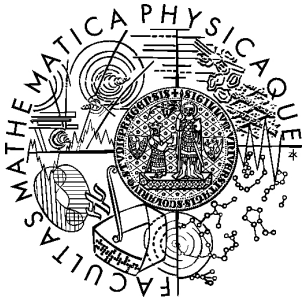
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Vladimír Vyskočil

Systém pro monitorování kvality připojení

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka

Studijní program: Informatika

2010

Děkuji RNDr. Ing. Jiřímu Peterkovi za ochotu a vynaložený čas při vedení mé diplomové práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 4. srpna 2010 Vladimír Vyskočil

Obsah

1 Úvod **6**

1.1 Cíle práce 8

1.2 Struktura práce 8

1.3 Dostupný software 9

1.3.1 Nedostatky dosavadních řešení 11

2 Analýza problému **13**

2.1 Charakteristiky kvality služby 14

2.1.1 Nestandardní hodnoty a jejich příčiny 15

2.2 Modelová síť 16

2.3 Protokoly modelové sítě 17

2.3.1 Ethernet 20

2.3.2 IP 20

2.3.3 TCP 23

2.4 Návrh metodiky měření 31

2.4.1 Výběr vhodných paketů pro měření 32

2.4.2 Porovnávání naměřených hodnot 35

2.4.3 Detekce ztráty paketů 36

2.5 Srovnání se současnými metodami měření 38

3 Implementace řešení **41**

3.1 Popis modulů 41

3.1.1 Hlavní program 42

3.1.2 Správa probíhajících spojení 44
3.1.3 Statistický modul pro zpracování naměřených hodnot z probíhajících a již ukončených spojení 47
3.1.4 Ukládání naměřených hodnot na disk 49
3.1.5 Agregace a analýza dat 50
3.2 Argumenty programu 52
3.3 Instalace monitorovací aplikace 54

4 Výsledky **56**
4.1 Ověření předpokladů 56
4.1.1 Zpracování odpovědi na ICMP echo paket 56
4.1.2 Zpracování odpovědi na SYN paket 58
4.1.3 Zpracování odpovědi na FIN paket 59
4.1.4 Vliv zpožděných ACK 60
4.2 Srovnání s aplikací ping 62
4.3 Stahování velkého souboru 65
4.4 Praktický test na síti 69

5 Závěr **72**
5.1 Možné rozšíření programu 72

Literatura **74**

Název práce: Systém pro monitorování kvality připojení
Autor: Vladimír Vyskočil
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka
e-mail vedoucího: peterka@ksi.mff.cuni.cz

Abstrakt: Cílem diplomové práce je navrhnut metodiku centralizovaného monitorování datového provozu u poskytovatele připojení k internetu, které by umožnilo detekovat přípojky vykazující problémy s kvalitou připojení a hodnotit míru těchto problémů. Metodika by minimálně měla sledovat a vyhodnocovat latenci a ztrátovost paketů mezi centrálně umístěným monitorovacím bodem v síti poskytovatele a počítači jednotlivých uživatelů. Dále je úkolem takto navrženou metodiku implementovat, ověřit a výsledky vyhodnotit.
Klíčová slova: latence, ztrátovost paketů, měření, kvalita služby, konektivita

Title: A system for monitoring the quality of connectivity
Author: Vladimír Vyskočil
Department: Department of Software Engineering
Supervisor: RNDr. Ing. Jiří Peterka
Supervisor's e-mail address: peterka@ksi.mff.cuni.cz

Abstract: The aim of the theses is to design a methodology for centralized monitoring of data traffic in an ISP network that would detect individual connection with connectivity problems, and evaluate these problems. The methodology should at least evaluate packet loss and latency between a centrally located monitoring point and endpoints at user premises. A further goal is to implement this methodology, test it and evaluate the results.
Keywords: latency, packet loss, measurements, quality of service, connectivity

Kapitola 1

Úvod

V posledních několika letech rozšíření bezdrátových sítí prošlo explozivním vývojem. Původní návrh standardu 802.11 počítal s nasazením těchto sítí především pro snadné propojení počítačů v kanceláři nebo domácnostech. Nejčastějším řešením mělo být přivedení internetového připojení na místo určení pomocí kabelových rozvodů (DSL, kabelová televize, optické kabely, ...). Teprve pak měla přijít ke slovu bezdrátová technologie v podobě přístupového bodu pokrývajícího všesměrovými anténami vnitřek budovy. Výhodou tohoto modelu je bezesporu přivedení internetové konektivity pomocí technologie poskytující minimální výpadky (např. kabel). Bezdrátové připojení tak zprostředkovává jen poslední spoj a jistá nespolehlivost je zde tolerována. Kratší výpadky tedy nevyžadují nutnou pozornost správce sítě, často stačí přesunout notebook na místo s lepším signálem, nebo počkat než zmizí zdroj rušení.

Česká republika má nicméně trochu jiný způsob využití bezdrátových sítí, kromě výše zmíněného se používají také pro rozvod internetu do domácností. Značný počet poskytovatelů připojení tohoto druhu je lokálním specifikem. Existující sítě jsou tvořeny kombinací mnoha technologií, často využívají různé radiové frekvence k přenosu dat. Kvalita takových spojů je závislá na místních podmínkách (viditelnost mezi body, přítomnost rušení od ostatních radiových zdrojů, vzdálenost mezi body, vysílací výkon, ...) a významně také na použité technologii. Velmi rozšířené je použití zařízení standardu 802.11 k poskytování připojení poslední míle k uživatelům

nebo k vzájemnému propojení routerů. Vůbec se však nejedná o nasazení, které bylo zamýšleno při jejím návrhu. Mezi hlavní nevýhody patří neexistence podpory řízení kvality přenosu dat přímo na fyzické vrstvě, neefektivní řešení problému skrytých stanic a nevhodné použití protokolu TCP pro přenos dat.

První problém měl být vyřešen standardem 802.11H, zajištěním podpory kvality přenosu dat. Jeho schválení se však dosud zdá být v nedohlednu, proto byla pro rychlé vyřešení problému přijata jen jeho podmožina s označením WMM. Hlavním důvodem tohoto kroku byl rozmach streamování audia a video přes internet.

Druhý problém zatím není ve standardu vyřešen, pravděpodobně zde není dostatečný tlak na výrobce. Projevuje se totiž především při připojení více stanic na větší vzdálenosti a hlavně při použití směrových antén. Existuje několik proprietárních řešení (časový multiplex TDMA), která ale nejsou součástí standardu.

Třetí problém je odbornou veřejností mnohdy zcela opomíjen, třebaže studie poukazující na nevýhody použití protokolu TCP pro bezdrátový přenos existují (např. [1] a [2]). Některé velké společnosti jako AOL dokonce provozují vlastní protokol, který zabaluje vhodně TCP pakety do UDP datagramů tak, aby minimalizoval nevýhody prvně jmenovaného protokolu. Tou hlavní je, že v případě výpadku paketu protokol TCP předpokládá přetížení některého ze spojů mezi stanicemi. V bezdrátové komunikaci je ale výpadek často způsobený spíše působením rušení nebo tím, že více stanic komunikuje zároveň (problém skryté stanice). Reakce na ztrátu paketu záleží značně na implementaci v operačním systému a v detailech není součástí protokolu TCP, nejčastěji jde ale o snížení přenosové rychlosti na polovinu a velmi pomalý opětovný návrat k vyšším rychlostem. I krátkodobý výpadek tak dovede na relativně delší časový interval snížit podstatně propustnost TCP spojení.

Jelikož dosud tyto tři oblasti nejsou uspokojivě řešeny, nabývá na důležitosti otázka měření kvality služeb poskytovaných zákazníkům. Obecně není takový úkol zcela triviální, jelikož kvalita připojení závislá na více faktorech, jměnovitě vzdálenosti, přímé viditelnosti a rušení. V sítích, kde se stanice může pohybovat, lze kvalitu měřit, ale interpretace výsledků je obtížná (uživatel mohl být schovaný za překážkou, která bránila průchodu signálu). Naštěstí v případě u nás tak častém, kdy se

jedná o stabilní spoj sestávající se ze směrových antén, jsou vzdálenost a viditelnost konstantní (anténa je upevněna na fasádě domu). Tím je celá věc zjednodušena.

Na druhou stranu však je monitoring stavu bezdrátové sítě náročnější, než jak je tomu v případě přenosu signálu po kabelu. Takové sítě totiž většinou buď fungují bezchybně, anebo nefungují vůbec, což je situace snadno identifikovatelná. Zato výkony bezdrátových sítí se mohou pohybovat téměř kdekoliv mezi těmito dvěma krajními možnostmi. Dále také zhoršení kvality služby může být způsobeno buď nahodilým krátkodobým rušením, nebo může jít o dlouhodobý trend. Proto je nutné vytvořit systém měření a analýzy, který dokáže možné problémy s kvalitou připojení dostatečně spolehlivě rozeznat a zároveň, aby bylo možné rozlišit krátkodobé náhodné výpadky od vážnějších poruch, bude moci být nasazen soustavně.

1.1 Cíle práce

Cílem této práce je analýza možností měření kvality připojení stanic neinvazivním způsobem a návrh monitorovací aplikace, která tento způsob implementuje. Řešení má být obecné a nezávislé na použitém bezdrátovém hardware. Výstupem analýzy je nová metoda a software, který tuto metodu aplikuje při měření. Software má přístup k datové komunikace mezi uživateli sítě a internetem, z nichž získává informace o kvalitě jednotlivých spojení. Implementace, jež ověřuje závěry analýzy, je provedena v jazyce Python.

1.2 Struktura práce

Struktura této práce je následující. Příští kapitola seznamuje čtenáře s aktuálně dostupným softwarem, který lze použít k monitorování stavu sítě. Dále jsou uvedeny nedostatky současných řešení. V kapitole „Analýza problému“ je shrnuto, jak vypadá modelová síť, pro kterou se navrhuje potřebný software. Poté jsou definovány charakteristiky kvality služby, které bude program sledovat. Následuje nutný úvod do protokolů IP a TCP, neboť jsou pro pochopení nové metody zásadní. Pokračuje se

vysvětlením metody samotné a nastíněním jejích výhod a nevýhod. Kapitola „Popis implementace řešení“ obsahuje detaily věnující se implementaci nové metody. Dále jsou pak ověřeny teoretické předpoklady na skutečném hardware a provedeno srovnání nové metody měření latence s metodou příkazu ping. V závěru je zhodnocen výsledek celé práce a navržené softwarové řešení.

1.3 Dostupný software

Hlavním požadavkem na dostupný software je, aby byl schopen monitorovat základní charakteristiky kvality připojení cílové stanice (síťová latence, ztráta paketů) v delších časových intervalech. Dále by měl získaná data analyzovat a upozorňovat na stanice se špatnou kvalitou služby. Celé řešení by mělo být schopno provozu na obecné síti poskytovatele internetu, a tedy řešení využívající protokolu SNMP není přípustné (některá zařízení jej nemusí podporovat).

Zabbix Tento nástroj[1] umožňuje monitorování sítě po dlouhou dobu a zobrazení měřených veličin do grafů. Aplikace je velmi robustní a dokáže testovat několik stovek tisíc stanic pomocí velkého počtu možných měření. Je například možné testovat dostupnost služby HTTP, FTP a dalších. Měření latence provádí buď pomocí příkazu ping nebo pomocí TCP pingu. Rozdíl mezi oběma metodami je ten, že TCP ping měří odezvu mezi vysláním TCP SYN paketu a jeho potvrzením. To umožnuje získat data i pro stanice, kde je protokol ICMP (běžný ping) filtrován. Kromě měření aplikace také dokáže zajistit zasílání upozornění v případě výpadku některé služby, nebo zhoršení její kvality. Ukázka rozhraní systému Zabbix je na obrázku 1.1.

Cacti Nástroj Cacti[2] je velmi podobný předchozímu, nicméně nepodporuje delegaci testů na další agenty a nedovede tak měřit velké množství stanic. Nabízí také vytváření grafů. Notifikace v případě výpadků nebo zhoršení kvality služby jsou jen velmi omezené. Stejně jako v předchozím případě je potřeba pro sledování stanice

[1] Více informací na *www.zabbix.com*.
[2] Viz také *www.cacti.net*.

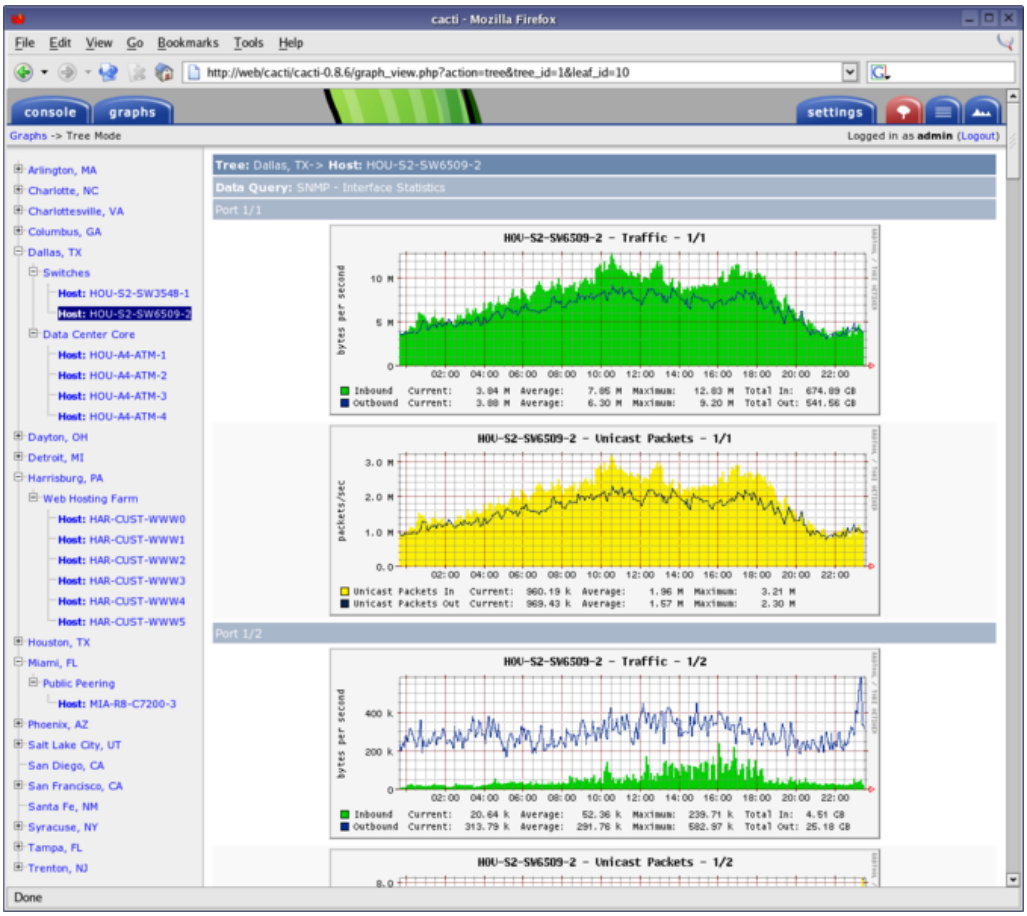
Obrázek 1.1: Rozhraní aplikace Zabbix

nejdříve provést konfiguraci. Ukázka webového rozhraní Cacti je na obrázku 1.2.

Nagios Tento software[3] v základním nastavení provádí pouze sledování dostupnosti stanic pomocí různých testů. Pro zobrazení grafů je potřeba instalovat rozšíření Nagiosgraph. Nagios je více zaměřen na krátkodobá měření a hlášení výpadku dostupnosti, Cacti a Zabbix pro změnu více sledují dlouhodobější vývoj služby.

Munin, PRTG, Argus, Orion,.. V oblasti monitorovacího software pro sítě existuje mnoho dalších systémů. Cacti a Zabbix patří aktuálně neaktivnější projekty. Ostatní nabízejí často méně obecné použití, nebo jsou to pouze moduly, z kterých je celý systém potřeba sestavit. Kompletní srovnání monitorovacího software lze nalézt zde [3].

[3]Domovské stránky *www.nagios.org*.



Obrázek 1.2: Rozhraní aplikace Cacti

1.3.1 Nedostatky dosavadních řešení

Všechny předchozí systémy provádějí aktivní testování cílové stanice. Pro malý objem generovaného provozu mohou být výsledky nepřesné (síť se může chovat jinak k malým a velkým paketům), zatímco při velkém objemu dochází k ovlivnění měření právě tímto provozem. ICMP pakety mohou být také filtrovány firewallem cílové stanice a měření tedy nemusí být možné. Většina provozu na internetu je tvořena protokolem TCP, který je zásadně odlišný od protokolu ICMP. Měření často ale probíhají právě pomocí protokolu ICMP, který se chová jinak (například TCP provoz toleruje za určitých podmínek ztrátu potvrzujících paketů, protokol ICMP nikoliv), měření tak nemusí věrně zobrazovat stav sítě. Dále také žádná ze zmíněných aplikací neposkytuje sledování aktuálních datových přenosů cílové stanice. Charakteristiky

kvality služby (např. síťová latence) jsou ale závislé na aktuální rychlosti přenosu dat.

Zmíněné nedostatky plynou především z obecnosti uvedených softwarových balíku. Předpoklad, že software má přístup k veškeré komunikaci mezi vnitřní sítí a internetem je příliš silný, ne vždy však splněný. Proto uvedená řešení nenabízejí možnost takovou komunikaci analyzovat. V následujících kapitolách je ale ukázáno, že využití těchto dat k měření může poskytnout velmi zajímavé výsledky.

Kapitola 2

Analýza problému

Navrhovaný software bude mít k dispozici pouze omezené zdroje, což si vynucuje rozdělit sběr a analýzu dat na dva samostatné úkoly. Díky tomu ani velké výpočetní zatížení vyvolané nároky vyhodnocování paketů nezaviní zpoždění provozu.

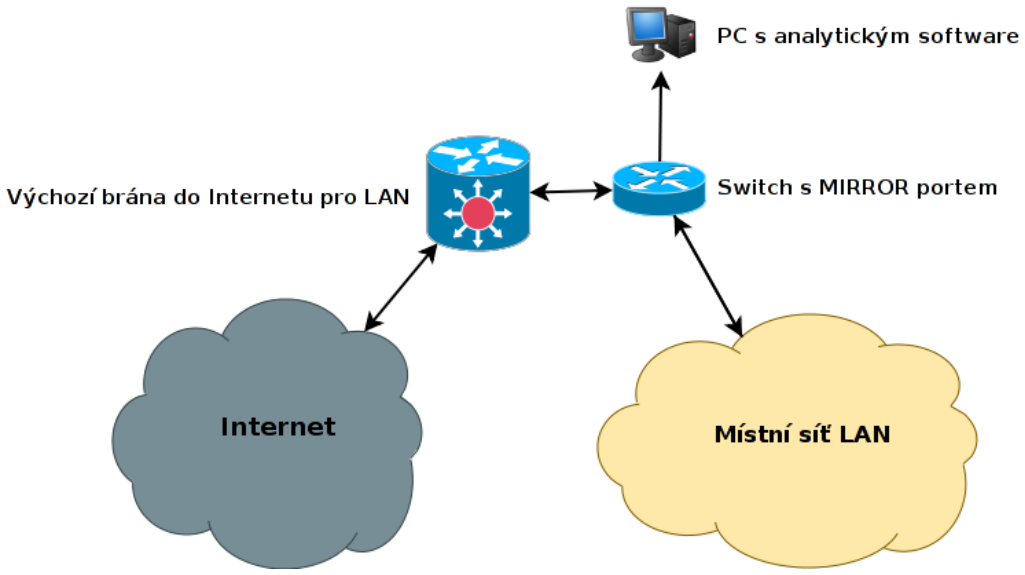
V síti musí být umístěn hardware schopný zachytávat průchozí data a tato data odesílat pro zpracování na počítač s nainstalovanou monitorovací aplikací. Některé switche mají speciální port (tzv. mirror port), na který jsou odeslány všechny pakety, které prošly předem specifikovaným rozhraním. Výhoda takového řešení tkví především v tom, že switche s touto vlastností se mohou chovat transparentně a u průchozího provozu zvýší latenci pouze tak, jako by jej zvýšil obyčejný switch. Další důležitou výhodou je, že tato zařízení mají velmi stabilní chod. Výpadek zařízení, přes které prochází veškerý provoz do internetu, by totiž snadno mohl způsobit nedostupnost připojení v celé síti. Pravděpodobnost chyby switche je ale řádově nižší než u monitorovací stanice se složitým operačním systémem.

Důležité je také umístění PC s vyhodnocovacím software co nejblíže k přeposílajícímu switchi. Jestliže máme k dispozici pouze jednoduchý switch, který přeposílá pakety a dále je nedoplňuje o hlavičku s datem detekce, tak je to dokonce nutnost. V případě umístění počítače na vzdálenější místa v síti by totiž docházelo k výraznému zkreslení dat vlivem dodatečné časové latence a ztráty paketů způsobené linkami mezi switchem zachytávajícím pakety a počítačem, který je zpracovává. Nepříliš doporučovanou alternativou je switch přeposílající pakety pomocí protokolu

TaZmen, který umožňuje zachycená data zabalit do protokolu UDP a poslat na libovolnou adresu v síti. Při tomto řešení získáváme flexibilitu v umísťování monitorovací stanice, nicméně protokol UDP je nespolehlivý a ztrátu UDP datagramu bychom tak mohli interpretovat jako ztrátu TCP paketu, který v něm byl zabalen.

Nyní je tedy testovací prostředí připravené. Sestává se ze switche, který provádí přeposílání průchozích dat na svůj speciální port. K tomuto portu je pomocí ethernetového kabelu připojen počítač s nainstalovaným software pro analýzu paketů. Umístění této sestavy závisí na části sítě, kterou chceme měřit. Musí být totiž takové, aby rozdělovala počítačovou síť na dvě oblasti. Na tu která podléhá měření a na zbytek. Pro internetové poskytovatele tak vychází vhodné umístění poblíž výchozí brány do internetu. Právě i z tohoto důvodu je vhodné pro zachytávání paketů použít zařízení s minimální pravděpodobností výpadku.

