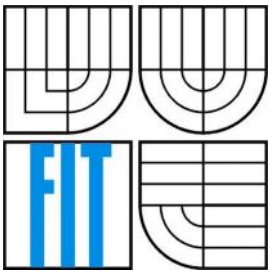
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TESTOVÁNÍ VÝKONNOSTI AKTIVNÍCH SÍŤOVÝCH PRVKŮ

PERFORMANCE TESTING OF ACTIVE NETWORK DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL VRTÍLEK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR MATOUŠEK, PH.D.

BRNO 2007

Zadání bakalářské práce/5454/2006/xvrtil00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Vrtílek Michal**
Obor: Informační technologie
Téma: **Testování výkonnosti aktivních síťových prvků**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s metodologií na vytváření testů pro síťová zařízení.
2. Pro konkrétní skupiny aktivních prvků (směrovače, přepínače, firewally) navrhněte množinu testů pro zjišťování výkonu těchto zařízení.
3. Navrhněte nástroje pro jednotlivé testy. Ukažte způsoby zapojení testovacích nástrojů a testovaných zařízení.
4. Implementujte navržené nástroje.
5. V případové studii konkrétního zařízení ukažte výsledky navržených testů.
6. Zhodnoťte přínos a využití vytvořených nástrojů.

Literatura:

- S.Bradner, J. McQuaid: Benchmarking Methodology for Network Interconnect Devices, IETF RFC 2544, 1999.
- M.Konečný: Tool for Generating Traffic with GUI, diplomová práce, FIT VUT v Brně, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Matoušek Petr, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Michal Vrtílek**
Id studenta: 84102
Bytem: Blodkova 3762/11, 636 00 Brno
Narozen: 22. 03. 1984, Brno
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Testování výkonnosti aktivních síťových prvků
Vedoucí/školitel VŠKP: Matoušek Petr, Ing., Ph.D.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

V současné době se rozvíjí hlasové a televizní služby přes sítě nad IP. Tyto služby kladou velké nároky na výkon aktivních síťových prvků. Pokud chceme zjistit, zda dané zařízení splňuje nároky na přenos, je potřeba jeho výkonnost otestovat. K testování výkonu jednotlivých zařízení se mohou použít hardwarové či softwarové aplikace. Testy by měly splňovat požadavky kladené standardem RFC 2544. V dnešní době neexistuje žádná volně dostupná aplikace, která by podle daného standardu prováděla všechny testy požadované standardem. Tato práce se zabývá návrhem a implementací testovací aplikace pro aktivní síťové prvky podle standardu RFC 2544.

Klíčová slova

aktivní síťové prvky, testování výkonnosti, RFC 2544

Abstract

At the present time voice and TV services deploy over IP networks. These services put high requirements on network devices. To find out if an device covers these requirement we need to pass efficiency tests on the device. Hardware and software tools can be used for testing. Tests should match requirements based by RFC 2544. Nowadays there isn't any free available application that should do all tests according to the standard. This work deals with proposal and implementation testing application for active network devices according to standard RFC 2544.

Keywords

active network devices, performance testing, RFC 2544

Citace

Michal Vrtílek: Testování výkonnosti aktivních síťových prvků, bakalářská práce, Brno, FIT VUT v Brně, 2007

Testování výkonnosti aktivních síťových prvků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Ing. Petra Matouška, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Vrtílek
15.5.2007

Poděkování

Děkuji tímto vedoucímu své bakalářské práce, Ing. Petru Matouškovi, Ph.D, za vedení, pomoc při řešení problémů a vstřícný přístup.

Obsah

Obsah ... 1
Úvod ... 3
1 Protokoly používané pro testování výkonnosti ... 4
1.1 Ethernet ... 4
1.1.1 Formát rámce ... 4
1.2 IP (Internet Protocol) ... 5
1.2.1 Formát paketu ... 5
1.3 UDP (User Datagram Protocol) ... 7
1.3.1 Formát UDP ... 7
2 Standardy pro testy ... 8
2.1 RFC 1242 – terminologie ... 8
2.2 RFC 2544 – metodologie testů ... 8
2.2.1 Test propustnosti ... 11
2.2.2 Test zpoždění (latence) ... 11
2.2.3 Test ztrátovosti rámců ... 12
2.2.4 Test back-to-back rámců ... 13
2.2.5 Zotavení se po přetížení ... 13
2.2.6 Zotavení se po restartu ... 14
3 Návrh testovací aplikace ... 15
3.1 Koncept aplikace ... 15
3.1.1 Klient ... 15
3.1.2 Server ... 16
4 Implementace ... 17
4.1 Nároky na software ... 17
4.2 Klient ... 17
4.2.1 Zpracování konfiguračního souboru (config.dat) ... 18
4.2.2 Provádění testů ... 18
4.3 Server ... 19
5 Instalace a použití ... 20
5.1 Instalace ... 20
5.2 Použití ... 20
5.2.1 Server ... 20
5.2.2 Klient ... 20
6 Testy zařízení ... 23

6.1 Přepínač 23
6.2 Směrovač 23
6.3 Test přepínače Cisco Catalyst 2950 24
7 Závěr 27
Literatura 28
Seznam příloh 29

Úvod

V současné době zažívají počítačové sítě velký rozvoj. Ve velkém se budují sítě podnikové s připojením k celosvětové počítačové síti, internetu. Rozvíjí se hlasové a televizní služby používající IP sítě. Také domácí uživatelé mají zájem o využití internetu, mnohdy se však již nespokojí s pomalejším připojením, tedy pomocí modemů. Naopak mají zájem o vysokorychlostní připojení. Všechny tyto požadavky kladou velké nároky na výkon aktivních síťových prvků. A výkon je právě to, o co se jedná v mé práci.

Pro zjištění výkonu síťových zařízení se v dnešní době používají buď hardwarové nebo softwarové testovací nástroje. Výhodou hardwarových nástrojů je vysoká přesnost a snaha o dodržování standardů, ale bohužel jejich pořizovací cena je vysoká, a proto nejsou dostupné v tak velké míře jako softwarové. V segmentu softwarových nástrojů je jich několik, které provádějí testy podle standardu. Ovšem tyto nástroje nejsou volně dostupné. Ostatní buďto standard nedodržují vůbec nebo jen sporadicky. Proto cílem této práce je vytvořit softwarový nástroj, který bude provádět testy výkonnosti podle standardu a bude volně dostupný.

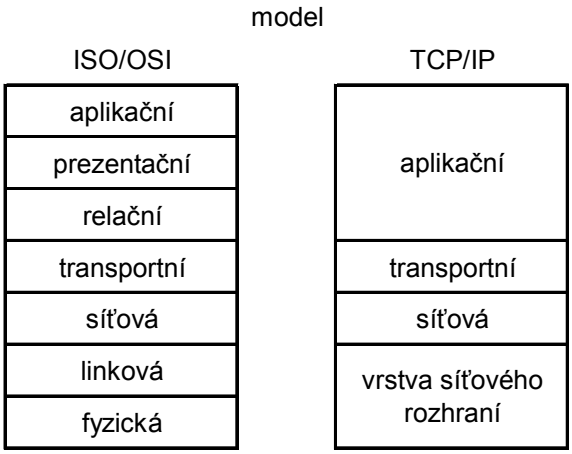
V následující kapitole popisuji protokoly (Ethernet, IP a UDP), které jsem využil při tvorbě aplikace. Druhá kapitola je věnována standardu pro tvorbu výkonnostních testů. Po této teoretické části přicházím se samotným návrhem testovací aplikace (kapitola 3). Bližší informace o implementaci jsou uvedeny v kapitole čtvrté. V páté popíši postup instalace a použití implementované aplikace. V šesté kapitole jsem navrhl různé formy testů jednotlivých zařízení a otestoval reálné zařízení.

1 Protokoly používané pro testování výkonnosti

V této kapitole jsou popsány síťové protokoly používané v programech pro testování výkonnosti.

Protokol je jazyk využívaný k dorozumění se mezi dvěmi zařízeními. Jeho syntaktická a sémantická pravidla určují strukturu informací obsažených v jednotlivých blocích.

V počítačových sítích se zavedlo rozdělení protokolů na vrstvy. Toto dělení je velice důležité, protože dokáže skrýt rozdílné typy hardwaru a softwaru. V ISO/OSI modelu existuje 7 vrstev, které jsou v TCP/IP modelu spojeny do 4 vrstev, přičemž každá z nich poskytuje své služby vrstvě nad ní. V každé vrstvě je definován jeden či více protokolů.



