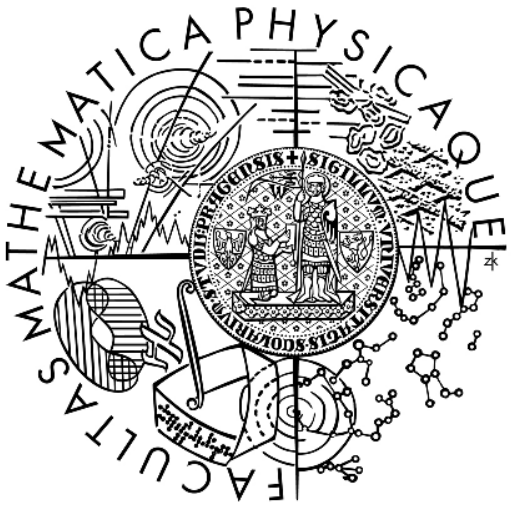
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Daniel Matteoni

Měření rychlosti a kvality datových přenosů

Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka
Studijní program: Informatika – softwarové systémy

Rád bych poděkoval panu RNDr. Ing. Jiřímu Peterkovi, za mnoho cenných rad a odpovědí na záludné otázky.

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

V Praze dne Daniel Matteoni

Obsah

Název práce: Měření rychlosti a kvality datových přenosů
Autor: Daniel Matteoni
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Ing. Jiří Peterka
e-mail vedoucího: Jiri.Peterka@mff.cuni.cz
Abstrakt:

Měření výkonnosti a kvality počítačových sítí a přesněji datových přenosu v nich má velký význam pro neustálý rozvoj a rozšiřování Internetu. Tato práce má za cíl prozkoumat, které prakticky využitelné parametry datových přenosů se dají měřit. Dále je důležité zjistit, jaké algoritmy jsou nejvhodnější pro tato měření v ohledech přesnosti a doby měření a také zátěže pro síť.

Výsledkem tohoto průzkumu je identifikace pěti důležitých veličin charakterizujících aktuální stav nebo obecné vlastnosti síťových cest. *Kapacita* je maximální rychlost, kterou mohou být přenášeny pakety od zdroje do cíle v síťové vrstvě. *Dostupná kapacita* je nevyužitá kapacita během určitého časového intervalu. *Propustnost TCP* je množství dat přenesených za jednotku času jedním TCP spojením. *Doba obrátky* je čas potřebný k tomu, aby se paket dostal ze zdroje do cíle a zase zpátky do zdroje. *Ztráta paketů* je podíl paketů ztracených během přenosu a všech odeslaných paketů.

Součástí této práce je popis postupů pro měření výše uvedených veličin a také návrh a vývoj aplikace pro jejich end-to-end měření.
Klíčová slova: měření, počítačová síť, datový přenos

Title: A system for measuring quality and throughput of data transmissions
Author: Daniel Matteoni
Department: Department of Software Engineering
Supervisor: RNDr. Ing. Jiří Peterka
Supervisor's e-mail address: Jiri.Peterka@mff.cuni.cz
Abstract:

Network performance measurements play a key role in the continuous development and expansion of the Internet. The first part of this work aims to identify the important parameters of data transmissions which are possible to measure. Next we try to find out which algorithms are the most suitable with respect for the accuracy and the time of the measurement and also for the load caused on the network.

As a result of that research we found five important measures describing the actual state or general characteristics of network paths. *Capacity* is the maximum rate that can be used to transfer the packets from source to destination in the network layer. *Available bandwidth* is the unused capacity during some time interval. *TCP throughput* is the amount of data per time unit that is delivered over a single TCP connection. *Round trip time* is the time required for a packet to get from source to destination and back to source again. *Packet loss rate* is the ratio of number of packets lost during the transmission and number of all sent packets.

This work includes the description of the algorithms used for the measurements of mentioned measures and also the design and the development of an application for these measurements.
Keywords: measurement, computer network, data transmission

Kapitola 1 - Úvod

Od začátku existence počítačových sítí klíčovou roli v jejich zlepšování hraje měření jejich výkonnosti. Navzdory několika desítkám let vývoje v této oblasti, měření výkonnosti je stále předmětem výzkumu.

V posledních letech se měření datových přenosů zaměřují hlavně na protokoly TCP/IP, na nichž je založen Internet. Mnohé internetové aplikace jsou totiž závislé na informacích o výkonnosti sítě a přesněji o parametrech síťové cesty, po které se aktuálně přenášejí data. Důsledkem toho je větší důležitost měření typu end-to-end (od jednoho ke druhému koncovému uzlu), a tedy těžiště měření se přesouvá z jádra sítě k jejím okrajům. Měření jsou typicky prováděna spíše koncovými uzly než prvky uvnitř sítě.

Cílem této práce je prozkoumání problematiky měření v počítačových sítích se zaměřením na datové přenosy a na technologii TCP/IP, identifikace veličin důležitých pro měření a známých metodologií pro jejich měření a implementace systému, který by umožňoval měřit identifikované důležité veličiny.

Kapitola 2 - Základy měření v sítích

Měření různých parametrů počítačových sítí má dlouhou tradici. Měření síťového provozu jsou nutná pro monitorování a zajištění *kvality přenosu* (QoS – *Quality of Service*) a jsou nezbytná pro mnohé další aktivity jako třeba plánování výstavby sítí.

2.1 Vývoj telekomunikačních sítí

Klasické telefonní sítě, nazývané *Plain Old Telephone System* (POTS) nebo také *Public Switched Telephone Network* (PSTN), začaly být silně regulovány několik desetiletí po vynálezu telefonu v roce 1876. Pro zajištění globální dostupnosti telefonie byla potřeba mezinárodní spolupráce při vytváření standardů pro zajištění kompatibility zařízení. Velkou roli zde hrála *International Telecommunication Union* (ITU). Snahy dodavatelů tradičních PSTN zařízení a služeb o uspokojení rostoucí potřeby po mobilní a datové komunikaci vyústily v množství nových technologií. Mezi nimi jsou kupříkladu standard pro mobilní komunikace *Global System for Mobile Communications* (GSM), *Integrated Services Digital Network* (ISDN) a jeho širokopásmová verze B-ISDN nebo *Asynchronous Transfer Mode* (ATM). ATM se mělo stát novým integrovaným řešením pro poskytování celé řady hlasových a datových komunikačních služeb, této technologii se však nepovedlo splnit očekávání.

Internet byl vyvinut za účelem zajištění jednoduché, efektivní a robustní komunikace založené na principu přepojování paketů. Výsledkem jeho vývoje byl mezi jinými vznik specifikací řady protokolů. Centrálním je *Internet Protocol* (IP), který přenáší data nazývaná *datagramy* mezi odesílatelem a příjemcem, kteří jsou identifikování pomocí adres pevné délky. IP spoléhá na služby nízkoúrovňových protokolů lokálních sítí a zajišťuje vlastní služby pro vysokoúrovňové protokoly typu *host-to-host*.

Zatímco tradiční dodavatelé telekomunikačního vybavení a služeb byli zaneprázdnění vývojem a standardizací nové síťové technologie, IP začal být používán prakticky na všech terminálech vlastněných zákazníky dodavatelů telekomunikačních služeb a stal se takto de-facto standardem. Dnes je IP nedílnou součástí všech významných operačních systémů počítačů, herních konzolí a mobilních zařízení.

Díky úspěchu aplikací jako e-mail a WWW a také díky robustnosti a dostupnosti v čase, kdy to potřeboval trh, se stal IP dominantní síťovou technologií moderních telekomunikací.

Výsledkem této evoluce je, že model telekomunikační sítě se oproti PSTN výrazně změnil. Namísto k jednomu standardnímu telefonnímu terminálu uživatelé přistupují k různým telekomunikačním službám prostřednictvím velkého množství terminálu – počítačů, mobilních telefonů, kapesních počítačů PDA, herních konzolí. Navíc existuje spousta technologií přístupů podle typu zařízení připojujícího se k telekomunikační síti.

2.2 TCP/IP

Kořeny IP hledejme v šedesátých letech dvacátého století, když byl zrozen program ARPANET – datová síť organizace Advanced Research Projects Agency. V sedmdesátých a začátkem osmdesátých let byla vyvinuta soustava protokolů kolem IP nazývána *TCP/IP Protocol Suite*. Je to kombinace řady protokolů komunikujících na čtyřech různých vrstvách – aplikační, transportní , síťové a síťového rozhraní.

Hlavním úkolem vrstvy síťového rozhraní je zajistit fyzické spojení mezi sousedními uzly. Typicky je realizovaná prostřednictvím ovladačů zařízení, které jsou součástí operačního systému a které se zabývají komunikací na nejnižší úrovni.

Síťová vrstva, často také nazývána internetová vrstva, je zodpovědná za doručování datových paketů jakémukoli uzlu v síti. Důležitou úlohou síťové vrstvy je směrování paketů. V TCP/IP je síťová vrstva implementována pomocí protokolů *Internet Protocol* (IP), *Internet Control Message Protocol* (ICMP) a *Internet Group Management Protocol* (IGMP).

Služby protokolu IP jsou využívány dvěma známými protokoly transportní vrstvy – *User Datagram Protocol* (UDP) a *Transmission Control Protocol* (TCP). Oba zajišťují datový proud mezi dvěmi stanicemi pro účely protokolů aplikační vrstvy. Mezi těmito protokoly je podstatný rozdíl. TCP zajišťuje spolehlivý tok dat, kdežto jednodušší UDP pouze přenáší datové pakety zvané *datagramy*, ale nezaručuje jejich doručení.

Více informací o TCP/IP lze získat např. z publikací [1] a [2].

2.3 Měření v telekomunikačních sítích

V sítích PSTN se měřící úsilí zaměřovalo na různé síťové prvky. Většina měření v moderních telefonních přepínačích je implementována pomocí čítačů událostí, které jsou periodicky čteny jinými uzly a aplikacemi pro správu sítí. Tato měření se obvykle zaměřují na dostupnost různých síťových zdrojů. Čítače počítají požadavky, úspěšné požadavky a selhání

z různých důvodů. Tato měření jsou typicky dostupná ve všech prvcích sítě, v praxi se však musí čelit různým omezením jako např. omezenému místu pro ukládání dat v síťových prvcích, nadbytečnému zatížení procesorů atd.

Někteří poskytovatelé PSTN služeb také provozují monitorovací systém nezávislý na síťových uzlech, který funguje na principu testovacích volání z určitých terminálů umístěných na rozlišných místech jejich sítě. Tyto systémy se snaží shromažďovat měření end-to-end výkonu a kvality tak, jak jsou vnímány uživateli.

Měření síťového výkonu v síti ARPANET v rané fázi vývoje datových komunikací se prováděla v podstatě identicky jako výše popsaná měření v sítích PSTN.