Schéma zapojení je znázorněno na následujícím obrázku.



Obrázek 2.1: Schéma zapojení

2.1 Charakteristiky kvality služby

Předtím, než se pustíme do hledání vhodných prostředků pro měření, bychom se měli zamyslet důkladněji nad cíli měření. Kvalitu připojení stanice je možné vyjádřit

pomocí následujících hodnot:

- Doba potřebná k přenosu paketu po síti mezi zdrojem a cílem a nazpět - budeme označovat termínem „síťová latence“
- Procento nedoručených paketů vztažené k celkovému počtu odeslaných paketů - tzv. ztrátovost paketů
- Jitter - kolísání velikosti zpoždění paketů při průchodu síti (především pro streamování audia, videa a VOIP služby)
- Kapacita spoje
- Intenzita a délka trvání výpadků
- Měřitelné veličiny bezdrátového spoje (síla signálu, šum pozadí, počty paketu s nepotvrzeným příjmem na fyzické vrstvě)

Navrhovaný způsob měření dokáže zachytit první tři hodnoty, tedy síťovou latenci, ztrátu paketů a variantu jitteru - směrodanou odchylku síťové latence od její střední hodnoty.

2.1.1 Nestandardní hodnoty a jejich příčiny

Příčin vzniku nestandardních hodnot může být několik, následující výčet prezentuje ty nejčastější:

- hardwarová chyba aktivního zařízení na síti (poškozený switch)
- interference s jiným zdrojem signálu (souběh metalických vodičů, současné radiové vysílání více zdrojů)
- nedostatečné dimenzování výkonu hardware, nebo jeho přetížení (slabý procesor routeru)
- softwarová chyba na routeru

- špatná instalace (vlhkost v optickém vedení), nebo prekročení doby životnosti zařízení (rezavějící kabely)
- přírodní vlivy (silná mlha nebo hustý déšť u optických bezdrátových spojů)

Vlivem těchto faktorů může v síti dojít ke snížení kvality poskytované služby. Některé z nich způsobí okamžitý výpadek, jejich detekce je tím jednodušší. Jiné, například radiová interference s jiným zdrojem vysílání, nelze snadno detekovat a je potřeba si definovat jistou hranici, při jejímž překročení označíme kvalitu poskytované služby za nedostatečnou. Z jednoho měření totiž není možné rozlišit, zda jde o vzácný případ nebo časté chování. Bude tedy potřeba vyhodnotit data z delšího intervalu a pomocí statistických metod prezentovat výsledky. Jedním z možných přístupů je stanovení limitní hranice pro určitou měřenou veličinu. Například hodnota síťové latence přes 200 milisekund. Při jejím překročení by měl správce diagnostikovat připojení a provést vyřešení problému. Srovnáním výsledků z určitého intervalu s touto hodnotou, lze získat informaci o tom, jak dlouho byla služba z celkového času nekvalitní.

2.2 Modelová síť

Součástí požadavku na řešení by měl být také popis sítě, pro kterou se předpokládá jeho použití. V předchozích odstavcích bylo zmíněno, že oblastí hlavního uplatnění budou bezdrátové sítě. Je ale možné i využití pro sledování klasických drátových LAN sítí. Modelová síť tedy může být tvořena buď výhradně jednou technologí nebo jejich kombinací.

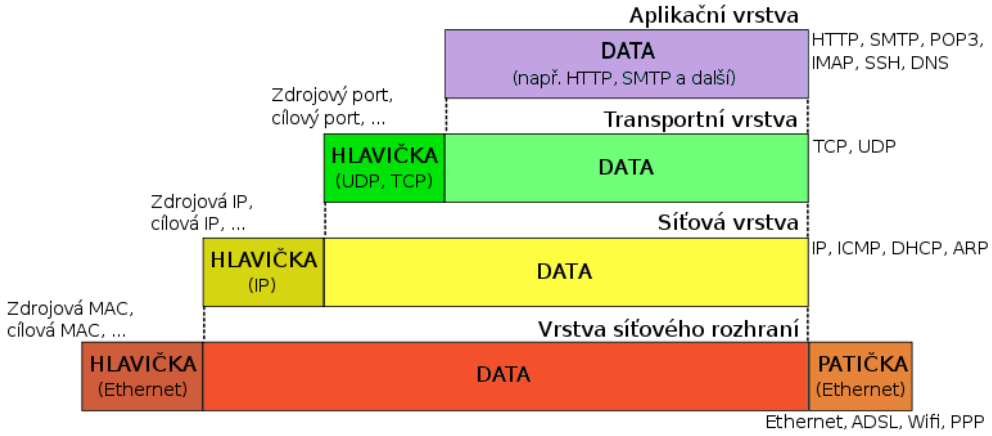
Základním požadavkem na síť je podpora IP protokolu ve verzi 4. V případě existence TCP provozu navíc bude možné provádět měření některých dalších charakteristik kvality služby (síťová latence, rozptyl latence a ztrátu paketů). U ostatních protokolů bude pouze zaznamenán objem přenesených dat. Dále pak je pro měření nezbytná přítomnost centrálního bodu v síti, přes který prochází veškerá komunikace lokální sítě s internetem. K tomuto bodu musí být připojitelné monitorovací

zařízení, které bude zachytávat průchozí provoz. Splnění uvedených požadavků na síť dovoluje provádět měření kvality služby poskytované stanicím na LAN.

2.3 Protokoly modelové sítě

Jestliže je tedy testovací prostředí připraveno, lze provést analýzu protokolů, které se používají k přenosu dat a identifikaci vhodných atributů, které lze sledovat a s jejichž pomocí lze měřit kvalitu připojení jednotlivých klientských zařízení. Důraz je kladen především na protokoly, s kterými je možné se setkat na datové síti providera.

Modelová síť je součástí internetu, v kterémž je drtivá většina komunikace založena na protokolech rodiny TCP/IP. Tyto protokoly jsou rozděleny do několika vrstev, které si data vzájemně předávají (obr. 2.2).



Obrázek 2.2: Vrstvy modelu TCP/IP [5]

Každá vrstva tohoto modelu přijme data od vyšší vrstvy, přidá k nim vlastní hlavičku a pošle je nižší vrstvě nebo již přímo síťové kartě (zapouzdření dat v jednotlivých vrstvách je znázorněno na obr. 2.2). U příjemce dojde k opačnému postupu. Paket se začne zpracovávat od nejnižší vrstvy a po odstranění zapouzdření a provedení kontroly obsahu je paket předán vyšší vrstvě.

Na modelové síti se do komunikace kromě PC odesílatele a PC příjemce zapojují

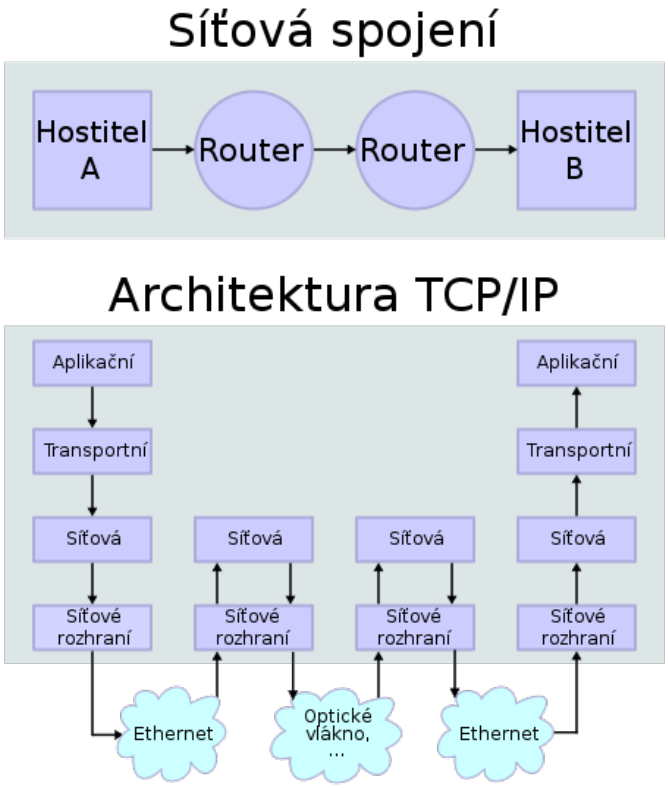
ještě další zařízení. Moderní verze technologie ethernet dovede realizovat pouze spojení bod - bod. Pro realizaci složitějšího zapojení je tedy do sítě nutné zahrnout tzv. přepínač (switch) nebo opakovač (repeater). Obě tato zařízení pracují na vrstvě síťového rozhraní modelu TCP/IP. Opakovač je velmi jednoduché ale rychlé zařízení, které sestává z několika datových zásuvek (tzv. portů). Příchozí paket na jednom z portů automaticky přepošle na všechny ostatní porty. Přepínač vypadá fyzicky velmi podobně, nicméně byl obdařen navíc i pamětí, takže data nepřeposílá zcela „hloupě“ na všechny porty, ale pro každý port si uchovává seznam fyzických adres, které přes tento port komunikovaly. Data pak přeposílá v ideálním případě pouze na tento jeden port.

V současnosti se od použití opakovačů upouští z důvodů požadavků kladených na efektivitu využití sítě (zbytečně zatěžují provozem části sítě, kde se nevyskytuje cílový PC). Navíc opakovač pouze přeposílá pakety zatímco přepínač kontroluje i hlavičku protokolu ethernet (pro zjištění fyzických adres odesílatele a příjemce).

Obě zařízení omezuje zásadně fakt, že dva počítače v této síti mohou komunikovat pouze jestliže mají IP adresu ze stejného síťového rozsahu. Tedy síť providera může být složená pouze z přepínačů, ale pak nebude možné komunikovat s IP adresami z jiných síťových rozsahů a tedy ani se zbytkem internetu. Na každé síti připojené do internetu se proto lze setkat s alespoň jedním zařízením označovaným jako router nebo layer3 switch. Výraz layer 3 označuje, že se jedná o switch, který rozumí i třetí (síťové) vrstvě modelu ISO/OSI (odpovídá druhé vrstvě modelu TCP/IP). Tato zařízení umí předávat pakety mezi sítěmi s různými IP rozsahy. V jakékoliv síti připojené k internetu je alespoň jeden router, a to výchozí brána do internetu.

Je-li síť rozdělena na více IP sítí, potom se router využívá i k propojení jejich komunikace. V obrázku 2.3 je znázorněno zabalení dat do paketu na odesílajícím PC (hostitel A), cesta sítí přes dva routery a opětovné rozbalení paketu na příjemci (hostitel B).

Pro vrstvu síťového rozhraní hostitele A se jeví ostatní sítě schované za prvním routerem jako jako stejné zařízení. Jestliže tedy rámce obsahují pouze hlavičku protokolu ethernet a nejsou v něm tedy zapouzdřeny hlavičky vyšších vrstvev, pak není



Obrázek 2.3: Cesta paketu sítí

možné tyto rámce předávat do další sítě pomocí routeru. Router totiž pracuje na síťové vrstvě a vrstvu síťového rozhraní používá pouze k přesunu dat uvnitř sítí, které spojuje. Po obdržení rámce router odstraní ethernetovou hlavičku (vrstva síťového rozhraní) a před odeslánim ji nahradí novou. Původní hlavička by totiž na nové síti neměla význam, neboť se zde vyskytují rozdílné fyzické adresy zařízení.

Tento fakt zásadně omezuje použití hodnot získaných z hlaviček protokolů síťové vrstvy. Jestliže je síť postavena pouze pomocí switchů a neobsahuje kromě internetové brány žádný router, bylo by možné těchto údajů využít. V opačném případě jsou ale hlavičky tvořeny údaji z nejbližších routerů a nenesou tedy informace vložené odesílatelem.

Co se týče protokolů, v bezdrátových sítích je přenos na linkové vrstvě realizován téměř výhradně pomocí ethernetu. Na něj se tedy zaměříme nejdříve. Nad vrstvou

síťového rozhranní je síťová vrstva. Zde se lze nejčastěji setkat s IP protokolem (verze 4, zřídka verze 6) a protokolem ARP. O úroveň výše nalezneme transportní vrstvu, kderá je tvořena především z protokolů TCP, UDP a ICMP. V následujících odstavcích bude v jednoduchosti vysvětleno fungování protokolu ethernet, IP a TCP.

2.3.1 Ethernet

Ethernet je protokolem vrstvy síťového rozhraní modelu TCP/IP, který zajišťuje přenos dat uvnitř jednoho síťového segmentu. Toto pojmenování nese také celý soubor technologií pro budování lokálních komunikačních sítích. V naší modelové sítí se jedná o základní protokol a formát každého zachyceného paketu je stanoven právě tímto protokolem. Základní jednotkou dat pro protokol ethernet je rámec. Obrázek 2.4 obsahuje jeho schéma pro protokol Ethernet II a 802.3 (liší se využitím jednoho pole). V následujícím textu je popsán význam jednotlivých polí v tomto protokolu.

Preambule	SFD	MAC cíle	MAC zdroje	Typ/délka	Data a výplň	CRC32	Mezera mezi rámci
7× oktet 10101010	1× oktet 10101011	6 oktetů	6 oktetů	2 oktety	46-1500 oktetů	4 oktety	12 oktetů
		64-1518 oktetů					
72-1526 oktetů							

Obrázek 2.4: Ethernetový rámec

Síť pracující na technologii ethernet je základním požadavkem navrhovaného software. Hodnoty z hlavičky tohoto protokolu nicméně k výpočtům charakteristik kvality služby nejsou použity. Neposkytují totiž vhodné informace a na sítích s routery jsou jejich hlavičky upravené při každém průchodu routerem.

2.3.2 IP

Protokol IP je základním dorozumívacím mechanismem v internetu. Na rozdíl od protokolů nižší vrstvy umožňuje přenos dat mezi dvěma libovolnými počítači uvnitř této sítě. Každý počítač, který se chce účastnit komunikace, musí mít přidělenou tzv. IP adresu. Jedná se o čtyřbajtovou hodnotu (u IP verze 6 jde o 16 bajtů), která musí být v dané síti jedinečná. Protokol IP negarantuje správné pořadí ani doručení dat,

nevylučuje duplicitu a ani nekontroluje integritu přenášených dat, tuto funkcionalitu přenechává protokolům vyšších vrstev. Přenos dat probíhá pomocí paketů. Ty jsou tvořeny vždy vlastní hlavičkou protokolu IP a částí obsahující přenášená data. Na obrázku 2.5 je znázorněn formát paketu.

Formát IP datagramu

Bajty	0		1	2	3
Bajt 0 až 3	Verze	Délka hl.	Typ služby	Celková délka	
Bajt 4 až 7	Identifikace			Příznaky	Offset fragmentu
Bajt 8 až 11	TTL		Protokol	Kontrolní součet hlavičky	
Bajt 12 až 15	Adresa odesílatele				
Bajt 16 až 19	Adresa cíle				
Bajt 20 až 23	Volby				Výplň
...	Data				

Obrázek 2.5: IP paket

Následující výčet představuje stručný popis jednotlivých polí IP paketu (datagramu) [4].

Verze protokolu IP, zde vždy 4.

Délka hlavičky ve čtyřbajtových slovech; hlavička může být kvůli volbám různě dlouhá.

Typ služby podle původních představ měl umožnit odesílateli, aby zvolil charakter přepravní služby ideální pro dotyčný datagram. Jednotlivé bity znamenaly např. požadavek na nejmenší zpoždění, největší šířku pásma či nejlevnější dopravu. Směrování pak mělo brát ohled na hodnotu TOS a volit z alternativních tras tu, která nejlépe odpovídala požadavkům datagramu. V praxi však k realizaci nedošlo. V současnosti se položka používá k podobným účelům - nese značku pro mechanismy zajišťující služby s definovanou kvalitou (QoS).

Celková délka datagramu v bajtech.

Identifikace představuje jednoznačný identifikátor přidělený odesíletelem každému paketu. Pokud byl datagram při přepravě fragmentován, pozná se podle této položky, které fragmenty patří k sobě (mají stejný identifikátor).

Příznaky slouží pro řízení fragmentace. Definovány jsou dva: *More fragments* ve významu „nejsem poslední, za mnou následuje další fragment původního datagramu“ a *Don't fragment* zakazující tento datagram fragmentovat.

Offset fragmentu udává, na jaké pozici v původním datagramu začíná tento fragment. Jednotkou je osm bajtů.

TTL představuje ochranu proti zacyklení. Každý směrovač zmenší tuto hodnotu o jedničku (případně o počet sekund, které datagram ve směrovači strávil, pokud zde čeká déle). Pokud tím TTL nabude hodnotu nula, datagram zahodí, protože vypršela jeho životnost.

Protokol určuje, kterému protokolu vyšší vrstvy se mají data předat při doručení.

Kontrolní součet hlavičky slouží k ověření, zda nedošlo k poškození. Počítá se pouze z hlavičky a pokud nesouhlasí, datagram bude zahozen.

Adresa odesílatele IPv4 adresa síťového rozhraní, které datagram vyslalo.

Adresa cíle IP adresa síťového rozhraní, kterému je datagram určen.

Volby zahrnují různé rozšiřující informace či požadavky. Například lze předepsat sérii adres, kterými má datagram projít. Volby obvykle nejsou v datagramu použity (v obrázku jsou barevně odlišeny).

Výplň nenese žádnou informaci, slouží k zaokrouhlení délky hlavičky na násobek čtyř bajtů, pokud jsou použity volby uvedené výše.

Data představují pochopitelně samu přenášenou informaci.

IP pakety jsou před odesláním zabaleny do protokoů nižších vrstev, které umožnují přenos jen určitého objemu dat. Tato hodnota se označuje jako MTU (Maximum

Transfer Unit) a například u sítě ethernet má velikost 1500 bajtů. Jestliže paket při cestou v síti narazí na router, z něhož vede k příjemci linka s menší velikostí, než má sám, dochází k nepříjemné situaci. Router ověří hodnotu pole příznaky (konkrétně DF bit). Je-li tato hodnota rovna nule, bude provedena fragmentace paketu. Jestliže ne, paket bude zahozen a dojde k odeslání ICMP zprávy o chybě na adresu odesílatele.

Při fragmentaci dojde k rozdělení dat jednoho paketu mezi více různých paketů. Při sestavování fragmentů do původního paketu se využívá hodnot polí Identifikace, Offset fragmentu a příznaku More fragments, z kterých lze jednoduše sestavit celý paket. Proces fragmentace je považován za nežádoucí, protože s vytvořením dalších fragmentů vzniká potřeba každému z nich přiřadit i novou hlavičku a přenos se stává méně efektivním. Navíc jediný, kdo je oprávněn paket složit opět dohromady, je jen příjemce. Routery, přes které paket prošel, jej sestavovat nesmí, neboť není zajištěno, že všechny pakety půjdou stejnou cestou.

Důležitou podmínkou na routery v síti je, že musí být schopny přenášet pakety minimálně velikosti 576 bajtů (minimální hodnota MTU). Ty mohou být dále fragmentovány, ale je to velmi vzácné. Maximální velikost IP hlavičky je 60 bajtů, TCP 60 bajtů a UDP 8 bajtů. Minimální velikost fragmentu je 68 bajtů [6] (60 bajtů pro hlavičku a 8 bajtů pro data). Obvyklá velikost IP hlavičky je však jen 20 bajtů a TCP hlavičky kolem 36 bajtů. Je-li tedy na paket použita fragmentace, lze stále s velkou pravděpodobností předpokládat, že první fragment bude obsahovat celou TCP nebo UDP hlavičku.

2.3.3 TCP

Protokol TCP je nejčastěji používaným protokolem transportní vrstvy modelu TCP/IP. Spolu s UDP tvoří dva hlavní protokoly, na kterých je postaven současný internet. K vlastnostem nižších vrstev (IP) přidává navíc garance spolehlivosti a doručení dat ve správném pořadí.

Pro svoji funkci vytváří trvalé spojení mezi odesílatelem a příjemcem. Aby bylo možné těchto spojení vytvářet více, bylo nutné rozšířit adresaci a tedy i možnost

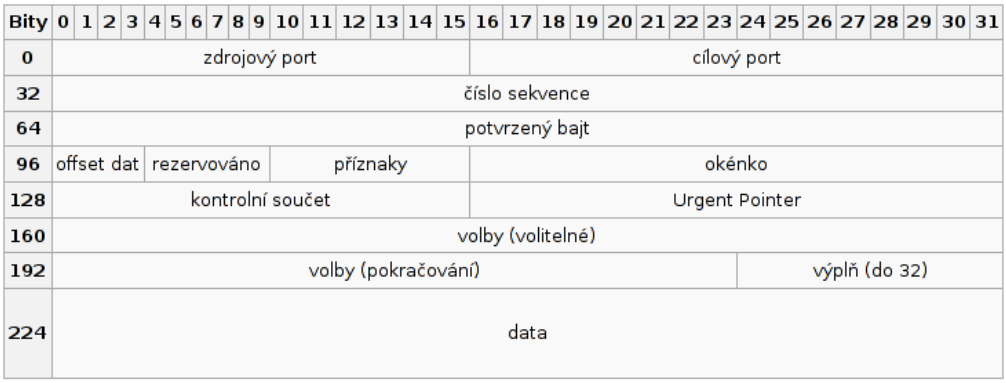
navázat mezi dvěma stejnými počítači více spojení. Toto rozšíření se nazývá port. Každé spojení má pak při svém vzniku přiřazený zdrojový a cílový port, na kterém probíhá komunikace. Součástí TCP hlaviček paketů je pak toto číslo, které umožňuje doručit data příslušné aplikaci, vlastníku daného portu. Porty nižší než 1024 se nazávají privilegované porty a na mnohých systémech je potřeba pro jejich otevření vyšších uživatelských oprávnění. Některé porty mají také přirazené známé služby, které se nich provozují. Je samozřejmě možné tyto porty používat i k jiným účelům, ale odporuje to zvyklostem.