1.1 Ethernet

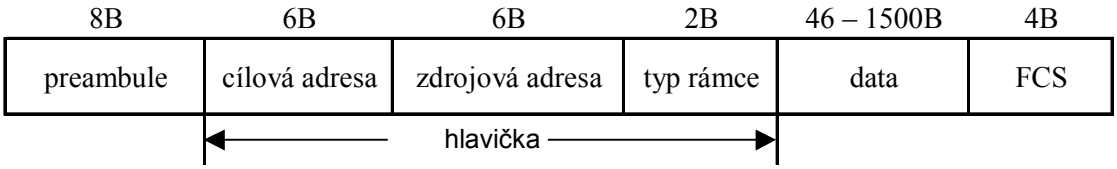
Ethernet je jedním z typů rámcově orientovaných lokálních sítí. Zajišťuje služby fyzické a linkové vrstvy ISO/OSI modelu.

Fyzická vrstva definuje síťová zařízení (např.: kabeláž, konektory, …) a signály. Pracuje s jednotlivými bity a spojuje fyzickou vrstvu jednoho zařízení s fyzickou vrstvou druhého zařízení.

Linková vrstva pracuje nad fyzickou vrstvou. Na této úrovni zařízení mají již adresu – MAC adresu (Media Access Control). MAC adresa je každému zařízení přidělena výrobcem, a proto je možné bloky informací neboli rámce doručovat cíleně.

1.1.1 Formát rámce

Nejvíce používaný formát ethernetového rámce je Ethernet II, zobrazený v následujícím obrázku.



- preambule – skládá se střídavě z binárních 0 a 1 sloužících k synchronizaci.
- zdrojová a cílová adresa (MAC adresa) – je unikátní.
- typ rámce – obsahuje 16 bitů identifikující typ použitého protokolu vyšší vrstvy. Např. 0x0800 definuje IPv4 nebo 0x0806 ARP.
- FCS (Frame Check Sequence) – obsahuje kontrolní součet, aby bylo možno detekovat, případně opravit, chyby způsobené přenosem. Obvykle se pro výpočet kontrolního součtu používá cyklický redundantní součet (CRC).

Více informací o Ethernetu lze nalézt v [1].

1.2 IP (Internet Protocol)

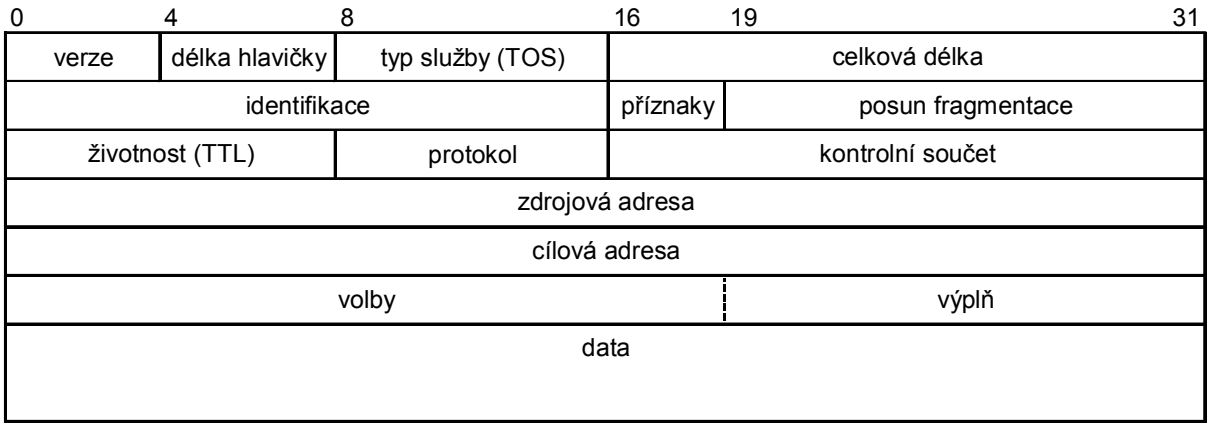
Protokol IP je základním protokolem síťové vrstvy ISO/OSI modelu a zároveň základním protokolem sítě Internet. Na této vrstvě neexistuje jen IP protokol, ale jsou zde i jiné. ARP/RARP protokoly překládají MAC adresu na IP adresu. Další jsou ICMP, IGMP. Postupem času bylo vyvinuto několik verzí IP protokolu. Dnes nejpoužívanější je verze 4, ale postupně se zavádí nová a to verze 6. Ta se od verze 4 liší mimo jiné velikostí IP adresy. Zatímco u verze 4 to bylo 32 bitů, zde je to již 128 bitů. Dále se budu zabývat pouze nejpoužívanější verzí 4.

Každému zařízení přiřazujeme IP adresu, což je unikátní 32 bitové (4 bytové) číslo. Obvykle se zapisuje ve formě čtyř desítkových číslic oddělených tečkami (např. 147.229.9.22). IP adresa se rozděluje na dvě části – síťovou a hostitelskou část. Síťovou část používají směrovače (routery) k určení následujícího uzlu v cestě bloku dat (blokem dat rozumíme IP paket). Hostitelská část identifikuje cílové zařízení v rámci jedné sítě určené síťovou částí.

IP protokol poskytuje nespolehlivou službu, negarantuje doručení paketu (spolehlivost doručení může být docílena např. protokolem TCP). Stále se jedná o spojení jednotlivých zařízení a ne aplikací na nich běžících.

1.2.1 Formát paketu

Paket obsahuje hlavičku a data z transportní vrstvy ISO/OSI modelu. Zde si vysvětlíme, co znamenají jednotlivé části hlavičky, jak uvádí [2]:



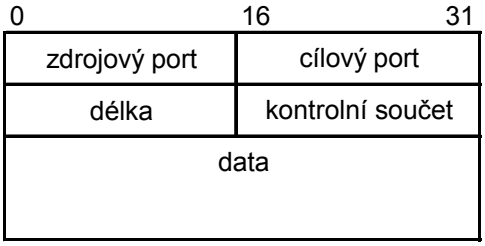
- verze (4b) – verze protokolu IP („4“ pro IPv4).
- délka hlavičky (4b) – délka hlavičky v 32 bitových slovech. Nejmenší hodnota je pět. Maximální možná velikost hlavičky tedy vychází na 480 bitů (60B).
- typ služby (8b) – slouží ke specifikaci zacházení s IP paketem během přenosu (možnost preference paketu před ostatními). Více v [3].
- celková délka (16b) – délka paketu zahrnující hlavičku a data v bytech. Maximální možná hodnota je 65535B.
- identifikace (16b) – jednoznačná identifikace paketů. Pozná se podle toho, které pakety patří k sobě, pokud došlo k fragmentaci.
- příznaky (3b) – slouží pro fragmentaci. Definovány jsou dva: More fragments (MF), za takto označeným paketem následuje další fragment původního paketu, a Don’t fragment (DF) zakazující tento paket fragmentovat.
- posun fragmentu (13b) – udává posun v paketu. Jednotkou je 8 bytů. První fragment má hodnotu 0.
- životnost (8b) – udává maximální čas, po který paket existuje. Každé zařízení pracující s IP paketem zmenší tuto hodnotu o jedničku (případně o počet sekund, které paket strávil v zařízení, pokud zde čeká déle). Pokud dosáhne hodnotu nula, paket se zahodí, protože vypršela jeho životnost.
- protokol (8b) – určuje protokol vyšší vrstvy, který je použit pro data. Hodnoty lze nalézt v [4].
- kontrolní součet (16b) – počítá se pouze z hlavičky. Je přepočítán a zkontrolován v každém bodě, kde je hlavička zpracovávána. Pokud nesouhlasí, je paket zahozen.
- zdrojová adresa (32b)
- cílová adresa (32b)
- volby (variabilní) – různé rozšiřující požadavky či informace. Můžou nebo nemusí být obsaženy v každém paketu.
- výplň (variabilní) – zajišťuje, že hlavička končí na násobku 32 bitové hranice.

1.3 UDP (User Datagram Protocol)

Protokol UDP je základním protokolem transportní vrstvy ISO/OSI modelu. Na této úrovni se již jedná o propojení různých aplikací přes síťovou infrastrukturu.

Protokol UDP poskytuje aplikacím posílání zpráv s minimálním protokolovým zatížením, ovšem bez garance doručení nebo duplikace zprávy. Díky tomu je UDP pro lehké a časově citlivé účely rychlejší a efektivnější. Pokud bychom požadovali garanci, museli bychom využít jiného protokolu, např. TCP (Transmission Control Protocol).

1.3.1 Formát UDP



Jak možno vidět na obrázku, hlavička protokolu UDP se skládá pouze ze čtyř částí:

- zdrojový port (16b) – nepovinný údaj. Slouží k identifikaci zasílající aplikace, aby jí mohla být doručena případná odpověď. Pokud je nepoužit, dosazuje nula.
- cílový port (16b) – identifikuje přijímací aplikaci.
- délka (16b) – celková délka zprávy včetně dat
- kontrolní součet (16b) – nepovinný údaj. Vypočítává se z hlavičky i dat.

Více o protokolu UDP lze najít v [5].

2 Standardy pro testy

Tato kapitola je věnována popisu standardů, podle kterých je v rámci této bakalářské práce navrhnut testovací program.