Role měření v moderních datových sítích založených na principu přepojování paketů zůstává podobná tradičním ideám, nicméně existují také některé významné odlišnosti. V těchto sítích rapidně vzrostl význam end-to-end měření - jedním z argumentů je potřeba zajištění vhodné kvality přenosu zákazníkům.

Modelování výkonu sítí založených na přepojování paketů se ukázalo jako velmi složitý úkol. Existující modely totiž nefungovaly tak spolehlivě pro zjišťování end-to-end výkonu a kvality přenosu jako tomu bylo v případě telefonních sítí založených na přepojování okruhů. End-to-end měření jsou takto jediným spolehlivým zdrojem informací a právem se staly dominantní měřící technikou používanou v Internetu.

Rozvoj počítačových technologií otevřel celou řadu nových příležitostí a zmenšila se tak důležitost integrování měřících služeb do síťového hardwaru. Ačkoliv dnešní síťové prvky, jako jsou např. směrovače a přepínače v jádrech datových sítí, stále podporují měření, end-to-end měření prováděna osobními počítači dominují.

2.4 End-to-end měření

Základní princip end-to-end měření je následující: Síťový provoz pocházející ze zdrojového uzlu je přenášen po síti do cílového uzlu. Monitory provozu (traffic monitors) zachycují provoz procházející monitorovanými body uvnitř sítě a zaznamenávají jeho vlastnosti potřebné pro měřící účely.

Obecně je velmi důležité nenarušovat měřený síťový provoz. Existují dvě základní metody duplikování provozu a doručování jej do monitorů provozu. První možností je použít přepínač nebo směrovač pro duplikaci datového proudu. Tyto síťové prvky typicky poskytují porty

vyhrazené pro účely měření, avšak tyto mají různá omezení na množství provozu. Režie potřebná pro doručení paketů na měřící port a omezení kapacity kombinované s tím, že vyšší prioritu má přenášení reálného provozu, mohou způsobit ztrátu paketů nebo narušení jejich časování v měřeném datovém proudu. Druhou možností je použít fyzickou duplikaci signálu. Nevýhodou tohoto řešení je, že může narušit signál původní linky a dále je zapotřebí složité instalace.

V dalším kroku měřicího procesu monitory provozu potřebují zachytit datový proud, který jim je předáván, což znamená přijmout pakety, zpracovat je a nakonec zapsat a uložit výsledky.

2.5 Pasivní měření

Pojmem *pasivní* měření se rozumí standardní přístup sledování chování a výkonu proudů paketů pomocí monitorování provozu procházejícího měřícími body. Aby bylo možné sledovat chování provozu pocházejícího z určitých zdrojových adres, je potřeba, aby tento provoz byl dostatečný pro získání potřebného množství dat. Provoz je různými způsoby filtrován za účelem izolace různých proudů a skupin proudů podle kritérií, jakými jsou cílová adresa, protokol nebo aplikace. Jedním z extrémů je ignorovat informace o adresách, čímž se získají agregované statistiky linky v daném bodě jako např. rozptyl velikostí paketů složení provozu podle protokolů. Tato třída síťových měření se obvykle označuje termínem *pasivní at-a-point* měření (měření v bodě).

Pasivní měření ve zdrojovém a v cílovém uzlu lze kombinovat a získat tak *pasivní end-to-end* měření. Takovéto metody dovolují měření např. počtu ztrát paketů nebo jejich zpoždění. Aby to bylo možné, je potřeba vyřešit problémy týkající se synchronizace časových razítek nebo identifikace paketů.

2.6 Aktivní měření

Pojmem *aktivní* měření se rozumí vysílání umělého sondovacího provozu do sítě a měření jeho vlastností v různých bodech. Počáteční struktura sondovacího provozu je známá, proto změřením toho, jak je tato struktura ovlivněna úsekem sítě, přes který sondy prošly, lze zjistit síťové podmínky. Tato práce se bude dále zabývat pouze *aktivními end-to-end* měřeními, kde sondy jsou posílány mezi párem zdroj-cíl a měření jsou prováděna pouze na zdrojovém a cílovém uzlu. Tímto typem měření je možné určit celou řadu zajímavých parametrů a přitom je

prováděno pouze na uzlech mimo vlastní jádro sítě, tj. je jednoduché na implementaci a správu. Zdroj proudu sond bude dále označován jako *Odesílatel*, zkráceně ***SND***, cíl jako *Příjemce*, zkráceně ***RCV***. Toto zkrácené označování bude využíváno i v následujících kapitolách, především v kapitolách popisujících implementační detaily jednotlivých měření.

Proces aktivního měření se skládá ze dvou částí. První částí je odesílání sond na odesílateli, druhou je monitorovací proces prováděný jak na odesílateli, tak na příjemci. Monitorovací proces je vlastně pasivní měření s filtrem pro výběr proudu očekávaných sond.

Klíčovou výhodou tohoto přístupu je velká flexibilita, s jakou je možné navrhovat proudy sond, aby jejich vlastnosti byly vhodné pro konkrétní měření. Parametry návrhu aktivního proudu sond jsou posloupnost typů paketů, velikosti paketů a časy mezi odesláním jednotlivých paketů. Tyto parametry mohou být vybrány v závislosti na jakémkoliv deterministickém nebo statistickém kritériu. Další výhody zahrnují možnost častého opakování a pozměňování experimentů, mnohem menší objem měřených dat než u pasivního monitorování hlavně v sítích s vysokou propustností a dále nemožnost jakéhokoliv narušení soukromí u citlivých dat.

Hlavní nevýhodou aktivních měření je jejich invazivní charakter. Sondy mohou narušovat ostatní síťový provoz nebo měnit směrovací podmínky. Ve snaze minimalizovat tyto efekty měřící projekty typicky používají proudy sond s průměrnou rychlostí 10 Kbps, což odpovídá provozu generovanému jedním uživatelem připojeným pomocí vytáčeného spojení [3].

Kapitola 3 - Veličiny

Hlavní oblasti internetových měření jsou podle [4] *topologie, zátěž, výkon* a *směrování.*

Globální infrastruktura Internetu podléhá neustálým změnám, je tedy velkou výzvou zmapovat tento extrémně komplexní systém. Cílem topologických měření je získat informace o tom, jak jsou páteřní uzly navzájem propojeny a určit jejich geografické umístění. *Cooperative Association of Internet Data Analysis* (CAIDA) monitoruje topologii Internetu pomocí sond odesílaných z mnoha zdrojů do desítek tisíc cílů rozmístěných po celé zeměkouli.

Měření zátěže se zaměřují na sběr statistik o využití směrovačů, přepínačů a jednotlivých spojů. Typicky jsou prováděná pasivně monitorováním provozu na různých místech sítě.

Měření výkonu se zaměřují na analýzu end-to-end chování a na diagnózu síťových problémů. Tato úsilí typicky znamenají sběr statistik jako end-to-end ztráta paketů (end-to-end packet loss), zpoždění (delay) a doba obrátky (round trip time) pomocí odesílání testovacího provozu do sítě.

Měření směrování poskytují náhled na dynamiku směrovacích protokolů a na změny ve směrovacích tabulkách.

3.1 Standardní veličiny

Nejběžnější veličiny typicky měřené poskytovateli internetových služeb (ISP – Internet Service Providers) jsou latence, ztráta paketů, propustnost, využití a dostupnost. Bohužel většina těchto veličin není dobře definovaná, kvůli čemuž porovnávání výsledků z různých zdrojů není vůbec triviální záležitostí. IP Performance Metrics Working Group (IPPM) patřící do Internet Engineering Task Force (IETF) vyvíjí *Requests For Comments* (RFC) pro měření v sítích. Úsilí je zaměřeno především na ISP, nicméně i přes tyto standardizační snahy se většina výzkumných aktivit nezakládá na těchto standardech.

3.2 Pasivní veličiny

3.2.1 Propustnost (throughput)

Propustnost je rychlost, s jakou jsou data přenášena spojem nebo síťovou cestou. Obvykle se vyjadřuje v bitech za sekundu (bps) nebo v bytech za sekundu (Bps). Propustnost je měřená

počítáním množství dat transportovaných během určitého časového intervalu. Propustnost se může odkazovat na celkové přenesené množství dat nebo se může jednat o filtrovaný provoz v závislosti na typu aplikace, protokolu, cílu, zdroji, atd.

Výběr časového intervalu pro tato měření není triviální. Volba krátkých intervalů snižuje důvěryhodnost, zatímco nadbytečně dlouhé intervaly skrývají nárazovost a nestacionárnost provozu v menších časových měřítcích.

3.2.2 Využití (utilisation)

Využití spoje může být definováno jako poměr propustnosti a kapacity spoje. Tato veličina dává poskytovatelům služeb informaci o tom, jak efektivně jsou využívány zdroje a jak blízko jsou k tomu, aby byly přetíženy.

3.2.3 Dostupnost (availability)

Dostupnost může být definována jako procento času, kdy služba byla dostupná pro normální použití v určitém časovém intervalu. Bohužel rozhodování o tom, kdy je služba dostupná není triviální.

3.2.4 Další pasivní veličiny

Výše uvedené veličiny jsou nejčastěji používány a dávají vysokoúrovňový pohled na výkon sítě. Pro detailnější analýzu jsou sbírány další statistiky. Například *rozptyl velikostí paketů* (*packet size distribution*), *délka řetěze paketů* (*length of packet trains*) a *rozptyl délky prefixů IP adres* (*IP address prefix length distribution*) jsou používány u návrhu a konfigurace směrovačů.

3.3 Aktivní veličiny

3.3.1 Kapacita (capacity)

Spoj ve vrstvě síťového rozhrání obvykle pracuje s konstantní *přenosovou rychlostí* (*bit rate* nebo také *transmission rate*), což je kupříkladu 10Mbps u 10BaseT Ethernetu. Přenosová rychlost je omezená šířkou pásma přenosového média a také odesílacím/přijímacím hardwarem.

V síťové vrstvě je dostupná nižší rychlost kvůli *zapouzdřování paketů* (*encapsulation*) a *synchronizaci* (*framing*). Předpokládejme, že nominální kapacita segmentu vrstvy síťového rozhraní je C_1. Pak doba přenosu IP paketu o velikosti L_2 bytů v síťové vrstvě je

$$\Delta_2 = \frac{L_2 + H_1}{C_1}, \quad (1)$$

kde H_1 je celková režie potřebná pro zapouzdření IP paketu v bytech. Tedy kapacita C_2 v síťové vrstvě je

$$(2) \qquad C_2 = \frac{L_2}{\Delta_2} = \frac{L_2}{\frac{L_2 + H_1}{C_1}} = C_1 \frac{1}{1 + \frac{H_1}{L_2}}.$$

Kapacita v síťové vrstvě tedy závisí na velikosti IP paketu a na režii vrstvy síťového rozhraní.

