijatá data.

2.1.6 TCP segment

Datová jednotka transportní vrstvy síťového modelu se nazývá *datagram* v případě použití protokolu UDP (User Datagram Protocol), nebo *segment* protokolu TCP. Oba dva lze souhrnně označit jako *Protocol Data Unit* (PDU). Segment se skládá z hlavičky, obsahující meta data o přenosu, a samotných dat. Obsah prvního je vidět v tabulce 2.2.

byte		0				1				2				3			
	bit	0 - 3			4 - 7	8 - 11			12 - 15	16 - 19			20 - 23	24 - 27			28 - 31
0	0	číslo zdrojového portu								číslo cílového portu							
4	32	sekvenční číslo (SYN)															
8	64	potvrzovací číslo (ACK)															
12	96	délka hlavičky			rezerva	NS	CWR	ECE	URG	ACK	PSH	PST	SYN	FIN	velikost okna		
16	128	kontrolní součet								ukazatel urgence							
20 ..	160 ..	další (volitelné) informace															

Tab. 2.2: Obsah hlavičky TCP segmentu

Délka hlavičky je udávána v počtu 32 bitových slov, minimem je pět, maximem patnáct. Rezervní bity jsou ponechány pro budoucí, zatím neznámé účely. Velmi důležitým parametrem je velikost okna, která udává, kolik bytů je možno přenést

před souhrnným potvrzením jejich přijetí. Tato hodnota může být na straně odesilatele a příjemce různá, neměla by být nastavována příliš nízko (pak by režie zabrala velkou část přenesených dat), ani příliš vysoko (omezila by počet potvrzení). Parametr lze zneužít k útoku.

Modře jsou v těle tabulky vyznačeny TCP příznaky (flags). Tyto jsou jednobitové hodnoty, které lze buď zapnout (nastavit), nebo ponechat neaktivní; protokol jejich pomocí kontroluje spojení. Nastavené pole `URG` značí, že se jedná o spěšná data. `ACK` a `SYN` oznamují platnost potvrzovacího, případně sekvenčního čísla, uvedeného v samostatném poli. Pomocí `PSH` je požadováno okamžité doručení segmentu vyšší vrstvě, `FIN` je žádost o ukončení spojení a `RST` spojení ukončuje bez jakékoli další domluvy. `NS`, `CWR` a `ECE` nejsou ani hojně používané, ani pro tuto práci důležité.

2.1.7 IP packet

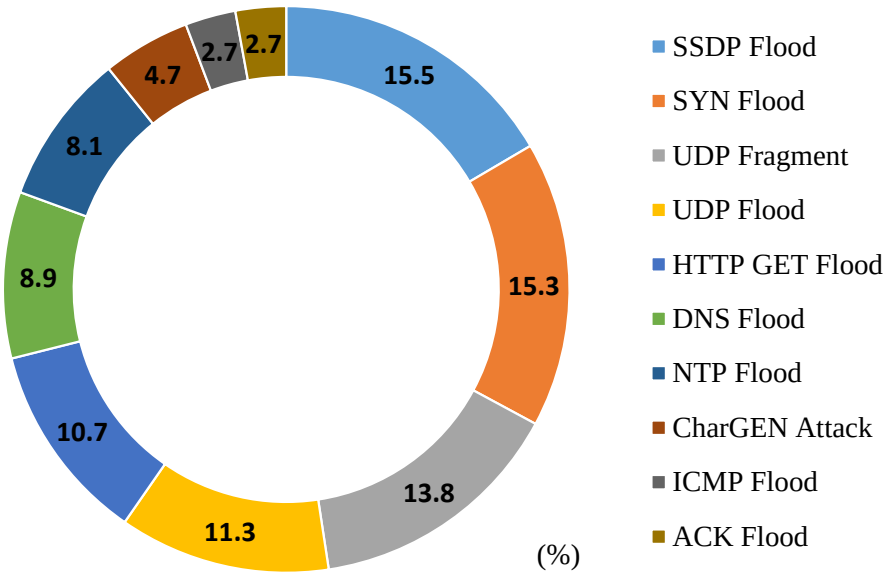
Stejně jako segment, má packet – přenosová jednotka síťové vrstvy – nejdříve informační hlavičku a pak data. Hlavička (header) obsahuje informace o packetu: zdrojovou a cílovou IP adresu, délku dat, kontrolní součet, určení protokolu další vrstvy a podobné, přesně popsáno tabulkou 2.3. Pro protokol IPv4 vypadá tak, jako na obrázku. Header má standardní velikost 160 bitů, dodatečné informace ho ale mohou zvětšit až 480 bitů. Délka celého packetu je standardně omezena na 65 535 bytů, i když se zavádí tzv. MTU (Maximum transmission unit) pro každý typ linky zvlášť.

<table>
<tr><td rowspan="2">byte</td><td colspan="2">0</td><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">3</td></tr>
<tr><td>bit</td><td>0 - 3</td><td>4 - 7</td><td>8 - 11</td><td>12 - 15</td><td>16 - 19</td><td>20 - 23</td><td>24 - 27</td><td>28 - 31</td></tr>
<tr><td>0</td><td>0</td><td colspan="2">verze IP</td><td colspan="2">délka hl.</td><td colspan="2">typ služby (DSCP)</td><td colspan="2">celková délka packetu</td></tr>
<tr><td>4</td><td>32</td><td colspan="4">identifikace</td><td>značky</td><td colspan="3">posun fragmentu</td></tr>
<tr><td>8</td><td>64</td><td colspan="2">TTL (Time to live)</td><td colspan="2">protokol</td><td colspan="4">kontrolní součet hlavičky</td></tr>
<tr><td>12</td><td>96</td><td colspan="8">zdrojová IP adresa</td></tr>
<tr><td>16</td><td>128</td><td colspan="8">cílová IP adresa</td></tr>
<tr><td>20
...</td><td>160
...</td><td colspan="8">další (volitelné) informace</td></tr>
</table>

Tab. 2.3: Obsah hlavičky IPv4 packetu

2.2 Metody útoků

Pod název DoS útoky, zahrnující i DDoS a DRDoS, patří mnoho různých typů provedení, lišících se zejména zneužitým protokolem, případně zneužitou vlastností daného protokolu. V této kapitole je tedy probráno deset nejčastějších metod DoS útočení, a to podle zastoupení na podzim roku 2015 podle dat společnosti Akamai, shromažďující tyto údaje, prezentovaných na webu State of the Internet[5]. Dle dostupných informací není třeba očekávat změnu těchto trendů ani v roce 2016.



Obr. 2.5: Typy útoků zachycených během října 2015

Typy vypsané v grafu zastupují zhruba 95% všech záplavových útoků, zbytek tvoří málo používané RESET Flood, FIN Flood, PUSH Flood nebo třeba Reserved Protocol Flood, kterým v práci nebude věnována pozornost.

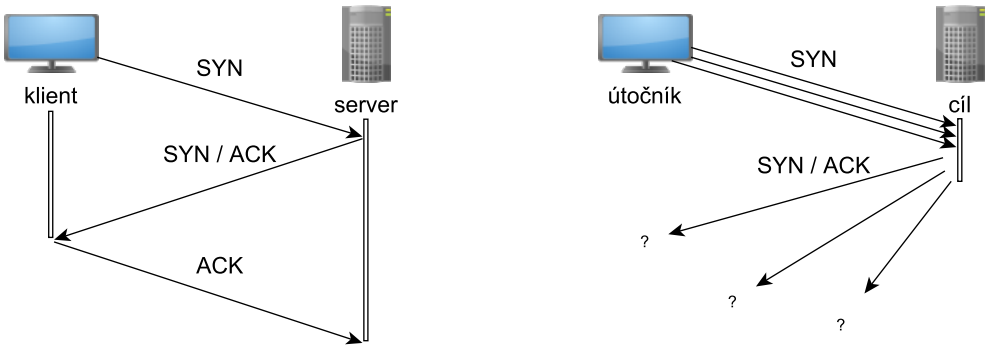
2.2.1 SSDP Flood

SSDP (Simple Service Discovery Protocol) je protokol pro zjištění připojení a snadnou inicializaci periferního zařízení připojitelného k počítači (UPnP, Universal Plug and Play). Velice často se používá pro tzv. zařízení Plug & Play – je tedy implementován a povolen v mnoha různých domácích a kancelářských nástrojích, jako jsou tiskárny a web kamery, a na serverech pro zjišťování síťových služeb. Funguje na základě zpráv o připojení a odpojení zařízení a na požadavcích o nalezení všech

dostupných připojených periferií (M-SEARCH). Pokud uživatel není chráněn řádným firewallem nebo access control listem, je pro útočníka snadné poslat do jeho sítě M-SEARCH dotazy se zpáteční adresou oběti. Pokud je v síti dostupných více UPnP zařízení, počet odpovědí vzroste a zamíří směrem k cíli útoku. Jestliže má agresor dostatečný seznam zneužitelných sítí, může snadno zahlcovat stanice bez vědomí majitelů zneužitých síťových prvků. Možnost zneužití SSDP k napadení není známa dlouho, nicméně je velice populární díky vysokému bytovému zesílení – po odražení v cizí síti míří na cíl až třiceti násobek původního provozu.

2.2.2 SYN Flood

Záplava SYN packety – SYN Flood – zneužívá vlastnosti protokolu TCP transportní vrstvy, a to třícestného podání ruky (three-way handshake). Tato synchronizace je potřebná pro bezchybné navázání plného TCP spojení. První stanice, která chce zahájit komunikaci, zašle druhé packet s nastaveným TCP SYN příznakem (pole „značky“ v tabulce 2.3). Pokud má dojít ke vzniku spojení, musí tato stanice odpovědět packetem se SYN/ACK a původní klient už jen potvrzuje s ACK. Takto mají oba konce synchronizována sekvenční čísla (jedno pro každý směr komunikace) a mohou začít výměnu dat. Zneužití tohoto procesu je možné díky tomu, že tázaná stanice, typicky server, čeká po obdržení žádosti určitou dobu na potvrzovací packet s ACK a po tento čas udržuje tzv. polootevřené spojení, kdy má předem alokovány zdroje pro očekávané plné spojení. Pokud útočník pomocí ACK spojení nepotvrdí, a naopak zašle další žádosti s SYN, může se server zahltit polootevřenými nevyřešenými záznamy a přestane odpovídat legitimním žádostem, na které nemá dostatek výpočetních zdrojů. Srovnání obou případů je vidět na obr. 2.6.



Obr. 2.6: Postup three-way handshake (vlevo) a jeho zneužití k útoku (vpravo)

V extrémním případě mohou tyto nepravé záznamy setrvávat v paměti serveru po neomezenou dobu, právě kvůli objevu možnosti SYN Flood jsou ale tyto časy vět-

šinou omezeny na úroveň stovek milisekund. Útočník navíc může modifikovat zdrojovou adresu v odesílaných SYN žádostech na adresu jiné oběti tak, že bude server odpovídat tomuto počítači, šetřit linku útočníka, zaplňovat spojení oběti a uchovávat pro něj větší anonymitu.

2.2.3 UDP Fragment

Fragmentace je proces rozdělení jedné datové jednotky na více menších. Provádí se, pokud je velikost přenášené buňky větší, než MTU pro daný protokol nižší vrstvy. Části se pak přenesou zvlášť a na druhé straně složí zpět dohromady podle identifikačních údajů v hlavičkách. Problém, využitelný k útoku, nastává při neočekávaném stavu u skládání fragmentů dohromady. Pokud jsou tyto záměrně špatně nastaveny nebo označeny, může dojít k situaci, kdy se podle pořadových čísel dva fragmenty částečně nebo úplně překrývají, kdy je výsledná složená jednotka větší, než je dovolený limit, nebo kdy ke kompletaci chybí některé části. Také se může zaplnit paměť cíle určená k dočasnému uchovávání fragmentů čekajících na sloučení. Systém cíle je tedy v situaci, k níž nezná řešení a často kolabuje.

2.2.4 UDP Flood

Při tomto typu záplavy útočník zasílá co největší množství packetů na adresu cíle s náhodně zvolenými cílovými porty, využívajíc bezstavového spojení při přenosu pomocí UDP. Oběť se pro každou takovou přijatou zprávu snaží zjistit, zda na daném portu poslouchá některá z lokálních aplikací. Jelikož jsou čísla portů náhodně generována, nejspíš zjistí, že ne a odpoví zprávou ICMP Destination unreachable. Záplava UDP packetů tedy tvoří záplavu ICMP packetů v opačném směru. Stejně jako u SYN Flood ovšem může útočník pozměnit zdrojovou adresu odesílaných dat a odklonit tyto odpovědi jinam.

2.2.5 HTTP GET Flood

Protokolem zneužitým k přetížení cíle je v tomto případě HTTP. To funguje na základě jednoduchých zpráv mezi klientem (typicky webový prohlížeč) a HTTP serverem. Nejpoužívanějšími příkazy jsou GET a POST. GET žádá o URL (Uniform resource locator) adresu odkazující na žádanou část dat, POST data odesílá. Při záplavě se útočník snaží zasypat server dostatečným počtem žádostí, aby vyčerpal jeho výpočetní nebo paměťové zdroje. Může navíc vložit odkaz na data na často navštěvované stránky a každý, kdo poté přejde na tento web, automatiky zašle GET požadavek na cílový server bez svého vědomí. Výsledkem je pak nedostupná stránka.

2.2.6 DNS Flood

DNS servery jsou důležitou částí internetu, starají se o překlad numerických IP adres do lidmi snadno zapamatovatelných doménových jmen. Pokud tato funkce lokálně nefunguje, jsou uživatelé, snažící se o přístup do internetu, neúspěšní. Cestou k úspěšnému zaplavení je překročení počtu DNS požadavků, které server zvládne obsloužit. Jde o princip podobný UDP Flood (DNS používá UDP jako transportní protokol), nicméně se odlišuje dotazováním na konkrétní službu.