Výrobci síťových prvků se snaží prodat co nejvíce svých vlastních výrobků. Ty podstupují testy výkonnosti, které jsou navrženy tak, aby testovaná zařízení dosahovala nejlepších výsledků. Proto je snaha o vytvoření standardů, podle kterých by se tyto testy prováděly. Touto činností se zabývá pracovní skupina BMWG (Benchmarking Methodology Working Group) [6], patřící do sdružení IETF (Internet Engineering Task Force). Snaží se popsat různé metodiky provádění testů včetně jednoznačné reprezentace naměřených hodnot. Své metodiky vydává jako dokumenty Request for Comments (RFC), což jsou de facto internetové standardy.

2.1 RFC 1242 – terminologie

Prvním dokumentem, který BMWG vydala, byl v roce 1991 *RFC 1242 Benchmarking Terminology for Network Interconnection Devices* [7]. Jak již název napovídá, tento dokument definuje pojmy, které jsou používány v dalších dokumentech BMWG (např.: v popisech výkonnostních srovnávacích testů nebo jejich výsledků). Kromě klíčových pojmů, jako jsou propustnost nebo zpoždění, obsahuje i definice známé: směrovač (router) nebo most (bridge). Tímto se snaží autoři předejít možným nejednoznačnostem při výkladu.

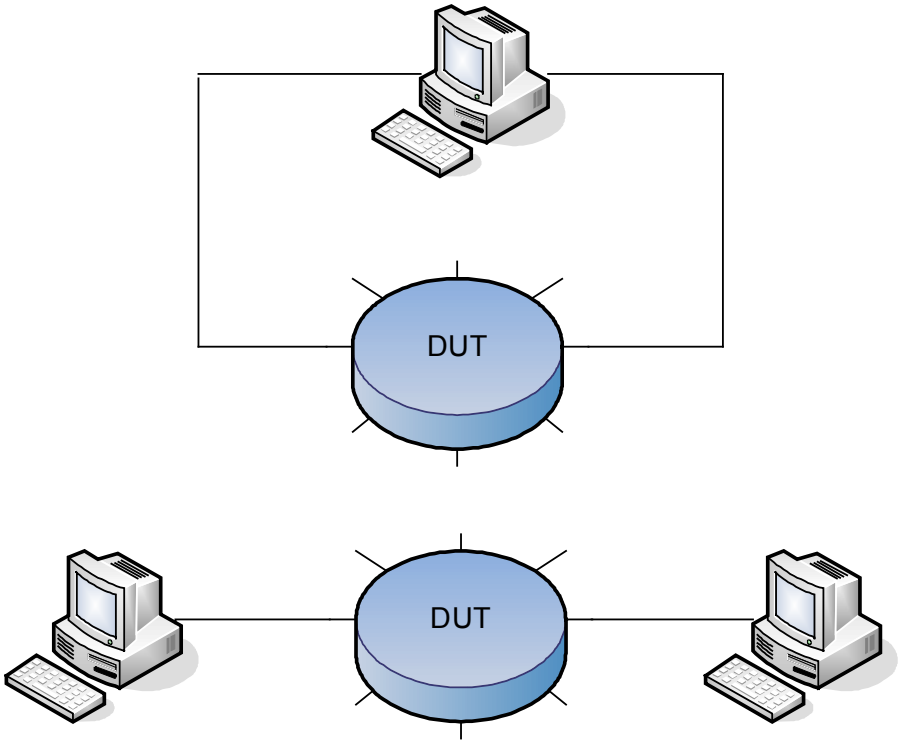
2.2 RFC 2544 – metodologie testů

V roce 1999 vyšel *RFC 2544 Benchmarking Methodology for Network Interconnect Devices* [8], který nahradil *RFC 1944* [9] z roku 1996. Obsahuje popis výkonnostních srovnávacích testů včetně jednoznačného formátu pro reprezentaci výsledků. Tyto výsledky zajišťují jednoduchou porovnatelnost zařízení různých výrobců.

Stejně jako v jiných dokumentech RFC se i zde rozlišuje mezi výrazy *musí*, *měl by* a *volitelné*, které definují požadavky implementace. Podle míry jejich splnění, pak testy částečně či úplně odpovídají specifikacím tohoto dokumentu.

Následuje popis zapojení testovaného zařízení (DUT, Device Under Test) a zamyšlení nad propojením různorodých sítí. V celém dokumentu se na testované zařízení pohlíží jako na skříňku (*blackbox*), protože nás nezajímá jeho vnitřní struktura ani jak zpracovává data, ale jeho výkonové charakteristiky. Nejideálnější cestou k zapojení je využití testovacího zařízení s vysílacím i přijímacím portem. Zasílání testovacích dat probíhá z vysílacího portu do testovaného zařízení a z něj pak do přijímacího portu. Zde pak dochází k porovnání vyslaných dat s přijatými, na jehož základě

prohlásíme jaký výkon má zařízení. Stejného efektu dosáhneme i oddělením vysílacího a přijímacího portu na dvě testovací zařízení, ale pak je nutné zasílat stavové informace.



Další část je věnována nastavení testovaného zařízení. Je očekáváno, že všechny podporované protokoly budou nakonfigurovány a spuštěny. Dále se očekává, že všechny testy provedeme beze změny v konfiguraci daného zařízení. Více informací podává příloha A standardu, který plní funkci průvodce, ale není závazný.

Dále se autoři zabývají formátem testovacích rámců. Pro TCP/IP přes ethernet jsou formáty uvedeny v příloze C standardu. Tyto přesné formáty mohou být použity pro testy popsané v tomto RFC. Jestliže použijeme jiné formáty než uvedené v příloze C standardu, musí být uvedeny ve výsledné zprávě.

Velikost rámců je tématem deváté kapitoly standardu. Všechny popisované testy by měly být spuštěny několikrát, nejméně však s pěti rozdílnými velikostmi rámců. Nejlépe by měly být do testů zahrnuty rámce s minimální a maximální povolenou velikostí pro dané médium. Pro různá média jsou určeny různé velikosti. Např. pro Ethernet jsou to: 64, 128, 256, 512, 1024, 1280 a 1518 bytů. Dále je diskutována velikost rámců na základě MTU (Maximum Transmission Unit) a fragmentace.

Dále je řešeno ověřování přijatých rámců. Testovací zařízení by mělo zahazovat rámce během probíhajícího testu, které nejsou aktuálně přeposílanými testovacími rámci. Ty pak nejsou započítány do celkového počtu přijatých rámců. Např. se může jednat o routovací aktualizace. V každém případě bychom měli kontrolovat, zda se velikosti vyslaných a přijatých rámců shodují. V některých testech záleží i na pořadí testovacích rámců. Pokud každý rámec obsahuje sekvenční číslo, pak můžeme kontrolovat, v jakém pořadí se vrátily zpět, zda nedošlo k duplikaci, atd.

V některých případech je dobré znát výkon testovaného zařízení. Např. v případě směrovače či mostu je dobré do testovacích rámců zahrnout i všesměrové rámce (broadcast frames). Ty pak musí být přeposlány na všechny rozhraní tohoto zařízení (mostu). Doporučuje se, aby takovýto rámec byl jeden ze sta. Nebo můžeme sledovat, jak se testované zařízení zachová na dotaz správy sítě (např. SNMP). Velkou roli také hraje úprava směrovacích tabulek nebo definování filtrů u směrovačů (mostů). Tato pravidla rozhodují na základě zdrojové a cílové adresy o průchodu paketu směrovačem a jsou většinou výpočetně náročné. Před testováním s dodatečnými podmínkami bychom měli testovat zařízení bez těchto podmínek a až poté s každou podmínkou zvlášť.

Další část standardu je zaměřena na adresy používané v testovacích paketech. Nejjednodušší je začít pouze s jedním tokem dat, který má jednu zdrojovou a jednu cílovou adresu. Ovšem v reálném světě je takových toků mnohem více. Proto by se mělo po otestování jedním tokem dat pokračovat s toky dat, které budou mít náhodně generované cílové adresy nebo rovnoměrné rozložení těchto adres. Bližší informace o rozsahu adres pro IP pakety lze nalézt v příloze C standardu.

Až doposud se autoři věnovali jednosměrnému provozu, ale v normální síti je obvyklé, že data tečou oběma směry. Pokud zařízení má více portů (obvyklé u směrovačů), měli bychom všechny porty otestovat každý zvlášť. Tím můžeme odhalit případné rozdíly. Také je doporučováno generovat takový provoz, aby jej zařízení přijalo na různých vstupních portech a pak jej poslalo na všechny výstupní porty v jednu chvíli.

V uvedeném dokumentu nejsou specifikovány žádné testy probíhající na různých protokolech či médiích. Stejně tomu je v případě různých velikostí rámců v testu. Jediný požadavek je kladen na použití velikostí rámců, tak jak bylo uvedeno v předcházejícím textu.

Na testování více propojených zařízení je nahlíženo jako na jedno zařízení (tzv. end-to-end přístup). V tomto případě musíme brát v úvahu, že některá zařízení mohou zvyšovat zpoždění.