Kapacitu přeskoku i (*capacity of a hop i*) C_i definujeme jako maximální možnou přenosovou rychlost v síťové vrstvě na tomto přeskoku. Podle rovnice (2) je maximální přenosové rychlosti v síťové vrstvě dosaženo pro pakety o velikosti MTU. Tedy *Kapacita přeskoku* (*kapacity of a hop*) je přenosová rychlost, se kterou jsou přenášeny IP pakety o velikosti MTU v síťové vrstvě.

Předchozí definici lze rozšířit pro síťovou cestu. *End-to-end kapacita* (*end-to-end capacity*) je maximální rychlost, kterou mohou být přenášeny pakety od zdroje do cíle v síťové vrstvě. End-to-end kapacita síťové cesty tedy vymezuje horní odhad propustnosti, kterou by bylo možné očekávat na této cestě v síťové vrstvě. End-to-end kapacita C je určena kapacitou minimálního spoje na cestě

$$(3) \qquad C = \min_{i=1,\ldots,H} C_i,$$

kde C_i je kapacita i-tého přeskoku a H je počet přeskoků na cestě. Přeskok s minimální kapacitou je nazýván *úzký spoj* (*narrow link*).

Dále je potřeba poznamenat, že některé technologie vrstvy síťového rozhraní nepracují s konstantní přenosovou rychlostí. Například bezdrátová technologie IEEE 802.11b odesílá rámce rychlostí 11, 5.5, 2 nebo 1 Mbps v závislosti na četnosti chyb. Předchozí definice kapacity spoje může být použitá pro tyto technologie během časových intervalů, ve kterých je konstantní.

3.3.2 Dostupná kapacita (available bandwidth)

Doslovný překlad této veličiny z angličtiny by zněl *dostupná šířka pásma*, pojem šířka pásma se však používá spíše v analogových technologiích pro označení rozsahu frekvencí, proto zde bude tato veličina označována jako *dostupná kapacita*.

Dostupná kapacita se týká nevyužité kapacity spoje během určitého časového intervalu. Přestože kapacita spoje závisí na přenosové technologii a médiu, dostupná kapacita navíc závisí na objemu provozu na spoji a typicky se mění v čase.

V jakémkoli časovém okamžiku spoj buďto přenáší paket plnou kapacitou linky nebo je nečinný, takže okamžité využití spoje může být pouze 0 nebo 1. Definice dostupné kapacity proto vyžaduje průměrování okamžitého využití spoje v čase.

Pokud C_i je kapacita přeskoku i a u_i je průměrné využití tohoto přeskoku v daném časovém intervalu, průměrná dostupná kapacita A_i přeskoku i je nevyužitým dílem kapacity, tj.

(4) $A_i = (1 - u_i)C_i$.

Rozšíření předchozí definice pro cestu s H přeskoky – end-to-end dostupná kapacita, je minimální dostupná kapacita ze všech H přeskoků

(5) $A = \min_{i=1,\ldots,H} Ai$.

Přeskok s minimální dostupnou kapacitou je nazýván *těsný spoj* (*tight link*).

Jelikož dostupná kapacita se může měnit v čase, je důležité, aby měření probíhalo rychle. Platí to zejména pro aplikace, které používají měření dostupné kapacity k přizpůsobení jejich přenosové rychlosti. Na druhou stranu kapacita zůstává konstantní v dlouhých časových intervalech, proto měření nemusí probíhat tak rychle jako pro dostupnou kapacitu.

3.3.3 Latence (latency)

Latence může být definována jako čas potřebný k tomu, aby se paket dostal ze zdroje do cíle. Speciálním případem je *doba obrátky* (*RTT – Round Trip Time*), kdy cíl je identický se zdrojem, avšak pakety překonávají cestu od zdroje k určitému vzdálenému uzlu a zpátky. RTT měřen mezi klientem a serverem je jedním z klíčových faktorů určujících výkon interaktivních aplikací a účinnost mechanismů řízených zpětnou vazbou.

3.3.4 Propustnost TCP (TCP throughput)

Další klíčovou veličinou v TCP/IP sítích je propustnost TCP spojení, protože TCP je dominantním protokolem Internetu. Propustnost TCP je ovlivněna několika faktory. Jedná se o velikost přenosu, počet konkurenčních TCP spojení, velikosti bufferů TCP soketů na straně odesílatele i příjemce, zahlcení podél zpětné (ACK) cesty, velikosti bufferů směrovačů a kapacitu a zatížení každého spoje na cestě.

Kupříkladu propustnost malého přenosu (např. webové stránky) závisí primárně na počáteční velikosti TCP okénka a době obrátky (RTT). Propustnost velkých TCP přenosů se

může významně měnit při použití různých implementací TCP, i když dostupná kapacita je totožná.

Zajímavé je porovnání myšlenek propustnosti TCP a dostupné kapacity. Dostupná kapacita totiž předpokládá, že průměrná zátěž je konstantní, a odhaduje dodatečnou kapacitu, kterou cesta může nabídnout až do nasycení těsného spoje, kdežto propustnost TCP závisí na tom, jak TCP sdílí kapacitu s ostatními aktuálně probíhajícími TCP toky.

V RFC 3148 [6] je zaveden pojem *Bulk Transfer Capacity* (*BTC*) a je definován jako maximální propustnost dosažitelná jedním TCP spojením. Spojení musí implementovat všechny algoritmy předcházení zahlcení tak, jak jsou specifikovány v RFC 2581 [7].

Byly provedeny i další pokusy o zavedení jakési standardní veličiny pro TCP (např. [6], [7], [8]). Nicméně různorodost implementací TCP činí všechny tyto pokusy velmi obtížnými.

3.3.5 Ztráta paketů (packet loss)

Ztráta paketů je podíl paketů ztracených během přenosu a všech odeslaných paketů, obvykle je vyjadřována v procentech. Na rozdíl od latence jednosměrná ztráta paketů je důležitější veličinou než ztráta paketů během obrátky. RFC 2680, které specifikuje veličinu *jednosměrná ztráta paketů* (*one-way packet loss metric*), to zdůvodňuje asymetrií cest v Internetu, asymetrií výkonu v různých směrech i pro symetrické cesty a závislostí některých aplikací (např. přenos souborů pomocí TCP) na výkonu v jednom směru.

Tato veličina je důležitá, neboť v mnoha aplikacích výkon drasticky klesá, pokud end-to-end ztráta paketů překročí určitou hranici. Tato hranice je samozřejmě závislá na konkrétní aplikaci, ale je zvláště důležitá pro real-timeové aplikace jako třeba IP telefonie, které se stávají nepoužitelnými při nadměrné ztrátě paketů.

3.3.6 Dostupnost (availability)

Dostupnost byla již zmíněna v sekci o pasivních veličinách. Zde je potřeba poznamenat, že aktivní měření je potřeba, když mezi příslušnými uzly není žádný přirozený provoz. Měření dostupnosti je často založeno na programu *ping*.

3.3.7 Topologie (Topology discovery)

Zjišťování topologie je už tradičně založeno na aktivním měření – klasickým příkladem je utilita *traceroute*, která funguje na principu posílání ICMP paketů s rostoucími hodnotami TTL (*Time To Live)*.

3.3.8 Dynamika směrování (Routing dynamics)

Dynamika směrování je další oblastí využití aktivních měření. Typicky se používají měření RTT, avšak lze také použít pasivní monitorování zpráv protokolu Border Gateway Protocol (BGP) [4].

Kapitola 4 - Úvod do aktivního měření

Základní myšlenkou aktivního měření je posílání řízeného proudu paketů, neboli sond, po síti a měření různých parametrů na cílovém uzlu (který může být identický se zdrojovým uzlem). Jelikož přesné vlastnosti odesílaného provozu jsou známé, odchylky způsobené sítí mohou být využity k získání přesných informací o stavu a podmínkách v síti.

Základními parametry jsou *protokol* použitý k přenosu sond (např. UDP, ICMP), *obsahy paketů* a *způsob odesílání* – zde se jedná především o pravidlo pro výpočet časů mezi odesláním jednotlivých paketů.

Klíčovým problémem je výběr protokolu, protože s různými protokoly sítě zacházejí odlišnými způsoby. Kupříkladu nelze zaručit, že směrovače se řídí pokyny v Option Field paketů, nebo ICMP pakety mohou být někdy záměrně zahazovány. Každý druh proudu sond má své vlastnosti vhodné pro měření různých veličin.

4.1 UDP

UDP tvoří malou část internetového provozu, jeho hlavní použití je v DNS (*Domain Name Server*). UDP pakety jsou častou volbou pro měření jednosměrných veličin jako třeba jednosměrné zpoždění, kolísání zpoždění a ztráta. Tok UDP paketů je totiž nespojovaný a nepoužívá se vyšší protokol řídící odesílání do sítě, což je v souladu s ideou nezávislých sond měřících spíše síť než jejich vlastní dynamiku. Pomineme-li problémy s časováním, vytváření a odesílání UDP sond není příliš složitou záležitostí na naprogramování. Velikost sond může být minimální, stačí když sondy obsahují sekvenční čísla pro jejich správnou identifikaci, další informace o stavu mohou být udržovány v koncových programech. Nevýhodou tohoto přístupu je, že na obou koncových uzlech musí být nainstalován specifický odesílací a přijímací software.

4.2 TCP

Měření pomocí TCP lze implementovat více způsoby. Je možné spustit obyčejné TCP spojení a pomocí měření různých veličin jako propustnost nebo charakteristika ztrát se pokusit vytvořit závěry týkající se podmínek v síti specificky pro TCP spojení. Problémem tohoto přístupu je velká různorodost implementací TCP s velkými rozdíly i v jejich výkonu. Dalším

možným přístupem jsou různé generátory sond podobných TCP nebo *proudy sond řízené pomocí TCP*.

Proudy sond řízenými pomocí TCP se rozumí TCP pakety vygenerované speciálními procesy. Kandidáty jsou SYN pakety pro jednosměrná měření a páry SYN / SYN-ACK pro obousměrná měření. Směrovače zacházejí s těmito pakety, jako kdyby se jednalo o začátky normálních TCP spojení. Proto takto naměřené hodnoty mají blízko k výkonu TCP, resp. k podmínkám na začátku spojení. Jelikož TCP je dominantním protokolem, takovéto sondy budou navíc zpracovávány typickým výkonem sítě. Dále páry SYN / SYN-ACK jsou automaticky dostupné na téměř všech koncových uzlech, což umožňuje obousměrná měření bez nutnosti instalace zvláštního měřícího softwaru, navíc generování SYN-ACK je řešeno na nízké úrovni, což redukuje zbytečná zpoždění, která narušují měřená zpoždění na síťové úrovni.