Spolehlivostí doručení paketů se rozumí, že aplikace která přenáší data pomocí TCP se nemusí starat o kontrolu, zda byla data doručena a zda nebyla v průběhu cesty poškozena. Tato práce je totiž v kompetenci TCP. Další vlastností je zaručení správného pořadí došlých dat. Internet je velmi rozlehlá síť a pakety protokolu IP mohou být doručeny v jiném pořadí než byly odeslány. Routery na cestě totiž mohou být nastaveny tak, aby upřednostňovaly například malé pakety nebo mohou vyrovnávat zátěž některého spoje tím, že data rozdělí mezi více spojů. Výsledkem je doručení paketů v nesprávném pořadí. Při použití TCP ale tento problém zcela odpadá. Oproti UDP nabízí také vyšší bezpečnost, podvrhnutí zdrojové adresy je výrazně obtížnější právě z důvodu vytváření trvalých spojení.

Byly zmíněny dvě velké výhody protokolu TCP, nyní tedy řekněme něco i k nevýhodám. Mezi tu hlavní patří vyšší režie protokolu, tedy je možné přenést méně dat za stejnou dobu. Dále je implementace TCP složitější a obtížnější. V případě přenosu audia nebo streamování videa je jeho použití také nevhodné. Výpadek dat totiž vyvolá jejich opětovné odeslání, které může způsobit zdržení, např. v telefonním hovoru. Pro aplikace, kde není nutné garantovat doručení všech dat, je TCP velmi nevhodné a nahrazuje se zde protokolem UDP.

Nyní se podíváme na segment(základní jednotka přenosu) protokolu TCP a jeho datová pole.

Zdrojový port je port odesílatele TCP segmentu, maximálně je možné až 65 535 portů.



Obrázek 2.6: TCP paket

Cílový port je portem adresáta TCP segmentu.

Číslo sekvence je pořadové číslo prvního bajtu TCP segmentu v toku dat od odesílatele k příjemci (TCP segment nese bajty od pořadového čísla odesílaného bajtu až do délky segmentu).

Potvrzený bajt je číslo následujícího bajtu, který je příjemce připraven přijmout (příjemce potvrzuje, že správně přijal vše až do pořadového čísla přijatého bajtu mínus jedna).

Offset dat vyjadřuje délku záhlaví TCP segmentu v násobcích 32 bitů.

Rezervováno rezervováno pro budoucí použití, nyní bez významu.

Příznaky obsahuje seznam příznaků, které slouží pro správu TCP spojení.

Okénko vyjadřuje přírůstek pořadového čísla přijatého bajtu, který bude příjemcem ještě akceptován.

Kontrolní součet jedná se o kontrolní součet IP záhlaví. Počítá se nejen ze samotného TCP segmentu, ale také z některých položek IP záhlaví.

Urgent pointer ukazuje na konec úseku naléhavých dat, je-li přičten k pořadovému číslu odesílaného bajtu. Odesílatel si přeje, aby příjemce tato data prednostně

zpracoval. Tento mechanismus se využívá například u služby telnet. Toto pole je nastané jen v případě, že je aktivní příznak URG.

Volby obsahují dodatečné nastavení protokolu.

Výplň doplňuje data tak, aby byla jejich délka zarovnaná.

Data obsahují přenášenou informaci.

Pomocí příznaků v TCP segmentu paketu lze řídit stav spojení. Následující seznam představuje výčet možných hodnot.

SYN odesílatel začíná s novou sekvencí číslování, tedy v TCP segmentu je možné nalézt pořadové číslo prvního odesílaného bajtu

ACK TCP segment má platné pole „Potvrzený bajt“. Tento příznak je nastaven ve všech segmentech spojení, kromě prvního v kterém se navazuje

RST odmítnutí TCP spojení

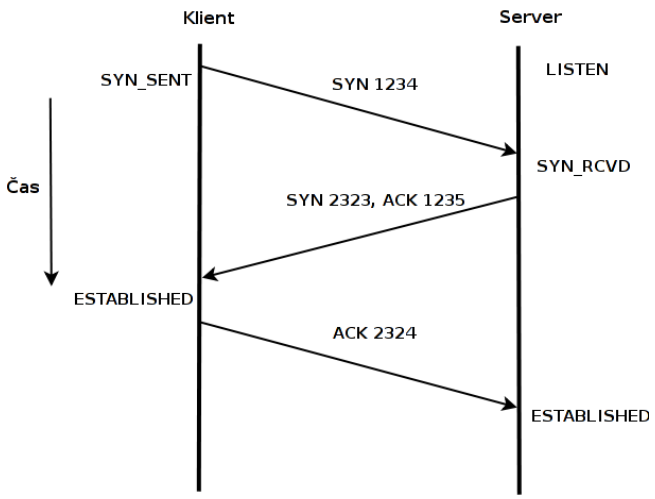
FIN odesílatel oznamuje, že ukončil odesílání dat. Přijetí segmentu s tímto příznakem ukončuje spojení pouze v jednom směru, komunikace v opačném směru je stále možná. Protokol TCP vytváří plně duplexní spojení a to je přijetím paketu s tímto příznakem uzavřeno v jednom směru. Není tedy možné odesílat další data, ale stále lze odeslat paket s příznakem ACK pro potvrzení příjmu dat od protistrany.

URG TCP segment nese naléhavá data

PSH vyjadřuje, že segment nese aplikační data, příjemce má tato data předávat aplikaci, použití ale není ustáleno

Pole volby může obsahovat dodatečná nastavení TCP spojení. Některé volby se mohou vyskytovat v segmentech pouze v určitém stavu spojení (například při jeho navazování).

Protokol TCP používá k transportu dat protokol IP, avšak nad tímto protokolem zřizuje spojovanou službu. Součástí popisu tedy musí být i řešení problémů s navázáním a ukončením spojení, potvrzováním přijatých dat, vyžádání ztracených segmentů, ale také problémy s průchodností cesty a udržování odpovídající rychlosti přenosu dat. V následujícím odstavci si popíšeme, jak takové činnosti probíhají, neboť popis TCP segmentu je jen malou částí TCP protokolu. Podstatně rozsáhlejší je proces výměny segmentů. Z těchto poznatků se poté pokusíme vytvořit postup, podle kterého by bylo možné měřit kvalitu přenosu dat na síti.



Obrázek 2.7: TCP navázání spojení

V předchozích odstavcích bylo už zmíněno, že protokol TCP využívá v adresaci kromě IP adresy také porty. IP adresa určuje cílový PC zatímco adresa portu určuje aplikaci, která na portu naslouchá a které mají být data doručena. Každé vytvořené TCP spojení lze jednoznačně identifikovat podle čtveřice (IP adresa odesílatele, IP adresa příjemce, port odesílatele, port příjemce).

Strana, která si přeje vytvořit připojení (nadále ji budeme označovat jako klient), vygeneruje náhodné 32 bitové číslo používané jako startovací pořadové číslo odesílaného bajtu. Vytvoří TCP segment bez dat a toto číslo vloží do pole Číslo sekvence. Nastaví příznak SYN, který oznamuje, že číslování dat během spojení začíná od čísla sekvence. V žádném dalším paketu tohoto spojení tedy není možné vygenerovat nové číslo sekvence a nastavit příznak SYN. Pole potvrzený bajt je vyplněno nulami a pří-

znak ACK není nastaven. Jedná se totiž o první bajt spojení, a tak zatím není co potvrzovat.

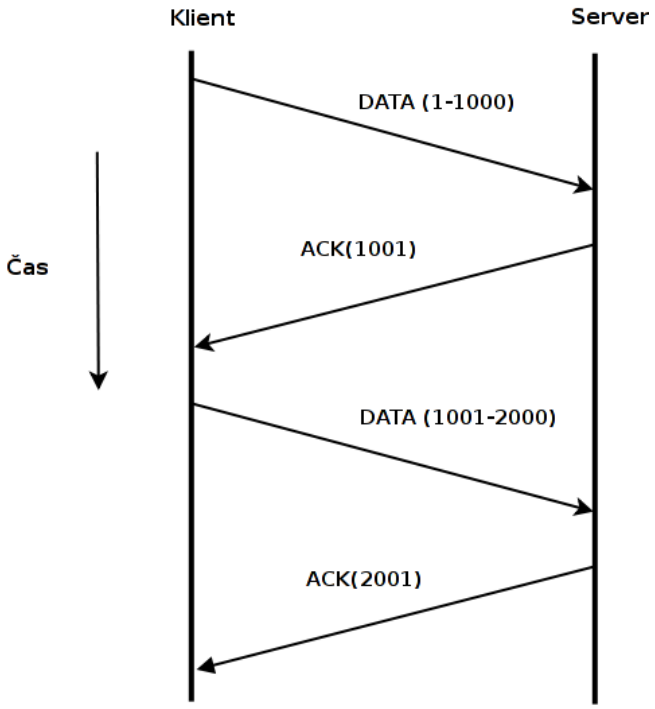
V případě, že protistrana odmítne navázat spojení na daném portu, tak odešle zpět prázdný segment s nastaveným příznakem RST, kterým oznamuje odesílateli, že port není přístupný a spojení nebude navázáno. V případě, že protistrana chce spojení uskutečnit, tak zpět odpoví prázdným paketem s příznakem SYN a ACK. Pro tento paket opět vygeneruje vlastní 32 bitové číslo jako startovací pořadové číslo odesílaného bajtu. Vzhledem k nastavenému příznaku ACK obsahuje pole Potvrzený bajt hodnotu Číslo sekvence z prvního paketu spojení zvýšenou o jedna. Pole totiž ukazuje na další bajt v datech, který je server ochoten přijmout. TCP má tedy pro každý směr spojení vlastní číslování odeslaných dat.

Posledním segmentem při navazování spojení je paket od klienta na server, který má v poli Číslo sekvence hodnotu o jedna vyšší než první paket a nastavený příznak ACK spolu s hodnotou o jedna vyšší než číslo sekvence přijaté v druhém paketu spojení. Třetím segmentem končí navazování spojení, často je proto tento proces označován jako „třífázový handshaking“. Ve chvíli, kdy jedna ze stran obdrží paket s příznakem ACK, může začít odesílat datové pakety.

Celý proces navázání spojení je znázorněn na obr. 2.7. Vzhledem k tomu, že proces vyžaduje odeslání několika segmentů, je nutné udržovat na každé straně informaci o aktuálním stavu spojení. Před začátkem spojení je server ve stavu LISTEN a naslouchá na daném portu. Klient odesílá první segment a přechází do stavu SYN_SEND. Po přijetí paketu se server přepne do stavu SYN_RCVD a obdrží-li následně i ACK paket od klienta, přechází do stavu ESTABLISHED a spojení je navázáno. Klient do stavu ESTABLISHED přechází ve chvíli, kdy obdržel paket s příznakem ACK na svůj SYN požadavek.

Je-li spojení úspěšně navázáno, může dojít k výměně dat. Protokol TCP vyžaduje, aby byl příjem TCP segmentů potvrzován protistranou. Ne všechny TCP segmenty proto musí nutně nést data. Často dochází k situaci, kdy jedna ze stran nepotřebuje odesílat data, přesto musí alespoň práznými pakety s nastaveným příznakem ACK odesílat potvrzení přijatých dat od protistrany, v takovém případě nesmí být

v prázdných segmentech nastaven příznak PSH. V nejjednoduším případě vypadá komunikace mezi klientem a serverem tak, jak je znázorněno na obrázku 2.8. Klient odesílá v prvním segmentu 1000 bajtů dat. Server odpovídá prázdným segmentem s příznakem ACK a pole Potvrzený bajt má hodnotu o jedna vyšší než poslední přijatý bajt od klienta. Klient odesílá dalších 1000 bajtů a přijímá od serveru potvrzení jejich příjmu. Ve skutečnosti může klient odesílat více paketů zároveň, ale jinak tento diagram vystihuje základní vlastnost TCP protokolu, kterou je potvrzování paketů.

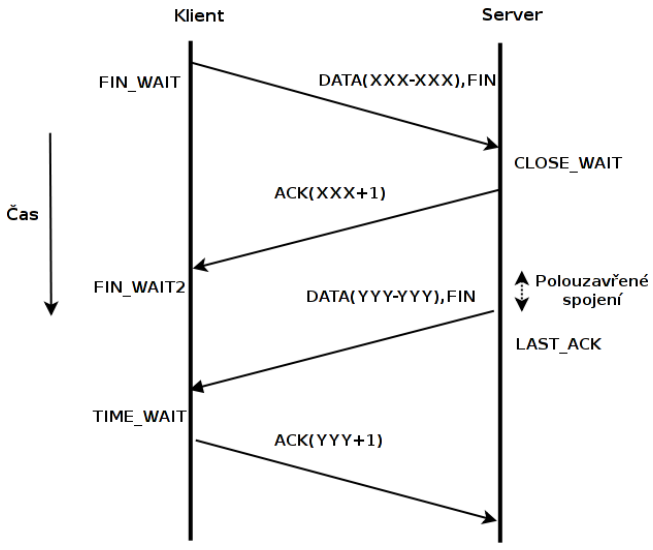


Obrázek 2.8: TCP přenos

V případě, že jedna strana dokončila přenos dat, může spojení jednostranně ukončit. K tomu stačí odeslat paket s nastaveným příznakem FIN, čímž dojde k takzvanému aktivnímu ukončení spojení. Druhá strana pak pouze potvrzuje a provádí tak pasivní ukončení spojení.

Pro správné ukončení spojení je nutné odeslat celkem čtyři segmenty. Průběh znázorněný na obrázku 2.9 opět popíšeme. Jestliže chce klient ukončit spojení jako první, odešle serveru paket s nastaveným příznakem FIN, poté sám přejde do stavu FIN_WAIT1. Tím signalizuje protistraně, že už byla odeslána všechna data. Ser-

ver po příjetí paketu odesílá jeho potvrzení a sám přechází do stavu CLOSE_WAIT (odpovídá pasivnímu uzavření spojení). Nyní již klient může odesílat pouze TCP segmenty bez dat. Po přijetí potvrzení od serveru klient přechází do stavu FIN_WAIT2. Zde zůstává do doby, dokud protistrana neodešle požadavek k ukončení spojení nebo systém sám po určité době nečinnosti uzavře toto spojení (přechod do stavu CLOSED). Ve chvíli, kdy se server rozhodne ukončit spojení odešle paket s příznakem FIN a sám přejde do stavu LAST_ACK. Klient obdrží paket, potvrdí jej prázdným paketem s nastaveným příznakem ACK a přechází do stavu TIME_WAIT. V tomto stavu setrvá po určitý čas (závislé na implementaci), po jehož uplynutí systém smaže záznamy a spojení a uvolní obsazené paměťové zdroje. Důvodem pro setrvání v tomto stavu je možná ztráta posledního ACK paketu od klienta na server. V tom případě, totiž server znovu odešle svůj paket s příznakem FIN. Kdyby ale klient po přechodu do stavu TIME_WAIT záznam o spojení ihned smazal, nerozuměl by požadavku serveru o ukončení spojení, pro klienta by totiž už byl záznam o spojení smazaný a vše by nasvědčovalo tomu, že takové spojení ani neexistovalo. Pravděpodobně by tedy místo potvrzení odeslal spíše paket s příznakem RST.



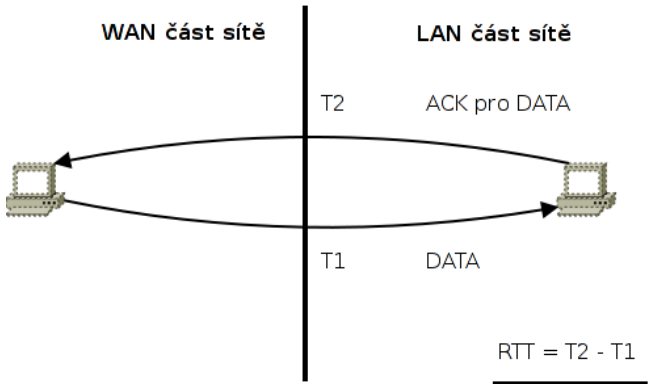
Obrázek 2.9: TCP ukončení spojení

S pakety, které mají nastavený příznak RST, se ale nesetkáváme jen tehdy, když je odmítnut pokus o navázání spojení (tedy odpověď na paket s příznakem SYN).

Často se totiž také používá k ukončení spojení v případě, že je zájem ukončit TCP spojení rychle a bez nutnosti setrvávat ve stavu TIME_WAIT. Je tedy možné, že je-li spojení v polouzavřeném stavu, tak strana která má stále otevřené spojení, může odeslat RST paket a spojení tak okamžitě ukončit. Případně klient, který odesílá potvrzení na FIN paket doplní toto potvrzení o RST příznak a dosáhne tím stejného výsledku.

2.4 Návrh metodiky měření

Základním principem, na kterém je postavena celá aplikace, je fakt, že každý TCP segment nesoucí data musí být po přijetí protistranou potvrzen bez zbytečného prodlení ACK paketem. Rozdíl časů mezi detekcí paketu s daty k cíli a přijetí potvrzujícího paketu je přesně čas, který byl potřeba pro cestu v síti tam, zpracování a cestu nazpět. Za určitých podmínek (viz dále) lze předpokládat, že čas potřebný pro zpracování paketu a vygenerování potvrzení je velmi malý (vzhledem k času potřebném pro průchod paketu sítí) a konstantní. To je dáno faktem, že ACK pakety jsou generovány přímo operačním systémem a kód je silně optimalizován pro rychlost. Rozhodnutí o odeslání potvrzení neprovádí cílová aplikace a je pouze v režii OS [7]. Rychlost odpovědi tedy není závislá na použitém programovacím jazyce programu ani na dokonalosti jeho algoritmu. Aplikace tedy bude umožňovat měřit čas, který potřebuje paket na průchod sítí (dále jen RTT).



Obrázek 2.10: Princip měření pomocí TCP ACK paketů

Ze sledování TCP spojení je možné také určit ztrátovost paketů při průchodu sítí. V běžném případě je tato hodnota přesná, nicméně za určitých podmínek může být zkreslena a odpovídá tak hornímu odhadu pro ztrátovost paketů. Vzácně se tedy může stát, že původně detekované špatné připojení je ve skutečnosti lepší, než hlásí aplikace.

Je důležité si také uvědomit, že není možné provádět měření na všech paketech přenesených na síti. Základním omezením je podmínka kladená na TCP protokol. Dále jsou některé pakety nevhodné pro měření, protože na ně byly aplikovány různé optimalizace TCP protokolu a poskytovaly by tak zavádějící informace. Je tedy za určitých nepříznivých okolností možné, že aplikace nebude schopna provést měření, třebaže klientský počítač po síti komunikuje.

Při praktické implementaci výše uvedeného jednoduchého principu programu se setkáme s celou řadou překážek. V následujících odstavcích probereme způsoby jejich odstranění.

2.4.1 Výběr vhodných paketů pro měření

Toto je nejzásadnější problém celého návrhu. Chceme-li totiž měřit IP adresy na vnitřní síti (dále jen LAN), potřebujeme TCP provoz takový, že počítač na LAN bude potvrzovat příjem dat od počítače z internetu. TCP protokol totiž potvrzuje pouze příjem TCP segmentů obsahujících data. TCP segmenty s nastaveným ACK příznakem, ale bez dat, potvrzované nejsou. Tedy pro měření je nutné, aby přicházela data do sítě a počítač v síti je potvrzoval. Z pohledu síťového zařízení na LAN se tedy jedná o stahování dat. Naštěstí situace na sítích poskytovatelů internetu je velmi vhodná pro měření, protože tyto sítě jen zřídka poskytují obsah a většina dat proudí dovnitř. Jakmile počítač na LAN data odesílá (např. maily nebo P2P sdílení souborů), lze k měření použít výrazně méně paketů z celého spojení.

Kromě ACK paketů potvrzujících data lze použít také ACK paket potvrzující pokus o navázání spojení. V kapitole o protokolu TCP bylo zmíněno, že k navázání spojení je potřeba třech paketů. První paket má nastavený SYN příznak a druhé straně tím oznamuje požadavek na otevření spojení. V kladném případě je potvrzen

ACK paketem. Hodnotu získanou z rozdílu časů detekce těchto paketů je možno použít pro měření.

Lze také využít paketu s příznakem RST, který oznamuje protistraně, že port není otevřen. Obě tyto odpovědi jsou totiž vytvořeny operačním systémem cílového počítače a nezávisí tedy na implementaci aplikace provádějící přenos dat[1].

V případné potvrzení ve formě paketu s nastaveným SYN a ACK příznakem musí být ještě potvrzeno protistranou. Tento paket je též vhodný, protože jeho odeslání podléhá pouze rozhodnutí operačního systému a je téměř automatické. Pro získání dat k měření tedy postačuje, aby počítač na LAN navázal spojení do vnější sítě. Těchto paketů je ale řádově méně, protože se vyskytují pouze při sestavování spojení.

Zbývá se ještě podívat na mechanismus ukončování spojení. Uzavření lze provést buď rychle pomocí odeslání RST paketu jednou stranou nebo pomaleji pomocí paketu s FIN příznakem. Při rychlé variantě zůstává RST paket nepotvrzen a tedy nelze využít k měření. U varianty s FIN paketem je situace opačná, je totiž okamžitě po přijetí potvrzen FIN ACK paketem. Odpověď je opět generována operačním systémem. Spojení je u FIN varianty ukončováno v každém směru samostatně. V optimálním případě, kdy je spojení ukončeno pomocí FIN paketů, tedy získáme alespoň jednu hodnotu při navazování spojení a jednu při ukončení.

Jakmile je TCP spojení navázáno, probíhá přenos dat mezi zdrojovým (na LAN) a cílovým počítačem (internet). Jak už bylo zmíněno, jestliže zdroj data pouze odesílá, nelze získat žádné pakety pro měření, příjem dat potvrzuje pouze cílový počítač. Většinou ale výměna dat probíhá oběma směry. Na zachytávacím rozhraní se tak objevují TCP segmenty s daty směrem do LAN sítě a TCP ACK segmenty z vnitřní sítě, potvrzující příjem dat. Ani zde však nelze využít každý paket.