Dále bychom měli v testech použít maximální rychlost zasílání rámců dané velikosti na daném médiu. Hodnoty těchto rychlostí jsou uvedeny v příloze B standardu. Ovšem u hodnot pro Ethernet bychom měli být opatrní. Jak uvádí Stuart Johnson ve svém komentáři [10], u Ethernetu je povolená odchylka od hodinového taktu $\pm 0{,}01\%$. Není tedy zaručeno, že každé zařízení zvládne pracovat s rychlostmi uvedenými v příloze B. Takže ztráta $0{,}02\%$ všech zaslaných rámců je povolená.

Další téma je věnováno shlukovému provozu (bursty traffic). Je zde zmíněno, že provoz s konstantní zátěží je nerealistický, a proto je doporučeno generovat i tento typ provozu. Rámce uvnitř shluku mají mít mezi sebou co nejmenší mezery. Cílem tohoto testu je nalézt nejmenší interval mezi shluky, kde by nedošlo ke ztrátě žádného rámce. Shluky mohou obsahovat 16, 64, 256 nebo 1024 rámců.

Posledním tématem před definicí samotných testů je uvedena skladba sady testů. Ta se skládá z několika dílčích testů. Postup provádění těchto dílčích testů je následující:

1. pokud je testované zařízení směrovač, pošleme směrovací aktualizaci na vstupní port a počkáme 2 sekundy, abychom si byli jisti, že směrovač tuto aktualizaci stihl zpracovat.
2. pošleme rámce, které „naučí“ zařízení kam má posílat testovací rámce a počkáme 2 sekundy.
3. spustíme samotný test a počkáme nejméně 60 sekund. U některých testů lze tuto dobu zkrátit, ale konečný test by měl trvat minimálně oněch 60 sekund.
4. počkáme 2 sekundy na poslední příchozí rámce.
5. nejméně 5 sekund necháme zařízení na stabilizaci.

Následuje popis postupu provádění a prezentace výsledků jednotlivých testů.

2.2.1 Test propustnosti

Tento test má za úkol zjistit propustnost zařízení definovanou v RFC 1242, tj. maximální rychlost zasílání rámců, při které není žádný zahozen.

Postup: Pošleme určitý počet rámců o určité rychlosti do zařízení a pak spočítáme počet přeposlaných rámců zařízením. Pokud je počet přijatých rámců roven počtu vyslaných, zvýšíme rychlost vysílání. Ale pokud je tento počet menší, snížíme rychlost vysílání. Poté test opakujeme. Propustnost je tedy nejvyšší rychlost (v rámcích za sekundu), při níž je roven počet vyslaných a přijatých rámců

Testovací zpráva: Výsledky testu by měly být prezentovány ve formě grafu. Na ose x jsou velikosti rámců a na ose y pak rychlost. V grafu by pak měly být zobrazeny nejméně dvě křivky, jedna pro teoretickou rychlost na daném médiu při různých velikostech rámců a druhá pro naměřené výsledky testu. Další křivky mohou ukazovat výsledky pro určitý testovací tok.

Doprovodný text pak informuje o tom, co musí závěrečná zpráva obsahovat a v jakých jednotkách lze interpretovat naměřenou rychlost.

Poznámka: V textu není udán přesný počet rámců, které bychom měli v testu zaslat. Autoři uvádí jen minimální délku trvání, a to 60 sekund. Z tohoto času a známé rychlosti zasílání lze tedy dopočítat počet rámců. Dále zde není přesně popsán způsob implementace. V praxi se však osvědčilo použití nějaké formy binárního půlení. Jeho časová složitost je logaritmická a tudíž se doba trvání značně zkrátí.

2.2.2 Test zpoždění (latence)

Cílem tohoto testu je nalezení zpoždění definované v RFC 1242. Zde je potřeba rozlišit různé typy zařízení, protože existují tři způsoby zpracování rámců (paketů).

Pokud je rámec nejprve celý načten a až poté je započato s přeposíláním, pak mluvíme o způsobu *store and forward*. V tomto případě je zpoždění definováno jako časový interval mezi průchodem posledního bitu na vstupním portu a objevením prvního bitu na výstupním portu.

Zařízení pracující na první vrstvě ISO/OSI modelu (např. opakovače), provádějí *bit forwarding*. Rámec je okamžitě přeposílán na výstupní rozhraní. Zpoždění je pak časový interval mezi průchodem prvního bitu vstupního rámce a objevením prvního bitu na výstupním portu.

Poslední způsob *cut-through* je „kombinací“ předchozích dvou. Rámec není načten celý, ale jen jeho hlavička, poté je již započato s přeposláním na výstupní port. Zde se pak za zpoždění považuje časový interval mezi průchodem posledního bitu hlavičky na vstupním portu a objevením prvního bitu na výstupním portu.

Postup: Před tímto testem je nutné nejprve zjistit propustnost pro rámce dané velikosti na daném médiu. Zašleme tok rámců o dané velikosti rychlostí naměřenou v testu propustnosti. Tento tok dat by měl trvat nejméně 120 sekund. Po 60 sekundách do toku přidáme „označený rámec“ a zaznamenáme čas, kdy byl zcela odeslán. Přijímací část pak musí rozeznat tento „označený rámec“ a zaznamenat čas příchodu tohoto rámce. Zpoždění je pak rozdíl těchto dvou časů podle výše uvedených definic. Tento test musí být proveden minimálně dvacetkrát a výsledná hodnota zpoždění je pak aritmetickým průměrem ze všech těchto naměřených hodnot.

Testovací zpráva: Použitá definice zpoždění z RFC 1242 musí být obsažena v testovací zprávě. Výsledky by měly být ve formě tabulky s řádkem pro každou velikost testovacích rámců. Kromě velikosti rámců obsahují sloupce i rychlost, jakou byly rámce vysílány, typ média a naměřenou hodnotu zpoždění.

Poznámka: Tento test klade vysoké nároky na přesnost času. Pokud budeme test provádět na dvou nezávislých strojích, pak je nutné nejprve je nějakým způsobem synchronizovat. K tomu nám může posloužit např. protokol NTP (Network Time Protocol) a některý z volně přístupných NTP serverů. Také tento test je náročný svou délkou trvání. Pokud zanedbáme čas potřebný k testu propustnosti a k ustálení, tak pro Ethernet se jedná o 7 dílčích testů, který každý trvá nejméně jednu hodinu (dvacet opakování po 120 sekundách).

2.2.3 Test ztrátovosti rámců

I zde se autoři odkazují na definici ztrátovosti (frame loss rate) uvedenou v RFC 1242. Ztrátovost je tedy procentuální údaj, který vyjadřuje kolik rámců mělo být přeneseno síťovým zařízením, ale z důvodu nedostatku systémových prostředků k tomu nedošlo.

Postup: Stejně jako v testu propustnosti i zde zašleme určitý počet rámců o určité rychlosti a zařízení by je mělo přijmout a přeposlat dále. Ztrátovost spočítáme takto:

$$\text{ztrátovost} = \frac{(\text{počet zaslaných rámců} - \text{počet přijatých rámců}) \cdot 100}{\text{počet zaslaných rámců}} [\%]$$

V prvním dílčím testu by měly být rámce zasílány stejnou rychlostí jako je maximální rychlost pro danou velikost rámce na daném médiu. Další dílčí testy jsou prováděny s postupně snižující se

rychlostí s krokem o maximální velikosti 10%, až do té doby, kdy ve dvou po sobě jdoucích dílčích testech nedojde ke ztrátě žádného rámce.

Testovací zpráva: Naměřené hodnoty by měly být vyneseny do grafu. Na ose x je rychlost vysílání v procentech maximální rychlosti daného média. Na ose y pak ztrátovost v procentech. Počátek souřadné soustavy musí být 0% a maximální hodnota na osách pak 100%. Do grafu je možné vynést více křivek, které oznamují např. ztrátovost pro různé velikosti rámců.

Poznámka: Jak již bylo popsáno v postupu, tento test se podobá testu propustnosti. V něm se hledala nejvyšší možná rychlost, při níž není ztracen žádný rámec. To znamená že se nejprve musela zjistit ztrátovost a až pak se podle ní zachovat.

2.2.4 Test back-to-back rámců

Tento test má za úkol zjistit, jak je zařízení schopno pracovat s back-to-back rámci. Pro vysvětlení pojmu back-to-back rámců opět použijeme RFC 1242. Jsou to rámce pevné délky vyslané za sebou. Interval mezi nimi má být minimální povolený na daném médiu.

Postup: Do testovaného zařízení zašleme shluk back-to-back rámců a sledujeme kolik těchto rámců bylo přijato v testovacím zařízení. Jestliže počet přijatých rámců je roven počtu vyslaných, zvýšíme počet rámců ve shluku a test opakujeme. Pokud je počet přijatých rámců menší než vyslaných, tak počet rámců ve shluku naopak snížíme. Výsledkem testu je počet rámců v nejdelším shluku, ve kterém nedošlo ke ztrátě ani jednoho rámce. Každý dílčí test musí trvat nejméně dvě sekundy a měl by být opakován nejméně padesátkrát. Výsledkem je pak průměr z naměřených hodnot dílčích testů.