Nevýhodou je, že servery mohou interpretovat sekvence SYN paketů jako útok a odfiltrovávat je. Další nevýhodou je to, že některé systémy generují SYN-ACK se zpožděním, které se mění v závislosti na mnoha podmínkách jako například zátěž serveru. Další informace jsou dostupné v [9].

4.3 ICMP

Na rozdíl od UDP a TCP byly ICMP pakety navrženy pro výměnu informací mezi stanicemi a směrovači v jádře sítě. Jejich hlavní použití v aktivním měření je v nástroji *ping* (*Packet InterNet Groper*), který testuje dostupnost uzlů a měří RTT, a *traceroute*, který používá inkrementaci TTL pro generování odpovědí od směrovačů, čímž rekonstruuje trasu paketů.

Nevýhodou měření založených na ICMP je, že někteří ISP omezují nebo blokují ICMP pakety. Navíc toto blokování je někdy realizováno za účelem skrytí vnitřní topologie. ICMP pakety mají často na směrovačích nižší prioritu než TCP a UDP pakety.

Ping a *traceroute* používají *ICMP Echo Request* a *ICMP Time Exceeded* pakety. Nástroj *pathchar* popsaný v [10] a [11] kombinuje ICMP zprávy s přístupem založeným na sondách, funguje totiž na principu ICMP zpráv generovaných směrovači v odpovědi na UDP sondy s omezeným TTL.

Kapitola 5 - Měřící techniky

Tato kapitola popisuje několik existujících měřících technik pro odhady kapacity a dostupné kapacity. Dále budeme předpokládat, že během měření cesty se nemění směrování a provozní zátěž je konstantní. Dynamické změny ve směrování nebo zátěži totiž mohou vyvolat chyby v jakékoli měřící metodě.

5.1 Sondování s variabilní velikostí paketů(Variable Packet Size – VPS)

VPS sondování se zaměřuje na měření kapacity každého přeskoku na cestě. Tato metodologie je zmíněna v [10], [11] a [14]. Klíčovým prvkem je měření RTT ze zdroje na každý přeskok jako funkce velikosti sondovacích paketů. VPS používá TTL pole v hlavičce IP paketu, aby paket byl zahozen na příslušném přeskoku. Směrovač paket zahodí a odesílateli vrátí ICMP Time Exceeded chybovou zprávu. Odesílatel využívá přijaté ICMP pakety k měření RTT na přeskoku.

RTT se skládá ze tří složek zpoždění na každém přeskoku – jedná se o *serializační zpoždění* (*serialization delays*), *fyzické zpoždění* (*propagation delay*) a *frontové zpoždění* (*queueing delay*). Serializační zpoždění paketu o velikosti L na spoji s přenosovou rychlostí C je čas potřebný k odeslání paketu do spoje. Tento čas je rovný L / C. Fyzické zpoždění paketu na spoji je čas, který zabere elektronům nebo fotonům projít skrz spoj a je nezávislý na velikosti paketu. Frontové zpoždění je složka zpoždění zapříčiněna frontováním v síťových prvcích. Zatímco serializační a fyzické zpoždění na dané cestě může být považováno za konstantní, frontové zpoždění je určeno vlastnostmi aktuálního provozu v síti a obvykle se mění v čase

VPS odesílá četné sondovací pakety dané velikosti do každého zařízení na síťové vrstvě po cestě od zdroje k cíli. Tato technika předpokládá, že alespoň jeden z těchto paketů společně s ICMP odpovědí, kterou generuje, nenarazí na žádné frontové zpoždění. Proto minimální RTT, které je naměřeno pro každou velikost paketů se skládá ze dvou složek: zpoždění nezávislého na velikosti paketu, většinou zapříčiněného fyzickými zpožděními, a zpoždění úměrného velikosti paketu, zapříčiněného časem obsluhy na každém spoji. Minimální RTT – $T_i(\mathrm{L})$ pro danou velikost paketu L do přeskoku i bude

$$(6) \qquad T_i(L) = \alpha + \sum_{k=1}^{i} \frac{L}{C_k} = \alpha + \beta_i L \,,$$

kde C_k je kapacita *k*-tého přeskoku, α jsou zpoždění do přeskoku *i*, která nezávisí na velikosti *L* paketu, β_i je vyjádřeno jako

$$\beta_i = \sum_{k=1}^{i} \frac{1}{C_k}. \quad (7)$$

ICMP odpovědi mají všechny stejnou velikost nezávislou na *L*, proto α zahrnuje jejich serializační zpoždění společně se součtem všech fyzických zpoždění na cestě tam i zpátky.

Opakováním měření minimálního RTT pro každý přeskok $i = 1,\ldots,H$, lze odhadnout kapacitu každého přeskoku jako:

$$C_i = \frac{1}{\beta_i - \beta_{i-1}}. \quad (8)$$

VPS sondování může významně podhodnotit kapacitu, pokud měřená cesta obsahuje store-and-forward přepínače ve vrstvě síťového rozhraní. Tato zařízení generují serializační zpoždění typu *L / C*, avšak neposílají ICMP Time Exceeded pakety, protože nejsou viditelné v síťové vrstvě.

5.2 Sondování pomocí disperze párů/řetězců paketů(Packet Pair/Train Dispersion – PPTD)

Sondování pomocí párů paketů se používá hlavně k měření end-to-end kapacity cesty. Zdroj odesílá četné páry paketů. Každý pár paketů se skládá ze dvou paketů stejné velikosti odeslaných těsně jeden za druhým (back-to-back). *Disperze* páru paketů na určitém spoji cesty je časová vzdálenost mezi posledními bity obou paketů. Tato technika byla zavedena a analyzována v [15], [16], [17] a [18].

Když spoj s kapacitou C_0 připojuje zdroj k cestě a sondovací pakety mají velikost *L*, pak disperze páru paketů na tomto prvním spoji je $\Delta_0 = L / C_0$. Obecně pokud disperze před spojem s kapacitou C_i je Δ_{in}, disperze páru paketů po opuštění spoje bude

$$\Delta_{out} = \max(\Delta_{in}, \frac{L}{C_i}) \quad (9)$$

za předpokladu, že spoj nepřenáší žádný další provoz.

Když pár paketů dorazí k příjemci, naměřená disperze Δ_R bude

$$\Delta_R = \max_{i=0,\ldots,H} (\frac{L}{C_i}) = \frac{L}{\min_{i=0,\ldots,H} (C_i)} = \frac{L}{C}, \quad (10)$$

kde C je end-to-end kapacita cesty. Takže kapacitu cesty může příjemce určit jako $C = L / \Delta_R$. Předpoklad, že na cestě není žádný další provoz (tzv. *křížující provoz – cross traffic*), je v praxi velmi obtížné splnit. Křížující provoz může zvýšit nebo snížit disperzi Δ_R, což způsobí příliš nízký nebo příliš vysoký odhad kapacity. Nízký odhad se objeví, když pakety křížujícího provozu jsou na určitém spoji odesílány mezi sondovacími pakety. Disperze bude v tomto případě totiž vyšší než L / C. Vysoký odhad se objeví, když křížující provoz zpozdí první paket páru paketů více než druhý paket na spoji, který následuje za úzkým spojem.

Následky křížujícího provozu lze zmírnit odesíláním mnoha párů paketů a použitím statistických metod pro odfiltrování nepřesných měření. Avšak standardní statistické postupy, jako např. počítání mediánu, nevedou vždy ke správným výsledkům – více lze najít v [17].

Sondování pomocí řetězce paketů je rozšířením sondování pomocí páru paketů. Disperze řetězce paketů na spoji je čas mezi posledním bitem prvního a posledního paketu řetězce. Jakmile příjemce změří end-to-end disperzi $\Delta_R(N)$ pro řetězec paketů délky N, může spočítat *míru disperze D* (*dispersion rate*)

$$D = \frac{(N-1)L}{\Delta_R(N)}. \tag{11}$$

Pokud na cestě není žádný křížující provoz, míra disperze bude rovna kapacitě stejně jako v případě páru paketů. Nicméně křížující provoz může způsobit, že míra disperze bude podstatně menší než kapacita.

Uvažme cestu se dvěma přeskoky, na které se používají FIFO buffery. Odesílatel odesílá řetězec paketů délky N skrz spoj s kapacitou C_0 bez křížujícího provozu. Sondovací pakety mají velikost L bytů. Druhý spoj má kapacitu $C_1 < C_0$ a přenáší křížující provoz s průměrnou rychlostí $R_c < C_1$. Disperze řetězce paketů za prvním spojem je $\Delta_1 = L(N - 1) / C_0$, kdežto za druhým spojem

$$\Delta_2 = \frac{(N-1)L + X_c}{C_1}, \tag{12}$$

kde X_c je velikost křížujícího provozu v bytech, který dorazí na druhý spoj v průběhu přenosu řetězce paketů na tomto spoji. Očekávaná hodnota X_c je

$$E[X_c] = R_c \Delta_1 = R_c \frac{(N-1)L}{C_0}, \tag{13}$$

a proto očekávaná hodnota míry disperze je

(14) $$E[D] = \frac{(N-1)L}{\Delta_2} = \frac{C_1}{1 + \frac{R_c}{C_0}}.$$

Když $R_c > 0$, $E[D]$ je menší než kapacita. $E[D]$ nezávisí na délce řetězce paketů, nicméně A má vliv na odchylku změřené míry disperze od její střední hodnoty $E[D]$ – delší řetězce paketů snižují odchylku D.

5.3 Self-Loading periodické proudy(Self-Loading Periodic Streams – SLoPS)

SLoPS je metoda pro měření end-to-end dostupné kapacity popsaná v [19]. Odesílatel odesílá K stejně velkých paketů (periodický proud paketů) příjemci určitou rychlostí R. Metodologie se týká monitorování odchylek jednosměrných zpoždění sondovacích paketů. Pokud je rychlost proudu R větší než dostupná kapacita A cesty, proud způsobí krátkodobé zahlcení fronty těsného spoje. Jednosměrná zpoždění sondovacích paketů se zvyšují, když každý paket čeká ve frontě na těsném spoji. Na druhou stranu když rychlost proudu R je nižší než dostupná kapacita A, sondovací pakety projdou cestou bez zvyšování obsazenosti fronty u těsného spoje, a proto jednosměrná zpoždění se nezvyšují.

Odesílatel se snaží, aby rychlost proudu R byla blízká dostupné šířce pásma A, k čemuž se používá iterativní algoritmus podobný binárnímu vyhledávání. Odesílatel sonduje cestu různými řetězci paketů a odesílatel mu zpátky oznamuje vývojovou tendenci jednosměrného zpoždění každého proudu. Odesílatel dále ověřuje, že na síti je vždy maximálně jeden proud a mezi následujícími proudy udržuje odstupy, aby sondovací provoz nepřekročil 10 % dostupné kapacity cesty.