2.2.7 NTP Flood

Protokol NTP (Network time protocol) se používá ke zjištění a synchronizaci hodin síťových zařízení, funguje již od roku 1985. Užitkem pro útočníka je příkaz „monlist“, který zadavateli vrátí IP adresy 600 posledních hostů připojených k danému NTP serveru. Při vyžádání velkého množství těchto seznamů s adresou oběti podvrženou na místo adresy zdroje v monlist zprávě zamíří na cíl mnohem větší objem dat, než útočník vyslal (žádost o seznam není tak velká, jako samotný seznam).

2.2.8 CharGEN Attack

Port 19 není často zmiňován a v některých přehledových tabulkách je dokonce vynechán. Je na něm dostupná zastaralá služba Character Generator Protocol, využívaná v minulém století pro testovací a ladící účely. Při použití TCP odpovídá nepřerušeným proudem náhodných znaků, trvajícím po dobu spojení, při UDP až 512 bytovými odpověďmi stejného obsahu. Je tedy velice snadno zneužitelná k odraženému útoku s vysokým faktorem znásobení, kdy krátký dotaz způsobí velkou odpověď. Ta se opět přesměrovává na adresu oběti.

2.2.9 ICMP Flood

Tento způsob záplavy je velice jednoduchý, zakládá se na ohromení oběti abnormálním počtem ICMP (Internet control message protocol) Echo request, tzv. pingů. V zásadě se dá použít jen na pomalejší cíle, které dokáže příval pingů ohromit a způsobit citelné zpomalení poskytované služby.

2.2.10 ACK Flood

ACK Flood, nebo také ACK Storm se od ostatních typů záplav liší svým startem. Při myšlené komunikaci mezi serverem a klientem, používající TCP, znají účastníci pro oba směry komunikace sekvenční čísla. Pokud při přijetí zprávy serverem číslo výrazně nesouhlasí, odešle se klientovi oznámení, že došlo k de-synchronizaci spolu

s očekávaným číslem. Packet s tímto varováním obsahuje sekvenční číslo z úhlu pohledu serveru, které je ovšem jiné, než očekává nesladěný klient. Klient tedy také varuje server o špatném čísle, nicméně zpráva, ve které tak činí, má jiné číslo, než server očekává. Tahle výměna varování zdánlivě nemá řešení a může trvat velice dlouho.

Pro zapříčinění popsané situace musí mít útočník přístup do sítě, ve které jsou oběti. Monitoruje provoz a jakmile zjistí TCP spojení stanice s obětním serverem, odešle jménem klienta serveru packet s RST příznakem, který říká, že chce ukončit spojení, a falešný SYN packet se změněnými sekvenčními čísly. Spojení serveru s klientem je přerušeno.

2.3 Existující možnosti tvorby záplav softwarem

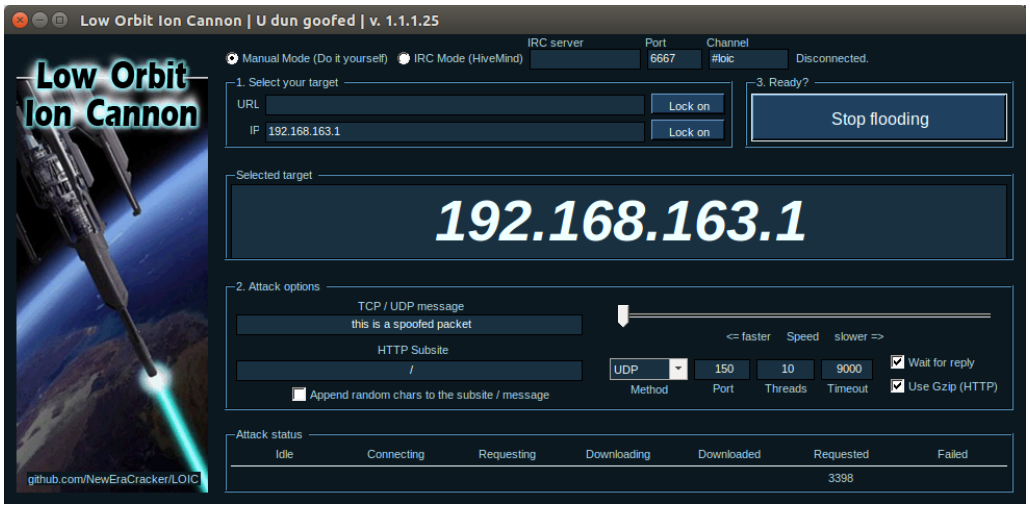
Bez nutnosti tvorby vlastního řešení lze pro vytvoření nebezpečného toku dat využít již existujících nástrojů. Tyto programy jsou většinou na internetu volně dostupné ke stažení a jsou hlavním důvodem nárůstu v četnosti útoků odepření služeb, protože výrazně zjednodušují práci s nimi. Těchto utilit existuje mnoho, zmíněny jsou pouze dvě nejznámější: hping a LOIC (Low Orbit Ion Cannon). Oba programy byly vyzkoušeny ve vlastní uzavřené testovací síti v operačním systému Ubuntu.

2.3.1 Low Orbit Ion Cannon

LOIC (Low Orbit Ion Cannon) je program s jednoduchým uživatelským prostředím, umožňující tvorbu UDP, TCP nebo HTTP DoS útoku jen se základními znalostmi fungování počítačových sítí, původně určený pro jejich testování. Existuje ve verzi pro většinu moderních operačních systémů, včetně mobilní platformy Android. Modifikovaná varianta LOIC byla použita i známou hackerskou skupinou Anonymous při útoku na finanční společnost Paypal při tzv. operaci Payback[6]. Prostředí programu je vidět na obr. 2.7.

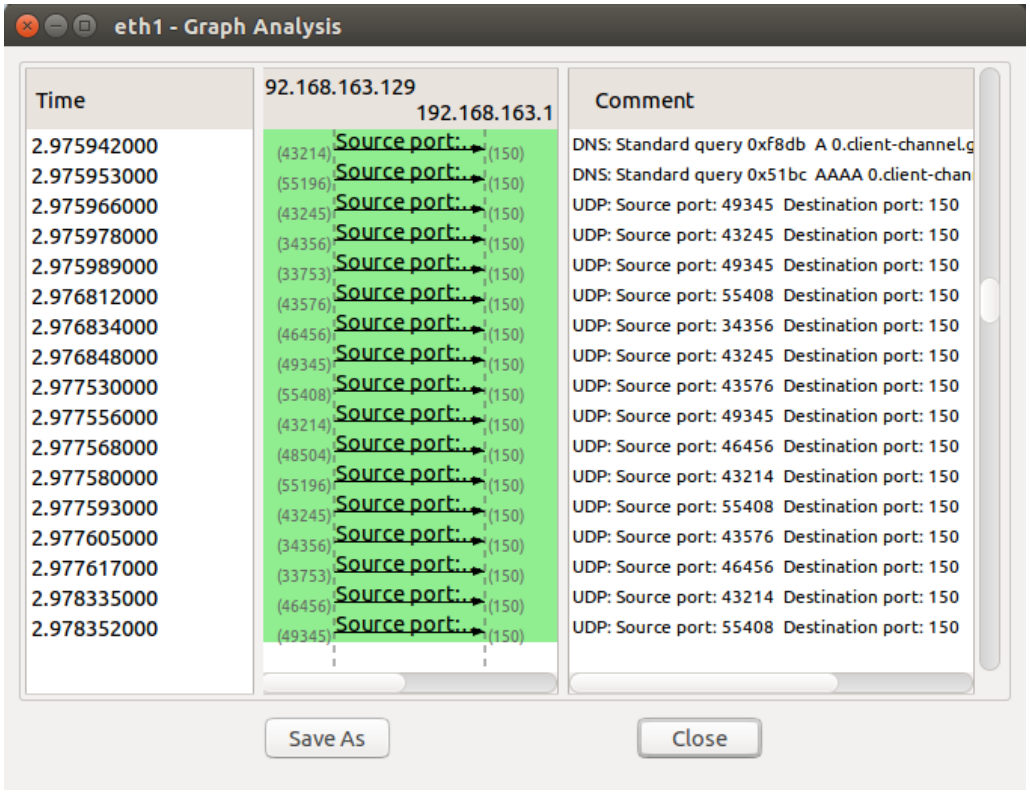
Pokud není známa IP adresa oběti, lze zadat URL a provede se pokus o překlad pomocí DNS. Dále je možné změnit typ útoku a zadat příslušné parametry, základní nastavení ale také funguje a uživatel tak nemusí znát tyto podrobnosti. Slabinou ve tvorbě případného napadení pomocí LOIC je absence krytí útočníka, program neprovádí žádný typ maskování a veškeré packety tak odchazí s původní adresou zdroje.

V případě provedeného testu šlo o lokální IP adresu vlastní síťové karty a útok typu UDP Flood (sekce 2.2.4). Na obr. 2.8 je vidět výpis provozu na napadeném zařízení, získaný pomocí programu Wireshark. V levém sloupečku je vidět čas přijetí zprávy, v pravém potom její podrobnosti. Šlo o více než 7000 packetů za



Obr. 2.7: Grafické prostředí programu LOIC

sekundu mířících z náhodně zvoleného portu (lze vidět měnící se čísla) na IP adrese 192.168.163.129 na port 150 adresy 192.168.163.1.



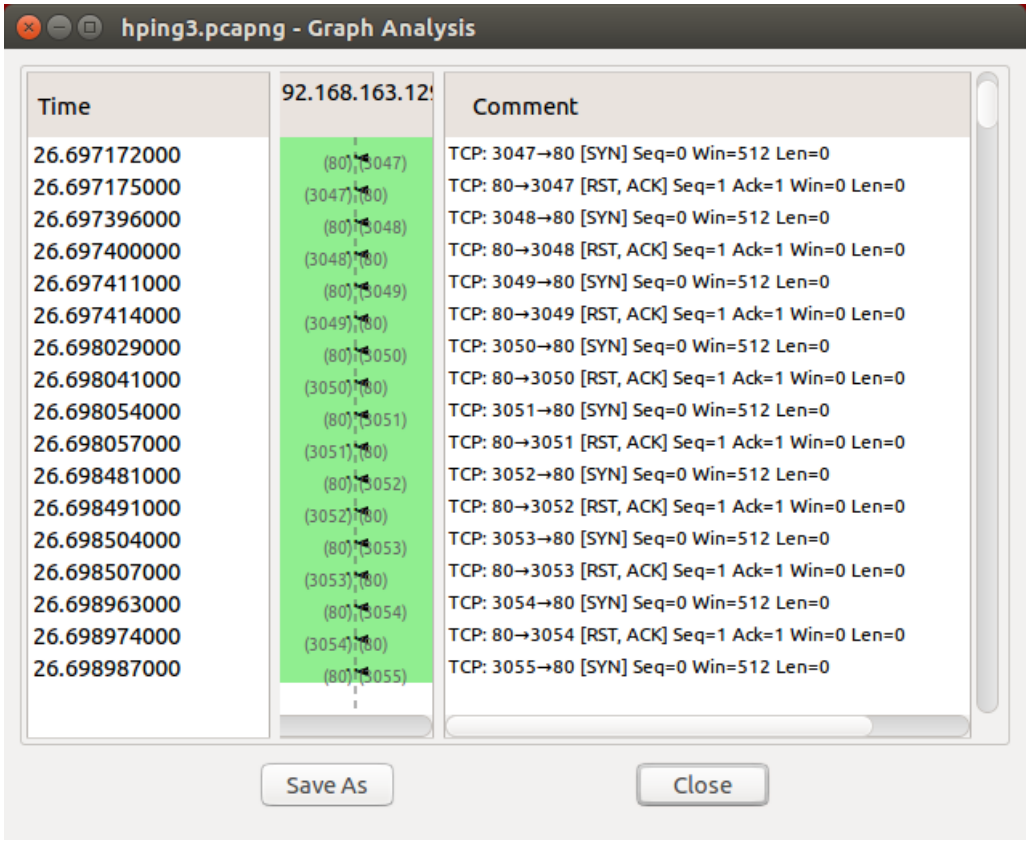
Obr. 2.8: Tok dat na zařízení napadeném UDP Flood pomocí LOIC

2.3.2 hping3

Program hping3 je dalším generátorem packetů, nemá však vlastní grafické prostředí. Spouští se z příkazové řádky podporovaných Unixových distribucí a poskytuje možnosti podobné těm u LOIC. Pomocí příznaků lze přepínat mezi verzemi IP, dostupnými typy útoků, kterými jsou SYN Flood (popsáno v 2.2.2) a ICMP Flood (2.2.9). V tomto případě byla použita varianta TCP SYN Flood. V Linuxovém terminálu spuštěno příkazem:

```
while 1 { hping send "ip(saddr = 192.0.0.13, daddr = 192.168.163.129)
            + tcp(sport = 1500, dport = 80, flags = s)},
```

který spouští záplavu TCP packetů s nastaveným SYN příznakem z adresy 192.0.0.13, port 1500 na cílovou adresu 192.168.163.129, port 80. Výhodou hping3 oproti LOIC je možnost upravení zdrojové adresy a tedy zamaskování identity útočníka. Obrázek 2.9 ukazuje pohyby na cílovém zařízení, výsledkem bylo více než 9 000 takovýchto packetů za sekundu.



Obr. 2.9: Tok dat na zařízení napadeném SYN Flood pomocí hping3

Dalšími programy podobného ražení jsou SlowIoris, Trinoo nebo Mausezahn, mající stejný účel, dosažený jinými prostředky.

3 SIMULACE ÚTOKU

Pro zjištění chování sítě, zažívající DoS útok, byla vytvořena simulace s odpovídajícím scénářem v prostředí významném pro daný účel – ns-3.