Testovací zpráva: Měla by být ve formě tabulky. Jeden řádek bude reprezentovat výsledek pro určitou velikost rámce. Sloupce by pak měly obsahovat velikost rámce, naměřený průměrný výsledek a také je vhodné doplnit standardní odchylku.

Poznámka: Stejně jako u testu propustnosti, i zde lze dobu trvání zkrátit pomocí algoritmu binárního půlení. Dolní mez určuje minimální délka trvání testu – dvě sekundy. Pokud je při této délce trvání ztracen nějaký rámec, nelze pomocí tohoto testu změřit testované zařízení.

2.2.5 Zotavení se po přetížení

Cílem tohoto testu je zjištění, jak rychle se umí zařízení zotavit z přetížení.

Postup: Pro tento test je nutné si nejprve zjistit propustnost zařízení pro danou velikost rámce. Poté zašleme tok rámců, trvající alespoň 60 sekund, 110%-ní rychlostí zjištěnou v testu propustnosti. V případě, že je tato rychlost vyšší než maximální, tak pak maximální rychlostí na daném médiu. Následuje snížení rychlosti vysílání na polovinu předchozí rychlosti. Zaznamenáme čas. Dále zaznamenáme čas poslední ztráty testovacího rámce. Výsledkem je rozdíl výše uvedených časů. Dílčí test by měl být opakován vícekrát, pak výsledkem bude průměr ze všech dílčích testů.

Testovací zpráva: Výsledky by měly být vyneseny do tabulky, kde řádky budou reprezentovat jednotlivé velikosti testovacích rámců a sloupce pak budou obsahovat velikost rámce, propustnost pro každý typ datového toku a čas nutný pro zotavení se.

2.2.6 Zotavení se po restartu

V tomto testu jde o zjištění rychlosti, jakou je testované zařízení schopno se zotavit po hardwarovém nebo softwarovém restartu.

Postup: Tak jako v předchozím testu, je i zde potřeba nejprve zjistit propustnost. Tentokrát ovšem jen pro nejmenší rámce povolené na daném médiu. Zašleme testovací proud složený z rámců minimální velikosti rychlostí, která byla naměřena v testu propustnosti. Provedeme restart testovaného zařízení a sledujeme výstupní port. Zaznamenáme čas, kdy byl přijat poslední rámec před restartem, a následně čas přijetí prvního rámce po restartu. Hledaný výsledek je rozdílem těchto dvou časů. Pokud chceme vyvolat restart pomocí přerušení napájení, měl by výpadek trvat deset sekund. Tento test lze spustit jen s adresami sítí připojených přímo k zařízení (není zde podmínka k čekání na aktualizaci směrovacích tabulek).

Testovací zpráva: Výsledek by měl být formou jednoduché sady údajů, jeden pro každý typ restartu.

3 Návrh testovací aplikace

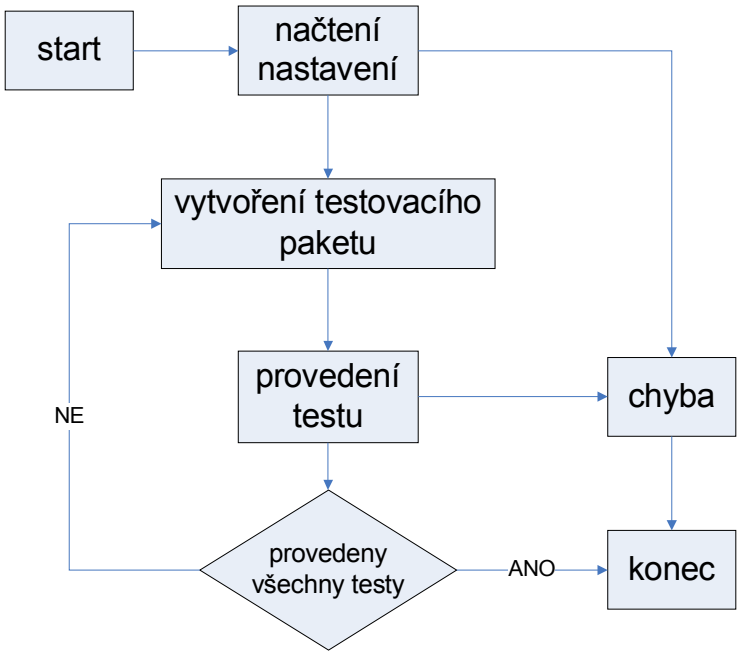
V této kapitole se věnuji návrhu testovací aplikace podle standardu RFC 2544 popsaného v předchozí kapitole.

3.1 Koncept aplikace

Nejprve bylo nutné si zvolit architekturu aplikace. Bylo možné si vybrat mezi jednou aplikací nebo rozdělením na dvě. V prvním případě by se jednalo o aplikaci pracující pouze na jednom testovacím stroji a realizující zasílání i příjem testovacích rámců. Druhé řešení umožňuje rozdělit tuto aplikaci na klienta a server. Toto řešení poskytuje větší volnost, avšak lze jej využít i na jednom testovacím stroji. Proto jsem také zvolil druhou možnost.

Dále se jednalo o to, jakým způsobem komunikovat s uživatelem. I zde jsem měl na výběr z minimálně dvou možností. Jednou je nastavení parametrů v příkazové řádce a druhou nastavení v konfiguračním souboru, což přináší jistý komfort. Vzhledem k tomu, že u serveru není nutné nastavovat příliš mnoho parametrů, byla zvolena první možnost. U klienta je tomu jinak, a proto jsem zvolil nastavení v konfiguračním souboru.

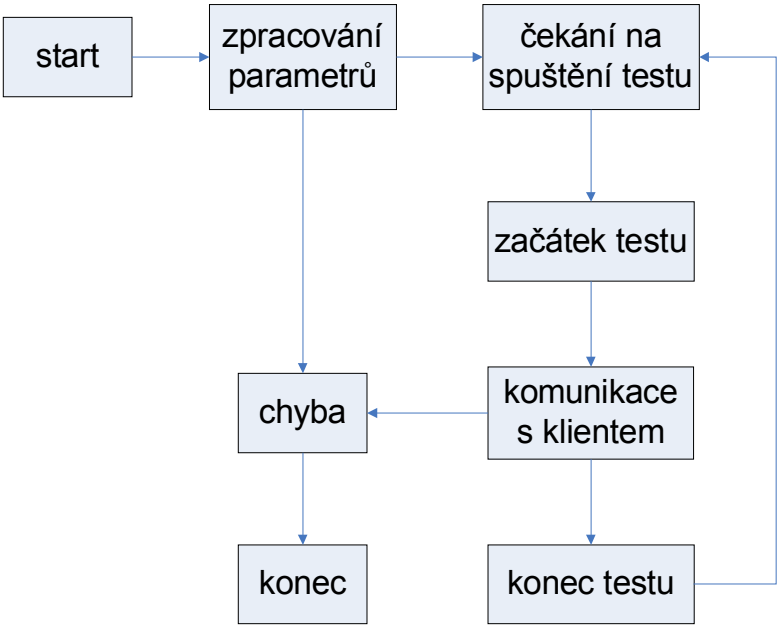
3.1.1 Klient



Klient generuje testovací pakety a provádí jednotlivé testy, jak je vidět v jednoduchém schématu. Z toho také vyplývá, že testy jsou prováděny jen jedním směrem. Načtení nastavení je provedeno z konfiguračního souboru. Jak je uvedeno v předchozí kapitole, testy mají být prováděny

pro různé velikosti rámce, proto je také v klientovi cyklus pro provedení více testů s různými velikostmi testovacích rámců (vytvořeny systémem z posílaných paketů).

3.1.2 Server



Server po spuštění zpracuje parametry z příkazové řádky a poté začíná čekat na spuštění testu klientem. Po spuštění testu přijímá testovací rámce a na konci zašle klientovi informace nutné pro vyhodnocení testu. Po ukončení testu pokračuje dále v čekání na zahájení dalšího testu, jak je vidět ve schématu. Pro ukončení serveru je mu zapotřebí zaslat signál, např. SIGTERM.

4 Implementace

Tato kapitola blíže popisuje aktuální implementaci testovacího nástroje.

4.1 Nároky na software

Pro implementaci jsem si zvolil operační systém Linux. Aplikace je naprogramována v jazyce C, používá standardní knihovny a BSD sokety. Proto by nebylo příliš těžké aplikaci upravit i pro jiné operační systémy.

Pro spuštění klienta je nutné mít práva uživatele „root“, protože používá RAW soketů.