Odhad dostupné kapacity A se může během měření měnit. SLoPS odhalí takové kolísání, když zjistí, že jednosměrná zpoždění nemají tendenci ani stoupat, ani být konstantní. V tomto případě metodologie oznámí *šedý region* (*grey region*), který se týká rozsahu, ve kterém se pohybují hodnoty A během měření.

5.4 Řetězce párů paketů(Trains of Packet Pairs – TOPP)

Tato metoda je zavedena v [20] a vylepšena v [21] a týká se měření dostupné kapacity síťových cest. TOPP odesílá mnoho párů paketů s postupně rostoucí rychlostí. Nechť je pár paketů odeslán s počáteční disperzí Δ_S a sondovací pakety mají velikost L bytů. V tomto případě je *nabízená rychlost* (*offered rate*) $R_o = L / \Delta_S$. Pokud je R_o větší než dostupná kapacita A, druhý paket bude umístěn ve frontě za prvním paketem, a proto *naměřená rychlost* (*measured rate*) na příjemci bude $R_m < R_o$. Když $R_o < A$, TOPP předpokládá, že pár paketů dorazí k příjemci stejnou rychlostí, jakou byl odeslán. Tato základní idea je podobná jako idea SLoPS. Největší rozdíly mezi těmito metodami se týkají statistického zpracování měření. TOPP zvyšuje nabízenou rychlost lineárně, kdežto SLoPS používá pro přizpůsobení rychlosti binární vyhledávání. Důležitým rozdílem mezi TOPP a SLoPS je, že TOPP může změřit také kapacitu těsného spoje cesty. Tato kapacita může však být větší než kapacita cesty, protože úzký a těsný spoj mohou být různé.

Uvažme cestu skládající se z jednoho spoje s kapacitou C, dostupnou kapacitou A a průměrnou rychlostí křížujícího provozu $R_c = C - A$. TOPP odesílá páry paketů s rostoucí nabízenou rychlostí R_o. Když R_o překročí A, naměřená rychlost páru paketů bude

$$R_m = \frac{R_o}{R_o + R_c} C . \quad (15)$$

TOPP odhadne dostupnou kapacitu A jako maximální nabízenou rychlost, pro kterou platí $R_o \approx R_m$.

5.5 Chirp Packet Trains(CPT)

Jedná se o metodu odvozenou od SLoPS a TOPP, avšak používající odlišný model řetězců paketů, zaměřující se na zmenšení latence měření.

V rámci řetězce paketů jsou exponenciálně zmenšovány mezery mezi jednotlivými pakety. Takovýto řetězec je označovány jako *chirp* (doslovný překlad je cvrkot – dále bude používán spíše původní anglický termín). Díky prudkému zvětšování rychlosti sondování v rámci každého řetězce má tato metoda k dispozici sadu informací pro široké spektrum rychlostí paketů.

Kapitola 6 - Aplikace *meas*

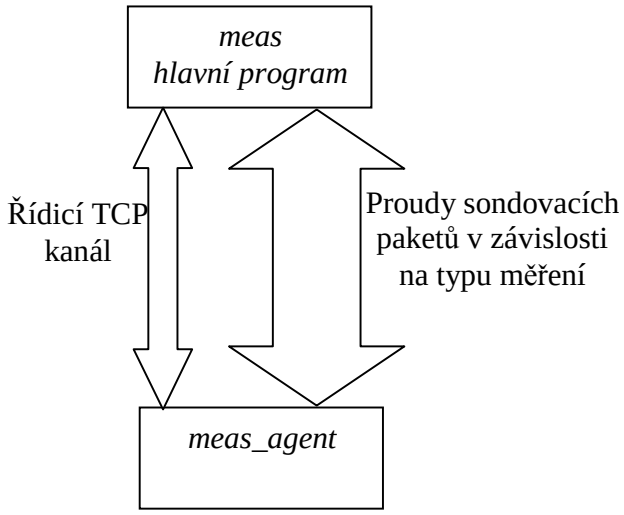
6.1 Cíle aplikace meas

Součástí této práce je vývoj aplikace pro měření důležitých vlastností datových přenosů. Tato aplikace je nazvána *meas* jako zkratka anglického slova *measuring* – měření.

Cílem bylo, aby aplikace umožňovala měření veličin důležitých pro datové přenosy. Informace o jednotlivých veličinách jsou uvedeny v kapitole 3. Na základě těchto informací jsem provedl výběr veličin pro implementaci v aplikaci *meas*. Byly vybrány následující veličiny: dostupná kapacita, kapacita, propustnost TCP, doba obrátky (čili latence), a ztráta paketů. Opomeneme-li tedy dynamiku směrování a topologii, *meas* lze použít přímo či nepřímo pro měření všech veličin z kapitoly 3.

Všechna tato měření jsou implementována jako end-to-end, tj. je měřená celá síťová cesta mezi dvěma koncovými uzly.

meas používá aktivní měřící techniky, jejichž výhody a nevýhody jsou popsány v kapitole 2.6. Aplikace *meas* se skládá z hlavního programu *meas* a z programu *meas_agent*. Každý z těchto programů je určen ke spuštění na jednom z koncových uzlů měřené cesty.



Obr. 1: Schéma komunikace mezi *meas* a *meas_agent*

meas_agent je navržen tak, aby mohl být spuštěn nepřetržitě a aby přijímal požadavky na měření a obsluhoval je. Představa je taková, že *meas_agent* bude spuštěn na počítači, ke kterému nemáme vždy přístup, ale chceme ho často používat jako jeden koncový uzel měřených síťových cest.

Oproti tomu hlavní program *meas* reprezentuje opačný koncový uzel měření a je potřeba ho spustit na počítači, ke kterému máme přístup. Při spouštění hlavního programu *meas* je potřeba vybrat požadované měření, zadat adresu stanice, na které je spuštěn *meas_agent*, k němuž chceme měřit síťovou cestu, a specifikovat, zda a jak se mají měření opakovat.

meas umožňuje:

- **Jednorázová měření** – měření je provedeno jednou při spuštění hlavního programu.
- **Periodická měření** - měření je spouštěno neustále v pravidelných časových intervalech, tj. například každých 30 minut.
- **Nepřetržitá měření** - jakmile měření skončí, je ihned spuštěno znovu. Pokud měření trvalo méně než minutu, je následující měření spuštěno až po uběhnutí minuty. Celý tento proces je opakován po určitý nastavený čas.

Agent naslouchá na řídicím TCP kanálu. Má-li začít měření, hlavní program pošle agentovi po tomto kanálu identifikaci typu měření a následně začne samotné měření.

meas_agent plní roli *SND*, hlavní program *RCV* (kromě měření ztráty paketů, kterého se agent neúčastní, takže hlavní program zde je jak *SND*, tak *RCV*). Toto rozdělení rolí jsem provedl s přihlédnutím k tomu, že *meas_agent* je vhodný k tomu, aby byl spuštěn na počítači, který bude plnit roli serveru nějaké síťové služby, kdežto hlavní program ve většině případů bude spouštěn spíše na klientských stanicích. Z toho je zřejmé, že zajímavější bude měřit síťovou cestu vedoucí z počítače se spuštěným agentem k počítači se spuštěným hlavním programem.

Dále budou popsány detaily měření jednotlivých veličin v *meas*.

6.2 Měření dostupné kapacity

Pro měření end-to-end dostupné kapacity lze použít metody PPTD, SLoPS, TOPP a CPT popsané v kapitole 5.

V [17] je ukázáno, že navzdory dřívějším pracím (např. [22]), disperze řetězců paketů není dostupnou kapacitou cesty, proto měření pomocí PPTD by nemuselo dávat požadované výsledky.

V [25] jsou empiricky porovnány metody SLoPS, TOPP a CPT. Závěr z těchto porovnání je takový, že CPT je nejlepší, co se týče přesnosti a účinnosti. Nejeví známky závislosti na velikosti paketů a má přijatelné chování i na cestách s mnoha přeskoky s různými modely křížujícího provozu. Výkon SLoPS není příliš horší než CPT. Metoda TOPP byla shledána jako velmi citlivá na velikost paketů křížujícího provozu a také velmi pomalá.

Na základě těchto informací a s přihlédnutím k tomu, že dostupná kapacita je nejdůležitější veličinou měřenou aplikací *meas*, jsem se rozhodl, že měření dostupné kapacity v *meas* bude implementováno dvěma způsoby - pomocí metod CPT a SLoPS. Dále budou popsány detaily těchto metod tak, jak jsou implementovány v *meas*.

6.3 Měření dostupné kapacity pomocí CPT

6.3.1 Dostupná kapacita cesty

Uvažme síťovou cestu P definovanou jako posloupnost kapacit spojů $P = \{C_0, C_1, ..., C_H\}$. Celkový objem křížujícího provozu na spoji i mezi časy a a b označíme jako $u_i[a, b]$. Pak dostupná kapacita cesty v časovém intervalu $[t - \tau, t]$ je

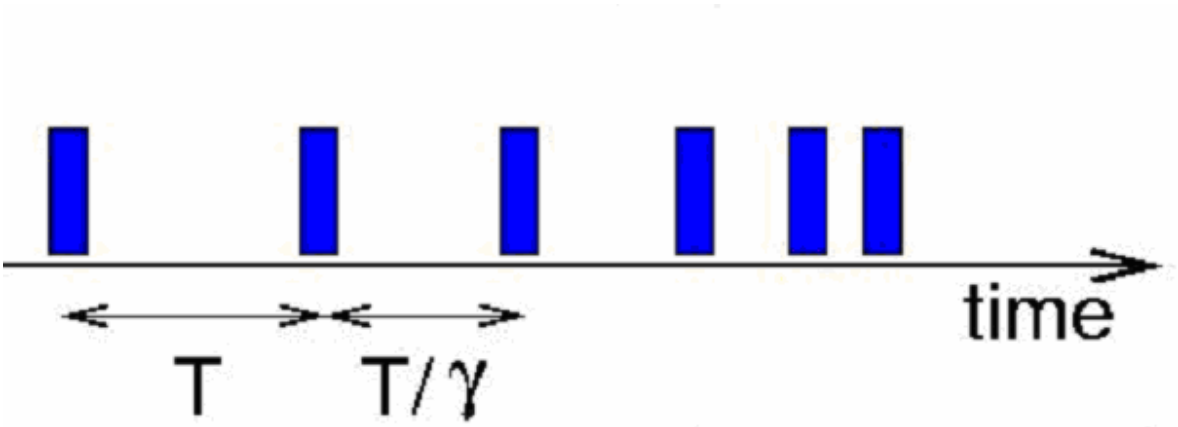
$$A[t-\tau,t] = \min_i (C_i - \frac{u_i[t-\tau+p_i, t+p_i]}{\tau}), \quad (16)$$

kde p_i je minimální čas potřebný k tomu, aby se paket dostal ze *SND* na směrovač i. Zpoždění p_i se skládá z fyzického a serializačního zpoždění.