3.1 Network simulator 3

Patříc mezi známé prostředky pro simulaci sítě, ns je simulátor diskrétních událostí (viz odstavec níže) vytvořený a používaný pro výzkum a výuku. Jeho třetí verze je vyvíjena pro Unixové operační systémy jako open source projekt. Neposkytuje grafické prostředí, po ukončení simulace je ale možné výstupní data graficky zobrazit pomocí programů třetích stran. Nabízí velkou obratnost, plynoucí z implementace procesů na síti tak, jak se dějí ve skutečnosti, za cenu zdlouhavé křivky učení.

Discrete Event Simulation (DES) je typ počítačové simulace, ve které jsou operace modelovány jako oddělené (diskrétní) události v probíhajícím čase. Při běhu každý z těchto případů může a nemusí způsobit změnu stavu celého systému, je však dáno, že v jiném čase, než v čase události, k tomu nemůže dojít. Po skončení jedné diskrétní posloupnosti je tedy možné posunout se v čase až na začátek další. Dle srovnání[7] s jinými druhy simulací poskytuje DES neomezenou flexibilitu v určování chování počítačových modelů.

Síť v ns-3 se popisuje programovacím jazykem C++, což je objektově orientovaný jazyk, ve kterém figurují abstraktní prvky vytvářené reality. Základními objekty v síti ns-3 obecně jsou:

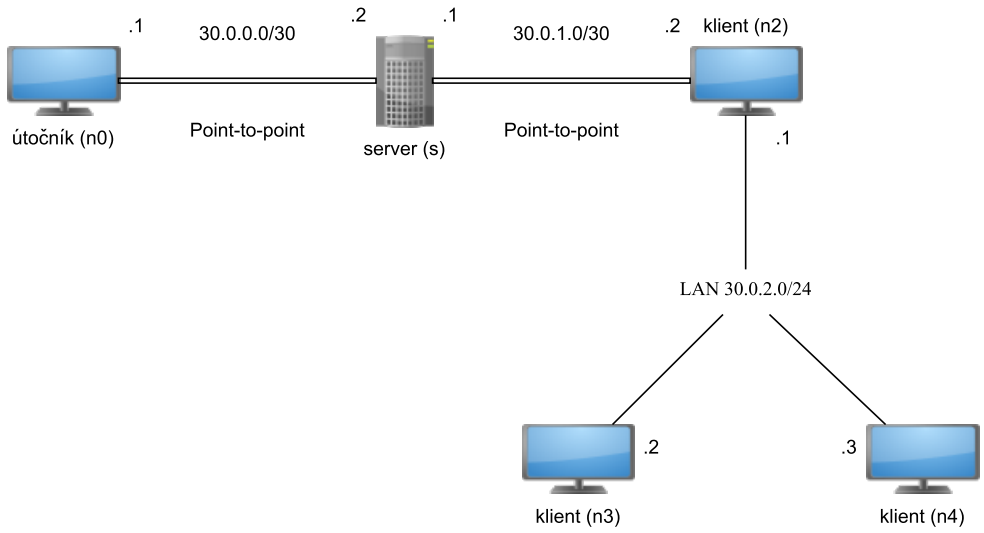
- uzel (node),
- aplikace (application),
- síťové zařízení (net device),
- kanál (channel).

Uzel představuje základní výpočetní jednotku. Je to zařízení, kterému lze přidávat funkce, jeho alternativou v síti internet by byla koncová stanice. Na uzel může být nainstalována aplikace, uživatelský program, která bude vytvářet aktivitu sledovanou v simulaci. V ns-3 se aplikace často vyskytují ve verzi pro server a pro klienta ke generování echo žádostí a odpovědí.

Aby byla uzlům s aplikacemi umožněna síťová komunikace, musí mít nainstalována síťová zařízení. Tato se chovají jako NIC (Network Interface Controller), tedy obdoba síťové karty klasických počítačů. Jakmile má uzel svoje síťové zařízení, může se připojit ke kanálu. Kanál je objekt představující jakýkoli spoj, v nejjednodušším případě tedy kabel. Od abstraktní třídy channel dědí jiné, konkrétnější typy spojení, např. CSMA, Wi-Fi nebo WiMAX kanály.

3.1.1 Simulace

Síť je naprogramována v Network simulatoru verze 3.24.1, pod operačním systémem Ubuntu 14.04 LTS. Použitý internetový protokol je pouze verze 4, na které probíhá většina DoS útoků. Vytvořená topologie odpovídá té na obrázku 3.1, kde je celkem pět uzlů – server `s` a čtyři koncové stanice, z toho jedna v roli útočníka (`n0`) a tři jako nevinní klienti (`n2`, `n3`, `n4`). Pro IPv4 existuje v ns-3 jistá forma automatického směrování podobná OSPF (Open Shortest Path First), která zajistí veškeré požadavky k vytvoření IP konektivity tím, že ze všech prvků vytvoří pseudo routery. Tohle výchozí řešení myšlenému účelu vyhovuje, v topologii tedy nejsou směrovače.



Obr. 3.1: Topologie vytvořené sítě

V hlavním souboru `dos.cc` jsou žádané uzly postupně vytvořeny pomocí:

```
Ptr<Node> n0 = CreateObject<Node> ();
Ptr<Node> s  = CreateObject<Node> ();
Ptr<Node> n2 = CreateObject<Node> ();
Ptr<Node> n3 = CreateObject<Node> ();
Ptr<Node> n4 = CreateObject<Node> ();
```

a následně sdruženy do kontejnerů po logických skupinách odpovídajících point-to-point mezi serverem a útočníkem, point-to-point mezi útočníkem a klientem `n2` a samostatné klientské LAN:

```
NodeContainer p2pAttackNodes (n0, s);
NodeContainer p2pUserNodes (s, n2);
NodeContainer csmaNodes (n2, n3, n4);
NodeContainer allNodes (n0, s, n2, n3, n4);
```

Dále jsou vytvořeny a nastaveny kanály, a to dva point-to-point a jeden CSMA představující LAN. Dvoubodová linka od serveru ke klientům má šířku pásma pouze 10 Mbitů, čehož později zneužije útočník `n0`.

```
PointToPointHelper p2pAttack;
p2pAttack.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
p2pAttack.SetChannelAttribute ("Delay", StringValue("2ms"));

PointToPointHelper p2pUser;
p2pUser.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
p2pUser.SetChannelAttribute("Delay", TimeValue (MilliSeconds (2)));

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (2000)));
```

K uzlům je potřeba připojit síťová zařízení, nainstalovat na ně tzv. TCP/IP stack a přiřadit IP adresy jejich rozhraním. Posledním příkazem v bloku je zprovoznění zmíněného globálního směrování:

```
NetDeviceContainer p2pAttackDevices;
p2pAttackDevices = p2pAttack.Install (p2pAttackNodes);

NetDeviceContainer p2pUserDevices;
p2pUserDevices = p2pUser.Install (p2pUserNodes);

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;
stack.Install (allNodes);

Ipv4AddressHelper address;

address.SetBase ("30.0.0.0", "255.255.255.252");
Ipv4InterfaceContainer p2pAttackInterfaces;
p2pAttackInterfaces = address.Assign (p2pAttackDevices);

address.SetBase ("30.0.1.0", "255.255.255.252");
Ipv4InterfaceContainer p2pUserInterfaces;
p2pUserInterfaces = address.Assign (p2pUserDevices);

address.SetBase ("30.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

Tím je vytvořena celá síť zobrazená v topologii na obr. 3.1. Pro účel simulace je třeba přidat aplikace, které na uzlech poběží. Je to aplikace `sUDP` nainstalovaná na serveru, běžící od času 1,5 s do 20,0 s a naslouchající na klientském portu, dále aplikace `cUDP` na třech klientech vysílající na server UDP echo zprávy (zprávy žádající o odpověď stejného rozsahu) o velikosti 1024 bytů pětkrát za sekundu v čase mezi 2. a 10. sekundou, a nakonec aplikace `aUDP` pro útočníka, která od 4. do 6. sekundy ruší legitimní komunikaci tisíci zprávami o velikosti `attackSize` za vteřinu (hodnota se bude měnit):

```
UdpEchoServerHelper sUDPhelper (clientPortUDP);

ApplicationContainer sUDP = sUDPhelper.Install(s);
sUDP.Start (Seconds (1.5));
sUDP.Stop (Seconds (20.0));

UdpEchoClientHelper cUDPhelper (s->GetObject<Ipv4>()->
  GetAddress(2, 0).GetLocal(), clientPortUDP);
cUDPhelper.SetAttribute ("MaxPackets", UintegerValue (1000));
cUDPhelper.SetAttribute ("Interval", TimeValue(MilliSeconds(200.0)));
cUDPhelper.SetAttribute ("PacketSize", UintegerValue (1024));

UdpEchoClientHelper aUDPhelper (s->GetObject<Ipv4>()->
  GetAddress(1, 0).GetLocal(), attackPortUDP);
ptrEcho = &aUDPhelper;

aUDPhelper.SetAttribute ("MaxPackets", UintegerValue(1000000));
aUDPhelper.SetAttribute ("Interval", TimeValue(MilliSeconds(1.0)));
aUDPhelper.SetAttribute ("PacketSize", UintegerValue(attackSize));

ApplicationContainer cUDP = cUDPhelper.Install (csmaNodes);
cUDP.Start (Seconds (2.0));
cUDP.Stop (Seconds (10.0));

ApplicationContainer aUDP = aUDPhelper.Install (n0);
aUDP.Start (Seconds (4.0));
aUDP.Stop (Seconds (6.0));
```

Na konci zdrojového kódu jsou sledovány Rx/Tx události uzlů odpovídající přijetí/odeslání packetu, zde v příkladu pro `n3`:

```
Config::Connect("/NodeList/3/$ns3::Ipv4L3Protocol/Tx",
  MakeCallback(n3_tx));
Config::Connect("/NodeList/3/$ns3::Ipv4L3Protocol/Rx",
  MakeCallback(n3_rx));
```

Spolu s výše uvedeným obsahuje hlavní soubor ještě příkazy pro vytvoření logů (záznamů), spuštění a zastavení samotné simulace a odkaz na tři další soubory,

které byly vytvořeny pro zjednodušení výpočtů, tvorby výstupních grafů a animace toku dat v síti.

Navíc bylo třeba upravit zdrojový kód ns-3 popisující enkapsulaci (sekce 2.1.5) na síťové vrstvě, obsaženou v souboru `/model/internet/ipv4-l3-protocol.cc` tak, aby byl zachycen jakýkoli packet odeslaný se zdrojovou adresou `30.0.0.1`, tedy adresou útočníka. Při detekci takového packetu je mu odebrána hlavička (sekce 2.1.7) a je nahrazena novou, s IP adresou zdroje nastavenou na `30.0.1.2` – adresu rozhraní klienta `n2`. Do souboru byl přidán následující kód:

```
Ipv4Header orig;
packet->PeekHeader(orig);
Ipv4Address addr = orig.GetSource();

if (addr.IsEqual (Ipv4Address ("30.0.0.1")))
{
    Ipv4Header spoofed;
    packet->RemoveHeader(spoofed);
    spoofed.SetSource(Ipv4Address ("30.0.1.2"));
    packet->AddHeader(spoofed);
}
```

na místo ve funkci `void Ipv4L3Protocol::SendRealOut`, od řádku 856.

Útokem v programu je modifikovaný UDP Flood (popsaný v sekci 2.2.4). Oproti známé verzi nemění útočník cílový port komunikace, protože by to v programu nemělo smysl, port již je nastaven na jiný, než server očekává. V souladu s DoS standardem ale dochází k podvržení zdrojové adresy útočníka na adresu klienta a odpovědi serveru tedy míří do klientské LAN, konkrétně na uzel `n2`.

3.1.2 Výsledky simulace

Výstup simulace je možný v podobě animace, trasovacích souborů a grafů. Animace je vytvořena pomocí knihovny `ns3/netanim-module.h`, uložena v xml souboru a může být přehrána v programu NetAnim (obr. 3.2). Trasovací soubory ve formátu pcap umí vytvořit samotný Network simulator a lze je prohlížet pomocí programu Wireshark. Jelikož jich pro každý uzel vznikne několik, je vhodné je sloučit do jednoho souboru zavoláním

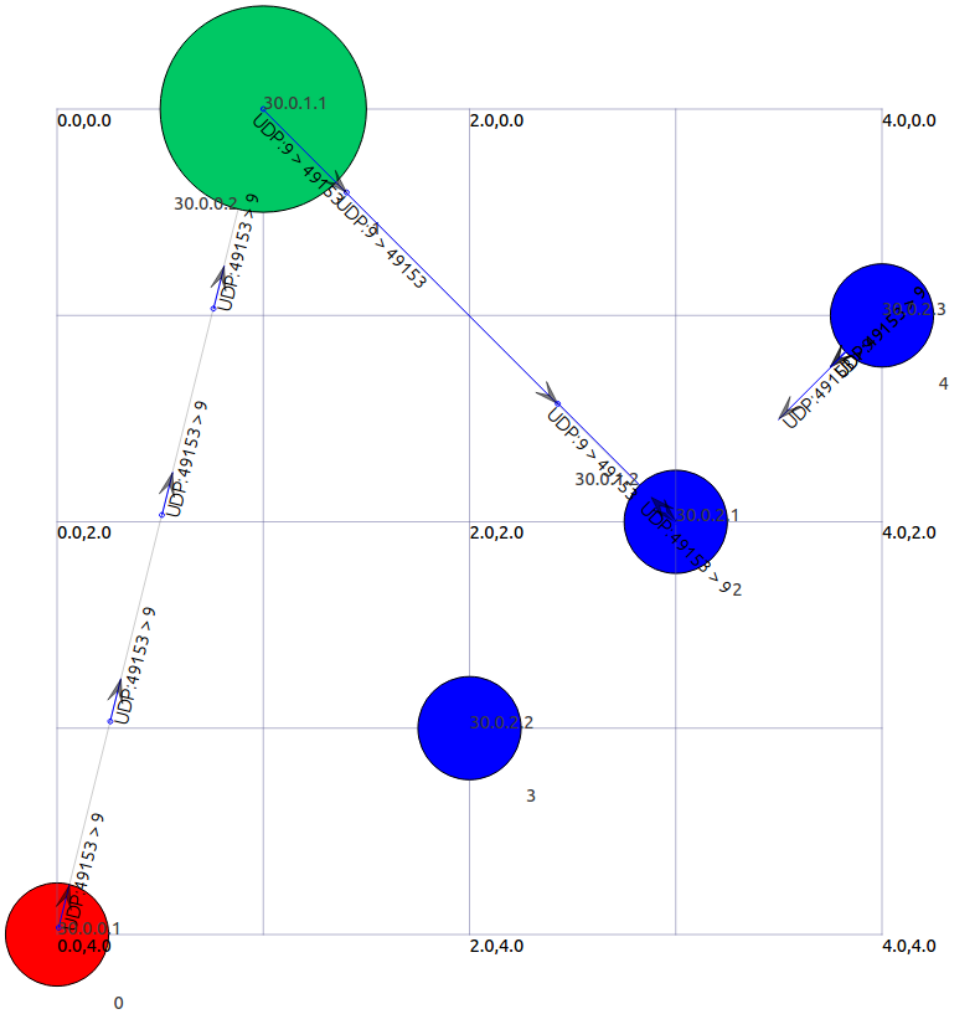
```
mergecap -T ether -w output.pcap *.pcap
```

v linuxovém terminálu. Data pro grafy zobrazují vývoj RTT (Round Trip Time) v čase, jsou výstupem knihovny `ns3/gnuplot.h` a do grafických závislostí jsou vizualizována pomocí příkazu