4.2 Klient

Klient je základní částí vyvinuté testovací aplikace. Jelikož je poměrně složitý, rozdělil jsem jej do modulů. Zde uvedu jejich seznam se stručným popisem:

- `externVariables.h` – hlavičkový soubor s globálními proměnnými (např. zdrojová IP adresa, port, aj.).
- `parser.c` – stará se o načtení nastavení z konfiguračního souboru do aplikace.
- `createIPpaket.c` – obsahuje funkce pro tvorbu IP paketů.
- `tests.c` – modul obsahující funkce s jednotlivými testy.
- `client.c` – hlavní část klienta. Volá jednotlivé funkce z ostatních modulů. Kontroluje nastavení, atd.

Pro tvorbu testovacích rámců jsem si zvolil RAW sokety. U nich lze nastavit všechny hodnoty IP hlavičky i hlavičky na transportní vrstvě (viz 1. kapitola). Pro mou práci byla nejdůležitější hodnota pro zakázání fragmentace IP paketu, aby byla zachována pevná velikost rámce. Dále jsem do RAW soketu přidal UDP hlavičku pro zjednodušení práce serveru. Ten pracuje nakonec jen s UDP pakety.

Po spuštění klienta se kontroluje, zda máme dostatek práv pro tvorbu RAW soketů (práva uživatele „root“). Následuje zavolání funkce `parse()` pro zpracování nastavení z konfiguračního souboru (viz 4.2.1). Pokud uživatel v souboru nastavil prioritu aplikace vyšší než nula, tak provedu optimalizace. Ta je provedena standardní funkcí `sched_setscheduler()`, která nastaví plánování procesu, a `mlockall()`, která zamezí odkládání z operační paměti do swapu. Poté si již můžeme vytvořit dva typy soketů, již zmíněný RAW soket a UDP soket pro odpovědi od serveru. Následují úkony nutné pro provádění testů (viz 4.2.2). Nakonec, pokud došlo k optimalizaci (priorita byla vyšší než nula), navrátím prioritu procesu na původní hodnotu a ukončím otevřené sokety.

Při testech zařízení se ukázalo, že občas může dojít i ke ztrátě startovacího a ukončovacího rámce, tím se test zastaví a už neprobíhá dále. Toto by se mohlo vyřešit změnou z UDP rámců na TCP nebo nastavením maximálního času po kterou se čeká na příjem rámce a následném znovu zasláním požadavku na začátek či konec testu.

4.2.1 Zpracování konfiguračního souboru (config.dat)

V konfiguračním souboru se nacházejí důležité parametry nutné pro správnou funkci aplikace. Uživatel v něm nastaví IP adresy a porty klienta a serveru, teoretickou maximální rychlost rozhraní, jaké testy chce spustit a případně i prioritu procesu. Funkce `parse()` nalézající se v modulu parser.c pak zajistí načtení a kontrolu hodnot zadaných uživatelem. Pokud nalezne jakoukoli chybu, vrátí jednu z chybových hodnot.

Pro správnou funkci je zapotřebí dodržet formát konfiguračního souboru. Řádek začínající znakem „#“ uvozuje komentář a po něm následuje řádek s názvem proměnné a její hodnotou.

4.2.2 Provádění testů

Před spuštěním samotného testu si aplikace vytvoří dva typy rámců. Prvním typem je rámec pro oznámení startu a konce testu (`IPpaketStartEnd`) a druhým pak samotný testovací rámec o určité velikosti (`IPpaketTest`), uvedené ve standardu. Následně dojde k zavolání funkce jednoho ze čtyř implementovaných testů, a to propustnosti, zpoždění, ztrátovosti nebo back-to-back rámců. Zbylé dva testy popsané ve standardu nejsou v aplikaci naprogramovány.

Jakýkoliv test vždy začíná posláním startovacího rámce serveru a čekáním na odpověď serveru, že je připraven k provedení testu. Následně dojde k zaslání testovacích rámců. Nakonec klient zašle serveru ukončovací rámec. Server odpoví rámcem, v jehož těle se nacházejí informace o počtu přijatých rámců. V případě testu zpoždění server ihned po příjmu „označeného“ rámce, zasílá rámec informující o této události.

Standard uvádí minimální délky trvání testů, které jsou v aplikaci dopočítány z teoretické maximální rychlosti zadané uživatelem a známé délky zasílaného testovacího rámce.

Kromě testu back-to-back bylo zapotřebí ovládnout rychlost vysílání testovacích rámců. Toto jsem vyřešil způsobem, jaký je použit v programu RUDE & CRUDE [11]. V něm je toto řešeno zjištěním aktuálního času a postupným přidáváním času potřebného k poslání testovacího rámce dané velikosti určitou rychlostí, tzn. naplánováním času zaslání rámce. Na tento naplánovaný čas se pak čeká ve funkci `waitForNext()`. Snižování rychlosti se děje v procentuálních krocích.

Na začátku zdrojové souboru `tests.c` jsou uvedeny konstanty týkající se délky trvání jednotlivých testů a jejich parametrů.

V první verzi byl test zpoždění řešen zaznamenáním času po přijetí „označeného“ rámce serverem a spolu s koncovým paketem poslán tento čas, ale toto se ukázalo při testech jako nevhodné

řešení z důvodu rozcházení systémového času na dvou testovacích strojích v řádech desítek milisekund.

4.3 Server

Jak jsem již psal v části o klientovi, server pracuje jen s UDP pakety. Má celkově menší velikost než klient, jde pouze o jeden zdrojový soubor `server.c`.

Po spuštění jsou zpracovány parametry předané z příkazového řádku. Pokud žádné nejsou, využije se hodnota výchozího portu a server bude naslouchat na všech rozhraních. Poté se již server dostává do nekonečné smyčky, ve které čeká na spuštění testu klientem.

Test začíná příjmem startovacího rámce, pokračuje čtením testovacích rámců a započítáváním jejich počtu. Jediná odlišnost je u testu zpoždění, kdy se může objevit „označený“ rámec. V případě příjmu takového rámce, je ihned zaslán rámec klientovi oznamující tuto událost. Jestliže klient zašle ukončovací rámec, tak server vytvoří tělo rámce s údaji o počtu přijatých rámců, a odešle jej klientovi. Tím je test ukončen a server může opět začít čekat na začátek dalšího testu.

5 Instalace a použití

Tato kapitola popisuje instalaci a použití implementované aplikace.

5.1 Instalace

Instalace klienta i serveru je jednoduchá. Obě aplikace jsou ve formě zdrojových textů, a proto je nutné je nejprve před použitím zkompilovat. Ke kompilaci je zapotřebí standardní vývojové prostředí jazyka C (gcc v4.0, make v3, …) operačního systému Linux. Kompilaci spustíte příkazem `make` v příslušném adresáři klienta nebo serveru. Poté je vytvořen spustitelný soubor `client` v případě klienta nebo `server` v případě serveru. Tím je instalace dokončena.

5.2 Použití

Aplikace je plně automatizovaná, stačí ji jen před spuštěním nastavit. Testy trvají řádově desítky minut až hodiny.

5.2.1 Server

Pro spuštění serveru je možné využít tři způsoby:

1. bez parametrů – server bude naslouchat na portu 5000 na všech rozhraních
2. s jedním parametrem PORT – server bude naslouchat na portu PORT na všech rozhraních
3. se dvěmi parametry PORT a ADRESA – server bude naslouchat na portu PORT na rozhraní s IP adresou ADRESA

Jako příklad uvedu spuštění serveru na portu 5555

```
# ./server 5555
```

nebo na portu 6666 a rozhraní s IP adresou 192.168.1.1.

```
# ./server 6666 192.168.1.1
```

5.2.2 Klient

Nejprve je potřeba si v některém z dostupných textových editorů otevřít konfigurační soubor `config.dat`, jehož obsah je zobrazen na následujícím obrázku.

```
# zdrojovy port
srcPort = 5001

# zdrojova IP adresa
srcIP = 172.16.2.1

# cilovy port
dstPort = 5004

# cilova IP adresa
dstIP = 172.16.2.3

# maximalni teoreticka rychlost testovaneho zarizeni [Mbit/s]
bandwidth = 10

# typy testu:
# 1 - propustnost
# 2 - zpozdeni
# 3 - ztratovost
# 4 - vyrovnavaci pamet zarizeni (test back-to-back ramcu)
# priklad: - provedeni testu zpozdeni + ztratovosti = 23
#          - provedeni testu propustnosti + vyrovnavaci pamet zarizeni + ztratovost = 143
test = 2

# priorita procesu (0 - 90, 0 nejnizsi)
priority = 0
```

Zdrojový port a IP adresa se vážou k testovacímu zařízení, na kterém bude spuštěn klient. Cílový port a IP adresa jsou parametry spuštěného serveru. Bandwidth určuje maximální teoretickou rychlost v Mb/s. Dále je zde možno nastavit k provedení jeden až čtyři nabízené testy. Posledním parametrem je priorita procesu klient, která je pro jistou optimalizaci přesnosti měření (popsána v kapitole 4).