U sondovacích paketů je potřeba počítat kromě minimálního zpoždění p_i také s frontovým zpožděním. Tedy sondy odeslané během intervalu $[t - \tau, t]$ mohou dorazit na směrovač i mimo časový interval $[t - \tau + p_i, t + p_i]$, a tedy přesně nezměří $A[t - \tau, t]$. Pro velké τ (>> RTT) se však efekty frontového zpoždění stávají zanedbatelnými.

6.3.2 Implementace odhadu dostupné kapacity

meas odhaduje dostupnou kapacitu cesty pomocí CPT tak, že vyšle určitý počet řetězců paketů (chirpů) ze *SND* na *RCV* a následně je provedena statistická analýza na *RCV*. Vysílané chirpy jsou číslovány jako $m = 1, 2, \ldots$



Obr. 2: Chirp paketů

Nechť chirp m se skládá z N paketů s exponenciálně klesajícími mezerami mezi následujícími pakety, každý paket o velikosti L. Definujeme *faktor rozpětí* γ jako poměr velikostí následujících mezer mezi pakety v rámci chirpu. Frontové zpoždění paketu k označíme jako $q_k^{(m)}$, čas odeslání paketu k ze *SND* jako $t_k^{(m)}$, velikost mezery mezi pakety k a $k + 1$ jako $\Delta_k^{(m)}$ a okamžitou rychlost paketu k jako:

$$R_k^{(m)} = \frac{L}{\Delta_k^{(m)}}. \tag{17}$$

Jelikož $\Delta_k^{(m)}$ a $R_k^{(m)}$ jsou stejné pro všechny chirpy, dále nebude potřeba je označovat číslem chirpu.

Předpokládáme-li, že křížující provoz má konstantní rychlost, platí:

$$q_k^{(m)} = 0, \quad \text{pokud } A[t_1^{(m)}, t_N^{(m)}] \geq R_k \tag{18}$$
$$q_k^{(m)} > q_{k-1}^{(m)} \quad \text{v ostatních případech ,}$$

z čehož se dá odvodit jednoduchý odhad:

$$\hat{A}[t_1^{(m)}, t_N^{(m)}] = R_{k*}, \tag{19}$$

kde k* je paket, na kterém se začne zvětšovat frontové zpoždění.

Nicméně předpoklad konstantní rychlosti křížujícího provozu je přílišným zjednodušením. Kvůli shlukům křižujícího provozu frontové zpoždění typicky není monotónní během celého chirpu. Průběh frontového zpoždění v čase během jednoho chirpu je označován jako *signatura frontového zpoždění*. Typická signatura obsahuje odchylky od nulové osy způsobené shluky křížujícího provozu. Dokud je R_k menší než kapacita úzkého spoje $C_{\min}$, po odchylce způsobené shlukem křížujícího provozu se signatura vrací na nulovou osu. Poslední odchylka obvykle končí s rostoucími frontovými zpožděními, protože $R_k > C_{\min}$, tj. pakety chirpu zahltí fronty.

meas používá tvar signatury k vytvoření odhadu $E_k^{(m)}$ *dostupné kapacity per paket* $A[t_k^{(m)}, t_{k+1}^{(m)}]$. Dále je pomocí váženého průměru $E_k^{(m)}$ vypočítán odhad $D^{(m)}$ *dostupné kapacity per-chirp* $A[t_1^{(m)}, t_N^{(m)}]$:

$$D^{(m)} = \frac{\sum_{k=1}^{N-1} E_k^{(m)} \Delta_k}{\sum_{k=1}^{N-1} \Delta_k}. \quad (20)$$

Nakonec je vypočítán odhad $\rho[t - \tau, t]$ dostupné kapacity $A[t - \tau, t]$ jako průměr odhadů $D^{(m)}$ získaných v časovém intervalu $[t - \tau, t]$.

6.3.3 Rozdělení na segmenty podle odchylek

Princip měření dostupné kapacity v *meas* pomocí CPT je založen na předpokladu, že zvětšující se frontová zpoždění znamenají, že dostupná kapacita je menší než okamžitá rychlost chirpu, zmenšující se zpoždění znamenají opak. To znamená:

$$E_k^{(m)} \geq R_k, \quad \text{pokud } q_k^{(m)} \geq q_{k+1}^{(m)} \quad (21)$$

$$E_k^{(m)} \leq R_k, \quad \text{v ostatních případech.} \quad (22)$$

Uvažujeme-li jeden přeskok, vztah (21) je vždy pravdivý, zatímco (22) nemusí vždy platit. Například mezera mezi pakety k a $k + 1$ může být enormně velká. V takovém případě fakt, že $q_k^{(m)} < q_{k+1}^{(m)}$ říká velmi málo o $E_k^{(m)}$. Pakety k a $k + 1$ totiž nemohou vyvolat zahlcení v síti a spíše představují nezávislé vzorky frontového zpoždění na cestě.

Aby bylo možné využít vztah (22), je každá signatura frontového zpoždění rozdělená na segmenty – pro každou odchylku v signatuře jeden segment. Vztah (22) je aplikován pouze na tyto segmenty. Zřejmě pokud se $q_k^{(m)}$ zvětší a zůstává větší než 0 pro několik následujících paketů, je pravděpodobné, že všechny tyto pakety jsou součástí fáze, ve které nějaká fronta na cestě není nikdy prázdná. V tomto případě lze očekávat, že vztah (22) platí. Je tedy potřeba najít segmenty v signatuře, pro které platí $q_k^{(m)} > 0$ pro několik následujících paketů.

V praxi se vyskytuje určitá odchylka mezi hodinami na obou koncových uzlech – není tedy možné používat $q_k^{(m)} > 0$ pro detekci segmentů. Místo toho *meas* používá relativní frontové zpoždění uvnitř chirpu.

6.3.4 Algoritmus pro rozdělení na segmenty podle odchylek

Cílem algoritmu pro rozdělení na segmenty je určit čísla *i* a *j* prvního a posledního paketu segmentu. Každý paket *i*, pro který platí $q_i^{(m)} < q_{i+1}^{(m)}$ je potenciálním prvním paketem segmentu. Konec segmentu *j* definujeme jako první paket, pro který platí:

$$q(j) - q(i) < \frac{\max_{i \le k \le j}[q(k) - q(i)]}{F}, \quad (23)$$

kde *F* je *zmenšující faktor*. Pokud rozdíl *j* – *i* je dostatečně velký, tj. segment signatury je dostatečně dlouhý, potom všechny pakety mezi *i* a *j* tvoří odchylku.

Poslední odchylka v signatuře obvykle není ukončena, tj. existuje paket *l*, pro který platí $q_l^{(m)} < q_{l+1}^{(m)}$ a neexistuje *j* > *l*, pro který platí (23) (namísto *i* dosadíme *l*).

6.3.5 Výpočet odhadů dostupné kapacity per-paket

Zbývá vypočítat odhady dostupné kapacity per-paket $E_k^{(m)}$. Každý paket *k* chirpu spadá do jedné z následujících tří kategorií:

1. Pokud *k* patří do odchylky, která je ukončena a $q_k^{(m)} \le q_{k+1}^{(m)}$, bude:

$$E_k^{(m)} = R_k . \quad (24)$$

Tato definice vyhovuje vztahu (22).

2. Pokud *k* patří do odchylky, která není ukončena, bude:

$$E_k^{(m)} = R_l, \forall k > l , \quad (25)$$

kde *l* je začátek odchylky.

Důvod, proč i v tomto případě se nepoužívá (24), je, že rychlost paketu *k* v této odchylce může být výrazně větší než $C_{\min}$. Dostupná kapacita je vždy menší než $C_{\min}$, proto musí platit $E_k^{(m)} < R_k$.

Nicméně podle (21) platí: $E_k^{(m)} > R_k > R_l$, pokud $q_k^{(m)} > q_{k+1}^{(m)}, k > l$. Pro takové *k* (25) vede k poněkud konzervativním odhadům.

3. Pro všechna *k*, která nejsou pokryta body 1. ani 2., nastavíme:

$$E_k^{(m)} = R_l . \quad (26)$$

Zde jsou zahrnuta *k*, která nepatří do odchylek a také *k* s klesajícím frontovým zpožděním patřící do odchylek. Pokud poslední odchylka signatury není ukončena, zvolíme *l* = *N* – 1.

Následuje úryvek kódu aplikace *meas* – implementace výše popsaných algoritmů pro výpočet dostupné kapacity jednoho chirpu:

```
  int cur_loc=0;/* aktuální pozice v chirpu */
  int cur_inc=0;/* aktuální pozice, kde se zvětšuje frontové zpoždění */
  int count;
  double inst_bw=0.0,sum_iat=0.0,max_q=0.0;

  /* nastav všechny av_bw_per_pkt na nulu */
  memset(av_bw_per_pkt, 0, (int)(num_interarrival) * sizeof(double));

  /* najdi první místo, kde roste frontové zpoždění */
  while(qing_delay[cur_inc]>=qing_delay[cur_inc+1] &&
cur_inc<max_good_pkt_this_chirp)
    cur_inc++;

  cur_loc=cur_inc+1;

  /* projdi všechna zpoždění */
  while(cur_loc<=max_good_pkt_this_chirp)
    {

      if (qing_delay[cur_loc]>NEG_THRESH+1.0)
      {
        /* najdi maximální frontové zpoždění mezi cur_inc a cur_loc*/
        if (max_q<(qing_delay[cur_loc]-qing_delay[cur_inc]))
          max_q=qing_delay[cur_loc]-qing_delay[cur_inc];

        if (qing_delay[cur_loc]-qing_delay[cur_inc]<(max_q/decrease_factor))
          {
            if (cur_loc-cur_inc>=busy_period_thresh)
            {
              for (count=cur_inc;count<=cur_loc-1;count++)
                {
                  /* všechny regiony s rostoucím zpožděním mezi cur_inc a
cur_loc mají dostupnou kapacitu rovnou rychlosti */
                  if (qing_delay[count]<qing_delay[count+1])
                  av_bw_per_pkt[count]=rates[count];
                }
            }
            /* najdi další bod, kde roste zpoždění */
            cur_inc=cur_loc;
            while(qing_delay[cur_inc]>=qing_delay[cur_inc+1] &&
cur_inc<max_good_pkt_this_chirp)
            cur_inc++;
            cur_loc=cur_inc;
            /* vynuluj max_q*/
            max_q=0.0;
          }
      }
      cur_loc++;
    }

  if(cur_inc==max_good_pkt_this_chirp)
    cur_inc--;

    /*dostupnou kapacitu během poslední odchylky nastav na rychlost na začátku
odchylky */
```

```
for(count=cur_inc;count<max_good_pkt_this_chirp;count++)
  {
    av_bw_per_pkt[count]=rates[cur_inc];
  }


/* všude jinde je dostupná kapacita popsaná poslední odchylkou */
for(count=0;count<max_good_pkt_this_chirp;count++)
  {
    if (av_bw_per_pkt[count]<1.0)
    av_bw_per_pkt[count]=rates[cur_inc];
    sum_iat+=iat[count];
    inst_bw+=av_bw_per_pkt[count]*iat[count];
  }

inst_bw=inst_bw/sum_iat;
```

K měření dostupné kapacity používá *meas* UDP chirpy paketů. Tyto pakety se pohybují ze *SND* na *RCV* a *RCV* počítá odhady. Tento model byl zvolen kvůli tomu, že posílání informací zpět na *SND* by mohlo narušit provoz chirpů ze *SND* na *RCV*. Toto by se mohlo stát na spojích, které nejsou plně duplexní.