Pozdržené ACK (Delayed ACK) je optimalizační mechanismus protokolu TCP definovaný v RFC2581 [8]. Jeho úkolem je snížení objemu generovanách TCP ACK paketů. Provádí to tak, že TCP ACK paket potvrzující příjem dat je odeslán vždy po uplynutí určitého času nebo po příjmu alespoň dvou TCP paketů s daty.

[1]Berkeley sockets API nabízí funkci accept pro navázání spojení. Volání této funkce způsobí, že u nových příchozích spojení bude automaticky proveden třícestný handshake. Po přijetí SYN paketu tedy o dalším pokračování v navazování spojení nerozhoduje cílová aplikace.

Tento čas je pro různé operační systémy často rozdílný[2]. Tímto jednoduchým postupem dochází k tomu, že při vyšších rychlostech je objem TCP ACK paketů až poloviční, časový limit totiž nestihne vypršet a potvrzuje se tak příjem každého druhého paketu.

Z výše zmíněného pro naši aplikaci plynou některé zásadní důsledky. Jestliže totiž zachytíme datový paket a pak i potvrzení jeho příjmu od protistrany, je stále možné, že potvrzení bylo uměle zpožděno. Vzhledem k faktu, že dokument RFC2581 nespecifikuje přesnou hodnotu, o které mají být potvrzení pozdržena, není ani možné tuto hodnotu automaticky od získaných hodnot odečítat. Návrh aplikace s tímto počítá a k měření využívá pouze paketů, které potvrzují alespoň dva přijaté pakety. Snižuje se tím počet získaných hodnot, ale je tak zajištěno, že nebudou zkreslené. Výhodnější je situace pro zařízení s operačním systémem Windows, neboť zde je časovač nastaven na relativně vysokou hodnotu 100 až 200 milisekund. Dorazí-li během tohoto úseku alespoň dva pakety, je vygenerováno potvrzení a měřící program použije zjištěnou hodnotu. Horší výsledky lze získat ze stanice s operačním systémem GNU Linux. Jeho časovač je totiž nastaven na interval mnohem nižší a je tedy méně pravděpodobné, že před jeho vypršením dorazí alespoň dva pakety. Některé operační systémy používají proměnnou hodnotu časovače[3], takže i se znalostí verze systému, by bylo obtížné zpoždění paketu odfiltrovat. Samo zjištění operačního systému síťové stanice by bylo nelehké, proto aplikace měří pouze potvrzení pro alespoň dva pakety. Tím si zachovává obecnost měření.

Mechanismus pozdržených ACK také není použit při úvodním navázání TCP spojení, při ukončení spojení a při potvrzení příjmu paketu, jestliže systém před ním přijal paket s vyšším pořadovým číslem (tedy v případě ztráty jednoho a více paketů z posloupnosti). Jeho použití při navázání a ukončení spojení by celý proces zablokovalo. V případě ztráty paketů a jeho přeposlání je důležité, aby protistanice přeposílala pouze opravdu ztracené pakety a to co nejrychleji. Zpožděné ACK by tak

[2]Např. systémy MS Windows jej nastavují rovnoměrně na hodnoty z rozsahu 100 až 200 milisekund [9], ale chování je možné změnit v registrech.

[3]Např. GNU Linux nebo MS Windows XP, které mají napevno nastavenou pouze horní a dolní mez.

způsobily buď zpomalení nebo nutnost přeposílat vždy dvojici paketů zároveň, což by vedlo k neefektivitě třeba v případě, kdy je potřeba znovu přenést pouze jeden paket.

Fragmentace IP protokolu je další možnou komplikací. Při přenosu IP paketu v síti mohlo dojít k jeho fragmentaci. Složení paketu je kompletně v režii cílového počítače. Zachytávací stanice tedy může obdržet jeden TCP segment rozprostřený do několika IP paketů a tyto pakety mohou dorazit v libovolném pořadí. Aplikace by tak byla postavena před úkol tyto fragmenty sestavit a vytvořit z nich původní TCP segment. Návrh počítá s jistým kompromisem. Pro správnou funkci měřící aplikace jsou totiž důležité pouze informace z IP a TCP hlaviček. Jestliže je tedy první fragment TCP segmentu dostatečně velký a obsahuje celou TCP hlavičku, pak je možné zbylé fragmenty ignorovat a zpracovat pouze ten první. IP hlavička je s malými rozdíly pro všechny fragmenty stejná. Tento kompromis se může zdát na první pohled omezující, nicméně velikost TCP hlavičky je 20 (nějčastější) až 80 byte a minimální velikost fragmentů je 576 bajtů.

2.4.2 Porovnávání naměřených hodnot

Při použití programu ping pro měření latence na sítí lze specifikovat velikost paketu, který bude odeslán do sítě. Cílová stanice odpoví přesně stejným paketem nazpět a program zobrazí dobu oběhu paketu. Při pasivním měření TCP spojení lze ale využít pouze zachycená data a tedy musíme pracovat s pakety různé délky. Chceme-li tedy měřit co nejpřesněji latence jednotlivých síťových zařízení, je nutné rozdělit pakety do kategorií podle jejich velikosti. Čím jemnější rozčlenění, tím přesnější hodnoty získáme. Bude-li úkolem aplikace pouze identifikovat velmi špatné připojení klientů, je možné použít pouze jednu kategorii. Rozdíly v časech mezi malými a velkými pakety jsou totiž často jen v řádu procent z celkového času na přenos. Je-li požadavkem přesnější měření, lze navrhnout například tři kategorie pro malé pakety (do 500B), střední pakety (501B - 1450B) a velké pakety (1451B - 1500B (nebo maximální velikost jednotky přenosu MTU)).

Druhým důležitým faktorem při porovnávání latence je její velikost v závislosti

na rychlosti datových přenosů cílové stanice. Nekvalitní spoj totiž může snadno způsobit, že i při malém zvýšení rychlosti se prudce zvedá latence. TCP protokol na to nejčastěji reaguje tak, že sníží rychlost přenosu dat. Vzhledem k faktu, že zachytávací aplikace má k dispozici veškerou komunikaci stanice, může z nasbíraných dat vypočítat i latenci v závislosti na aktuální přenosové rychlosti stanice. Získané výsledky je pak možné prezentovat v grafu nebo použít jednoduchý algoritmus pro vyhledávání stanic s příliš rychlým nárůstem latence.

2.4.3 Detekce ztráty paketů

Kromě výpočtu latence je možné využít data také k měření ztrátovosti přenosu paketů. Připomeňme si, že původní návrh TCP protokolu umožňoval opakovaný přenos paketu s daty v případě, že byl již jednou odeslán a ve stanoveném čase nebyl potvrzen příjem protistranou. Později se objevila vylepšení (např. mechanismus Fast retransmit [8]), kdy může příjemce informovat odesílatele o tom, že některý paket ze sekvence paketů nepřijal, ať jej tedy pošle znovu. Základní princip je však postavený na faktu, že data musí být znovu odeslána. Sledováním paketů je tedy možné detekovat opakovaný přenos dosud nepotvrzeného segmentu. Plně dostačuje evidovat dosud nepotvrzené TCP segmenty a dále mít dvě proměnné pro celkem přenesená data a opakovaně přenesená data. Podíl této dvojice je pak ztrátovost paketů na síti.

Získaná hodnota je ovšem vytvořená pouze nad vzorkem dat. Není totiž možné detekovat ztrátu dat při odesílání ze stanice na LAN. Na zachytávací stroj totiž takový paket ani nedorazí. Takové chování nemusí být podezřelé ani pro příjemce, z jeho strany to vypadá, že odesílatel jen nechce právě nic poslat. Směr přenosu dat od stanice na LAN směrem do internetu je tedy pro měření zastoupen pouze TCP ACK pakety, které generuje stanice při příjmu dat. V měření je tedy zastoupeno odesílání i příjem dat stanice, ale ztrátovost paketů můžeme určit jen pro celou dvojici (není možné rozlišit zda paket dorazil na stanici a následné potvrzení se ztratilo nebo zda už nedorazil přímo na stanici).

Další komplikací je fakt, že v případě ztráty ACK paketu, který často neobsa-

huje žádná data a je tedy velmi malý, je nutné odeslat celý TCP segment znovu, protože příjemce neobdrží ACK paket a předpokládá tedy, že data nebyla doručena. Vypočtená hodnota ztrátovosti paketů tak spíše odpovídá tomu, jak situace vypadá z pohledu uživatele stanice. Při ztrátě ACK paketu nelze poznat, že šlo pouze o ACK paket a počítadlo ztráty paketů se navýší o součet velikosti odeslaných dat a jejich ztraceného potvrzení. Stahování dat se tedy zpomalí po dobu přeposílání ztracených dat i přes to, že je už příjemce dříve obdržel, odesílatel to kvůli ztrátě potvrzení netuší.

Ztrátu ACK paketu by bylo možné částečně odfiltrovat. Moderní operační systémy implementují funkci Fast retransmit (FR) a SACK [10], kdy pomocí série duplicitních TCP ACK paketů upozorní odesílatele, že v přijatých datech chybí určitý úsek. Zachytávací stroj by mohl detekovat, zda opakovanému odeslání dat předcházelo použití mechanismu FR, a jestliže ne, tak to vyhodnotit jen jako ztrátu ACK paketu místo ztráty celé dvojice. Také by bylo možné využít předpokladu, že nejvýše každá dvojice paketů musí být potvrzena TCP ACK paketem. Jestliže tedy zachytávací stanice přijme TCP ACK paket potvrzující například čtveřici paketů a předtím nedošlo k opakovanému přenosu těchto paketů, pak došlo k ztrátě TCP ACK paketu, který potvrzoval první dvojici. Tyto metody jsou nicméně jen upřesňující a nikoliv dokonalé. Často spojené s řadou omezení. Například systém MS Windows 95 nemá implementovánu metodu FR ani SACK, takže by měření detekovalo vyšší ztrátovost paketů.

Téměř stejný výsledek vzniká i v situaci, kdy paket cestuje od odesílatele v internetu příliš dlouho. Jestliže totiž do určité doby neobdrží potvrzení, pokusí se znovu o přenos stejného segmentu i přes to, že se paket vlastně neztratil. Zachytávací aplikace tak detekuje stejný paket dvakrát a inkrementuje objem ztracených paketů. Tato situace je však velmi vzácná díky tomu, že moderní implementace TCP protokolů nastavují časovače vypršení podle odezvy spojení s velkou rezervou a dynamicky [11]. Dalším důvodem je, že LAN sítě jsou vzhledem k rozlehlosti internetu většinou malé a latence uvnitř také. Je tedy více než pravděpodobné, že stanice na LAN generuje TCP ACK potvrzení příjmu dat a odešle jej rychle nazpět. To po malé době

projde přes zachytávací stroj a ten vyjme paket z fronty těch dosud nepotvrzených. Je-li poté přijat stejný paket, aplikace už jej neoznačí za pokus o opakovaný přenos.

Důležitou vlastností je také fakt, že ztrátovost paketů je měřena ve směru ke stanici datovými pakety a v opačném směru povětšinou pouze prázdnými TCP ACK pakety. Měření tedy vypovídá spíše o ztrátovosti paketů směrem ke stanici, protože pro malé TCP ACK pakety je vyšší pravděpodobnost, že budou bezchybně přeneseny. To vyhovuje, protože většina provozů v sítích lokálních providerů je právě ve směru ke stanicím.

2.5 Srovnání se současnými metodami měření

Současný postup, pomocí kterého je možné odhalit zhoršenou kvalitu připojení stanice do sítě, je v obecném případě založen na dlouhodobé a pravidelné kontrole pomocí dvojice paketů ICMP echo request/reply. Jejich zpracování je implementováno v programu ping, který je dostupný na většině operačních systému. Jeho spuštěním dojde k zaslání paketu libovolné velikosti na zadanou IP adresu. Jestliže firewall na cílovém počítači tento paket nefiltruje, dostane odesílatel nazpět paket se stejnými daty. Z rozdílu časů odeslání požadavku a přijetí odpovědi je vypočítána síťová latence. V případě, že nedojde k přijetí odpovědi v určitém limitu od odeslání požadavku, považuje se daný paket za ztracený. Pravidelným spouštěním tohoto příkazu na cílovou stanici je možné detekovat změnu v latenci nebo ztrátě paketů. Měření je aktivní, takže spotřebovává zdroje (dostupnou kapacitu spojů), které by jinak mohly být v síti využity.

Navrhovaná aplikace také provádí měření latence a ztráty paketů. Oproti klasickému pingu má několik výhod, ale i jisté nevýhody. Především metoda přináší nový způsob měření kvality služby na síti, při kterém není potřeba generovat umělá data. Vše je prováděno pasivním měřením síťového provozu, který vytváří cílová stanice. Z toho plyne zásadní přínos, neboť je možné sledovat chování při reálném provozu. Alternativní dostupné metody měření pomocí protokolu ICMP posílají do sítě vlastní pakety. Tím ji nejen zatěžují, ale poskytují také zkreslené údaje. Poskytovatele služby

totiž obvykle nezajímá, jak se chová daný spoj bez zátěže, ale spíše jak reaguje na průchozí provoz. Metoda ICMP tak opticky zlepšuje výsledky dlouhodobého měření, neboť jich může být mnoho provedeno právě v době, kdy je stanice neaktivní.

Další zásadní metodou nového způsobu měření je, že dovede měřit nejen stav přístupové sítě poskytovatele (až k bodu, kde dochází k předání služby zákazníkovi), ale měří i parametry vlastní sítě zákazníka, která je schována za službou NAT (Network address translation). Aktivní metody (ICMP) tohoto měření nejsou schopny, neboť přes NAT neprojdou. To je mocná výhoda, protože zákazník obvykle nerozlišuje, zda je připojení špatné skutečně kvůli přístupové síti poskytovatele, anebo zda je na vině jeho vlastní bezdrátový spoj. Se službou je pak nespokojen, aniž by si toho jeho ISP byl vědom, neboť aktivní měření problém neodhalí.

Ještě další výhodou je skutečnost, že oproti programu ping naše metoda umožňuje věrněji zachytit vliv rušení v bezdrátovém přenosu.

TCP protokol je maximálně efektivní, to znamená, že při dostatku dat mají přenášené pakety co největší možnou velikost. Právě u takových paketů je však vyšší pravděpodobnost, že budou při přenosu sítí ztraceny, neboť jejich přenos bezdrátovou sítí trvá déle a je větší šance, že bude přerušen. V ideálním případě je k měření použit každý třetí paket procházející sítí (první dva nesou data, třetí je potvrzuje). Tedy například při stahování dat rychlostí čtyři mbps (500 kilobajtů za vteřinu, maximální velikost dvojice paketů je na síti ethernet 3 kilobajty) získáme pro měření 180 hodnot za sekundu. Metoda ping (ICMP) by pro stejný počet měření musela vygenerovat dodatečných 360 paketů (180 požadavků a 180 odpovědí). Takový objem dat by ale výrazně zkreslil měřené hodnoty.

Následující body obsahují ve zkratce seznam výhod a nevýhod.

Výhody

- Pasivní měření - aplikace nespotřebovává volnou kapacitu sítě a neovlivňuje tak vlastním provozem ani výsledky měření
- Zobrazuje reálné chování sítě - na rozdíl od pingu se nejedná o umělé měření, vypovídá o tom, jak skutečně síť fungovala v době pozorování. Tedy například

o rychlosti, kterou na počítačích klientů nabíhají webové stránky.

- Možnost měření hostů za NATem - ping nemá možnost vidět stav připojení počítačů za NATem. Navrhovaná aplikace nedovede rozpoznat více hostů za jednou IP adresou, ale dovede určit, že připojení počítačů za touto adresou je špatné. Slouží tedy jako upozornění pro správce sítě na možnou potíž.
- Měření kvality služby u stanice, která filtruje odpovědi na ICMP protokol.
- Latence i ztráta paketů se vztahují k přenosové rychlosti cílového počítače. Obecně ping nemá přístup k informacím o ostatních probíhajících síťových spojeních
- Pro měření využívá různé velikosti paketu, včetně těch nejvyšších. Aktivní měření by pro získání stejných výsledků zatěžovalo síť.
- Na spojích s rozdílnou kapacitou pro odesílání a příjem dat není možné jejich maximální vytížení pomocí příkazu ping. Ten je totiž symetrický a velikost odeslaných dat je stejná jako velikost přijatých dat. Zatímco tedy v jednom směru projdou data snadno, ve druhém to už bude obtížnější a může dojít k vypršení časového limitu.
- Spolu s prezentací naměřené síťové latence a ztráty paketů program zobrazuje také grafy datových přenosů cílové stanice. To je důležitý faktor ovlivňujícím interpretaci získaných dat.
- Aplikace má velmi jednoduché nastavení, stačí nastavit IP rozsah sítě, která se má sledovat. Není nutné přidávat jednotlivé stanice do konfigurace.

Nevýhody

- Pasivní měření - data jsou k dispozici jen tehdy, když je aktivní sama stanice. Jestliže právě nic nepřenáší, nelze naměřit latenci ani ztrátu paketů. Nelze tak například předvídat zhoršení kvality některého spoje.
- Systém musí být umístěn na bod, kde procházejí všechna data.

Kapitola 3

Implementace řešení

3.1 Popis modulů

Před sestavením návrhu monitorovací aplikace je nutné stanovit její hlavní úkoly. Program není určen pro produkční prostředí, záměrem této práce je otestovat možnosti nového typu měření a jeho dlouhodobé sledování, proto se požaduje od pouze rozumná rychlost výpočtu dat. Program by měl být samozřejmě maximálně efektivní, ale vzhledem k možným úpravám během testování je také vhodné, aby tyto změny bylo možné rychle implementovat a ověřit tak jejich vliv na výsledek.

Z těchto důvodů byl vybrán programovací jazyk Python. Přestože se jedná o skriptovací jazyk, poskytuje při výpočtech rozumný výkon a jeho velké množství knihoven dovoluje rychlou implemetaci konceptu a jeho ověření na skutečných datech. Python také nabízí přenositelnost, aplikaci je možné provozovat na různých operačních systémech (v offline módu se využívají pouze standardní knihovny Pythonu a rozšíření rrdpy pro práci s RRDtool databází). Nicméně zamýšlený operační systém pro běh démona je GNU Linux.

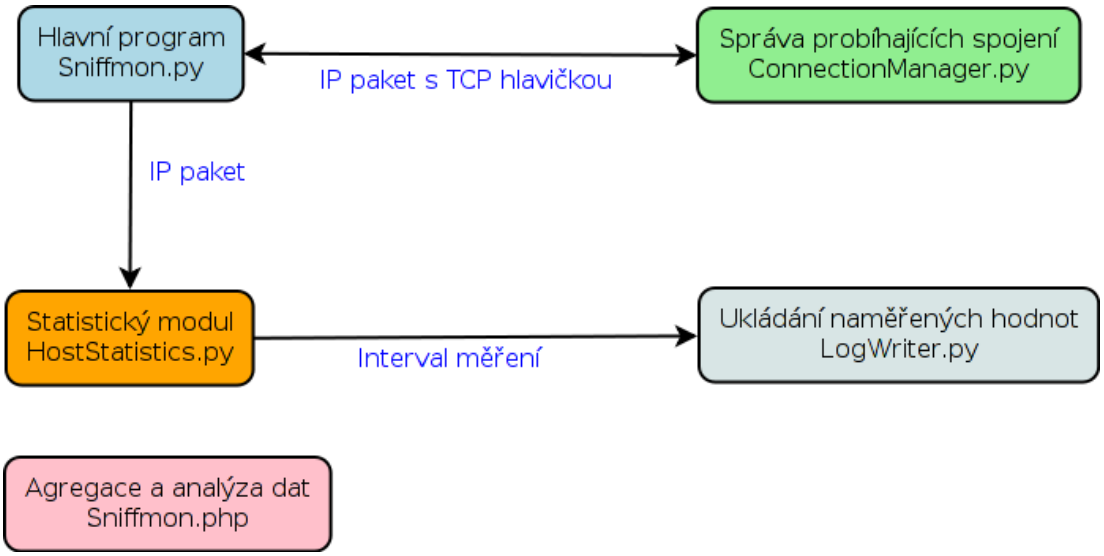
Hlavním úkolem programu je sledování procházejících paketů na určeném síťovém rozhraní a jejich následná analýza. Ta se pokusí identifikovat vhodné TCP ACK pakety, které vyhovují požadavkům z předchozí kapitoly. Dále aplikace vyčíslí ztrátovost přenosu paketů. Tato data jsou shromážďována po určitý časový interval, po jehož uběhnutí se získané informace uloží do souboru a měření započne znovu. Při

přechodu mezi měřícími úseky se smažou pouze statistická data z posledního intervalu, databáze probíhajících spojení zůstane zachována.

Následuje výčet modulů, do kterých byl program rozčleněn.

- Hlavní program
- Správa probíhajících spojení
- Statistický modul pro zpracování naměřených hodnot z probíhajících a již ukončených spojení
- Ukládání naměřených hodnot na disk
- Agregace a analýza dat

Vztah jednotlivých modulů je znázorněn na obrázku 3.1.