Po nastavení všech parametrů klienta spustíte příkazem `./client` z příkazové řádky. Po spuštění již klient pracuje automaticky. Výsledky testů jsou vypisovány průběžně na standardní výstup (obrazovku).

Výstup testů je určen pro čtení člověkem. Výpis začíná oznámením o jaký test se jedná a hodnoty testu, např. velikost rámce, počet vyslaných rámců a rychlost vysílání v procentech uživatelem zadané teoretické maximální rychlosti. Dále jsou uvedeny informace o zaslání a příjmu startovacího a ukončovacího rámce. Následuje výsledek dílčího testu. Na konci všech dílčích testů jednoho testu (např. testu zpoždění) je koncová naměřená hodnota. V následujících obrázcích jsou ukázky výstupu z jednotlivých testů.

```
test frameloss
ramce: velikost 64B, pocet vyslanych 2048000, rychlost vysilani 100%
start send...receive
end send...receive
pocet prijatych ramcu 1980901
ztrata 3.2763%
-----------------------------------------------------------------
test frameloss
ramce: velikost 64B, pocet vyslanych 2048000, rychlost vysilani 90%
start send...receive
end send...receive
pocet prijatych ramcu 1995704
ztrata 2.5535%
-----------------------------------------------------------------
test frameloss
ramce: velikost 64B, pocet vyslanych 2048000, rychlost vysilani 80%
start send...receive
end send...receive
pocet prijatych ramcu 1995204
ztrata 2.5779%
```

```
test throughput
ramce: velikost 1518B, pocet vyslanych 86340, tolerance ztraty 18, rychlost vysilani 100%
start send...receive
end send...receive
pocet prijatych ramcu 86340
10.634545 sec, 8118.824078 ramcu/s, 12035.522412 kB/s, 94.027519 Mb/s
--------------------------------------------------------------------------------
Vysledna hodnota propustnosti
10.634545 sec, 8118.824078 ramcu/s, 12035.522412 kB/s, 94.027519 Mb/s
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
test latency
ramce: velikost 1518B, pocet vyslanych 86340, 1. dilci test
start send...receive
end send...receive
zpozdeni 3.8750 msec
--------------------------------------------------------------------------------
test latency
ramce: velikost 1518B, pocet vyslanych 86340, 2. dilci test
start send...receive
end send...receive
zpozdeni 2.6480 msec
--------------------------------------------------------------------------------
test latency
ramce: velikost 1518B, pocet vyslanych 86340, 3. dilci test
start send...receive
end send...receive
zpozdeni 3.9875 msec
--------------------------------------------------------------------------------
test latency
ramce: velikost 1518B, pocet vyslanych 86340, 4. dilci test
start send...receive
end send...receive
zpozdeni 2.2985 msec
--------------------------------------------------------------------------------
test latency
ramce: velikost 1518B, pocet vyslanych 86340, 5. dilci test
start send...receive
end send...receive
zpozdeni 3.4655 msec
--------------------------------------------------------------------------------
Vysledna hodnota zpozdeni
prumer 3.2549 msec
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