Meas se snaží řešit praktický problém přepínání kontextu procesoru. Když dojde k přepnutí kontextu na počítači přijímajícím sondovací pakety, pakety jsou dočasně bufferovány, dokud procesor obsluhuje ostatní procesy. Takto vznikají zpoždění mezi pakety těsně před přepnutím kontextu a těsně po něm. Navíc bufferované pakety po přepnutí kontextu rychle dorážejí na aplikační vrstvu. Toto chování může ovlivnit prováděné měření, proto, je-li rozdíl mezi dvěma následujícími časovými razítky přijetí paketů menší než prahová hodnota *d*, je detekováno přepnutí kontextu a příslušný chirp je vyřazen. *d* je nastaveno na 30 μs, což je méně než čas přenosu 1000 bytového paketu na spoji s kapacitou 100 Mbps (80 μs).

6.4 Měření dostupné kapacity pomocí SLoPS

6.4.1 Úvod do metody SLoPS

Periodický proud v metodě SLoPS se skládá z K paketů velikosti L odesílaných konstantní rychlostí R. Pokud rychlost R je větší než dostupná kapacita A, jednosměrná zpoždění následujících paketů mají na *RCV* rostoucí tendenci.

Uvažme cestu ze *SND* na *RCV*, která se skládá z H spojů, $i = 1,\ldots,H$. Kapacita spoje i je C_i. Předpokládejme, že křížující provoz je nezávislý na čase. Tedy pokud dostupná kapacita spoje i je A_i, potom využití je $u_i = (C_i - A_i)/C_i$ a v jakémkoliv intervalu délky τ skrz spoj i projde $u_i C_i \tau$ bytů křížujícího provozu. Dále předpokládejme, že všechny spoje se chovají jako *First-Come*

First-Served a že obsahují dostatečný počet bufferů k zabránění ztrátám. Budeme ignorovat fyzická a další fixní zpoždění na cestě, protože tato nemají vliv na kolísání zpoždění mezi pakety. Dostupná kapacita *A* cesty je určena těsným spojem $t \in \{1,...,H\}$, pro který platí:

(27) $$A_t = \min_{i=1...H} A_i = \min_{i=1...H} C_i(1-u_i) = A.$$

Předpokládejme, že *SND* odesílá periodický proud *K* paketů na *RCV* rychlostí R_0. Velikost paketů je *L* bytů, a tedy pakety jsou odesílány s periodou $T = L/R_0$. *Jednosměrné zpoždění* (*One-Way Delay* - OWD) D^k paketu *k* ze *SND* na *RCV* je

(28) $$D^k = \sum_{i=1}^{H}(\frac{L}{C_i} + \frac{q_i^k}{C_i}) = \sum_{i=1}^{H}(\frac{L}{C_i} + d_i^k),$$

kde q_i^k je velikost fronty na spoji *i* při příchodu paketu *k* (q_i^k nezahrnuje paket *k*), $d_i^k = q_i^k / C_i$ je frontové zpoždění paketu *k* na spoji *i*. Rozdíl OWD mezi dvěma následujícími pakety *k* a *k* + 1 je

(29) $$\Delta D^k = D^{k+1} - D^k = \sum_{i=1}^{H}\frac{\Delta q_i^k}{C_i} = \sum_{i=1}^{K}\Delta d_i^k,$$

kde $\Delta q_i^k = q_i^{k+1} - q_i^k$ a $\Delta d_i^k = \Delta q_i^k / C_i$.

Uvažujme první spoj. Nechť $R_0 > A_1$. Předpokládejme, že t^k je čas příchodu paketu do fronty. Během intervalu $[t^k, t^k + T)$, kde $T = L/R_0$, je spoj zaneprázdněn, protože rychlost je větší než kapacita ($R_0 + u_1C_1 = C_1 + (R_{.0} - A_1) > C_1$). Během tohoto intervalu do spoje dorazí $L + u_1C_1T$ bytů a je přeneseno C_1T bytů. Tedy

(30) $$\Delta q_1^k = (L + u_1C_1T) - C_1T = (R_0 - A_1)T > 0,$$

a tedy

(31) $$\Delta d_1^k = \frac{R_0 - A_1}{C_1}T > 0.$$

Paket *k* + 1 opouští první spoj Λ časových jednotek po paketu *k*, kde

(32) $$\Lambda = (t^{k+1} + d_1^{k+1}) - (t^k + d_1^k) = T + \frac{R_0 - A_1}{C_1}T,$$

což není závislé na *k*. Tedy pakety proudu opouštějí první spoj konstantní rychlostí R_1, kde

(33) $$R_1 = \frac{L}{\Lambda} = R_0\frac{C_1}{C_1 + (R_0 - A_1)}.$$

Rychlost R_{i-1} budeme označovat jako *rychlost vstupu* na spoj *i*, R_i jako *rychlost výstupu* ze spoje *i*.

Jelikož $R_0 > A_1$ a $C_1 \geq A_1$, je zřejmé, že rychlost výstupu ze spoje *1* je větší nebo rovna než A_1 a menší než rychlost vstupu

(34) $A_1 \leq R_1 < R_0$.

Nyní nechť $R_0 \leq A_1$. V tomto případě je rychlost příchodu paketů na spoj v intervalu $[t^k, t^k + T)$ rovná $R_0 + u_1 C_1 \leq C_1$. Tedy paket k je obsloužen před tím než paket $k + 1$ dorazí do fronty. Tedy platí

(35) $\Delta q_1^k = 0, \Delta d_1^k = 0, R_1 = R_0$.

Výše ukázané výsledky pro první spoj lze nyní indukcí aplikovat na každý spoj cesty. Dostaneme následující vztahy pro rychlosti vstupu a výstupu spoje i:

(36) $R_i = R_{i-1} \dfrac{C_i}{C_i + (R_{i-1} - A_i)}$ pokud $R_{i-1} > A_i$

$R_i = R_{i-1}$ v ostatních případech.

Dále:

(37) $A_i \leq R_i < R_{i-1}$ když $R_{i-1} > A_i$.

V důsledku toho rychlost výstupu ze spoje i je:

(38) $R_i \geq \min\{R_{i-1}, A_i\}$.

Rozdíl frontových zpoždění mezi následujícími pakety na spoji i je

(39) $\Delta d_i^k = \dfrac{R_{i-1} - A_i}{C_i} > 0$ pokud $R_{i-1} > A_i$

$\Delta d_i^k = 0$ v ostatních případech

Pokud $R_0 > A$, lze použít (37) rekurzivně pro i = 1, …, (t – 1) a ukázat, že proud dorazí na těsný spoj t rychlostí $R_{t-1} \geq A_{t-1} > A_t$. Tedy podle (39) platí $\Delta d_t^k > 0$ a rozdíl OWD mezi následujícími pakety bude kladný, tj. $\Delta D^k > 0$ pro k = 1, ..., K – 1.

Pokud $R_0 \leq A$, pak $R_0 \leq A_i$ pro každý spoj i. Tedy rekurzivním použitím (38) pro všechny spoje cesty lze ukázat, že $R_i < A_i$ pro i = 1, …, H. Tedy podle (39) platí $\Delta d_i^k = 0$ pro každý spoj i, a tedy rozdíly OWD jsou nulové, tj. $\Delta D^k = 0$ pro k = 1, ..., K – 1.

Z (36) dále plyne, že rychlost R_H řetězce paketů na *RCV* je v obecném případě funkce C_i a A_i pro všechna i = 1, …, H. Není tedy vhodné počítat dostupnou kapacitu cesty A přímo z R_H.

Pro odhad A lze použít dále popsaný algoritmus založený na principu binárního vyhledávání. Nechť *SND* odesílá periodický proud n rychlostí $R(n)$. *RCV* analyzuje kolísání OWD proudu za účelem určení, zda $R(n) > A$. Následně *RCV* oznámí *SND* vztah mezi $R(n)$ a A. Pokud $R(n) > A$, *SND* odešle další periodický proud $n + 1$ s rychlostí $R(n + 1) < R(n)$. V opačném případě rychlost proudu $n + 1$ bude $R(n + 1) > R(n)$.

Konkrétně je $R(n + 1)$ počítáno následovně:

(40) Pokud $R(n) > A, R^{\max} = R(n)$;

Pokud $R(n) \leq A, R^{\min} = R(n)$;

$$R(n+1) = \frac{R^{\max} + R^{\min}}{2};$$

$R^{\min}$ a $R^{\max}$ jsou dolním a horním odhadem dostupné kapacity po proběhnutí proudu n.

6.4.2 Implementace SLoPS v *meas*

Pro periodické proudy paketů je používáno UDP, navíc mezi *SND* a *RCV* je vytvořeno TCP řídicí spojení. Řídicí spojení přenáší zprávy týkající se vlastností každého proudu, přerušení nebo ukončení měřícího procesu, apod.

Důležitými parametry měření jsou perioda T a velikost paketů L. Rychlost R proudu je

(41) $$R = \frac{L}{T}.$$

Máme-li dáno R, je potřeba zvolit L a T tak, aby platil vztah (41).

Existují však praktická omezení pro volbu L a T. L nemůže být větší než MTU cesty, aby nedocházelo k fragmentaci. Na druhou stranu L nemůže být příliš malé, neboť tzv. *zero-padding*, neboli doplnění „vycpávky“, může způsobit významné rozdíly ve velikosti paketů. *meas* zajišťuje, aby L bylo větší než 96 bytů, menší než MTU cesty a násobkem 48 bytů, aby nedocházelo k „vycpávání“ na ATM spojích.

Perioda T by měla být pokud možno co nejmenší, protože s rostoucím T roste délka trvání každého proudu. V ideálním případě by měl být přenos každého proudu dokončen před tím, než nastane přepnutí kontextu na *SND* nebo *RCV*. Navíc menší hodnota T vede ke kratšímu trvání celého měřícího procesu. Zdola je T omezeno možnostmi hardwaru a operačních systémů na *SND* a *RCV* – konkrétně T musí být určitě větší než čas na přenos paketu s minimální velikostí. V *meas* je T nastaveno na 100μs.