Obrázek 3.1: Vztah modulů programu

3.1.1 Hlavní program

Úkolem hlavního programu je načítání paketů ze sítě, extrakce IP a TCP hlaviček z paketů a jejich předání modulu pro zaznamenávání spojení a statistickému modulu. Program nemá grafické rozhraní a je určen pro běh v příkazovém řádku. Podle

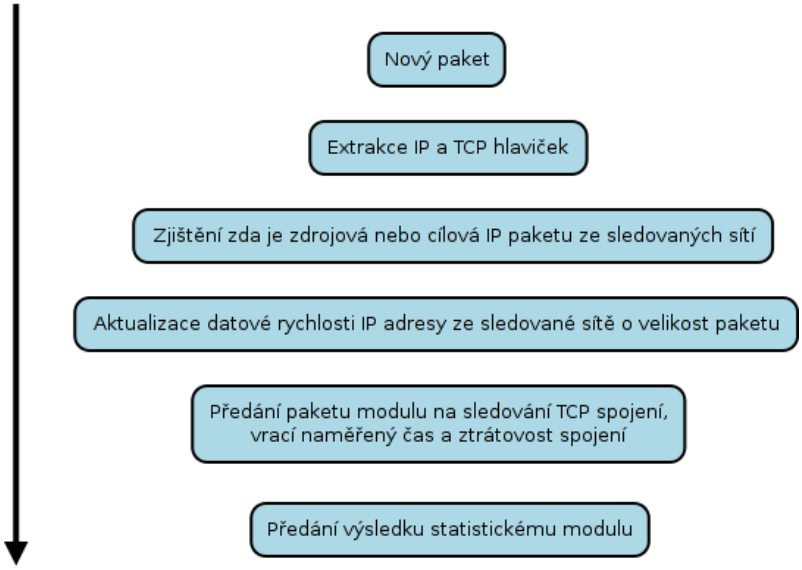
parametrů zadaných při startu programu lze zvolit tzv. online nebo offline mód. Online mód způsobí načítání průchozích paketů přímo ze síťového rozhraní, zatímco v offline módu se data načítají ze souboru ve formátu lipcap [12]. Ten lze vytvořit například pomocí programu tcpdump, který dovede zachycené pakety ukládat přímo do souboru.

Aplikace je také schopna při použití argumentu pcapdir používat specifikovaný adresář jako vyrovnávací buffer mezi programem zachytávajícím pakety a jí samotnou. Schopnost využívat buffer nabízí dvě velké výhody. Především je možné data nejdříve zachytit a při detekci nějaké nestandardní události si později prohlédnout uložené pakety v analytických programech. Existence dvou oddělených procesů (zachytávání a analýza) také dovoluje nastavit jejich prioritu, zpomalení analýzy paketů totiž neohrožuje měření. V případě, že by se zpomalilo načítání paketů ze síťového rozhraní a vyrovnávací paměť rozhraní by pak přetekla, pakety by byly nenávratně ztraceny, což by způsobilo chyby v měření. Odeslaná data by totiž buď nebyla potvrzena (lepší případ), nebo by byla potvrzena až dalším v dalším TCP ACK paketu, který by tak způsobil chybu v měření. Adresář ve funkci vyrovnávací paměti tak může v období velké zátěže (např. během dne) růst a při jejím snížení (např. v noci) se opět zmenšit. Formát souboru libpcap je velmi jednoduchý a v současnosti tvoří víceméně standard v ukládání paketů.

Po spuštění programu se provede inicializace proměnných programů a validace vstupních parametrů. Je-li vše v pořádku, spustí se načítání jednotlivých paketů ze souboru nebo síťového rozhraní. Pro práci se souborem je k dispozici třída PcapParser a její hlavní metoda parse. Předání jména funkce této metodě způsobí, že se tato funkce bude volat s daty paketu jako jejím parametrem.

Jestliže je požadavek na načítání dat přímo ze síťového rozhraní, využívá se knihovny pylibpcap [13]. Tato knihovna umožňuje využívat rozhraní libpcap z jazyka Python. Libpcap (**Lib**rary for **p**acket **cap**ture) je knihovna implementující rozhraní pro zachytávání paketů na síťovém rozhraní. Pomocí její funkce open_live se specifikuje rozhraní, na kterém se budou zachytávat pakety. Pak už stačí pouze v cyklu volat funkci dispatch a v argumentu jí předat jméno funkce, která bude spuštěna při

příchodu paketu na síťové rozhraní (processPacket). Tento cyklus je přerušen pouze při ukončení programu.



Obrázek 3.2: Schéma běhu hlavního programu

Na obrázku 3.2 je znázorněn průchod paketu hlavním programem. V první fázi dojde k extrakci IP a TCP hlaviček z paketu. Nejedná-li se o TCP paket, funkce vrací pouze IP hlavičku. V takovém případě je paket použit jen k aktualizaci údaje o přenosové rychlosti dané stanice na síti. Naopak TCP pakety jsou předány dále modulu pro sledování spojení. Ten vrací nazpět několik údajů. Nejdůležitější je právě naměřená latence a informace, zda je byl tento paket už dříve zachycena a zda se tedy jedná o opakovaný přenos dat spojený s jejich předchozí ztrátou. Tyto dvě hodnoty se v dalším kroku předávají statistickému modulu. Zde se běh programu vrací zpět do hlavní smyčky a volá se dispatch pro získání dalšího paketu.

3.1.2 Správa probíhajících spojení

Tento modul je zastoupen třídami ConnectionManager a Connection. Connection Manager obsahuje databázi všech probíhajících spojení. Jeho rozhraní se skládá z metody add, která nejdříve vyhodnotí zda paket patří ke známému spojení, případně pro daný paket založí nové spojení (objekt Connection). Každé TCP spojení je de-

finováno čtveřicí Ip adresa zdroje, IP adresa cíle, TCP port zdroje, TCP port cíle. Další metodou je cleanOldConnections, po jejímž zavolání dojde k odstranění starých a neaktivních spojení z paměti. Tato metoda je spouštěna v pravidelných intervalech.

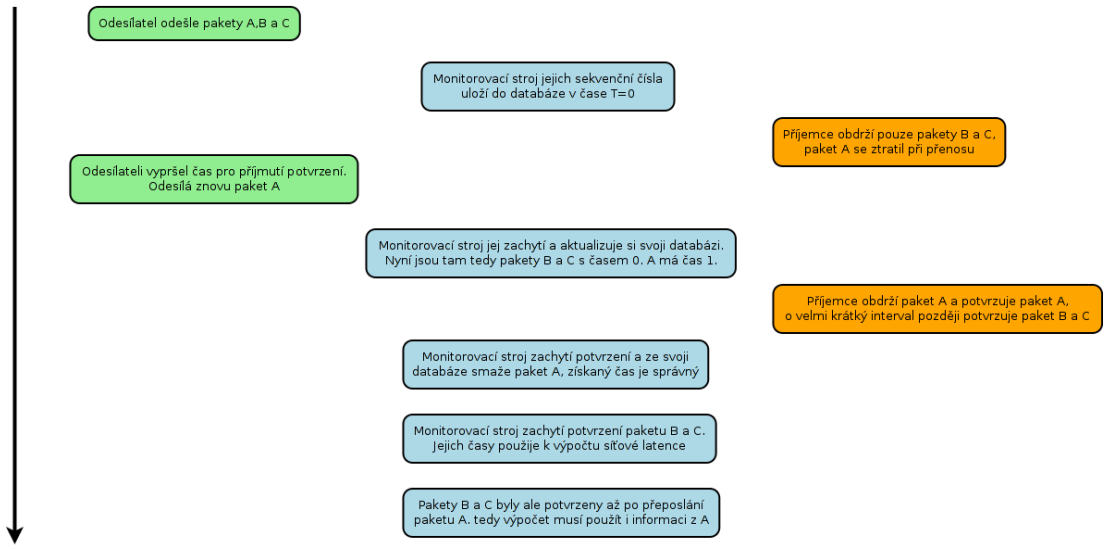
Třída Connection obsahuje dvě dynamická pole pro evidenci paketů příslušného TCP spojení v obou směrech. Dalším atributem této třídy je čas posledního zachyceného paketu ve spojení. Často se totiž stává, že není některé spojení korektně ukončeno, a v paměti by se tak hromadila spojení, která jsou už dávno uzavřena. Při znalosti času poslední aktivity spojení je možné provádět údržbu databáze a neaktivní položky po určité době smazat. Součástí třídy je metoda isRetransmit, která se pro hlavičku paketu v argumentu pokouší nalézt stejný záznam v databázi dosud nepotvrzených paketů. Jestliže je nalezena, znamená to, že stejný paket už byl alespoň jednou odeslán, ale dosud nebyl potvrzen, jedná se tedy o ztrátu paketu.

Hlavní metodou třídy Connection je add, která provádí analýzu paketu v kontextu jeho TCP spojení. Vzhledem k tomu, že jsou potvrzovány pouze datové pakety nebo ty s příznakem SYN, jsou právě jen tyto pakety ukládány do databáze spojení. V úvodu metody se testuje, zda se jedná o opakovaný přenos paketu. Jestliže ano, paket není přidán do databáze[1], je ale pouze aktualizován čas přijetí původního ztraceného paketu. Potvrzuje-li paket příjem dat od protistrany (nastavený ACK příznak), provede se průchod databáze těchto zaznamenaných paketů a všechny, které jsou tímto paketem potvrzené jsou z databáze vymazány. Jako čas pro výpočet rozdílu je použita nejvyšší hodnota z časů přijetí potvrzených paketů. Vždy totiž platí, že posledně přijatý paket z těch právě potvrzených je ten, po kterém příjemce zaslal potvrzení všech předchozích. Jestliže šlo o potvrzení SYN paketu, případně alespoň dvojice datových paketů, je možné použít naměřený čas. Vrací se tedy rozdíl času zaznamenání TCP ACK paketu na síťovém rozhraní a času posledně zachyceného a nyní potvrzeného paketu v opačném směru.

Tento jednoduchý algoritmus se může zdát na první pohled dostačující. Nicméně po prvních testech bylo nutné ještě pro každý směr spojení přidat jednu proměnnou.

[1]Ve skutečnosti databáze neobsahuje celý paket, ale pouze jeho sekvenční číslo, čas přijetí a velikost.

Ta obsahuje čas do té doby nejnovějšího smazaného paketu. Představme si totiž situaci, kdy bylo odesláno několik paketů a řekněme, že ten první se ztratí. Protistrana se pokusí o přeposlání ztraceného paketu, na což příjemce reaguje odesláním TCP ACK paketu (potvrzení přeposlaného paketu). Pravděpodobně z důvodu špatné implementace TCP stacku nebo kvůli velmi velkému zatížení systému ale odešle pouze potvrzení pro přeposlaný paket, a to i přesto, že kromě něj předtím už přijala i několik dalších paketů. To si v zápětí uvědomí a tyto pakety potvrdí dalším ACK paketem. Bez přidaných proměnných by první ACK paket způsobil smazání přeposílaného paketu z databáze. Jenže právě tento paket zdržel všechny ostatní a ACK bylo odesláno až na jeho základě. Bez přeposílaného paketu v databázi zbývají už pouze starší pakety, které byly odeslány spolu s tím prvním ztraceným. Druhý ACK paket by tedy bez přidaných pomocných proměnných způsobil, že by se použil čas přijetí posledně zaznamenaného paketu, což je nesprávné. Toto zvláštní chování se při měření na reálné síti objevovalo pouze u několika stanic a jen zřídka (tehdy když docházelo k ztrátě paketů). Ukázka chybného měření, které by vzniklo bez dodatečných proměnných je na obrázku 3.3.



Obrázek 3.3: Vznik chyby měření

3.1.3 Statistický modul pro zpracování naměřených hodnot z probíhajících a již ukončených spojení

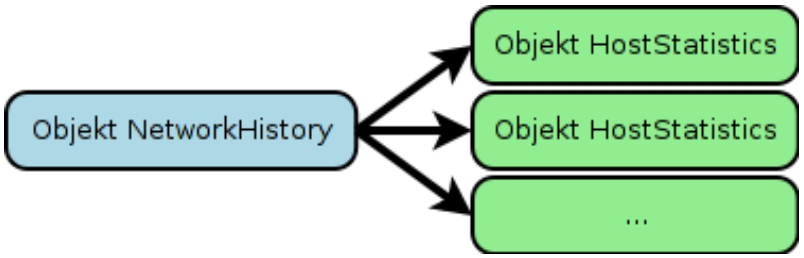
Hlavním úkolem tohoto modulu je základní agregace dat z právě probíhajících spojení. Hodnoty latence a ztráty paketů je totiž potřeba někde uchovávat do doby, než budou zapsány na disk. Objekty Connection existují pouze po dobu trvání TCP spojení a poté jsou z paměti vymazány. Navíc je nutné data z jednotlivých spojení složit dohromady a vytvořit tak statistiku pro danou stanici. Proto bylo nutné do návrhu přidat další třídu.

Statistický modul pracuje v intervalech, během kterých sbírá data. Po uplynutí intervalu jsou data zapsána do souboru a veškeré atributy tohoto modulu smazány a začíná se opět od začátku. Rozdělením na úseky a ukládáním dat pouze z těchto úseků, tak vzniká možnost nezávislého porovnání měření z různých intervalů. Tento modul se skládá ze dvou tříd.

- NetworkHistory - obsahuje databázi všech stanic, které během aktuálního časového intervalu komunikovaly v síti. Obsahuje metodu write, která provede uložení dat modulu do souboru a následný reset modulu. Metoda update se pokusí vyhledat objekt HostStatistics, který se vztahuje k zadané síťové adrese. Jestliže není nalezen je vytvořen nový.

- HostStatistics - obsahuje databázi celkových (pro všechny přenosové rychlosti) naměřených hodnot pro danou síťovou stanici a aktuální časový interval (přenesená data ve směru od a k stanici, pole naměřených hodnot síťových latencí a objem ztracených dat, celkový počet uskutečněných měření)

Třída NetworkHistory tedy obsahuje naměřená data pro všechny stanice, které v aktuálním časovém intervalu komunikují. Každá taková stanice je reprezentována v paměti jedním objektem HostStatistics. Jeho trvání v paměti je omezeno délkou měřících intervalu. Při přechodu mezi intervaly jsou objekty HostStatistics z paměti smazány (reprezentují totiž měření z předchozího intervalu). Vztah jednotlivých tříd je znázorněn na obrázku 3.4.



Obrázek 3.4: Vztah tříd ve statistickém modulu

Tento modul obsahuje metodu update, která provádí přidání nového měření do statistického souboru a metodu write, která provede nejdříve odstranění velmi odlehlých hodnot ze statistického souboru, výpočet charakteristik kvality služby (objem přenesených dat, průměrná latence, směrodatná odchylka latence a ztrátovost paketů pro daný interval a danou stanici) a jejich uložení do souboru pomocí objektu LogWriter (viz oddíl 3.1.4).

Proces odstranění velmi odlehlých hodnot využívá poznatků z Čebyševovi nerovnosti. Pro libovolnou náhodnou veličinu X se střední hodnotou $E(X)$ a rozptylem $D(X)$ je totiž pravděpodobnost, že absolutní hodnota $|X - E(X)|$ nabude hodnoty menší než libovolné $\varepsilon > 0$, omezena právě vztahem 3.1. Pro $\varepsilon = 4 * \varsigma$ lze vzorec 3.1 upravit na tvar 3.2. Hodnota ς je směrodatná odchylka náhodné veličiny X.

$$P(|X - E(X)| < \varepsilon) \geq 1 - \frac{D(X)}{\varepsilon^2} \tag{3.1}$$

$$P(|X - E(X)| < 4 * \varsigma) \geq 0,9375 \tag{3.2}$$

Hodnoty vzdálenější o více než $4 * \varsigma$ od průměru měření jsou tedy velmi odlehlé, neboť v intervalu $4 * \varsigma$ pro libovolnou nádhodnou veličinu leží téměř 94 procent všech hodnot. Tato metoda je doporučována pro filtraci odlehlých dat [14].

Odstraňování vzdálených hodnot nebylo součástí první verze programu, nicméně následná měření prokázala, že některé stanice na měřené síti občas zdrží svoji odpověď o neobvykle dlouhý interval. Stejné chování bylo pozorováno také u měření pomocí příkazu ping. Předpokládá se, že tyto výkyvy jsou způsobené buď chybou

v implementaci TCP protokolu, nebo náhlým velkým zatížením cílové stanice.

3.1.4 Ukládání naměřených hodnot na disk

Tento modul je zastoupen třídou LogWriter. Zavoláním její jediné metody write se provede uložení dat ze statistického modulu na disk. Data jsou pro každou stanici a den měření zaznamenána ve vlastním souboru, jehož název obsahuje IP adresu stanice a den meření. Rozdělení podle dní umožňuje snažší vyhledávání a jednoduchou archivaci starých dat. Každý soubor tedy obsahuje jen měření pro jednu IP adresu a pouze ta, která byla započata v daném dni. Soubory s daty měření mají koncovku snf a ukládají se do adresáře specifikovaného přepínačem outdir nebo defaultně do tmp.

Pro každé měření je do souboru uložena právě jedna řádka obsahující tyto hodnoty:

- čas počátku měření
- celkový objem přenesených dat v bajtech (směrem k cílové stanici) za dobu měření
- celkový objem přenesených dat v bajtech (směrem od cílové stanice) za dobu měření
- počet celkem přenesených paketů (směrem k cílové stanici) za dobu měření
- počet celkem přenesených paketů (směrem od cílové stanice) za dobu měření
- objem ztracených dat v bajtech z celkového objemu
- počet naměřených hodnot latence
- průměrná síťová latence vypočítaná ze všech měření daného intervalu
- směrodatná odchylka průměrné síťové latence

Aplikace používá pro ukládádání dat také soubory ve formátu RRDtool [15]. RRDtool je systém pro průběžné ukládání hodnot, jejich agregaci a vytváření grafických výstupů. Monitorovací aplikace nejdřívě ověří, zda je v cílové složce (parametr -r nebo –rrddir) soubor s databází pro danou stanici. Jestliže tento soubor neexistuje, je vytvořen. Potom již stačí při každém měření pomocí funkce update (knihovna pyrrd) pouze zapsat do databázového souboru hodnoty měření. K vytvoření grafu s hodnotami pak použijeme funkci graph.

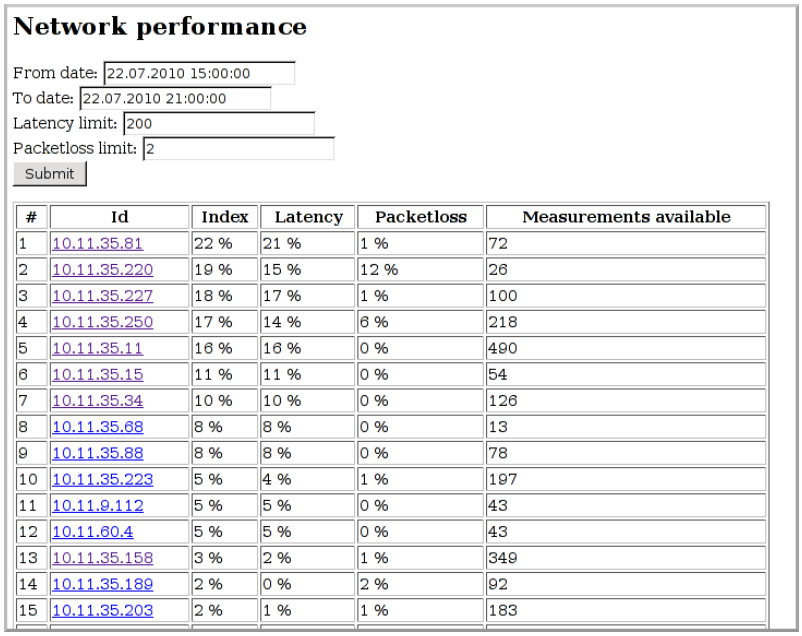
Knihovna pyrrd [16] poskytuje rozhraní k aplikaci RRDtool dostupné z jazyka Python. Hlavní výhodou systému RRDtool je vytváření velmi pěkných grafů, automatické vynechání chybějících hodnot, agregace dat a pevná velikost datových souborů. Databáze je totiž cyklická a při jejím založení se uvede maximální interval, který v ní může být uložen. Po překročení tohoto intervalu se data začnou zapisovat opět od začátku, čímž se smažou nejstarší hodnoty. Systém tak zajišťuje stabilní velikost datových souborů. Další výhodou je automatická agregace dat. RRDtool totiž v databázi ukládá nejnovější hodnoty velmi přesně (zcela bez agregace), u starších hodnot se agregace stupňuje. Snadno je tak možné i s velmi malou velikostí databáze sledovat vývojový trend za poslední rok a přesto mít přesné informace z posledních třiceti minut měření.

3.1.5 Agregace a analýza dat

Posledním důležitým modulem je skript pro analýzu naměřených dat. Jeho úkolem je načíst data měření z výstupního adresáře démona a zpracovat je do výstupu, kterému lze snadno porozumět. Modul není součásti zdrojových kódů démona a je celý napsán v jazyce PHP. Svůj výstup totiž zobrazuje jako webovou stránku. Spuštění skriptu se provádí zadáním jeho adresy do webového prohlížeče.

Načtením odkazu se uživateli zobrazí velmi jednoduchý formulář pro zadání rozmezí datumů, maximální hodnoty latence a maximální procentuální ztráty paketů, pro které chce získat analýzu. Po odeslání žádosti skript provede načtení všech měření z daného časového rozmezí pro všechny stanice. Pro každé měření porovná jeho hodnotu s maximální povolenou hodnotou z formuláře. Jestliže byl limit překročen,

označí kvalitu připojení dané stanice pro dané měření za špatnou. Po načtení všech měření jedné stanice stanoví procentuální podíl měření se špatnou kvalitou služby vůči všem měřením a výsledek uloží. Tento postup je opakován pro všechna měření z daného časového intervalu. V závěru se provede setřídění hodnot podle kvality služby a výstup se zobrazí uživateli. Na obrázku 3.5 je ukázka výstupu prezentačního rozhraní.