```
gnuplot *.plt,
```

který převede hodnoty RTT všech uzlů ze souboru plt do grafů v obrázku formátu png. Round Trip Time je čas mezi odesláním packetu s žádostí o odpověď a přijetím odpovědi na něj, jde o dobrý ukazatel přetížení sítě.

Na obrázku z programu NetAnim (obr. 3.2) je vidět vytvořená síť v čase desítek milisekund po čtvrté sekundě. Tedy v čase, kdy již útočník (znázorněn červeně) započal vysílání podvrhnutých zpráv na server (v obrázku zeleně) a ten, v domnění, že jde o žádost od klienta, posílá odpověď směrem do klienstké LAN (modré barvy). Tím dochází k simulovanému zahlcení linky a v praxi může nastat i vyčerpání výpočetních prostředků samotného serveru. Zaroveň klient na obrázku vpravo odesílá přes prostředního klienta serveru svou skutečnou periodickou žádost.

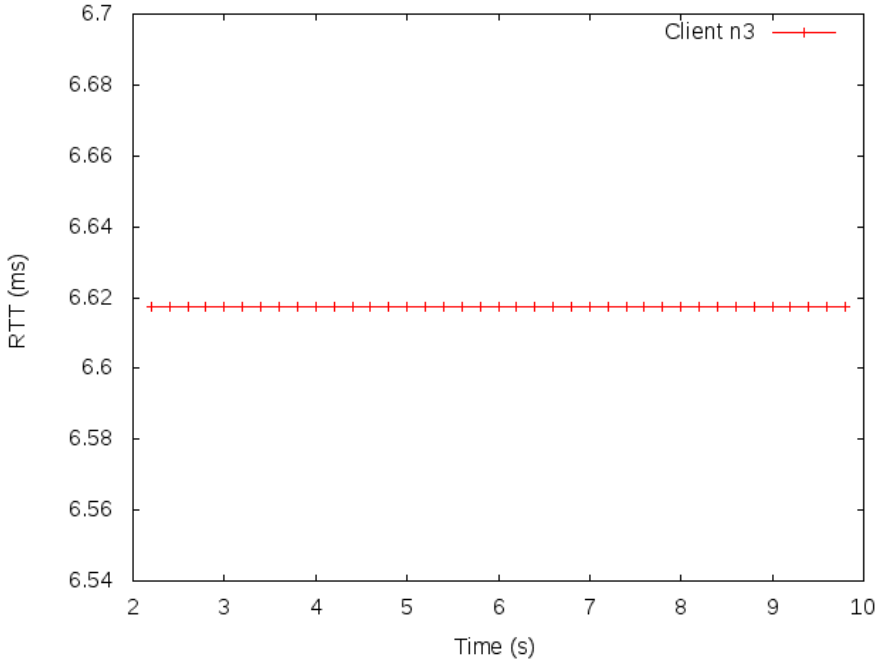


Obr. 3.2: Animace provozu vytvořené sítě programem NetAnim

Pro prezentování výsledků bude jistě nejlepší zmíněný graf. I přes to, že program nashromáždí data pro všechny uzly, mají vypovídací hodnotu pouze ty z klientů

`n3` a `n4`. Je to způsobeno tím že se hodnoty RTT počítají jako rozdíl v čase mezi odesláním zprávy serveru a přijetím odpovědi na ni. Všechen provoz odeslaný útočníkem má ale zaměněnu zdrojovou adresu, tedy na jeho původní adresu v průběhu celé simulace žádná data nepřijdou. Veškerý pozměněný tok dat od `n0` prochází přes server a tím zkresluje i jeho statistiky a míří na klienta `n2`, který naopak dostává spoustu zpráv, které si nevyžádal, tedy ani graf z jeho RTT nebude správný. A jelikož je na poli Rx/Tx událostí (přijetí, odeslání) rušivým elementem pouze útok, který nadmíru vyvolává tyto události jen na útočníkovi, serveru a prvním klientovi, je možné dopad na síť sledovat „z povzdálí“, na klientech `n3` a `n4`.

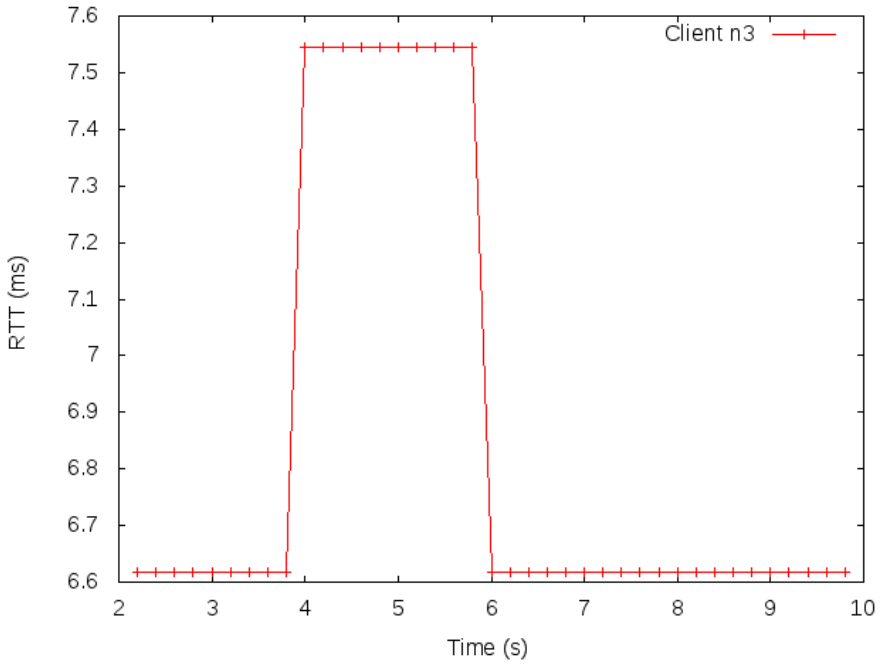
Stav sítě v situaci, kdy útočník neposílá žádná data, je vidět na obrázku 3.3. Síť funguje tak, jak byla navržena, útočník po celou dobu neposílá data a v žádném



Obr. 3.3: Vývoj RTT klienta `n3` v čase, síť v bezpečí

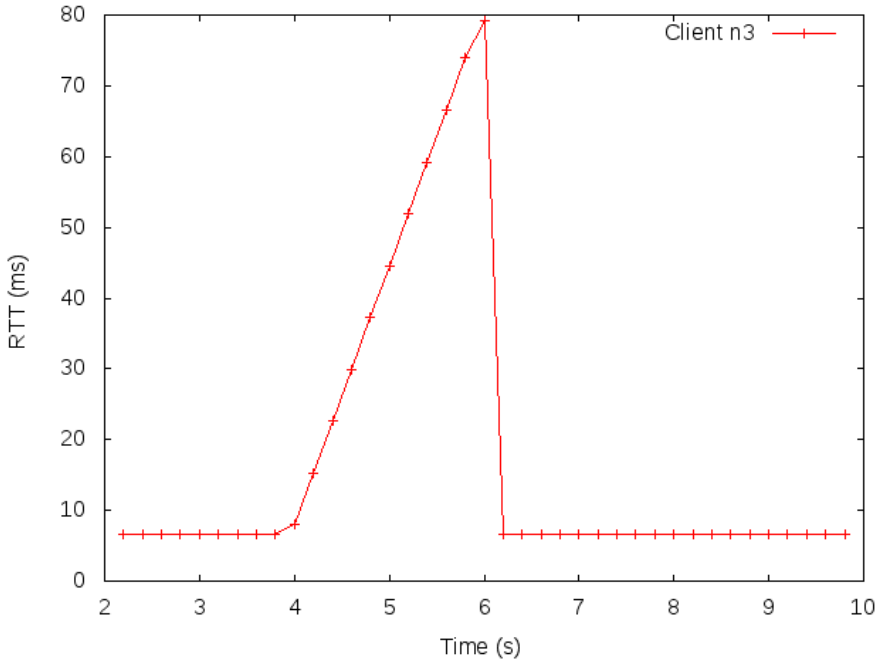
místě nedochází k přetížení. RTT packetů klienta je přes celou simulaci konstantní.

Při nastavení hodnoty `attackSize` na 1024 bytů za sekundu je z grafu na obr. 3.4 vidět, že 10 Mbitový point-to-point spoj mezi serverem a klienty začíná působit jako škrtidlo provozu. Šířka pásma kritického kanálu je 10 Mb/s, neboli 1 250 kB/s. Odpovědi serveru, vyvolané agresí útočníka, mají velikost 1,024 kB a frekvenci 1000 Hz, jejich datový tok je tedy 1 204 kB/s. Pro provoz UDP mezi klienty a serverem zbývá jen 46 kB/s, což se projeví prodloužením čekání na odezvu zhruba o jednu milisekundu. Fakt, že je tohle způsobeno útokem trvajícím od 4. do 6. sekundy, je v grafu jasně vidět.



Obr. 3.4: Vývoj RTT klienta `n3` v čase, síť pod útokem

Nakonec je velikost zpráv útočníka nastavena na 1 250 bytů, tedy hodnotu, která při tisíci jednotkách za vteřinu přesně vyplní šířku pásma cílené linky. Průběh odezvy na obr. 3.5.



Obr. 3.5: Vývoj RTT klienta `n3` v čase, síť zahlcena

Oproti případu s 1 024 byty, kdy se RTT útočníkovi podařilo zvýšit o necelou milisekundu, zde jde již skoro o stonásobek. Takové zpoždění už může být na koncové

stanici uživatelem pociťováno, útok na fungování služby se tedy dá považovat za úspěšný.

3.2 Riverbed Modeler

Tento komerční síťový simulátor je založen na původním produktu společnosti OPNET a používá diskrétní časové události (popsáno v kap. 3.1 o Network simulatoru), stejně jako ns-3. Oproti němu ale nabízí plné grafické rozhraní, umožňující tvorbu topologie sítě i nastavení simulace intuitivnějším způsobem bez nutné znalosti jakéhokoli programovacího jazyka. Jelikož se ale jedná o proprietární closed-source program, který navíc neposkytuje žádnou formu dokumentace ke zdrojovým souborům, jsou možnosti tvorby omezeny na schopnosti aplikace a obzory zúženy záměry vývojářů.

Není tedy možné zasáhnout do procesu zapouzdřování dat (kap. 2.1.5) a vytvořit tak plnohodnotné odepření služby stejně, jako to dovolil ns-3. Simulace v tomto programu tedy nebyla vyrobena pro nemožnost dosažení uspokojivých výsledků dostupnými prostředky.

4 TEORIE K TVORBĚ ÚTOKU

Tato kapitola popisuje teoretické skutečnosti, přímo se týkající úkolu v praktické části práce.

4.1 FPGA

Field-programmable Gate Array (Programovatelné hradlové pole) je technologie digitálních integrovaných obvodů s uživatelsky upravitelnými vlastnostmi (*Configurable Logic Block, CLB*) spojenými konfigurovatelnou strukturou (*Input/Output Block, IOB*). Na bázi jednotlivých vstupních a výstupních pinů je pomocí logických hradel možné určit vztah mezi nimi a za použití všeho, co daný programovací jazyk dovolí, řešit libovolné vypočitatelné problémy. To je umožněno přítomností vlastního mikroprocesoru na desce. Osazené obvody jsou tedy rekonfigurovatelné a na rozdíl od logických hradel (například AND, OR) nemají fixní funkci. Při výrobě jim není dán přesně definovaný účel a před použitím musí být uživatelem naprogramovány, tedy překonfigurovány (*programovatelné*). Tento proces je zařízení schopno provést v neomezeném počtu opakování.

Typické případy použití FPGA zahrnují zpracování digitálního signálu, rozpoznání řeči nebo výpočet kryptografických entit v řešeních, která nejsou dostatečně rozsáhlá pro návrh vlastního, nového a často drahého hardwarového integrovaného obvodu, a zároveň jsou příliš složitá pro použití konvenčních softwarových systémů. Pomocí této technologie je nicméně možné vytvořit vysoce funkční hardwarové řešení, uchovávající si výbornou možnost konfigurovatelnosti, zatímco zůstává nízkonákladové a efektivní. Další významnou výhodou je velmi vysoký stupeň paralelizace – správně naprogramovaná vývojová deska dokáže vykonávat mnoho různých příkazů v jednom časovém okamžiku, mnohem více, než zvládnou obecné vícejádrové desktop procesorové jednotky.

Významným výrobcem FPGA desek spolu se softwarovou podporou na pozadí je společnost Xilinx.

4.2 VHDL

Významem VHSIC Hardware Description Language, kde VHSIC značí Very High Speed Integrated Circuit, VHDL je programovací jazyk určený pro popis daného hardwaru. Umožňuje vytvářet a simulovat chování reálných zařízení, včetně programovatelných polí a rozdělením své základní struktury na dvě části – entitu a architekturu – vytváří možnost velmi efektivního znovupoužívání kódu. Systémová entita

představuje skutečné rozhraní designové jednotky, která může samostatně implementovat více architektur. Jde o pohled na entitu ze zbytku systému, definovány jsou pouze základní náležitosti, jako vstupy a výstupy. Naopak architektura určuje, jak se budou konkrétní události zpracovávat a udává tak funkci entity. Zatímco architektura popisuje činnost dílu jako celku, entita zajišťuje přenosové operace. Díky tomuto je možné vytvořit jeden pracovní oddíl s více funkcemi, využívajícími stejné entity.

Klíčovou roli v celém programu hraje tzv. *top level modul*. Je to VHDL soubor se zdrojovým kódem podobný ostatním, jeho pozice je však na vrcholu pomyslné pyramidy hierarchie projektu. Má tedy přístup ke všem dalším souborům daného řešení a určuje jejich vzájemné vztahy. V jednom projektu může být těchto modulů více a jejich střídání je běžným úkonem pro dosažení různých výstupů stejného celku. Zároveň je také určujícím pro taktování všech zdrojových souborů – definuje *hodinovou funkci*, čtvercový signál o přesné frekvenci. Jednotlivé události běhu programu se pak váží na tento signál a jsou s ním synchronní, rychleji než s touto frekvencí pracovat nemohou a mezi dvěmi sousedícími změnami těchto „hodin“ k žádné změně (ani např. výpočtu) v obvodu nedochází. Zde je třeba pamatovat na paralelizaci, tedy že mnoho odlišných „procesů“ s tímto kmitočtem může běžet současně.

4.2.1 Životní cyklus kódu VHDL

Mezi dokončením prvotního návrhu FPGA-VHDL aplikace a jejím úspěšném spuštění na desce stojí kroky *kompilace*, *simulace*, *syntézy*, *implementace* a *generování bitového toku*.

Kompilace zdrojového kódu je obecně překlad algoritmů ve vyšším programovacím jazyce a jejich převod do jazyka nižšího. Pro VHDL platí pouze překladová část, zdrojový kód je zkontrolován pro syntaktické chyby.

Simulace ověří korektnost navrženého modelu a nabídne uživateli možnost zkontrolovat jeho funkčnost a sémantiku kódu prostřednictvím přehledu změn stavů vstupů a výstupů v čase. Pro to je potřeba mít již vytvořený tzv. *testbench* – top level modul určený k otestování dané části funkce projektu. Tento soubor musí obsahovat přinejmenším alespoň deklaraci signálů a vstupně/výstupních pinů a určení kmitočtu hodinové funkce. Ukázka malé části jednoho z testbenchů prostředí NetCOPE (kapitola 4.3) je zobrazena zde:

```
WriteData( mi_address,
           mi_data,
           mi_be,
           status(0),
           0);
      wait for 5*clkper;

ReadData( mi_address,
          mi_data,
          mi_be,
          status(0),
          0);
      wait for 5*clkper;
```

Jakkoli se podstata příkazů v tomto úryvku může zdát být bezpředmětná, při použití v testbenchi má smysl. První polovina zapíše předem definovaná data `mi_data` na adresu určenou v `mi_address` za použití příznaku `mi_be`, hned poté druhá polovina (od `ReadData`) ze stejné adresy tato data vyčte. Díky tomuto je možné zapisované a přečtené hodnoty porovnat a tím ověřit funkčnost zápisu a čtení. Tomuto překážkou je skutečnost, že stanice, používaná pro simulaci (typicky osobní počítač), vykonává procesy sekvenčně a je schopna jen malého počtu současně běžících úkolů. Aby bylo možno vyhodnotit vysoce paralelizovaný VHDL program, simulační software vytvoří *event list* (seznam událostí) tak, aby je mohl vykonávat po řadě, ale přesto byl výsledek uspokojivě podobný reálnému použití.

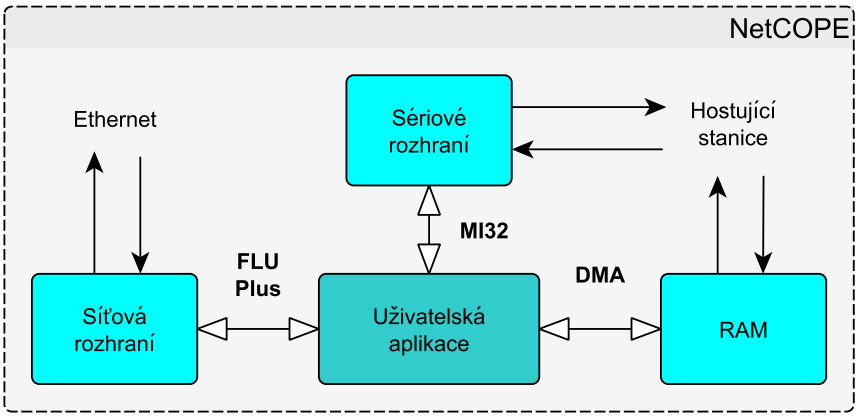
Syntéza představuje proces přeložení popisu hardwarového zařízení na vyšší úrovni abstrakce na optimalizované provedení v nižší vrstvě abstrakce. Dříve se jednalo o ruční práci, nyní je zautomatizována. Podle algoritmů ve VHDL jsou určeny potřebné spoje (IOB) mezi sestavenými výpočetními bloky (CLB) požadované FGPA karty tak, aby výsledná supersíť těchto jednotek plnila kýžený účel. Tím vznikne *netlist* – vypočítaná mapa vnitřní struktury. Existuje mnoho různých variant syntetizačního softwaru, jejichž výsledky (netlisty) mohou být svým provedením velmi odlišné, měly by se však vždy dostat do stejného cíle. U všech verzi se jedná o náročnou a zdlouhavou operaci.

Implementace používá jako svůj vstup syntetizovaný netlist. V krocích jej optimalizuje pro jednodušší nasazení, případně i pro snížení spotřeby výsledného řešení, tuto novější verzi designu umístí na cílové zařízení, kde se může znovu pokusit o optimalizaci již fyzických spojů a nakonec v návrhu vytvoří logické trasy.

Generování bitového toku zakončuje úspěšný proces programování hardwaru tím, že vytvoří sadu příkazů pro konfiguraci procesoru karty, již lze již do zařízení nahrát.

4.3 NetCOPE

Prostředí NetCOPE poskytuje nadstavbu nad uživatelskou FPGA aplikací a výrazně urychluje a zjednodušuje vývoj síťových programů na této platformě. Zavádí další vrstvu abstrakce mezi projekt, vytvořený vývojářem, a hardware používané desky s jejími rozhraními. Tato vrstva se stará o komunikaci mimo vlastní program – prostřednictvím protokolů popsaných v kapitole 4.3.1 zajišťuje aplikaci spojení s jejím okolím tak, jak je znázorněno na obrázku 4.1.



Obr. 4.1: Zjednodušená stavba prostředí NetCOPE

Tato rozhraní jsou navíc obecná, díky čemuž poskytují jednoduché připojení, nezávislé na použitém hardwaru.

Pro integraci do projektu je NetCOPE přizpůsoben. Ve svém vlastním top modulu nechává místo pro top level aplikaci uživatele, již je třeba do tohoto prostoru přesunout a spojit tím funkci tvořeného programu s funkcemi prostředí.

4.3.1 Komunikační prostředky NetCOPEu

Firmware rozeznává pro přenos konfigurace a dat následující komunikační rozhraní.

FrameLink je dvoubodovým synchronním protokolem, adaptací původního LocalLinku firmy Xilinx. Pro svou činnost definuje sadu signálů, jež mu umožňují přenášet packet rozdělený na části. Je schopen přenést libovolná data, přesto byl v dalších verzích prostředí nahrazen FLU (FrameLinkUnaligned).

Jeho devět signálů tvoří vlastní hodinová funkce (CLK) pro synchronizaci, signály určující začátek a konec celého rámce (SOF_N, EOF_N), začátek a konec konkrétní části rámce (SOP_N, EOP_N), znamení, že jsou zdroj a/nebo cíl připraveni vysílat, případně přijímat (SRC_RDY_N, DST_RDY_N), kanál pro samotná data (DATA) a signál pro určení velikosti platných dat v poslední části přenášeného packetu, která nemusí být vždy plně využita (DREM). Ty ze signálů, které v názvu nesou „_N“, jsou považovány za aktivní ve stavu logické nuly – uplatňují negativní logiku. Přenos dat probíhá, pouze pokud jsou zdroj i cíl zároveň indikováni jako připravení, v opačném případě je třeba vyčkat.

FrameLinkUnaligned (Plus) představuje novou verzi FrameLinku, oproti předchůdci by měl být efektivnější. Nepodporuje fragmentaci packetů na části, používá jinou soustavu signalizačních kanálů a je veden v pozitivní logice (tedy logická jednička značí aktivní stav).

Svou pozici si zachovaly signály CLK a DATA, pohotovost zdroje a cíle indikují SRC_RDY, DST_RDY. Stejně tak i SOP a EOP jsou přítomny, neoznačují ale začátek a konec části (part), nýbrž hranice packetu. Fragmentace dat musí být zajištěna jinak, je-li potřeba. Přibyly naopak kanály pro SOP_POS a EOP_POS. Tyto nejsou pouze signalizační, přenášejí údaj o poloze začátku nebo konce packetu ve slově datového proudu, jde tedy o adresu. Tento systém, ač efektivněji využívajíc dostupných prostředků, není zpětně kompatibilní s protokolem FrameLink.

Verze FLU Plus zavádí do pravidel další signál CHANNEL, který slouží ke snažší identifikaci přenášených dat mezi komunikujícími stranami firmwaru. Měnit jeho hodnotu může pouze odesílatel. FLU Plus je kompatibilní s FLU.

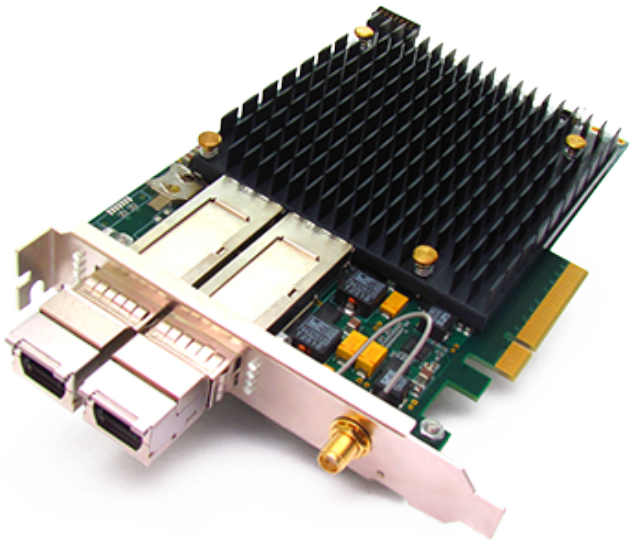
Protokoly rodiny FrameLink jsou NetCOPEem využívány ke komunikaci se síťovými rozhraními (Ethernet), a to přostřednictvím mezilehlého síťového modulu.

DMA popisuje techniku přímého přístupu periferně připojených zařízení k RAM paměti (Direct Memory Access) hostitelské stanice (počítač, vývojový server). Tím je výrazně sníženo zatížení vzdáleného CPU a přenos je rychlejší, s menším zpožděním. Prostředí NetCOPE tuto funkci implementuje.

MI32 sběrnice (Memory Interface 32-bits) je pro svou nižší propustnost používána pouze k přenosu konfiguračních dat z hostitelské stanice do vývojové karty. Používá sadu signálů podobnou té FrameLinku, zahrnuje indikaci připravených dat (DRDY) a připraveného čtení (ARDY), žádost o zápis, čtení (WR, RD), příznak byte-enable (BE), data k zápisu (DWR) a ke čtení (DRD) a obojetnou adresu, sloužící k předání žádané lokace příslušné operace (ADDR).

4.4 COMBO-80G Karta

Pro praktickou realizaci záplavového zařízení je k dispozici FPGA karta COMBO-80G (na obr. 4.2). Je to dvouportová síťová karta, využívající hardwarové akcelerace, s podporou 10G a 40G Ethernetu na každém rozhraní. Pro komunikaci s hostitelskou stanicí slouží rozhraní PCI-e. Podrobnější specifikace je uvedena na stránkách mateřského projektu Liberouter[8].



Obr. 4.2: Karta COMBO-80G[9]

5 TVORBA ÚTOKU

Generování datového provozu je možné dvěma způsoby, a sice:

- tvorbou nových, vlastních packetů, které mohou být uzpůsobeny co největšímu výslednému objemu nebo chování reálné sítě, tedy reflektovat určité fluktuace svých parametrů, nebo
- zachytáváním skutečného provozu na síti, jeho uložením a pozdějším přeposláním.

Pro použití v cíleném řešení byly nastudovány dva projekty, každý z nich volí jiný přístup k problému. Jsou to:

- *Flexible, extensible, open-source and affordable FPGA-based traffic generator*[10, 11], na který bude dále odkazováno pomocí zkratky FEOSA,
- *VHDL precise packet generator and RTT calculator*[12], jehož název bude zkracován na PPG.

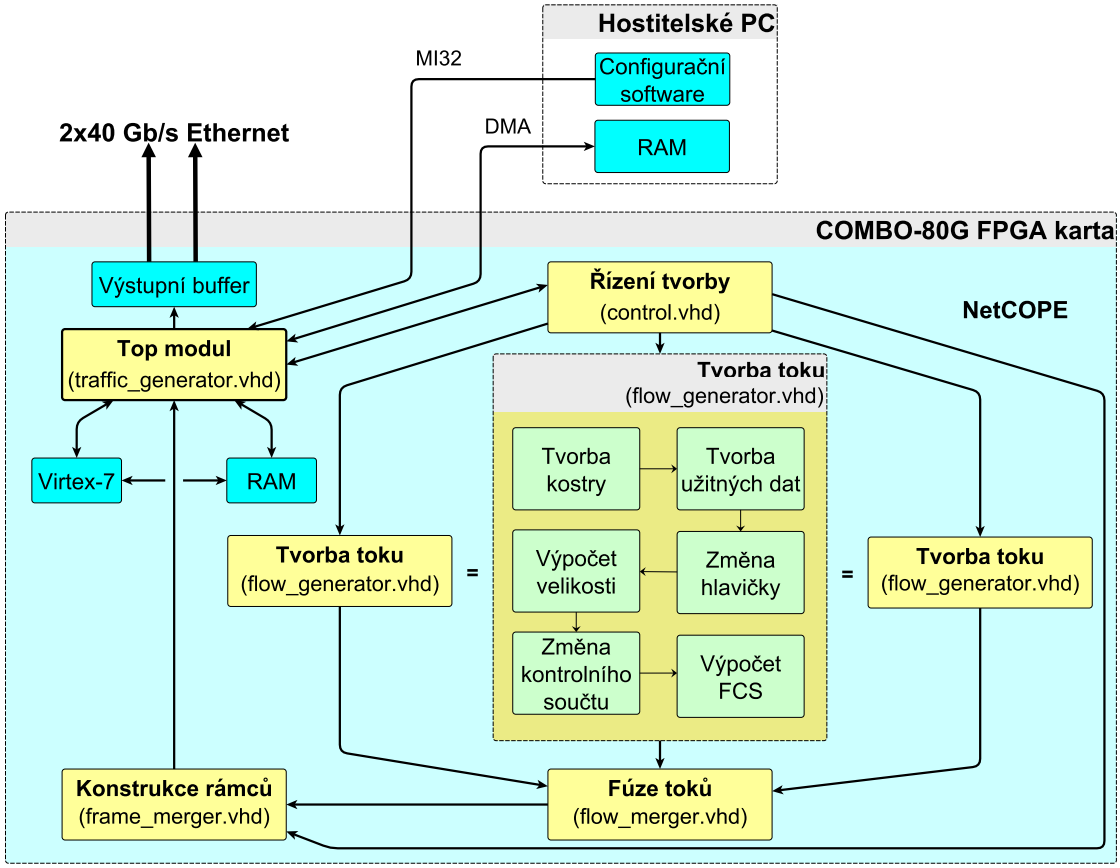
Jejich implementace je popsána v následujících kapitolách.

5.1 Řešení I

Řešení I využívá projektu FEOSA. Tento program byl vytvořen v letech 2011–2013 šesti francouzskými vývojáři coby síťový tester, poté uvolněn jako open-source. Zaměřuje se na výbornou konfigurabilitu, schopnost vyplnit 10 Gbitovou linku i malými packety a vysoký stupeň modularity, která teoreticky umožňuje externí použití, adaptaci a rozšíření pomocí zásuvných modulů. Program využívá služeb prostředí NetCOPE, v obrázku 4.1 (kap. NetCOPE) zastává funkci uživatelské aplikace, vnořené do okolní abstrakční vrstvy firmwaru.

Způsob implementace a použití souborů projektu jsou zobrazeny na obrázku 5.1. Pro maximalizaci výkonu je umožněno využití více procesů generování dat najednou tím, že je úkon rozdělen více generátorům toku, jejichž výstupy jsou později spojeny. Top modul funguje jako vstupní brána pro konfiguraci, přijatou z hostující stanice, a jako výstupní bod pro vytvořená záplavová data. Zároveň operuje s fyzickými komponentami použité FPGA desky. Přijatou konfiguraci odesílá do části pro řízení tvorby, jež pokyny distribuuje požadovanému počtu jednotek pro tvorbu toku. Tyto samy (a všechny zaráz) začnou vyrábět data, odesílaná jako tok přes dvojici programů *flow_merger* a *frame_merger*, které jednotlivé toky spojí do jednoho proudu a vytvoří výsledné rámce, legitimně označené a pojištěné správným kontrolním součtem.

Samotný projekt obsahuje sadu testbenchů, určených pro zjištění funkčnosti některých jeho částí. Pro otestování úkolu generování dat byl vytvořen nový, samo-



Obr. 5.1: Způsob implementace FEOSA generátoru

statný testbench, zaměřený na proces generování dat. Pracuje tak, že nahradí vstup programu pro generování toku (*flow_generator.vhd*), tedy simuluje výstup řízení tvorby (*control.vhd*), a zároveň z generátoru odebírá jeho výstupní hodnoty. Do testované jednotky odešle simulovanou vstupní konfiguraci a svoje výstupy ztotožní s jejími. Protože jde o sběrnici s protokolem FrameLink (popsán v kapitole 4.3.1), jsou tyto hodnoty následující:

```
signal RX_DST_RDY_N : std_logic;
signal TX_DATA      : std_logic_vector(63 downto 0);
signal TX_REM       : std_logic_vector(2 downto 0);
signal TX_SOF_N     : std_logic;
signal TX_EOF_N     : std_logic;
signal TX_SOP_N     : std_logic;
signal TX_EOP_N     : std_logic;
signal TX_SRC_RDY_N : std_logic;
```

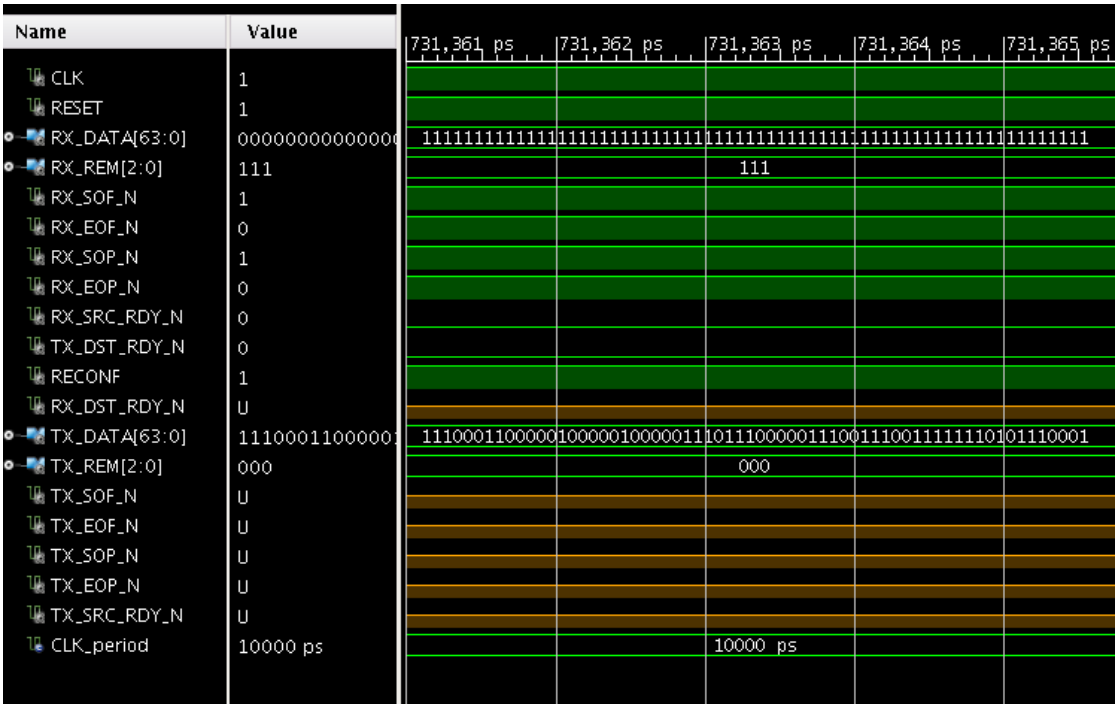
Z pohledu generátoru toku jde tedy v každém hodinovém cyklu o signál indikující schopnost cokoli přijímat (RX_DST_RDY_N), výstupní proud dat – 64 bitové slovo v TX_DATA a zbylé režijní signály NetCOPEu. Dále je vytvořena instance

testované jednotky a začíná proces odesílání konfigurace. Ta byla vytvořena podle vzorové jako binární proud, zahrnující všechny informace, které generátor toku potřebuje ke své základní činnosti – údaje do hlaviček IP a Ethernetu, omezení velikosti požadovaných dat, jejich objem, rozestup, počet a další. Vytvořená konfigurace jednoho generátoru toku obsahuje 176 bytů digitální informace, rozdělených do 22 řádků po 8 bytech. Příklad odeslání jednoho řádku v průběhu hodinového cyklu, konkrétně vymezení oblasti pro kontrolní součet síťové vrstvy (IP checksum):

```
RX_SOF_N <= '1';
RX_EOF_N <= '1';
RX_SOP_N <= '1';
RX_EOP_N <= '1';
RX_DATA  <= "00001100000000000000000000000000
             00000011000000011100000010000100";
wait for CLK_period;
```

5.1.1 Zjištěný maximální výkon

Výsledek simulace projektu pomocí vytvořeného testbenche je vidět na obrázku 5.2. Pravý sloupeček obsahuje názvy všech proměnných testbenche, vlevo jsou jejich hodnoty. Dva nejdelší řetězce obsahují poslední konfigurační slovo (horní linie), a aktuálních 64 bitů vytvořených výstupních dat (spodní řada). Simulovaná doba činí 1 ms, hodinová perioda byla ponechána na výchozích 10 ns.



Obr. 5.2: Výstup simulace generátoru FEOSA vytvořeným testbenchem

Při nezohlednění času potřebného pro konfiguraci, která probíhá jednorázově, jde tedy o 8 bytů dat v každém hodinovém cyklu. Při simulaci byla nastavena cílená FPGA deska na model Virtex-7 VC709[13], která používá hodinový oscilátor o frekvenci až 200 MHz a nabízí k použití 430 000 LUT (LookUp Table, pravdivostní tabulka, která udává chování konkrétní logické funkce výsledného programu). Z článku o projektu FEOSA[10] je zároveň možné zjistit, že použití jednoho generátoru toku potřebuje zhruba 600 LUT v zařízení. Pokud by bylo 5 % kapacity desky potřeba k běhu NetCOPEu a dalších 5 % pro režii generátoru, bylo by do zbylých 90 % z 430 000 (387 000) možno uložit až 645 funkčních generátorů toku. Při 8 bytech za cyklus, který trvá při maximální frekvenci 5 ns, a 645 paralelních tocích, činí teoretický maximální výstup více než 1 TB dat za sekundu. Je však třeba pamatovat, že uvažovaná deska patří ke špičce (i vzhledem ke své pořizovací ceně – zhruba 120 tisíc korun[13]), a že je tento výpočet pouze teoretickým odhadem nejvyššího možného výstupu generátoru. V uvedených hodnotách tedy nejsou uvažována režijní data NetCOPEu, Ethernetu, ani další přípustné negativní vlivy.

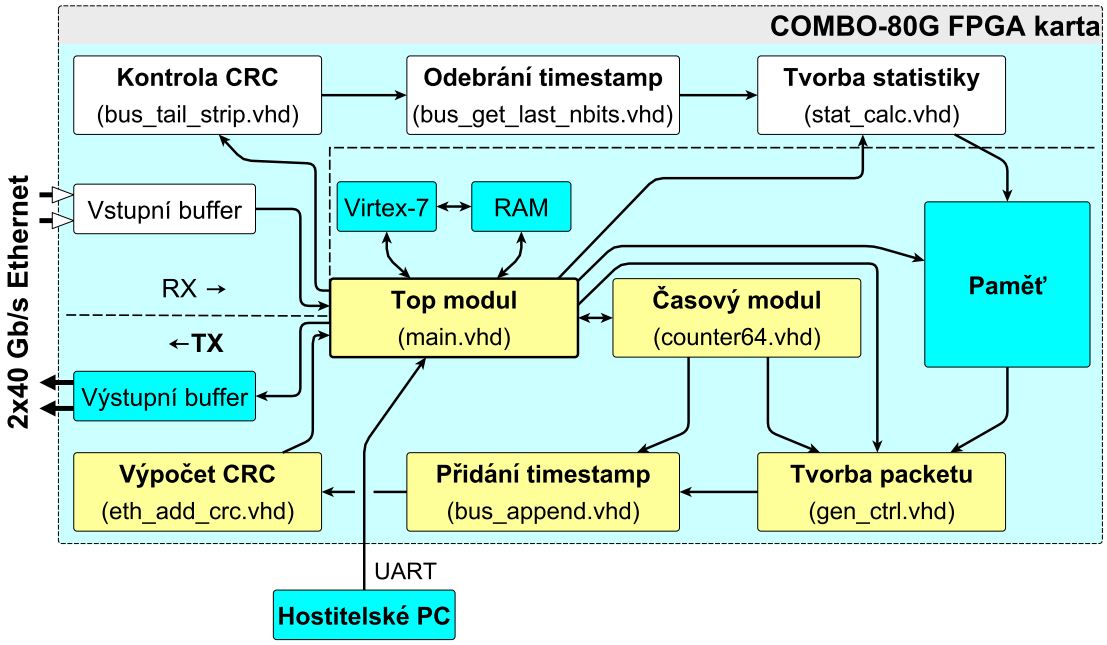
Lze nicméně říci, že generátor FEOSA je velice schopným řešením pro generování záplavových útoků.

5.2 Řešení II

Druhá odnož hardwarového generování dat je založena na projektu PPG. Tento řeší tvorbu packetů, jejich odeslání a příjem a následné změření doby této cesty (RTT). Jako takový představuje spíše síťový testovací nástroj, jehož nedílné části – tvorby zapouzdřených dat – je v této práci využito.

Schéma použití souborů projektu je zobrazeno na obr. 5.3. Protože jde o program přeposílající přijatá data, je zahrnuta i část zodpovědná za příjem síťového provozu, jeho analýzu a uložení do paměti (ve schématu značeno bíle, horní část „RX“). Ve spodní části jsou pak soubory potřebné k odesílání dat, jmenovitě *gen_ctrl.vhd*, který čte data z paměti RAM a tvoří z nich packety, těm je vzápětí přidána přesná časová známka (pro pozdější výpočet RTT) při průchodu *bus_append.vhd*, vypočteno kontrolní CRC (*eth_add_crc.vhd*) a jsou zařazeny do výstupní fronty pro odeslání. Stejně jako v případě FEOSA se jedná o jednu hlavní „rouru“, kterou data prochází jednotlivými modifikátory.

Tato linie ve vstupním i výstupním směru prochází přes top modul programu (*main.vhd*), pro zjištění schopností PPG byl tedy vytvořen testbench zaměřený právě na tuto část. Tento nový testbench je jednoduchým simulátorem běhu programu, majícím za úkol inicializovat testovanou jednotku a – jakmile je připravena – odeslat na její vstup data, zastávající příchozí síťový provoz. V této situaci začne top modul



Obr. 5.3: Způsob implementace PPG generátoru

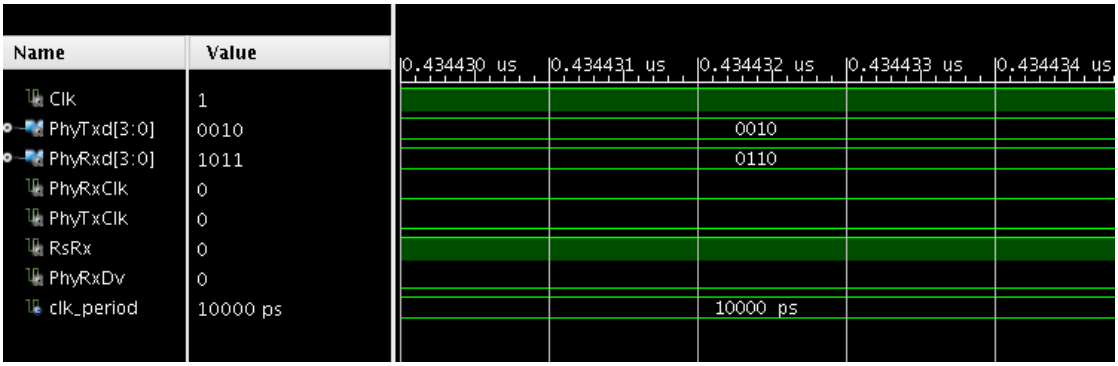
vykonávat svoji práci, spustí veškeré ostatní části programu a až tyto provedou svoje zadané úkoly, opět přijme vytvořená data. Ty následně odešle do výstupní fronty, na což čeká vytvořený testbench, který data zachytí.

Část exekuční části testbenche, ve které probíhá nastavení top modulu a začátek odesílání dat, je vidět v následujícím výpisu:

```
stim_proc: process
   begin
      RsRx <= '1';
      wait for 100 ns;
      RsRx <= '0';
      wait for CLK_period*10;

      PhyTxClk <= clk;
      PhyRxClk <= clk;

      PhyRxD <= "0110";
      wait for clk_period;

      PhyRxD <= "0010";
      wait for clk_period;
```

Stimulační (podněcovací) proces v testbenchi nejdříve provede „restart“ testované jednotky nastavením `RsRx` na 1 a zpět na 0. Poté do frekvence pro příjem (`PhyRxClk`) a vysílání (`PhyTxClk`) přiřadí stejnou hodnotu, jako je frekvence běhu programu

a nakonec spustí periodické vysílání dat na vstupní rozhraní (`PhyRxD`) top modulu. Tato data jsou čtyřbitová (nibble) slova v počtu šedesáti čtyř, celkem tedy program



Obr. 5.4: Výstup simulace generátoru PPG vytvořeným testbenchem

ze simulované sítě přijme 32 B, které později odešle, aby mohl změřit RTT. Výsledek této simulace je zachycen na obr. 5.4, ve kterém jsou vidět v aktuálním cyklu přijímaná (0110) a odesílaná (0010) data.

5.2.1 Zjištěný teoretický výkon

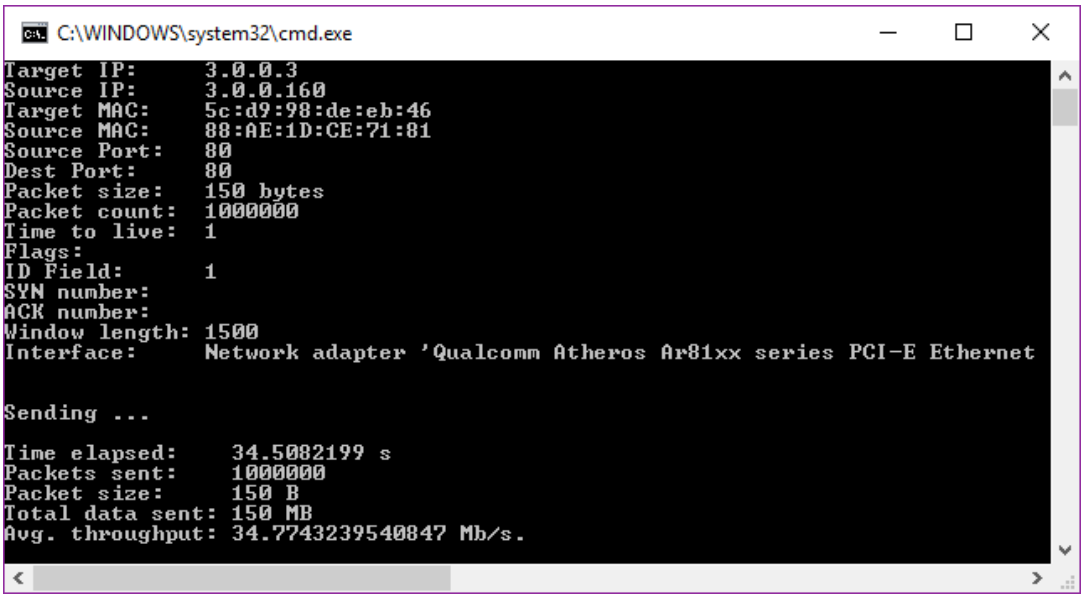
Při myšlené periodě odesílání nibblů 10 ns (odpovídá frekvenci čipu 100 MHz) činí výkon generátoru 50 MB vygenerovaných dat každou sekundu. Při uvážení faktu, že PPG neumí tuto hodnotu násobit množením procesů tvorby dat, jde v porovnání s FEOSA o nízkou hodnotu. To je způsobeno nejspíše tím, že byl tento projekt vyvíjen pro měření RTT, nikoli tvorbu internetových záplav. Pro toto použití tedy není příliš vhodný.

5.3 Softwarové řešení

Poslední z navržených řešení generování packetů není založeno na FPGA, nejedná se totiž ani o hardwarovou aplikaci. Pro účely srovnání výhodnosti hardwarového a softwarového řešení pro generování packetů byla vytvořena počítačová aplikace v jazyce C#, běžící v prostředí .NET operačního systému Windows a využívající open-source knihoven Pcap.net[14] pro práci se síťovými rozhraními a Command Line Parser Library[15], výrazně zjednodušující práci s parametry v příkazovém řádku (vše v přiloženém optickém disku). Výstup programu při zadání parametrů