```
test back-to-back
ramce: velikost 64B, pocet vyslanych 409613
start send...receive
end send...receive
pocet 409613, prijato 409061
--------------------------------------------------------------------------------
test back-to-back
ramce: velikost 64B, pocet vyslanych 409607
start send...receive
end send...receive
pocet 409607, prijato 409059
--------------------------------------------------------------------------------
maximalni pocet ramcu ve shluku beze ztraty 409604
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
test back-to-back
ramce: velikost 128B, pocet vyslanych 204800
start send...receive
end send...receive
pocet 204800, prijato 204282
--------------------------------------------------------------------------------
podle standardu RFC2544 nelze urcit maximalni pocet ramcu ve shluku
--------------------------------------------------------------------------------
```

6 Testy zařízení

V úvodu této kapitoly se věnuji návrhu testů pro různé aktivní síťové prvky. Navržené testy jsou pro implementovanou aplikaci. Slovo test se v této části nevztahuje k V závěru ukáži reálné výsledky navržených testů na jednom ze zařízení (přepínači).

6.1 Přepínač

Přepínač (switch) je zařízení pracující na linkové vrstvě ISO/OSI modelu. Přepínač pracuje s rámci a MAC adresami, podle kterých rozeznává, na jaký výstupní port má příchozí rámce zasílat. Pokud na vstupu dostane broadcastový rámec, tak jej zašle na všechny výstupní porty. Dále lze porty přepínače rozdělit do virtuálních sítí (VLAN), kde pro propojení mezi nimi je zapotřebí inteligentnějšího zařízení, směrovače.

V prvním testu jde o naměření hodnot, kdy je zařízení bez zatížení. Tento test nám ukáže, jaké maximální výkony zařízení dokáže dosáhnout.

V druhém testu se jedná o zjištění toho, zda zařízení pracuje v režimu full nebo half duplex, tzn. jestli jeho výkon nějak ovlivňuje obousměrný provoz na jednom portu. K tomuto je zapotřebí spustit klienta i server dvakrát, na každém ze dvou testovacích zařízení.

V dalším testu se zaměřuji na zjištění, jak se zařízení bude chovat při plném zatížení na dvou nezávislých linkách. Toto nám ukáže, zda a jak ovlivňují dva nezávislé toky dat výkonnost zařízení. Teoreticky by se nemělo téměř žádné ovlivnění projevit, vyplývá to z funkce přepínače. Tento test lze modifikovat s použitím VLAN, kdy jednu linku zapojíme do jiné VLAN než tu druhou. Pak zařízení bude více zatěžovat kontrola, zda se jedná o rámce směřující do stejné VLAN nebo do jiné.

Dalším velice zajímavým testem je zjištění práce přepínače při zasílání rámců ze dvou vstupních portů na jeden výstupní. Zde můžeme sledovat, jak se zařízení zachová, když dvojnásobný maximální tok rámců budeme zasílat na port, který má poloviční kapacitu.

Všechny uvedené testy lze doplnit o zasílání broadcastových rámců jednou za určitý časový úsek. K tomuto ovšem již nelze využít implementovanou aplikaci, ale poslouží k tomu velmi dobře některý z generátorů rámců (např. generátor vyvinutý v rámci diplomové práce Matěje Konečného [12]).

6.2 Směrovač

Směrovač (router) pracuje na síťové vrstvě ISO/OSI modelu, tedy ne s rámci, ale s IP pakety. Stará se o doručení IP paketů podle uvedené cílové IP adresy, přesněji podle síťové části z IP adresy. Pro zajištění této práce se využívají směrovací tabulky, které mohou být tvořeny staticky nebo

dynamicky. V případě dynamické tvorby se jedná o použití některého ze směrovacích protokolů (např. RIP, OSPF, …). Dále směrovače dokážou překládat veřejné IP adresy na privátní pomocí NATu a ještě mají možnost nastavení filtrování provozu, tzv. seznamy pro řízení přístupu (ACL, access control list).

Tak jako u přepínače i zde se první test provede bez zatížení.

Druhý test se zaměřuje na chování zařízení při použití nadefinovaných ACL listů. Je možné např. testovací provoz povolit až na několikáté pozici v ACL. Tímto se zjistí, jaké zpoždění může nastat. Doporučoval bych provedení pro různý počet ACL.

V třetím testu se jedná o zjištění výkonu při zapnutém statickém či dynamickém směrování. Tento test lze upravit tak, že během probíhajícího testu se odpojí některé rozhraní. Tím dojde k nutnosti úpravy směrovací tabulky, což určitým způsobem zatíží směrovač.

Další test je zaměřen na zatížení směrovače při použití překladu adres (NATu).

Kromě prvního testu by mělo stačit pro otestování zařízení, využití pouze testů propustnosti a zpoždění.

6.3 Test přepínače Cisco Catalyst 2950

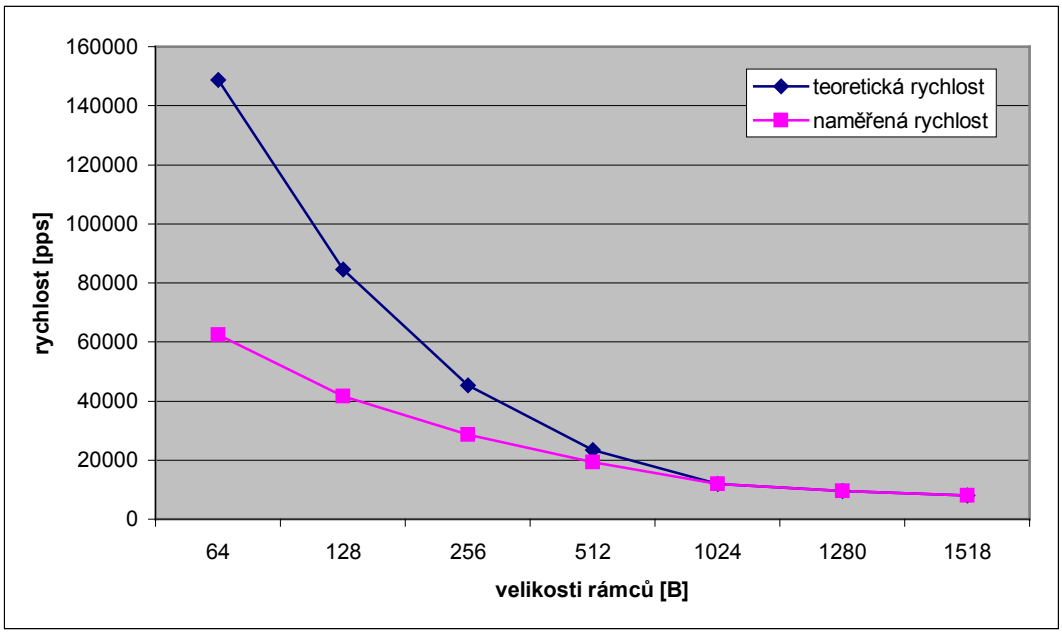
Před provedením testu jsem si nejprve zvolil formu zapojení přepínače. Následně jsem propojil dva počítače přes testovaný přepínač. V obou počítačích byly nainstalovány stejné 100Mb ethernetové síťové karty. Na přepínači bylo využito 10/100Mb portů. Všechny výsledky lze nalézt na přiloženém CD.

Pro výpočet teoretické rychlosti jsem použil následující vzorec.

$$\text{teoretická rychlost rámce o velikosti X}[\text{b}] = \frac{\text{teoretická rychlost zařízení}[\text{bps}]}{20 + \text{X}}$$

Podle výše navržených testů jsem provedl nejdříve test bez zatížení. V testu propustnosti se ukázalo, že testovaný přepínač je navržen a optimalizován pro rámce větších velikostí (1024, 1280 a 1518 bytů), při nichž docházelo k nejvyšším naměřeným hodnotám. Souhrnné výsledky přehledně ukazuje následující tabulka a graf.

velikost rámce [B]	teoretická rychlost [pps]	naměřená rychlost [pps]
64	148800	62499
128	84450	41666
256	45280	28571
512	23490	19230
1024	11970	11973
1280	9610	9615
1518	8120	8129



Naopak test zpoždění pro rámce malé velikosti vycházel lépe, což je logické. Přenos většího rámce totiž trvá déle. V následující tabulce naměřených hodnot sloupec „rychlost“ udává rychlost vysílání testovacích rámců.

velikost rámce [B]	rychlost [pps]	zpoždění [ms]
64	62499	0,0468
128	41666	0,0513
256	28571	0,0621
512	19230	0,0835
1024	11973	3,4626
1280	9615	3,5937
1518	8129	2,4956

Na měřená ztrátovost je v následující tabulce. Ztráta 20% rámců velikosti 64B je způsobena nejspíš chybou v zařízení. V jiném přepínači stejného typu se takto vysoká hodnota nevyskytovala.

rychlost vysílání [%]	ztrátovost [%] pro velikosti rámců [B]						
	64	128	256	512	1024	1280	1518
10	20,6170	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
20	0,2623	0,0114	0,0000	0,0000	0,0000	0,0000	0,0000
30	1,6491	0,0122	0,0047	0,0242	0,0000	0,0000	0,0000
40	2,2696	0,0189	0,0107	0,0000	0,0000	0,0000	0,0000
50	2,6108	0,0271	0,0160	0,0043	0,0000	0,0000	0,0000
60	2,7021	0,0336	0,0230	0,0102	0,0000	0,0000	0,0000
70	2,8599	0,0489	0,0311	0,0146	0,0000	0,0000	0,0000
80	2,5779	0,0520	0,0375	0,0238	0,0000	0,0000	0,0000
90	2,5535	0,0519	0,0439	0,0285	0,0031	0,0000	0,0000
100	3,2763	0,0063	0,0000	0,0000	0,0000	0,0000	0,0000

Test back-to-back rámců prokázal, že v případě malých velikostí rámců je nepoužitelný, jelikož standardem definovaný test musí trvat nejméně dvě sekundy a nesmí se při něm ztratit ani jeden rámec. Naopak u velikosti rámce 1280 nebo 1518 bytů tento shluk back-to-back rámců dosahuje vysokých hodnot.

Z výše popsaných výsledků, vyplývá, že shluky rámců malé velikosti jsou pro zařízení problémem.

Tímto se dostávám k dalším testům. Jedním z nich je test, při kterém se měřilo jak ovlivní výše naměřené hodnoty obousměrný provoz. Zde jsem narazil na problém, že se ztráceli ukončovací rámce. Z tohoto důvodu nešlo tento test úspěšně dokončit. Navržené řešení je v kapitole 4.

Dalším navrženým testem bylo zjištění chování přepínače při zasílání rámců na dva vstupní porty s přeposíláním na jeden. Zde se přepínač zachoval spravedlivě. Propustnost na vstupních portech poklesla na polovinu. I zde však test je ovlivněn výše zmíněným problémem. Výsledek se mi podařilo získat jen pro 1518 bytové rámce.

Poslední sadou testů jsem se zaměřil na porovnání provozu na dvou nezávislých linkách. K ovlivnění dochází jako obvykle u rámců o malé velikosti.

velikost rámce [B]	2 nezávislé linky - rychlost [pps]			
	bez VLAN		s VLAN	
	1.	2.	1.	2.
64	86429	86508	86433	86418
128	41666	41666	79614	41666
256	31249	25641	45217	25641
512	22222	19230	23483	19230
1024	11970	11969	11969	11971
1280	9614	9614	9613	9613
1518	8129	8128	8129	8127

7 Závěr

Cílem této práce bylo vytvoření testovací aplikace pro aktivní síťové prvky podle standardu RFC 2544. Před samotným návrhem a implementací však bylo nutné se nejprve seznámit s uvedeným standardem. Tomu jsem se věnoval v rámci teoretické části práce.

Navržená a implementovaná aplikace se dle mých výsledků jeví jako vhodný nástroj pro testování aktivních síťových prvků. Snažil jsem se také přispět svojí prací v oblasti volně dostupných testovacích aplikací, kde je podle mého názoru prostor. Mnou navrženou testovací aplikaci lze nadále rozšiřovat, navrhoval bych úpravu výstupu testů do formátu pro zpracování programy určenými pro tvorbu tabulek nebo grafů.

Literatura

[1] IEEE 802.3 Working Group: IEEE 802.3 CSMA/CD (ETHERNET) [on-line]. Poslední modifikace: 25 March 2007 [cit. 2007-04-30] Dostupné na URL: http://www.ieee802.org/3/.

[2] Postel, J.: Internet Protocol. RFC 791 (Standard), Září 1981. Dostupné na URL: http://www.ietf.org/rfc/rfc791.txt.

[3] Nichols, K, et. al.: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), Prosinec 1998, updated by RFCs 3168, 3260. Dostupné na URL: http://www.ietf.org/rfc/rfc2474.txt.

[4] Protocol numbers [on-line]. Poslední modifikace: 12 February 2007 [cit. 2007-04-30]. Dostupný na URL: http://www.iana.org/assignments/protocol-numbers.

[5] Postel, J.: User Datagram Protocol. RFC 768 (Standard), Srpen 1980. Dostupné na URL: http://www.ietf.org/rfc/rfc768.txt.

[6] Benchmarking Methodology (bmwg) [on-line]. Poslední modifikace: 18 December 2006 [cit. 2007-05-02]. Dostupný na URL: http://www.ietf.org/html.charters/bmwg-charter.html.

[7] Bradner, S.: Benchmarking terminology for network interconnection devices. RFC 1242 (Informational), Červenec 1991. Dostupné na URL: http://www.ietf.org/rfc/rfc1242.txt.

[8] Bradner, S.; McQuaid, J.: Benchmarking Methodology for Network Interconnect Devices. RFC 2544 (Informational), Březen 1999. Dostupné na URL: http://www.ietf.org/rfc/rfc2544.txt.

[9] Bradner, S.; McQuaid, J.: Benchmarking Methodology for Network Interconnect Devices. RFC 1944 (Informational), Květen 1996. Dostupné na URL: http://www.ietf.org/rfc/rfc1944.txt.

[10] Johnson, S.: Komentář k RFC 2544 [on-line]. Poslední modifikace: 1 February 2005 [cit. 2007-04-30] Dostupné na URL: http://www.faqs.org/qa/rfcc-1646.html.

[11] Laine, J.: program RUDE & CRUDE. Poslední modifikace: 13 August 2002 [cit. 2007-04-30] Dostupné na URL: http://rude.sourceforge.net/.

[12] Konečný, M: Tool for generating network traffic with GUI. 2006. Diplomová práce. FIT VUT.

[13] Bartoň, J.: Performance Testing Tools [on-line]. Poslední modifikace: 30 October 2003 [cit. 2007-04-30] Dostupné na URL: http://www.cesnet.cz/doc/techzpravy/2003/perftools/.

Seznam příloh

Příloha 1. CD