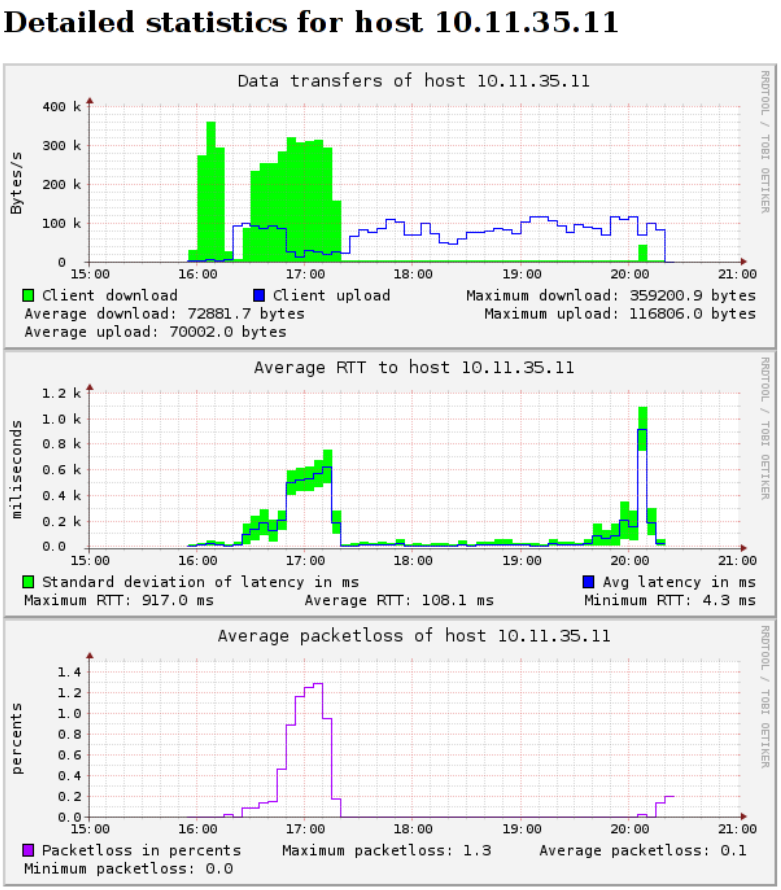
Network performance

From date: 22.07.2010 15:00:00
To date: 22.07.2010 21:00:00
Latency limit: 200
Packetloss limit: 2
Submit

#	Id	Index	Latency	Packetloss	Measurements available
1	10.11.35.81	22 %	21 %	1 %	72
2	10.11.35.220	19 %	15 %	12 %	26
3	10.11.35.227	18 %	17 %	1 %	100
4	10.11.35.250	17 %	14 %	6 %	218
5	10.11.35.11	16 %	16 %	0 %	490
6	10.11.35.15	11 %	11 %	0 %	54
7	10.11.35.34	10 %	10 %	0 %	126
8	10.11.35.68	8 %	8 %	0 %	13
9	10.11.35.88	8 %	8 %	0 %	78
10	10.11.35.223	5 %	4 %	1 %	197
11	10.11.9.112	5 %	5 %	0 %	43
12	10.11.60.4	5 %	5 %	0 %	43
13	10.11.35.158	3 %	2 %	1 %	349
14	10.11.35.189	2 %	0 %	2 %	92
15	10.11.35.203	2 %	1 %	1 %	183

Obrázek 3.5: Hlavní stránka prezentančního rozhraní

Správce sítě si tak v úvodu skriptu nastavil hranici, po jejímž překročení je už kvalita služby nevyhovující. Z výstupu lze potom snadno odvodit, jak často jsou tyto limity překročeny. V tabulce srovnávající kvalitu připojení měřených stanic je možné kliknout na odkaz u IP adresy dané stanice. Tím dojde k načtení grafů, které byly při kliknutí na odkaz vytvořeny aplikací RRDtool. Změnou časového intervalu lze také měnit data zobrazená v grafech. Obrázek 3.6 zobrazuje ukázkové grafy.



Obrázek 3.6: Grafy prezentančního rozhraní

3.2 Argumenty programu

Možné argumenty při spuštění programu shrnuje následující výčet.

- **-t, –tazmen** - jestliže je na příkazovém řádku uveden tento přepínač, program analyzuje pouze UDP datagramy ve formátu protokolu TaZmen. Tento protokol umožňuje zachytávání dat například na routerech nebo switchovacích zařízeních v sítí a jejich přenos na libovolnou IP adresu. Získané pakety jsou zabaleny dovnitř UDP datagramů pomocí, kterých jsou přeneseny na libovolnou IP adresu. Protokol UDP negarantuje doručení paketů, je tedy žádoucí, aby bylo spojení mezi zachytávací aplikací a příjemcem stabilní a bez ztrátovosti dat. Kvůli maximální velikosti přenosové jednotku (MTU) může docházet

k jejich oříznutí a paket tedy může být zkrácen. Pomocí tohoto protokolu lze uskutečnit analýzu dat, která procházejí přes vzdálený router prostým přeposláním na monitorovací stanici. Hlavičky protokolu bohužel neobsahují čas zachycení, takže se musí použít čas jejich doručení na cílový stroj. Výsledné časy jsou tedy zkreslené o dobu zpoždění mezi zachytávacím routerem a démonem (tento spoj musí tedy mít minimální síťové zpoždění při doručování paketů). Dalším omezením je snížení výkonu monitorovací aplikace, neboť je potřeba nejdříve načíst UDP hlavičku s daty a pak z dat paketu extrahovat ethernet, IP a TCP hlavičku. Úvodní práce s paketem tak zabírá dvakrát více času. Protokol libpcap také nabízí možnost určit počátečních X byte z každého paketu, které se mají při zachytávání načítat do paměti (nebo ukládata do souboru). Aplikace potřebuje pro svoji práci pouze IP a TCP hlavičku, takže je ji stačí prvních z každého paketu pouze prvních 48 bajtů (14 bajtů pro ethernet hlavičku, 20 bajtů pro IP hlavičku (maximum je až 60 bajtů, ale drtivá většina paketů má jen 20 bajtů) a 14 bajtů z úvodní části TCP hlavičky, zbytek paketu program ke své činnosti nepoužívá). Při použití protokolu TaZmen je ovšem potřeba načíst minimálně 95 bajtů (2x ethernet hlavička, 2x IP hlavička, 1x TaZmen hlavička(5 bajtů), 1x UDP hlavička (8 bajtů) a 1x TCP hlavička). V případě, že jsou zachycené pakety nejdříve ukládány do souboru, použití TaZmen protokolu, zvětší jeho velikost téměř na dvojnásobek.

- **-f, –pcapfile** - jméno souboru v libpcap formátu, který obsahuje zachycené pakety. Po zpracování souboru se program ukončí.

- **-d, –pcapdir** - specifikuje adresář, kde jsou uloženy soubory se zachycenými pakety. Aplikace se pokusí soubory otevřít v pořadí podle jejich stáří. Začne tím nejstarším. Po zpracování paketů ze souboru jej smaže a vyhledá aktuálně nejstarší soubor v adresáři. Celý proces se opakuje do chvíle, kdy zůstává v adresáři jen jeden soubor. Tehdy se aplikace uspí na deset vteřin a po jejich uběhnutí si ověří zda v adresáři přibyl nový soubor. Jestliže ano, proces se opakuje, jestliže ne, probouzí se každých deset vteřin, aby ověřila zda v adresáři

něco přibylo. Na první pohled toto chování může působit zvláštně, ale jedná se o mechanismus spolupráce s programem tcpdump. Ten totiž zachytává pakety na síťovém rozhraní a získaná data ukládá do souborů s libpcap formátem. Pomocí přepínačů lze nastavit, aby vytvářel soubory jen určité maximální velikosti. Tcpdump ukládá data ihned po jejich zachycení do souboru. Je tedy nanejvýš rozumné počkat s jeho zpracováním až do chvíle než jej tcpdump opustí a začne s vytvářením nového. Aplikace se tedy stranní nejnovějšího souboru v adresáři oprávněně.

- **-o, –outdir** označuje adresář, do kterého budou ukládány výstupy aplikace
- **-i, –interface** označuje síťové zařízení určené k zachytávání paketů. Jestliže je použit tento přepínač, aplikace načítá data přímo z rozhraní v promiskuitním režimu. K dispozici jsou tedy i data, jejichž adresátem není monitorovací počítač. Na operačním systému GNU Linux je ale nutné, aby byla spouštěna s právy roota.
- **-n, –networks** obsahuje seznam IP rozsahů, které jsou na místní LAN síti a budou tedy podléhat měření. Formát zadání jsou dvojice oddělené čárkou, kde každá dvojice obsahuje počáteční a koncovou adresu oddělené pomlčkou. Například obsahuje-li měřená síť IP adresy od 10.0.0.1 do 10.0.0.20 a 192.168.0.1 do 192.168.0.254, pak bude formát následující: –networks 10.0.0.1-10.0.0.20,192.168.0.1-192.168.0.254. Tento seznam by měl být co nejkratší, protože se při příjmu každého paketu musí ověřovat jeho příslušnost k některému z rozsahů v lokální síti
- -r –rrddir označuje adresář, který obsahuje soubory s RRDtool databázemi přenosů jednotlivých stanic na síti

3.3 Instalace monitorovací aplikace

Instalace aplikace je velmi jednoduchá. Předpokládaný operační systém je GNU Linux, nicméně s malými úpravami je možný díky přenositelnosti Pythonu provoz i

na ostatních systémech. Samotnou aplikaci není nutné instalovat, stačí pouze spustit příkazem Sniffmon.py. K běhu bude potřebovat jazyk Python, knihovnu pyrrd (v systému Debian balíček python-pyrrd), knihovnu pypcap (jen v případě použití obline módu, v systému Debian balíček python-pypcap) a aplikace RRDtool (v systému Debian balíček rrdtool). Prezentační webové rozhraní vyžaduje nainstalovaný webový server s podporou jazyka PHP. Skript prezentačního rozhraní stat.php se zkopíruje do adresáře webového serveru. Skript sám o sobě neobsahuje správu přístupu uživatelů. Lze ji buď snadno implementovat v jazyce PHP nebo využít služeb webového serveru (např htaccess u služby Apache). Po spuštění monitorovacího procesu je možné kdykoliv přistupovat k webovému rozhraní a dotazovat se na aktuální stav měření.

Kapitola 4

Výsledky

4.1 Ověření předpokladů

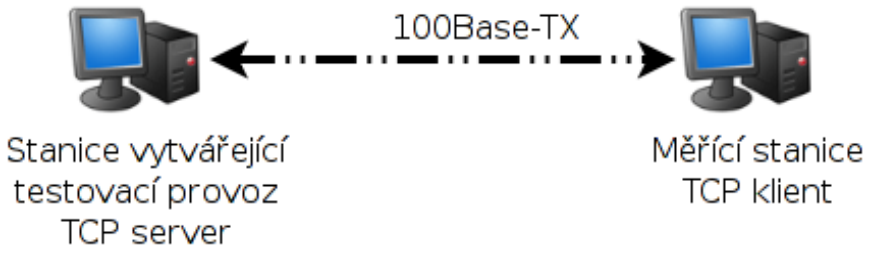
Během návrhu algoritmu démona bylo stanoveno několik předpokladů, které je nutné pomocí měření také potvrdit. Jedná se o následující.

- Hodnota získaná z rozdílu času detekce SYN ACK paketu a ACK paketu potvrzujícího tento SYN ACK odpovídá síťové latenci
- Hodnota získaná z rozdílu času detekce FIN paketu a ACK paketu, který jej potvrzuje, odpovídá síťové latenci
- Zpožděné ACK neumožňují použít libovolný TCP ACK paket k měření latence

Testy jsou provedeny na síti tvořené dvojicí počítačů propojených mezi sebou metalickým ethernetem (100Base-TX). Jeden z nich (TCP server) odesílá data na druhou stanici (TCP klient). Na té je spuštěn program Wireshark, který ukládá veškerou síťovou komunikaci do souboru. Obrázek 4.1 znázorňuje zapojení sítě.

4.1.1 Zpracování odpovědi na ICMP echo paket

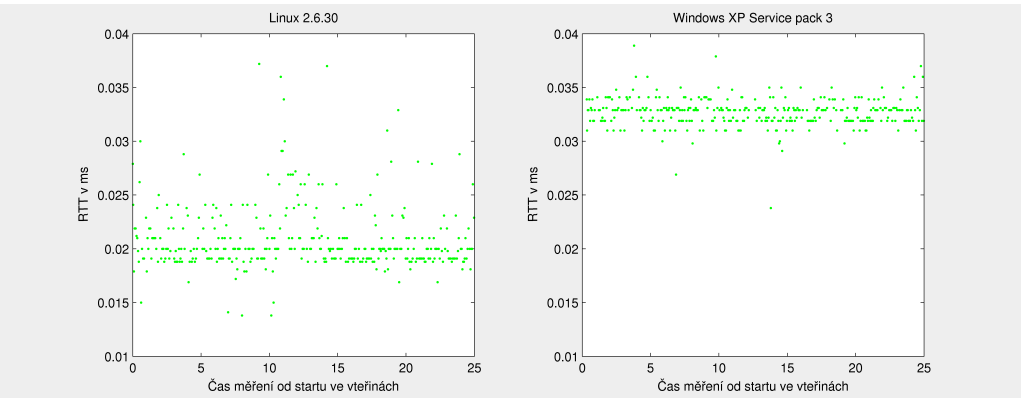
V následujících odstavcích provedeme srovnání hodnot měřených monitorovacím démonem s hodnotami získanými pomocí paketu ICMP echo. Ten je používán nástrojem ping a slouží k zjištění síťové latence a ztrátovosti paketů na cestě mezi ode-



Obrázek 4.1: Schéma sítě pro ověření předpokladů

sílatelem a příjemcem. Program ping odesílá v pravidelných intervalech požadavek na vzdálenou stanici. Vytvořený paket může obsahovat také data. Pokud požadavek není u příjemce blokován, odpovídá nazpět paketem se stejnými daty, která byla v přijaté v žádosti.

Pro potřeby srovnání výsledků monitorovacího démona s výsledky programu ping byla vytvořena aplikace měřící čas, který systém potřebuje pro vytvoření odpovědi na ICMP echo paket. Měření proběhla na systému MS Windows SP3 a GNU Linux s jádrem 2.6.30. Zjištěné hodnoty obsahují pouze dobu zpracování na cílovém počítači bez časů, které paket potřebuje pro cestu v síti. Tabulka a graf 4.2 zobrazuje výsledky testu. Získané časy budou použity v následujících měření pro srovnání výsledků s monitorovacím démonem.



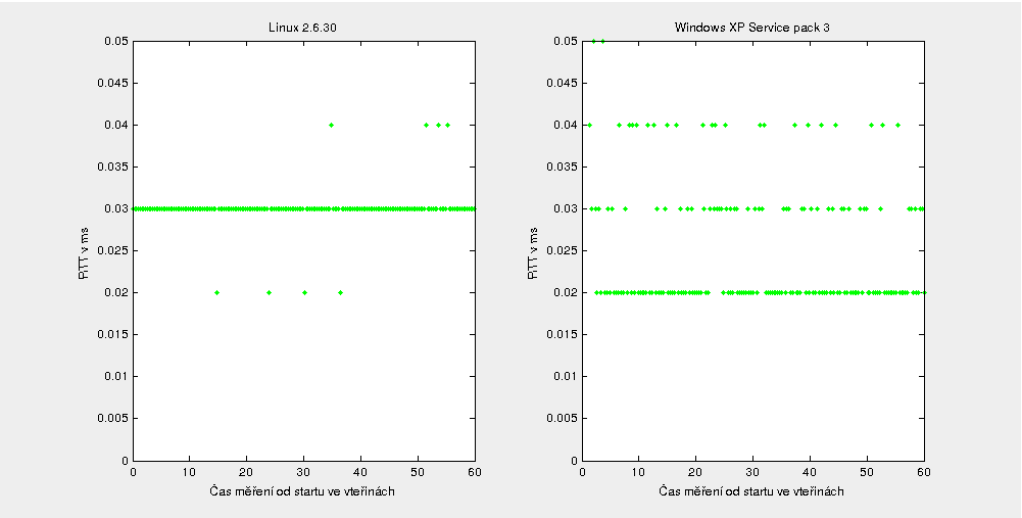
Obrázek 4.2: Čas potřebný ke zpracování odpovědi na Icmp echo paket

Tabulka 4.1: Čas potřebný ke zpracování odpovědi na Icmp echo paket

OS	Průměr v ms	Směrodatná odchylka v ms	Počet měření
Linux 2.6.30	0,0208	0,0031	300
Win XP SP3	0,0327	0,0016	300

4.1.2 Zpracování odpovědi na SYN paket

Malou změnou v monitorovacím démonu lze zajistit, aby aplikace zpracovávala jen pakety, které mají nastavený buď příznak SYN, nebo potvrzují jiný SYN paket. Tím je zajištěno, že vrátí pouze požadované hodnoty. Generování paketů probíhá pomocí dvojice jednoduchých skriptů (TCP server a TCP klient) v Pythonu a data jsou zachytávána pomocí programu Wireshark na stanici, kde běží TCP klient skript. Klientská aplikace se pouze připojí na server a po chvíli se zase odpojí. Testy byly provedeny na operačním systému GNU Linux (jádro 2.6.30) a MS Windows XP Service Pack 3. Zobrazené časy odpovídají době mezi přijetím potvrzení SYN paketu a jeho potvrzením (tedy SYN ACK a ACK paket). Získané hodnoty tak reprezentují pouze čas strávený zpracováním požadavku bez cesty paketu v síti. Výsledky jsou obsaženy v grafu 4.3 a tabulce 4.2.



Obrázek 4.3: Čas potřebný ke zpracování odpovědi na TCP SYN ACK paket

Srovnáním časů v tabulce 4.2 s hodnotami získanými pomocí příkazu ping lze

Tabulka 4.2: Čas potřebný ke zpracování odpovědi na TCP SYN ACK paket

OS	Průměr v ms	Směrodatná odchylka v ms	Počet měření
Linux 2.6.30	0,0297	0,0024	300
Win XP SP3	0,0255	0,0076	300

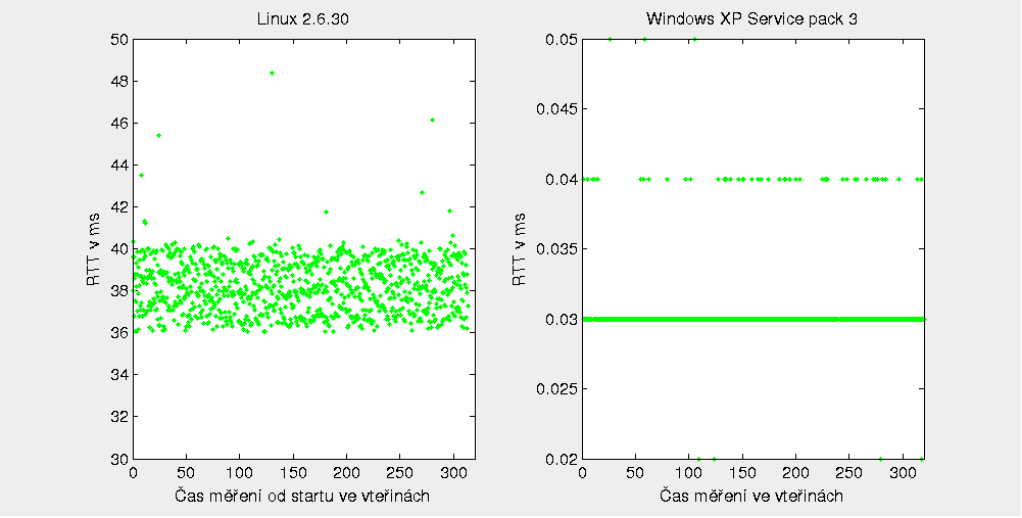
Tabulka 4.3: Čas potřebný ke zpracování odpovědi na TCP FIN paket

OS	Průměr v ms	Směrodatná odchylka v ms	Počet měření
Linux 2.6.30	38,3057	1,765	300
Win XP SP3	0,0305	0,0025	300

dojík k závěru, že časy potřebné k zpracování odpovědi jsou velmi podobné a zanedbatelné vzhledem k času, který paket potřebuje na cestu sítí. Monitorovací démon tedy s pomocí SYN paketů získává hodnoty, které odpovídají síťové latenci.

4.1.3 Zpracování odpovědi na FIN paket

Pro měření byla použita data z předchozího experimentu. Monitorovací démon byl upraven tak, že měří pouze reakci potřebnou na zpracování potvrzení FIN paketu. Výsledky pokusu jsou obsaženy v grafu 4.4 a tabulce 4.3.



Obrázek 4.4: Čas potřebný ke zpracování odpovědi na TCP FIN paket

Z tabulky 4.3 lze jednoznačně odvodit velký rozdíl v časech potřebných ke zpracování FIN paketů. U obou měření bylo zajištěno, že všechny pakety dorazily včas a v pořadí v jakém byly odesláno. Není tedy možné, že v případě Linuxu bylo potvrzení FIN paketu pozdrženo kvůli čekání na předchozí dosud nedoručená data. Chování Linuxu je tedy poněkud zvláštní, neboť dle standardu je na FIN požadavek jediná možná odpověď, a to je jeho potvrzení. Pravděpodobně Linux předpokládá, že po ukončení spojení z jedné strany bude následovat okamžité ukončení spojení i z druhé strany. Proto vyčkává po určitou dobu a je-li spojení opravdu ukončeno i z druhé strany, pak potvrzení FIN paketu odesílá spolu s RST příznakem pro okamžité ukončení spojení. V optimálním případě tak lze ušetřit jeden TCP ACK paket. Chování systému Windows je trochu jiné, potvrzení odesílá vždy a okamžitě. Tento rozdílný přístup tak nedovoluje použít TCP FIN pakety k měření síťové latence, neboť by zvyšoval její hodnotu u stanic s operačním systémem Linux.

Možné použití by se ale našlo u programů pro detekci operačních systémů (např. nmap). Stačilo by navázat spojení na libovolný otevřený port a změřit odpověď na SYN paket v jeho úvodu a FIN paket při ukončení spojení. Případně by šlo využít toho, že Windows XP odpovídají na TCP FIN požadavek vždy FIN ACK paketem následovaným RST paketem pro uzavření druhé strany spojení. Linux odpoví buď zpožděně ACK paketem nebo RST paketem okamžitě. Rozdílné je i ukončování spojení u těchto systému. Windows reagují na požadavek program na ukončení TCP spojení FIN paketem, zatímco Linux provádí rychlé ukončení resetem spojení (paket TCP RST).