```
SW_Generator.exe -c 1000000 -z 100 -w 1500 -t 1 -i 1 -d 3.0.0.3
                 -m 5c:d9:98:de:eb:46 -p 80
```

je vidět na obrázku 5.5. Všechny hodnoty byly potvrzeny analyzátorem Wireshark.



Obr. 5.5: Výstup C# generátoru

Program byl tvořen se záměrem možnosti vysoké konfigurace. Ta – krom nutných parametrů, jako jsou adresa cílové stanice – umožňuje nastavit i proměnné, jako jsou TTL (Time to Live), délka okna, příznaky i s hodnotami čísel SYN a ACK (vše vysvětleno v kapitole 2.1.6), podvrhnutou zdrojovou IP a MAC adresu, čísla portů obou stran i velikost posílaných rámců. Data (payload) jsou generována náhodně. Vše je uzpůsobeno snaze o co nejrychlejší odesílání dat. V první části programu jsou nakonfigurovány zmíněné vlastnosti packetu, zjištěna dostupná síťová rozhraní a vybrané z nich připraveno k akci. Je vytvořena kýžená hlavička packetu, podle zadané velikosti automaticky vygenerována data. Druhá fáze běhu programu pak sestává ze smyčky, ve které se již předem vytvořený packet repetitivně odesílá daným rozhraním. Doba trvání tohoto úkonu je zároveň měřena, je tedy možné z této hodnoty a známého objemu odeslaných dat po jeho skončení vypočíst průměrnou rychlost tohoto provozu.

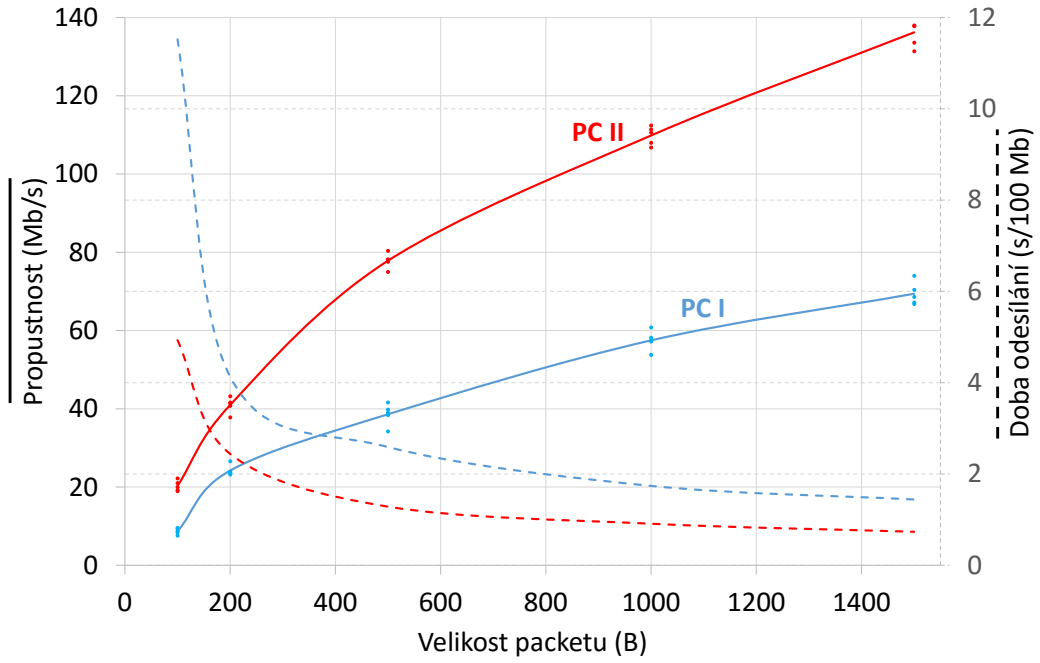
<table>
<tr><td>Počet</td><td>1 000 000</td><td>Cílová IP adresa</td><td rowspan="2">adresa výchozí brány</td></tr>
<tr><td rowspan="2">Velikost (B)</td><td rowspan="2">100, 200, 500, 1000, 1500</td><td>Cílová MAC adresa</td></tr>
<tr><td>Zdrojová IP adresa</td><td rowspan="2">adresa použitého rozhraní</td></tr>
<tr><td>Velikost okna (B)</td><td>1500</td><td>Zdrojová MAC adresa</td></tr>
<tr><td>Příznaky</td><td>žádné</td><td>Cílový port</td><td rowspan="2">80</td></tr>
<tr><td>Time to Live</td><td>1</td><td>Zdrojový port</td></tr>
<tr><td colspan="4">Objem dat v jednom měření: 100 MB – 1.5 GB</td></tr>
</table>

Tab. 5.1: Nastavené vlastnosti odesílaných packetů v testu

Maximální velikost packetu, přijatého z TCP vrstvy, je 65 535 bytů, největší velikost rámce pro přenos Ethernetem ale činí pouhých 1 500 bytů. Protože vytvořený program neumí dělit packety na části, je pro velikost jednotlivých zpráv převzata hranice 1 500 bytů. Při použití této hodnoty zabírají „užitná“ vygenerovaná data 96,4 % přenášeného objemu (velikost hlavičky 54 B).

5.3.1 Výsledky softwarové alternativy

Běh této aplikace byl se stejně zadanými parametry (podle tab. 5.1) vyzkoušen na dvou různě výkonných PC s velikostí packetů odškálovanou na 100, 200, 500, 1000 a maximálních 1500 B. Výsledky, představované průměrem hodnot získaných z pěti měření, jsou vidět v grafu na obr. 5.6, ze kterého je patrná závislost výstupní rychlosti dat na výkonu použitého PC a na velikosti odesílaných dat.



Obr. 5.6: Výkon C# generátoru v závislosti na velikosti packetů pro dvě různé stanice

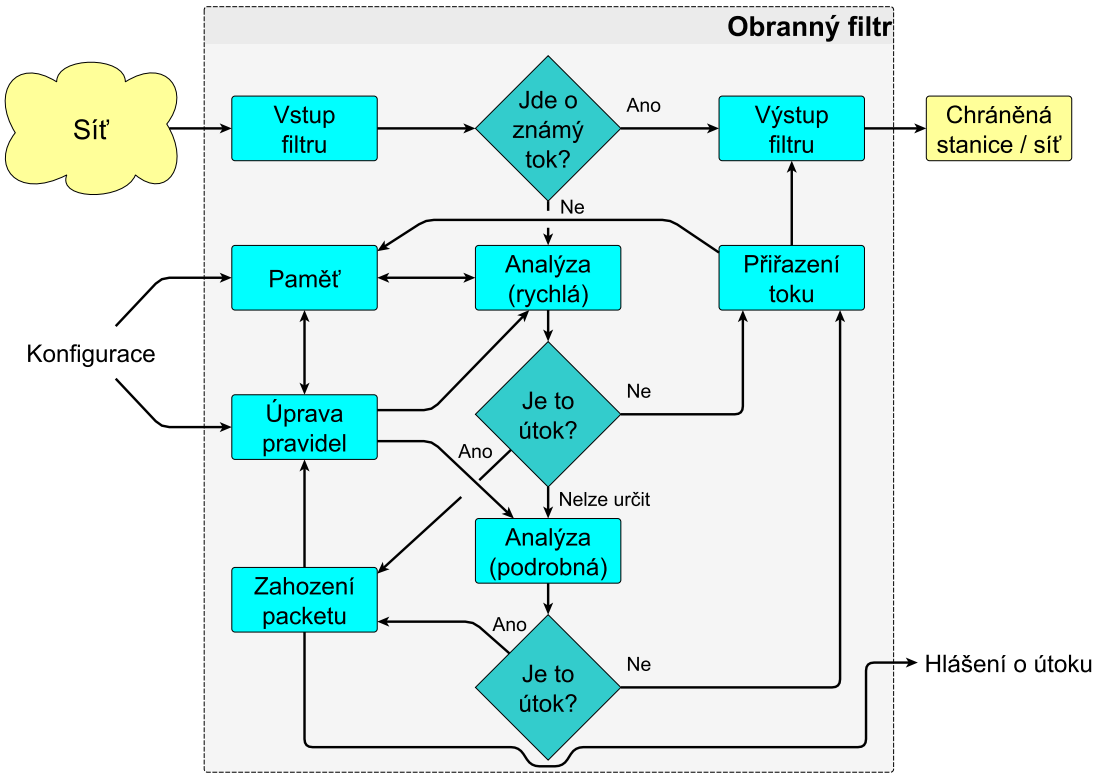
Při testech bylo zjištěno, že výkon zkoušené aplikace je závislý téměř výhradně na výkonu CPU použitého počítače, na ostatní komponenty proto nebyl brán zřetel. PC I je osazeno čipem AMD Turion II P540 (dvě procesorová jádra, 2.4 GHz), PC II používá Intel Core i7-3632QM (čtyři jádra, 2.2 GHz), jenž je podle dostupných měření[16] o 50–80 % výkonnější. Nejen že je tedy účinnost softwarového řešení generování packetů výrazně závislá na použitém CPU, ani při ideálních podmínkách navíc není dosaženo rychlostí hardwarových variant.

Výsledkem srovnání je tedy potvrzení obecně známého faktu, že FPGA je pro jednoúčelové použití svým výkonem, dosaženým vysokou paralelizací, vhodnější.

6 NÁVRH ZABEZPEČENÍ

Komplikací ve snaze bránit datovým záplavám je fakt, že zákeřné packety mají většinou navzdory velmi špatnému úmyslu naprosto správný obsah a formát. Nelze je tedy obecně odchytávat na základě kontrolního součtu, specifických sekvencí v užitné části nebo podle určení zdroje. Současná obrana proti záplavovým útokům se často zaměřuje na konkrétní typ (výčet nejčastějších v kap. 2.2), u nějž potom neguje možnost zneužití vlastnosti daného protokolu, např. omezením maximálního počtu současně uložených polootevřených spojení při SYN záplavě (2.2.2). Takovéto řešení útoku nezabraňuje, nýbrž zmírňuje jeho dopady. Pokud ovšem nemá být počet spojení omezen neustále a snižovat tím kapacitu serveru, musí dojít ke správné detekci, z níž předchozí vyústí jako dočasná, obranná akce.

Vlastní návrh takového filtru je zobrazen na obr. 6.1. FPGA zařízení, postavené podle tohoto modelu, by fungovalo jako prostředník mezi klientskou stanicí nebo chráněnou doménou a potenciálně nebezpečným zbytkem veřejné sítě. Jde o jedno-



Obr. 6.1: Návržený model zabezpečení

směrný filtr, který propouští packety známých a ověřených – důvěryhodných – toků, a data bez tohoto označení testuje na legitimnost. Pokud nepozná škodlivý úmysl, vytvoří pro packet nový tok a všechny další zprávy daného toku (popsaného například použitým protokolem transportní vrstvy, zdrojovou a cílovou IP adresou a cí-

leným číslem portu), již projdou filtrem bez přídavné kontroly. Touhle cestou jsou tedy zdrženy pouze ty packety, které je potřeba ověřit, nikoli ty, jimž již totožnost prokázaly předchozí. Celý navržený systém se zároveň učí z již zjištěných okolností, k čemuž mu také slouží paměť. Tento proces je možné konfigurovat, stejně jako měnit hladiny podezření, používané dvěma analyzátory. Vyhodnocení útoku a následné zahození packetu je podnětem pro zapsání do paměti a pravidel a je hlášeno případným logovacím kanálem.

Samotný proces analýzy packetů a metody učení filtru nicméně nejsou předmětem návrhu, jde pouze o design filtru jako celku pro použití v FPGA.

7 ZÁVĚR

V teoretické části bakalářské práce jsou popsány druhy záplavových útoků a jejich nejčastěji používané typy. Je vysvětleno, v čem se liší, jaké slabiny využívají a jakou měrou mohou na oběť působit, včetně statisticky nejpodstatnějších motivů potenciálního útočníka. Zmíněny jsou také možnosti generování záplavy s pomocí dostupného softwaru. Je vytvořena počítačová simulace chování napadené nano sítě, ze které je vidět dopad na cílené uživatele, popis tohoto programu a vysvětlení jeho částí spolu se seznámením s použitými nástroji.

V praktické části byla otestována dvě dostupná řešení pro tvorbu záplavových dat na platformě FPGA. Vlastní řešení nebylo pro svou složitost vyrobeno. Je zde také popsána vytvořená C# aplikace, zastávající softwarový program v kontrastu s hardwarovým přístupem FPGA. Možnosti všech tří řešení jsou změřeny a porovnány. Bylo zjištěno, že pro jednoúčelovou úlohu generování dat je FPGA výrazně vhodnější, než CPU běžného počítače. Ze dvou testovaných řešení prokázal markantně lepší vlastnosti generátor FEOSA. Na konci této části je uveden návrh zabezpečení proti záplavovým útokům v podobě adaptabilního filtru síťového provozu.

S příspěvkem založeným na této práci se autor zúčastnil studentské konference EEICT s pozitivním hodnocením.

LITERATURA

[1] United states computer emergency readiness team: Understanding Denial-of-Service Attacks. *How do you know if an attack is happening?* [online]. [cit. 2015-11-01]. Dostupné z: https://www.us-cert.gov/ncas/tips/ST04-015

[2] Technopedia. *Denial-of-Service Attack (DoS)* [online]. [cit. 2015-10-31]. Dostupné z: https://www.techopedia.com/definition/24841/denial-of-service-attack-dos

[3] World's largest DDoS attack reached 400Gbps. *Techworld* [online]. [cit. 2015-11-28]. Dostupné z: http://www.techworld.com/news/security/worlds-largest-ddos-attack-reached-400gbps-says-arbor-networks-3595715

[4] DDoS attacks: Understanding the Threat. *Arbor Networks: The Security Division of NETSCOUT* [online]. [cit. 2016-05-23]. Dostupné z: https://resources.arbornetworks.com/h/i/21694420-ddos-attacks-understanding-the-threat

[5] State of the internet: Security visualizations. *Attack types* [online]. [cit. 2015-11-01]. Dostupné z: https://www.stateoftheinternet.com/trends-visualizations-security-real-time-global-ddos-attack-sources-types-and-targets.html

[6] Anonymous cyber-attacks cost PayPal £3.5M, court told. *The Guardian* [online]. [cit. 2015-12-01]. Dostupné z: http://www.theguardian.com/technology/2012/nov/22/anonymous-cyber-attacks-paypal-court

[7] Comparing Three Simulation Model Using Taxonomy: System Dynamic Simulation, Discrete Event Simulation and Agent Based Simulation General Terms-Simulation. *Academia* [online]. [cit. 2015-11-28]. Dostupné z: http://www.academia.edu/4844456/Comparing_Three_Simulation_Model_Using_Taxonomy_System_Dynamic_Simulation_Discrete_Event_Simulation_and_Agent_Based_Simulation_General_Terms-Simulation

[8] COMBO-80G. *www.liberouter.org: Liberouter* [online]. [cit. 2016-04-03]. Dostupné z: www.liberouter.org/combo-80g/

[9] NetCOPE FPGA Boards. *NetCOPE Technologies* [online]. [cit. 2016-04-03]. Dostupné z: http://www.netcope.com/en/products/fpga-boards

[10] GROLÉAT, Tristan, Alban BOURGE, Matthieu ARZEL, Yannick LE BALCH, Sandrine VATON a Hicham BOUGDAL. *Flexible, extensible, open-source*

and affordable FPGA-based traffic generator [online]. 2012, , 1-8 [cit. 2016-04-03]. Dostupné z: https://portail.telecom-bretagne.eu/publi/public/fic_download.jsp?id=16866

[11] Hardware traffic generator. *GitHub* [online]. [cit. 2016-04-03], pod licencí Creative Commons (http://creativecommons.org/licenses/by-nc-sa/4.0/legalcode. Dostupné z: https://github.com/twisterss/hardware-traffic-generator

[12] VHDL Precise Packet Generator. *GitHub* [online]. [cit. 2016-04-09]. Dostupné z: https://github.com/kuba-moo/VHDL-precise-packet-generator

[13] Xilinx Virtex-7 FPGA VC709 Connectivity Kit. *Xilinx: All Programmable* [online]. [cit. 2016-05-23]. Dostupné z: http://www.xilinx.com/products/boards-and-kits/dk-v7-vc709-g.html

[14] Pcap.Net. *GitHub* [online]. [cit. 2016-04-21]. Dostupné z: https://github.com/PcapDotNet/Pcap.Net

[15] Command Line Parser Library. *GitHub* [online]. [cit. 2016-04-21]. Dostupné z: https://github.com/gsscoder/commandline

[16] Intel Core i7 3632QM vs AMD Turion II P540. *Cpuboss* [online]. [cit. 2016-04-22]. Dostupné z: http://cpuboss.com/cpus/Intel-Core-i7-3632QM-vs-AMD-Turion-II-P540

SEZNAM ZKRATEK

ACK	Acknowledge
CLB	Configurable Logic Block
CSMA	Carrier Sense Multiple Access
DDoS	Distributed Denial of Service
DMA	Direct Memory Access
DNS	Domain Name System
DoS	Denial of Service
DRDoS	Distributed Reflected Denial of Service
DSCP	Differentiated Services Code Point
FCS	Frame Check Sequence
FLU	FrameLinkUnaligned
FEOSA	Flexible, Extensible, Open-source and Affordable FPGA-based Traffic Generator
FPGA	Field-programmable Gate Array
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IOB	Input/Output Block
IP	Internet Protocol
ISO/OSI	International Standards Organization / Open Systems Interconnection
LOIC	Low Orbit Ion Cannon
LUT	LookUp Table
MAC	Media Access Control
MI32	Memory Interface 32-bits
MTU	Maximum Transmission Unit
NIC	Network Interface Controller
NTP	Network Time Protocol
ns-3	Network simulator 3
OSPF	Open Shortest Path First
PDU	Protocol Data Unit
PPG	Precise Packet Generator
RAM	Random Access Memory
RTT	Round Trip Time
SaaS	Software as a Service
SSDP	Simple Service Discovery Protocol

SYN	Synchronize
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URL	Uniform Resource Locator
US-CERT	United States Computer Emergency Readiness Team
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

A OBSAH PŘILOŽENÉHO CD

- **ns-3** ..... soubory simulace v ns-3
  - output ..... výstupní grafy
    - video
  - src ..... zdrojové kódy
- **text** ..... soubory k tvorbě výsledného PDF
  - latex ..... soubory VUT šablony
    - loga
    - obrazky
    - pdf
    - text
  - office ..... vytvořené tabulky a grafy
  - yed_graph_editor ..... vytvořená schémata
- **visual studio** ..... vytvořená C# aplikace
  - packages ..... použité knihovny třetích stran
    - CommandLineParser 1.9.71
      - lib
        - net35
        - net40
        - net45
    - Pcap.net
  - SW_Generator ..... soubory aplikace
    - bin
      - Debug
      - Release
    - libs
    - obj
      - Debug
      - Release
    - Properties
- **vivado** ..... soubory vytvořené pro VHDL
  - feosa
  - ppg