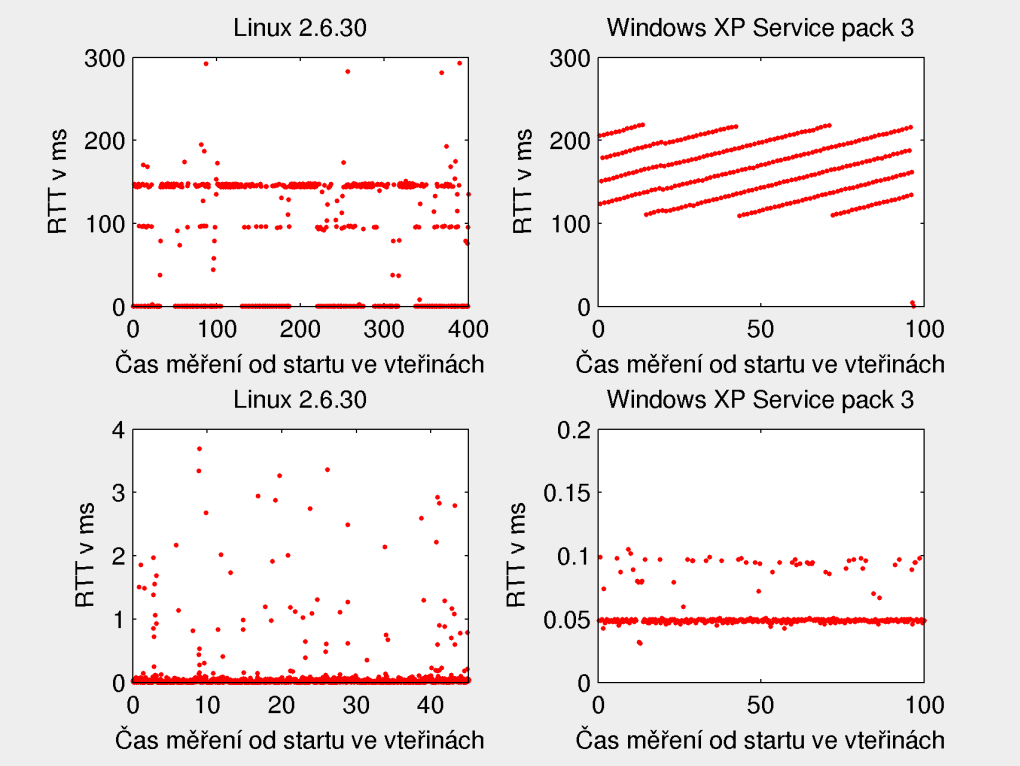
4.1.4 Vliv zpožděných ACK

V následujícím odstavci bude provedeno srovnání mezi časy, které potřebuje operační systém pro zpracování ACK odpovědi na přijatá data. Předpokladem měření je, že obecně lze využít pouze časy, které jsou získané z potvrzení alespoň dvojice paketů libovolné délky. Úprava monitorovacího démona umožnila měřit jen hodnoty, které jsou získané z rozdílu času detekce paketu a jeho potvrzení. K testu byl použit tcp server a klient skript z předchozích pokusů. K získání hodnot byly potřeba dvě

Tabulka 4.4: Čas potřebný ke zpracování ACK odpovědi na data

OS	Průměr v ms	Směrodatná odchylka	Počet měření
Linux 2.6.30 1 paket	101,7223	73,9261	1520
Win XP SP3 1 paket	163,5422	33.9851	1000
Linux 2.6.30 2 pakety	0,0467	0,2424	4940
Win XP SP3 2 pakety	0,0561	0,0163	1000

fáze. V první TCP server odesílal malý objem dat (jeden paket) v pravidelných intervalech na klientskou stanici (TCP klient). Zde byl spuštěn program Wireshark pro zachytávání paketů a příslušných vygenerovaných potvrzení. Naměřené hodnoty, stejně jako v předchozích testech, neobsahují čas potřebný pro cestu paketu sítí. V druhé fázi testu byly podmínky pozměněny pouze tak, že server odesílal větší objem dat (dva a více paketů) zároveň. Tím bylo umožněno měření reakce systému na jeden a více paketů a získány tak hodnoty pro graf 4.5 a tabulku 4.4.



Obrázek 4.5: Čas potřebný ke zpracování ACK odpovědi na data

Graf 4.5 znázorňuje v levém sloupci čas, který potřeboval během testu Linux k vy-

generování potvrzení přijatých dat. Horní grafy zobrazují hodnoty získané v případě, že ACK paket potvrzoval jen jeden paket a dolní pro dva a více paketů. V pravém sloupci jsou hodnoty pro systém Windows XP. Z tabulky je zřejmé, že obecně nelze použít časy získané pouze z potvrzení jednoho paketu. Windows zde totiž jednoznačně pozdržují odpověď. Linux vykazuje komplikovanější chování, někdy totiž potvrzuje okamžitě a někdy potvrzení odloží. Z grafu lze vyvodit, že časy pro potvrzení dvou a více paketů jsou podobné časům získaným pomocí příkazu ping a vzhledem k latenci sítě zanedbatelné. Měření síťové latence tedy lze realizovat pomocí této metody.

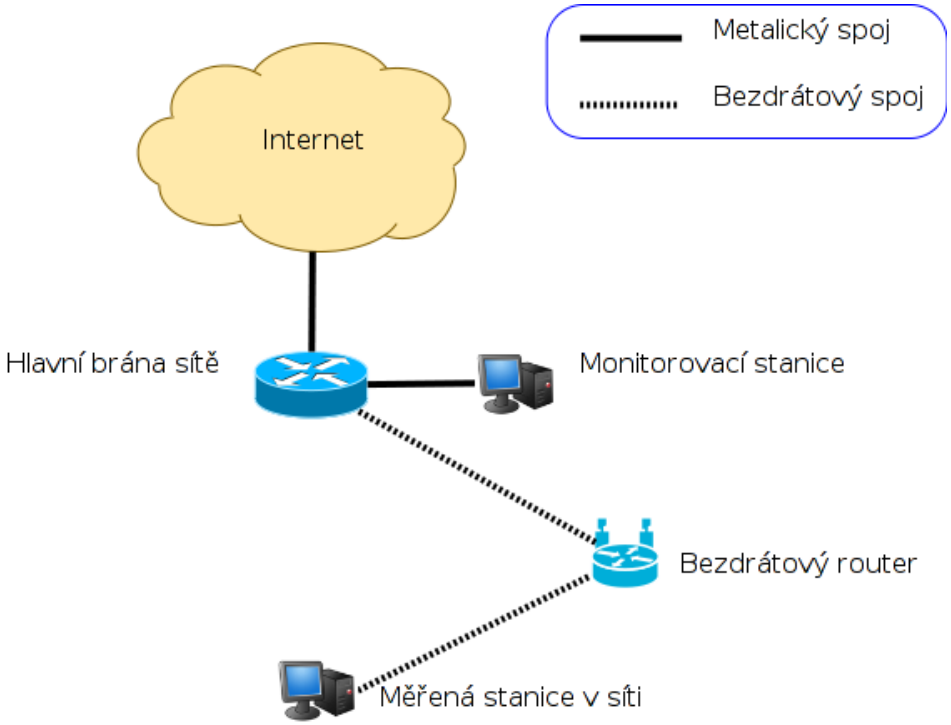
4.2 Srovnání s aplikací ping

V předchozí kapitole byl ověřen předpoklad, že reakce systému je pro ICMP echo pakety i TCP ACK pakety shodná. Provedená měření testovala dobu vytvoření odpovědi operačním systémem bez započtení času pro cestu paketu sítí. Nyní bude tedy provedeno srovnání výsledků obou metod na reálné síti.

Pro potřeby měření se využije část přístupové sítě lokálního poskytovatele internetu. Router na hlavní bráně do internetu provádí přeposílání všech dat na monitorovací stanici. Toto spojení disponuje dostatečnou kapacitou a je realizováno pomocí přímého propojení obou strojů metalickým ethernetem. Tím je zajištěno téměř konstantní sítově zpoždění. Část sítě mezi hlavní branou a měřenou stanicí je tvořena dvojicí bezdrátových spojů. Zapojení sítě je znázorněno na obrázku 4.6.

Před začátkem měření byla na monitorovacím stroji spuštěna dvojice příkazů ping. Každý z nich byl vysílán v intervalu 200ms s cílovou adresou měřené stanice a velikostní přenášených dat 1500 bajtů a 54 bajtů. Volba těchto hodnot není náhodná, neboť 54 bajtů je minimální velikost TCP ACK paketu a 1500 bajtů je maximální velikost paketu na síti typu ethernet. Dvojice těchto pingů tedy simulovala odeslání plného TCP paketu a jeho potvrzení příslušným TCP ACK paketem (v obou směrech). Dále bylo na cílové stanici spuštěno stahování velkého souboru z internetu (průměrná rychlost stahování byla kolem 900 kbps, rychlost byla během testu sta-

bilní). Tímto nastavením bylo zajištěno, že cesta ICMP paketů bude shodná jako měřená cesta TCP paketu (viz obrázek 4.6.

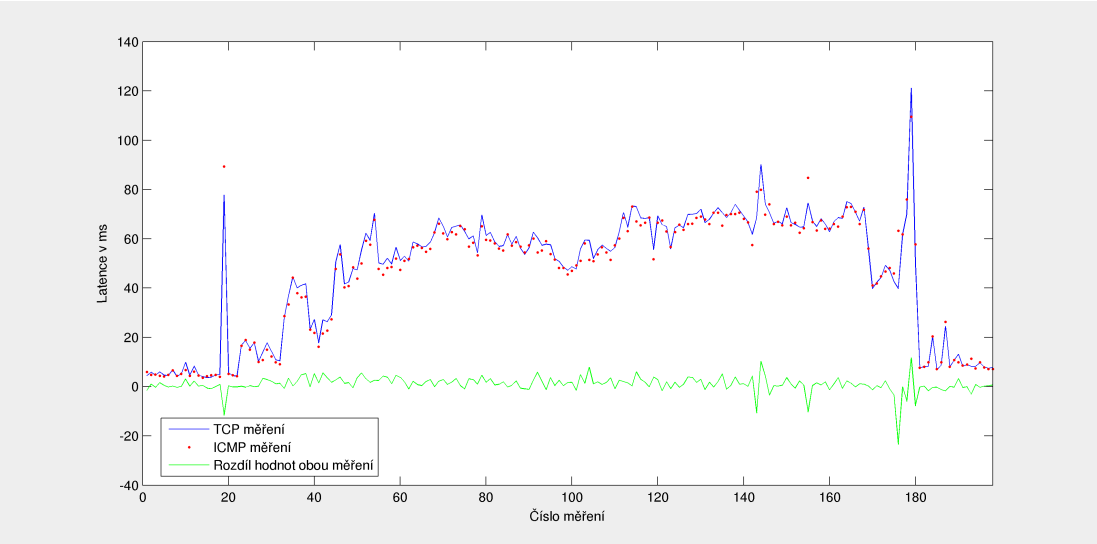


Obrázek 4.6: Umístění monitorovacího stroje a měřené stanice v síti

Jakmile běželo stahování z internetu i dvojice příkazů ping, byl spuštěn program tcpdump, který během intervalu 100 minut shromažďoval hlavičky paketů veškeré síťové komunikace cílové stanice. Takto získaná data byla následně analyzována.

Pro účely měření byla navržena aplikace, která podobně jako monitorovací démon projde soubor zaznamenaných paketů a vyhledá všechny dvojice ICMP paketů (požadavek, odpověď). Ze vstupních dat tak získáme průměry latence a ztrátu paketů v po sobě jdoucích třicetisekundových intervalech. Tyto intervaly jsou zcela shodné s výstupem TCP monitorovacího démona. Byla tak získána data pro stejné intervaly, ale pomocí rozdílných měřících metod. Srovnání obdržených výsledků je znázorněno na grafech 4.7 a 4.8.

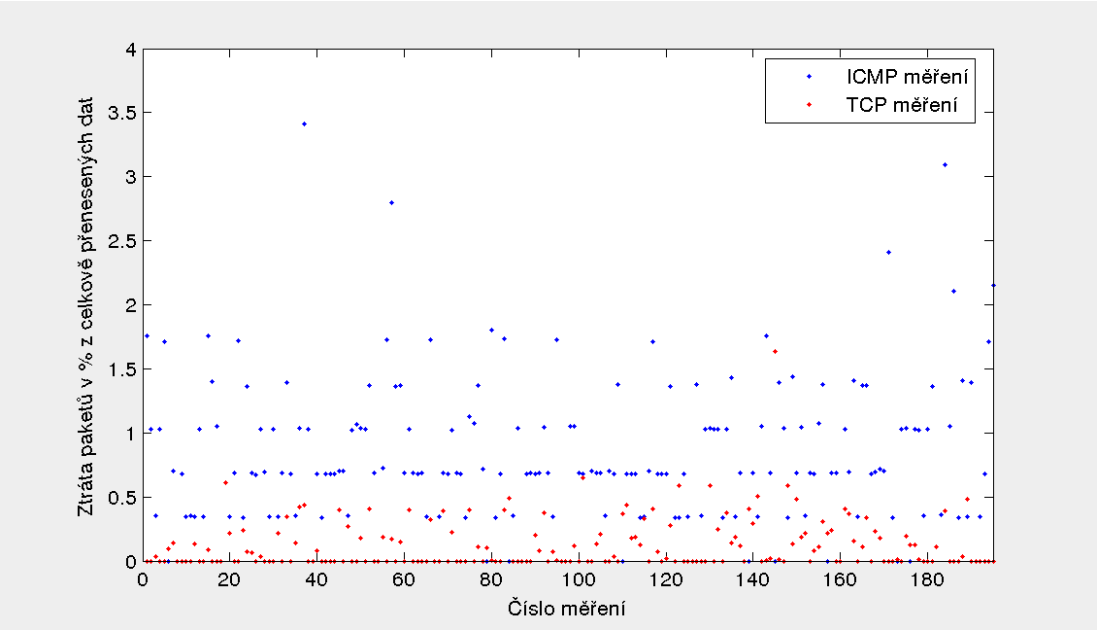
Graf 4.7 srovnává výsledky 197 měření (každé trvá třicet vteřin) získaných během jednoho souvislého časového intervalu. Zelená křivka znázorňuje rozdíly měření latence u obou metod. Průměrná hodnota rozdílu ve sledovaném intervalu je 0,82 mi-



Obrázek 4.7: Srovnání výsledků měření sítové latence pro obě metody

lisekundy (směrodatná odchylka 4,8 milisekundy). Z výsledků měření tedy plyne, že obě metody na reálné síti poskytují téměř shodné hodnoty průměrné síťové latence.

Graf 4.8 zobrazuje výsledky měření ztráty paketů pomocí stejné dvojice metod. V tomto případě jsou výsledky ale velice rozdílné, průměrná hodnota u metody ICMP je 0,85% zatímco u TCP jen 0,15%. Zde nás výsledek ale nemůže překvapit. TCP metoda měření ztráty paketů totiž dovede detekovat spolehlivě pouze ztrátu paketů ve směru od měřícího bodu k cílové stanici. V opačném směru je situace složitější. TCP protokol se totiž dovede se ztrátou TCP ACK paketu vyrovnat, jestliže v krátké době po tomto paketu je odeslán další TCP ACK paket. Ztráta prvního z nich bude vykompenzována doručením druhého paketu, který z principu musí potvrzovat alespoň data předchozího TCP ACK paketu. Lze tedy říci, že hodnota ztráty paketů měřena TCP metodou odpovídá za určitých podmínek chybovosti jen v jednom směru (z pohledu měřené stanice jde o příjem dat). Výsledek záleží na objemu přenášených dat, vzdálenosti paketu od předchozího paketu, koncových operačních systémech a latenci konkrétního TCP spojení. Minimální získaná hodnota je ztrátovost pouze v jednom směru, maximální možná hodnota je ztrátovost v obou směrech (tedy stejné číslo jako u ICMP metody). Převodní vztah mezi mezi výsledky obou metod lze tak jen velmi obtížně odhadovat. Je tedy vhodné měřené hodnoty u obou metod nesrovnávat, ale



Obrázek 4.8: Srovnání výsledků měření ztráty paketů pro obě metody

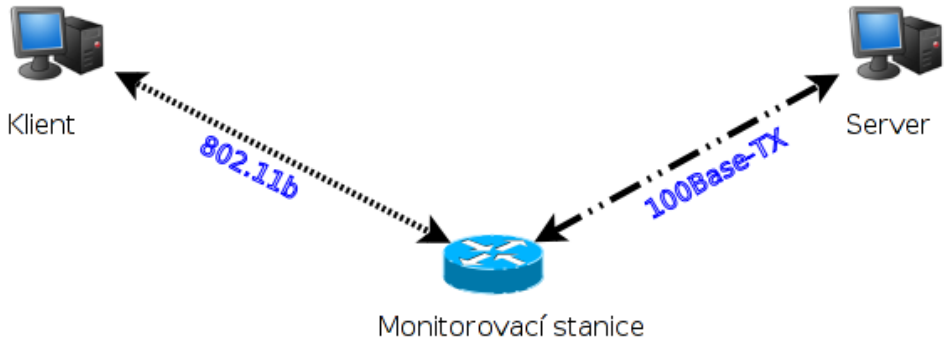
uvažovat je jako samostatné veličiny popisující kvalitu služby.

4.3 Stahování velkého souboru

Stahování velkého souboru je test, který si klade za cíl předvést měřící schopnosti programu v modelové situaci. Během tohoto testu byl vypnut statistický modul a nedochází tedy k agregaci dat přes delší časové intervaly. Výstupem jsou tedy přímo hodnoty naměřených síťových zpoždění a počty použitých a celkově přijatých paketů. Takto lze pozorovat okamžitou reakci programu na změnu síťových parametrů.

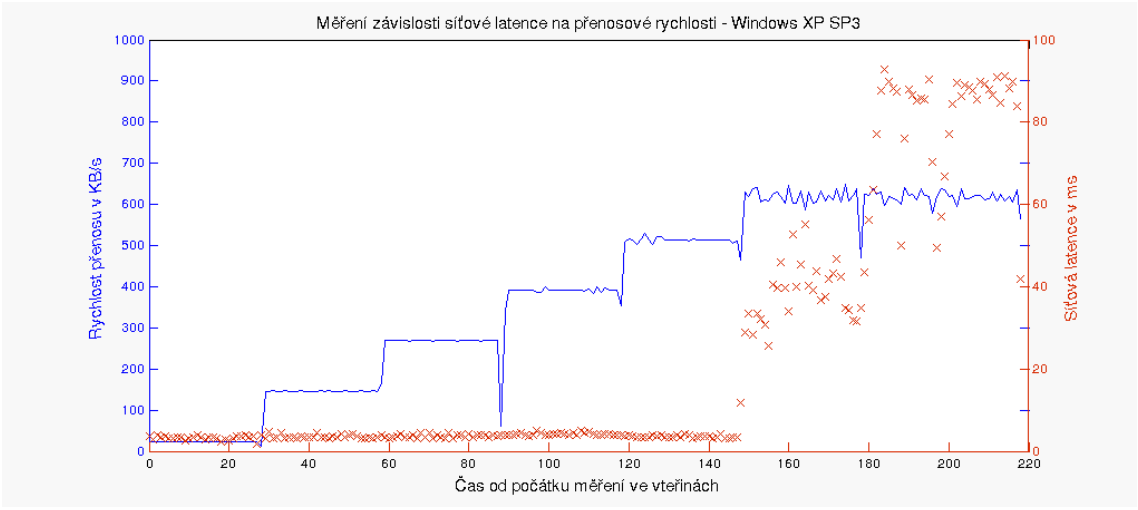
V následujícím textu bude ověřen vztah mezi síťovou latencí a přenosovou rychlostí na bezdrátovém rozhraní standardu 802.11b a též velikost podílu paketů použitelných k měření na celkovém počtu serverem přijatých TCP ACK paketů.

Zapojení sítě pro následující měření je zobrazeno schématem 4.9. Během testu byl stahován velký soubor pomocí protokolu HTTP ze serveru na klientskou stanici. Mezi serverem a klientem je umístěna monitorovací stanice, která zachytává průchozí pakety. Spoj mezi monitorovací stanicí a serverem je metalický ethernet, zatímco



Obrázek 4.9: Zapojení sítě pro test stahování velkého souboru

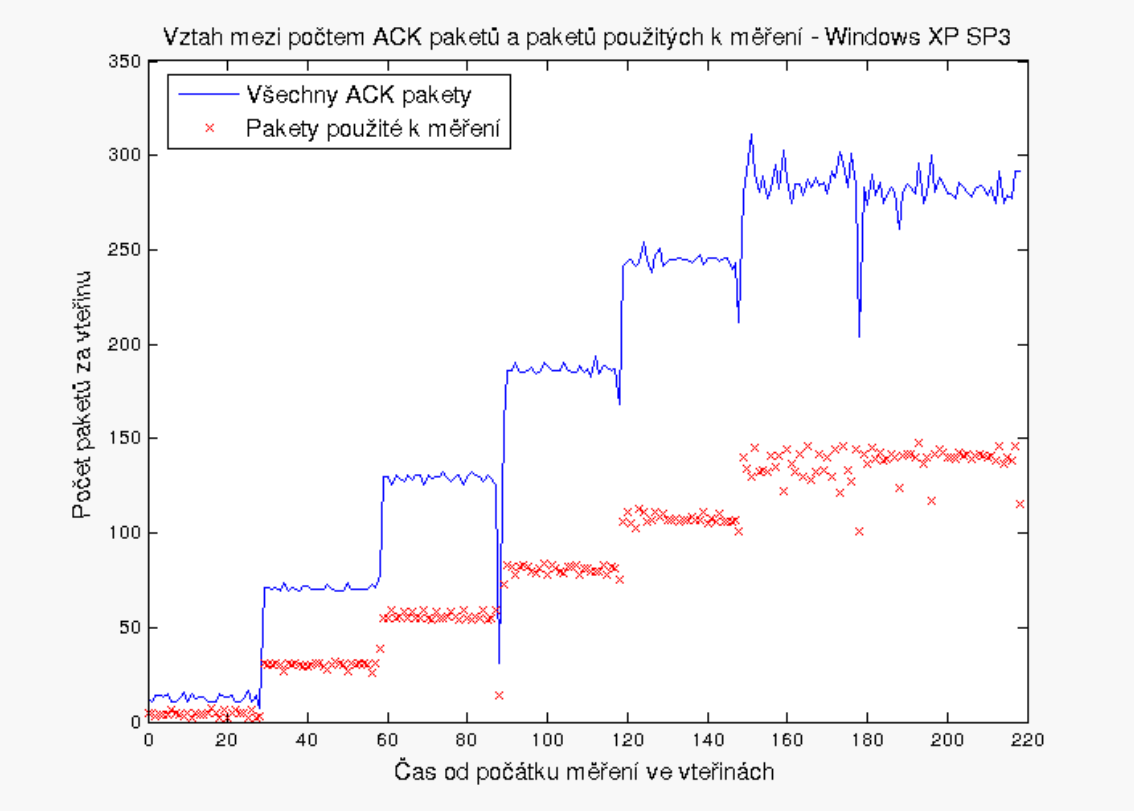
propojení mezi monitorovací stanicí a klientským počítačem je realizováno pomocí bezdrátového spoje standardu 802.11b. Vytvořená aplikace měří síťovou latenci právě na tomto spoji. Webový server také disponuje softwarem pro omezování rychlosti stahování. Ta je v úvodu nastavena na 200 kbit/s. Každých 30 vteřin je rychlost zvýšena o 1000 kbit/s. Jakmile je rychlost zvýšena na 7200 kbit/s, test je ukončen.



Obrázek 4.10: Závislost síťové latence na přenosové rychlosti

Na obrázku 4.10 je znázorněna závislost síťové latence (měřené pomocí monitorovací aplikace) na rychlosti stahování souboru na klientskou stanici pro systém Windows XP SP3. Z obrázku je dobře vidět maximální rychlost bezdrátového rozhraní kolem 5000 kbit/s. Po jejím dosažení se prudce zhoršila síťová latence při přenosu. K další změně došlo ve chvíli, kdy je serverový omezovač rychlosti navýšen na 6200

kbit/s.

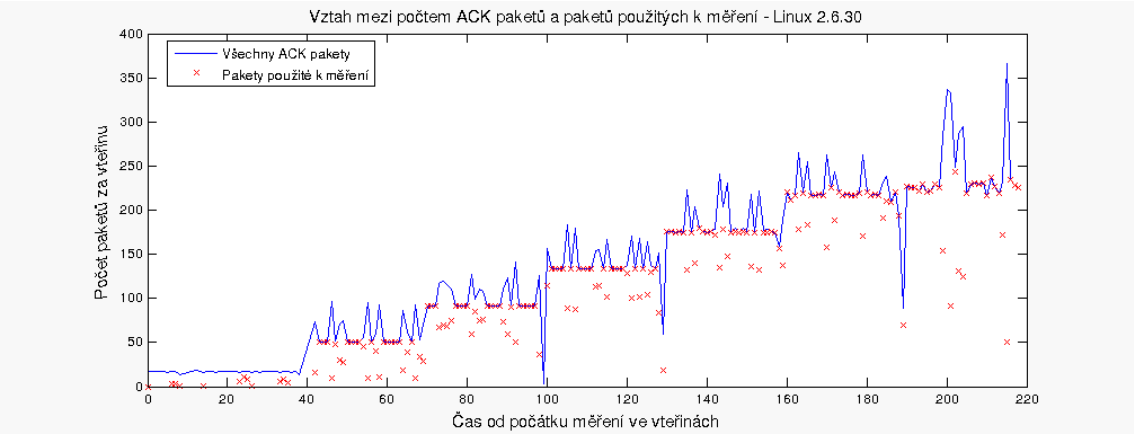


Obrázek 4.11: Podíl paketů pro měření na celkovém počtu příchozích TCP ACK paketů na server

Obrázek 4.11 zobrazuje počet zaznamenaných příchozích TCP ACK paketů na server a také objem paketů, který byl využit pro měření. Data jsou stejná jako v předchozím případě. Z grafu je patrné, že výsledný podíl je téměř konstatní, a to i při změnách rychlosti.

Grafy 4.12 a 4.13 zobrazují výsledky pro stejné zapojení sítě jako v předchozím případě. Jedinou změnou je operační systém na klientské stanici, MS Windows XP SP3 byl zaměnen za GNU Linux 2.6.30. V obrázku 4.12 vykazuje tento systém agresívnější chování. To vede k mírně vyšší přenosové rychlosti ale také velmi výraznému nárůstu síťové latence. Využitelnost paketů v tomto testu vyšla lépe než pro systém Windows XP.

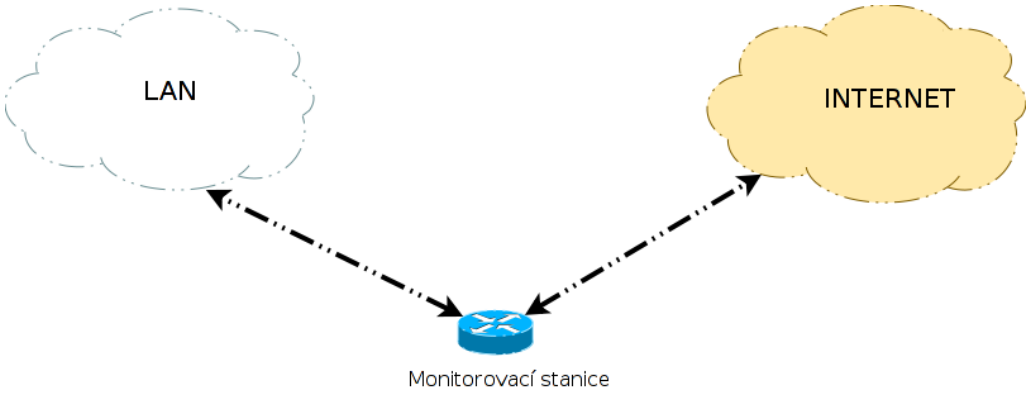
Obrázek 4.12: Závislost síťové latence na přenosové rychlosti



Obrázek 4.13: Podíl paketů pro měření na celkovém počtu příchozích TCP ACK paketů na server

4.4 Praktický test na síti

Tento test zobrazuje výsledky monitorovací aplikace z části sítě lokálního poskytovatele internetu. Data jsou sesbírána během 24 hodin provozu. Síť je velmi nehomogenní. K připojení je použito metalického ethernetu 100Base-TX, bezdrátových standardů 802.11b/g/a/n nebo optických kabelů(100Base-FX). Schéma zapojení monitorovací aplikace je na obrázku 4.14. Ukázkové grafy jsou získány z výstupu webového rozhraní aplikace.

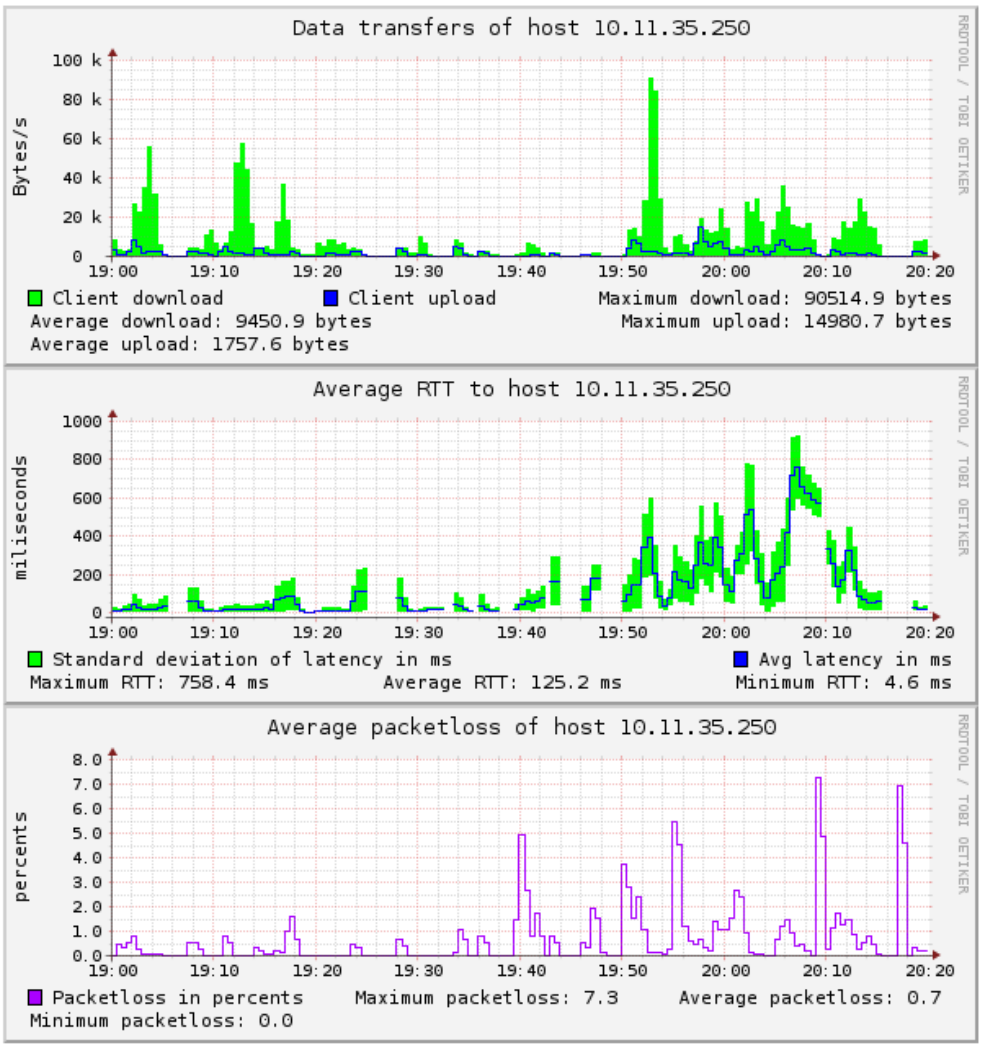


Obrázek 4.14: Umístění měřící stanice v reálné síti

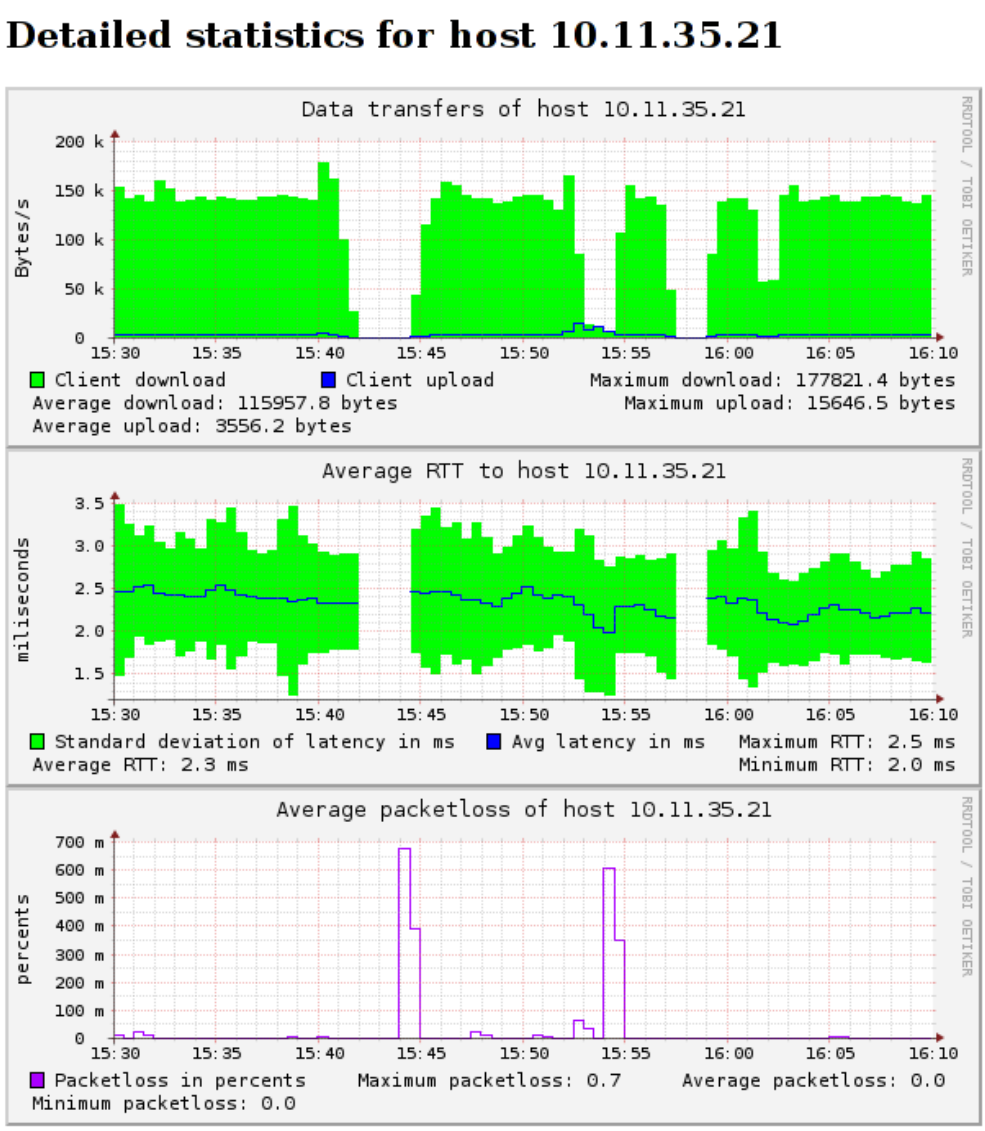
Obrázek 4.15 představuje statistiku pro stanici v síti, která nastavenou hranici síťové latence 200 milisekund a ztrátu paketů 2 % překonala ve 22% případů ze všech provedených měření v daném intervalu(latenci v 19% a ztrátu paketů ve 4%). Z grafu je patrné, že se latence zvedá nezávisle na přenosu dat a vysoké hodnoty se objevují pro malé přenosové rychlosti kolem 250 kbps. Stejně tak hodnota ztrátovosti paketů je velmi vysoká.

Další obrázek 4.16 zobrazuje měření na stanici, které ve sledovaném intervalu nepřekročila limitní hranice v žádném měření. Přestože zařízení stahovalo data rychlostí 1200 kbps, spojení si zachovávalo nízkou síťovou latenci a minimální ztrátu paketů.

Z dlouhodobého sledování charakteristik kvality připojení síťových stanic lze získat údaje, které pomohou správci sítě rozhodovat o nutnosti případných servisních zásahů.



Obrázek 4.15: Graf stanice se špatnou kvalitou poskytované služby



Obrázek 4.16: Graf stanice s dobrou kvalitou poskytované služby

Kapitola 5

Závěr

Cílem práce byl návrh systému na monitorování kvality služby. Jeho velkou část zabírá popis nové metody měření kvality služby na síti. Ta byla následně použita při implementaci softwarového systému a na praktickém příkladu srovnány její výsledky se současnými metodami. Mezi hlavní požadavky patřila jednoduchá a srozumitelná prezentace naměřených výsledků a automatické kontinuální měření bez zásahů správce systému. To bylo splněno, systém lze jednoduše aktivovat spuštěním skriptu a dále již běží sám. Prezentace pomocí webového rozhraní umožnuje přístup k datům odkudkoliv z lokální sítě nebo dokonce internetu. Oproti současným aplikacím pro monitorování kvality služby nabízí několik vylepšení plynoucích z nové metody měření. Implementovaný systém má tedy základní přepoklady k tomu, aby správcům sítě pomohl lépe identifikovat možné problémy v poskytované kvalitě služby.

5.1 Možné rozšíření programu

Přidání aktivního měření Toto vylepšení spočívá v začlenění programu ping do monitorovacího systému. V případě, že by pasivní systém detekoval velmi špatnou kvalitu služby pro určitou stanici, spustil by v pravidelných intervalech příkaz ping na danou stanici. Poté by správci sítě na webovém rozhraní prezentoval výsledky aktivního i pasivního měření. Z rozdílu obou měření by bylo možné rozpoznat, zda

je problém na síti poskytovatele nebo až na domácí síti, která je skrytá za NATem.

Přidání odhadu kapacity linky Systém by bylo možné také rozšířit o jednu z metod [17] odhadu kapacity síťového spojení. Většina dostupných způsobů je totiž velmi ovlivněna provozem, který generují jiné stanice v síti. Přijmeme-li předpoklad, že většina paketů v sítí prochází přes hlavní bránu, pak má naše monitorovací aplikace přehled o stavu provozu v jednotlivých částech sítě a může tedy naměřené hodnoty opravovat o vzniklou chybu. Systém by tak mohl měřit další veličinu charakterizující kvalitu nabízené služby.

Sledování zatížení páteřních spojů Doplněním systému o možnost zadávání struktury sítě by bylo možné zobrazovat informace o aktuální zátěži v celé síti. Opět je zde stejný předpoklad jako u předchozího bodu, a sice že významná většina provozu je mezi vnější a vnitřní sítí a objem přenosů jen uvnitř lokální sítě je zanedbatelný. Výhoda oproti protokolu [1]SNMP je, že data jsou vždy aktuální a nezatěžují se přitom routery ani switche častými SNMP dotazy. Další výhodou je také fakt, že některá síťová zařízení stále protokol SNMP neimplementují nebo je provedení natolik chybné, že ohrožuje stabilní chod zařízení.

Vlastní limitní hodnoty pro každou stanici Každý IP paket obsahuje pole TTL, které uchovává informaci o počtu přeskoků, kterými prošel při ceste v síti. Tuto hodnotu by bylo možné ukládat do datového souboru pro každou stanici a provedené měření. V prezentačním rozhraní by pak nastavený limit pro síťovou latenci označoval maximální hodnotu pro jeden přeskok. Stanice, které bude například vzdálená pět přeskoků, tak nebude znevýhodněna ve výsledné statistice oproti stanici, která je od brány vzdálena jen jeden přeskok.

[1]Simple Network Management Protocol (SNMP) je součástí sady internetových protokolů. Umožňuje průběžný sběr nejrůznějších dat pro potřeby správy sítě, a jejich následné vyhodnocování.

Literatura

[1] KIN, K. L.; et al Methods to Improve TCP Throughput in Wireless Networks With High Delay Variability. In *VTC2004-Fall Los Angeles* [online]. 4. Los Angeles (California) : IEEE, 2004 [cit. 2010-08-04]. s. 3015-3019. Dostupné z WWW: <http://www.commsp.ee.ic.ac.uk/ kkleung/papers/improve.tcp.doc>. ISBN 0-7803-8521-7.

[2] BALAKRISHNAN, H.; et al Improving TCP/IP performance over wireless networks. In *Proceedings of the 1st annual international conference on Mobile computing and networking* . Berkeley (California) : ACM New York, 1995. s. 2-11. Dostupné z WWW: <http://nms.lcs.mit.edu/ hari/papers/mcn.ps>.

[3] Comparison of network monitoring systems. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9 November 2007, last modified on 26 July 2010 [cit. 2010-08-04]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems>.

[4] IPv4. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 19.7.2005, last modified on 24.7.2010 [cit. 2010-08-04]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/IPv4>.

[5] TCP. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 15.10.2004, last modified on 31.5.2010 [cit. 2010-08-04]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/TCP/IP>.

[6] *The Internet Engineering Task Force* [online]. Marina del Rey(California) : 1981 [cit. 2010-08-04]. RFC 791: Internet Protocol. Dostupné z WWW: <http://www.ietf.org/rfc/rfc791.txt>.

[7] BINDU, A. *IBM developerWorks* [online]. 6 November 2007 [cit. 2010-08-04]. Know your TCP system call sequences. Dostupné z WWW: <http://www.ibm.com/developerworks/aix/library/au-tcpsystemcalls/index.html>.

[8] ALLMAN, M.; PAXSON, V.; STEVENS, W. *The Internet Engineering Task Force* [online]. 1999 [cit. 2010-08-04]. RFC 2581: TCP Congestion Control. Dostupné z WWW: <http://www.ietf.org/rfc/rfc2581.txt>.

[9] PSARAS, I.; TSAOUSSIDIS, V. The TCP minimum RTO revisited. In *Proceedings of the 6th international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet*. Atlanta(Georgia) : Springer-Verlag, 2007. s. 981-991. Dostupné z WWW: <utopia.duth.gr/~ipsaras/minrto-networking07.pdf>. ISBN 978-3-540-72605-0.

[10] MAHDAVI, J.; et al. *The Internet Engineering Task Force* [online]. 1996 [cit. 2010-08-04]. RFC 2018: TCP Selective Acknowledgment Options. Dostupné z WWW: <http://www.ietf.org/rfc/rfc2018.txt>.

[11] LORD, C. C. *Chris Lord's website* [online]. 2003 [cit. 2010-08-04]. Chris Lord Lectures. Dostupné z WWW: <http://www.chrisandtrudi.com/-Chris/Portfolio/TCP%20Connection%20Survivability%20Paper.pdf>.

[12] *Wireshark Wiki* [online]. 2005 [cit. 2010-08-04]. Libpcap File Format. Dostupné z WWW: <http://wiki.wireshark.org/Development/LibpcapFileFormat>.

[13] *Pylibpcap: python module for libpcap* [online]. 2008 [cit. 2010-08-04]. Dostupné z WWW: <http://pylibpcap.sourceforge.net/>.

[14] KÁBA, B. Identifikace odlehlých pozorování ve statistických datech. In *Sborník konference - Agrární perspektivy VI.*. PEF ČZU Praha : Česká zemědělská

univerzita v Praze. Provozně ekonomická fakulta, 1997. s. 439-441. Dostupné z WWW: <http://www.agris.cz/etc/textforwarder.php?iId=125823>. ISBN 80-213-0368-9.

[15] *RRDtool* [online]. 2009 [cit. 2010-08-04]. Dostupné z WWW: <http://oss.oetiker.ch/rrdtool/>.

[16] *Pyrrd* [online]. 2004 [cit. 2010-08-04]. Dostupné z WWW: <http://code.google.com/p/pyrrd/>.

[17] PRASAD, R. S.; et al. Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. *IEEE Network*. 2003, 17, s. 27-35. Dostupný také z WWW: <http://www.cc.gatech.edu/ dovrolis/Papers/NetDov0248.pdf>.