Za účelem dosažení požadované rychlosti proudu *R meas* zvolí minimální možnou periodu *T*=100μs a následně podle vztahu (41) vypočítá příslušnou velikost paketů *L*. Pokud je výsledná velikost paketů menší než minimální přípustná hodnota 96 bytů, je *L* nastaveno na tuto hodnotu a *T* je vypočteno dle (41).

Dále je potřeba zvolit počet paketů *K* v proudu. Pokud je *K* příliš velké, proud může přeplnit frontu těsného spoje, když *R* > *A*, což způsobí ztráty paketů proudu i křížujícího provozu. Na druhou stranu pokud *K* je příliš malé, *RCV* nemusí mít k dispozici dostatečné množství informací k určení, zda OWD mají rostoucí tendenci.

Určení, zda *R* > *A* však není založeno na jednom proudu, ale na posloupnosti *N* proudů. Každý proud se skládá z *K* paketů velikosti *L* odesílaných periodicky každých *T* sekund. Všechny proudy v posloupnosti mají stejnou rychlost *R* = *L* / *T*. Každý proud je odeslán až poté, co byl předchozí proud potvrzen. Toto zavádí interval „klidu“ o délce jedné doby obrátky Δ mezi proudy.

Existují dva hlavní důvody, proč použít *N* proudů po *K* paketech namísto jedné posloupnosti *N* x *K* paketů. Prvním je, že *N* proudů dovoluje prověřit, zda *R* > *A* *N*-krát, protože *RCV* zjišťuje rostoucí tendenci z naměřených OWD nezávisle pro každý proud. Druhým důvodem je, že použití mnoha proudů oddělených intervalem „klidu“ Δ umožňuje, aby se fronty v síti vyprázdnily a zotavily se tak z krátkodobého přetížení způsobeného proudem. Jako *A* je v *meas* použita hodnota 12 proudů.

N, *K* a *T* určují dobu trvání *U* posloupnosti proudů, kde

(42) $$U = N \times (K \times T + \Delta).$$

Detekce rostoucí tendence

Důležitým algoritmem je algoritmus pro detekci rostoucí tendence OWD. Na začátku se *K* měření OWD $\{D_1, D_2, ..., D_K\}$ rozdělí na $\Gamma = \sqrt{K}$ skupin po Γ následujících OWD měřeních. Následně se z Γ zpoždění v každé skupině *i* spočítá medián $\hat{D}_i$ skupiny.

Pro zjišťování rostoucího trendu se používají dvě statistické veličiny. *Pairwise Comparison Test* (PCT) je definována následovně:

(43) $$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}_k > \hat{D}_{k-1})}{\Gamma - 1},$$

kde $I(X) = 1$, pokud X platí, v opačném případě $I(X) = 0$. Je zřejmé, že $0 \le S_{PCT} \le 1$. Pokud jsou hodnoty OWD nezávislé, očekávaná hodnota S_{PCT} je 0,5. Pokud mají OWD silnou rostoucí tendenci, S_{PCT} se blíží k 1. *meas* ohlásí „rostoucí tendenci“, pokud $S_{PCT} > 0{,}66$, „nerostoucí tendenci“, pokud $S_{PCT} < 0{,}54$. V ostatních případech ohlásí „nejednoznačnou tendenci“.

Druhá veličina – *Pairwise Difference Test* (PDT) - je definována následovně:

$$S_{PDT} = \frac{\hat{D}_\Gamma - \hat{D}_1}{\sum_{k=2}^{\Gamma} \left| \hat{D}_k - \hat{D}_{k-1} \right|}. \tag{44}$$

PDT určuje, jak velký je rozdíl OWD na konci a na začátku v porovnání s absolutními hodnotami rozdílů OWD uvnitř proudu. Zřejmě platí $-1 \le S_{PDT} \le 1$. Pokud jsou hodnoty OWD nezávislé, očekávaná hodnota S_{PDT} je 0. Pokud mají OWD silnou rostoucí tendenci, S_{PDT} se blíží k 1. *meas* ohlásí „rostoucí tendenci“, pokud $S_{PDT} > 0{,}55$, „nerostoucí tendenci“, pokud $S_{PDT} < 0{,}45$. V ostatních případech ohlásí „nejednoznačnou tendenci“.

Když PCT nebo PDT ohlásí „rostoucí tendenci“, zatímco druhá veličina hlásí „rostoucí“ nebo „nejednoznačnou“, proud je označen jako „rostoucí“. Obdobně ohlásí-li jedna veličina „nerostoucí tendenci“, zatímco druhá hlásí „nerostoucí“ nebo „nejednoznačnou“, proud je označen jako „nerostoucí“. Hlásí-li obě veličiny „nejednoznačnou tendenci“ nebo hlásí-li jedna „rostoucí“, zatímco druhá „nerostoucí“, proud je zahozen.

Po přijetí všech N proudů *RCV* určí zda $R > A$. Pokud velká část f z N proudů v posloupnosti je „rostoucí“, celá posloupnost má rostoucí tendenci a z toho se usoudí, že $R > A$. Obdobně pokud velká část f z N proudů v posloupnosti je „nerostoucí“, celá posloupnost nemá rostoucí tendenci a z toho se usoudí, že $R < A$. Pokud se stane, že méně než $N \times f$ proudů je „rostoucích“ a zároveň méně než $N \times f$ proudů je „nerostoucích“, rychlost R posloupnosti patří do tzv. „*šedé oblasti*“ dostupné kapacity, což budeme označovat $R \approx A$. V *meas* je f nastaveno na 70%.

Algoritmus pro přizpůsobení rychlosti

Po proběhnutí posloupnosti n s rychlostí $R(n)$ *meas* určí, zda $R(n) > A$, $R(n) < A$ nebo $R(n) \approx A$. Iterativní algoritmus, který určuje rychlost $R(n + 1)$ další posloupnosti je podobný přístupu pomocí binárního hledání popsanému vztahem (40). Nicméně existují významné rozdíly. Kromě horního a dolního odhadu dostupné kapacity $R^{\max}$ a $R^{\min}$ *meas* počítá také horní a dolní

hranici „šedé oblasti“ $G^{\max}$ a $G^{\min}$. Když $R(n) \approx A$, jedna z těchto hranic je aktualizována v závislosti na tom, zda $G^{\max} < R(n) < R^{\max}$ nebo $G^{\min} > R(n) > R^{\min}$. Pokud doposud nebyla objevena „šedá oblast“, rychlost $R(n + 1)$ je vybraná jako v (40) ve středu mezi $R^{\max}$ a $R^{\min}$. Pokud šedá oblast byla objevena, $R(n + 1)$ je nastavena na hodnotu ve středu mezi $G^{\max}$ a $R^{\max}$, pokud $R(n) \geq G^{\max}$ nebo $R(n + 1)$ je nastavena na hodnotu ve středu mezi $G^{\min}$ a $R^{\min}$, pokud $R(n) \leq G^{\min}$.

Měření končí, když dostupná kapacita je odhadnuta s jistým rozlišením ω, tj. $R^{\max} - R^{\min} \leq \omega$ nebo když $R^{\max} - G^{\max} \leq \chi$ a $G^{\min} - R^{\min} \leq \chi$, tj. oba odhady dostupné kapacity nejsou vzdáleny o více než χ od příslušných hranic „šedé oblasti“.

Nakonec je oznámen interval $[R^{\min}, R^{\max}]$ jako **výsledný odhad dostupné kapacity**.

Po přijetí každého proudu *RCV* měří ztráty paketů v tomto proudu. Pokud proud obsahuje nevelké ztráty (méně než 3 %), je označen jako ztrátový. Pokud větší množství proudů je označeno jako ztrátové, *RCV* oznámí *SND*, aby zrušil odesílání zbývajících proudů v posloupnosti. Pokud proud obsahuje nadměrné množství ztrát (více než 10 %), posloupnost je okamžitě zrušena. Rychlost *R* zrušené posloupnosti se stane novým horním odhadem $R^{\max}$.

Obdobně jako v případě metody CPT i zde je snaha o detekci přepnutí kontextu procesoru. Konkrétně je zde kontrolováno, zda se objevilo přepnutí kontextu na *SND* během odesílání proudu. Nechť t_i je čas odeslání paketu *i* ze *SND*. t_i je zaznamenán v paketu *i*. *RCV* porovnává časy odeslání následujících paketů a zjišťuje, zda $t_{i+1} - t_i > T + W$, kde *W* je maximální povolená odchylka od periody *T*. Pokud $t_{i+1} - t_i > T + W$, předpokládáme, že na *SND* byl přepnut kontext po odeslání *i*-tého paketu proudu. Následně *RCV* rozdělí přijatý proud na dva proudy – první obsahuje pakety 1 až *i*, druhý pakety *i* + 1 až *K*.

Dále je prováděna kontrola, zda došlo k přepnutí kontextu na *RCV* během přijímání proudu. Nechť a_i je čas přijetí paketu *i* na *RCV*. Pokud je proces *meas* na *RCV* přepnut v průběhu přijímání proudu, některé pakety proudu budou umístěny v bufferu jádra operačního systému. Když *meas* zpátky dostane řízení, obdrží tyto pakety s rozestupy *Q* μs, kde *Q* je latence systémového volání *recvfrom*. *Q* je typicky několik mikrosekund a je možné to změřit před samotným začátkem měření. *RCV* tedy může odhalit přepnutí kontextu pomocí porovnávání časů přijetí následujících paketů. Pokud $a_{i+1} - a_i \approx Q$, pakety *i* a *i* + 1 byly bufferovány v jádře operačního systému a jsou vyřazeny z analýzy OWD.

6.5 Měření kapacity

Měření kapacity je možné provádět pomocí metod VPS a PPTD popsaných v kapitole 5. VPS měří kapacitu jednotlivých spojů síťové cesty, kdežto metody PPTD měří end-to-end kapacitu tak, jak je definována v kapitole 3.3.1.

Jelikož *meas* se zaměřuje na end-to-end měření, pro určování kapacity síťových cest byla použita metoda PPTD, která bude dále podrobněji popsána. Detaily budou popsány tak, jak jsou implementovány v *meas*.

Měření kapacity v *meas* vyžaduje spolupráci obou koncových uzlů cesty – *SND* i *RCV*. Existují flexibilnější přístupy vyžadující přístup pouze na *SND*, *RCV* je donucen odpovědět na každý přijatý paket pomocí ICMP, UDP-echo nebo TCP-FIN paketů. Nevýhodou těchto metod je to, že zpětná cesta od *RCV* k *SND*, po které jsou přenášeny odpovědi, může ovlivnit měření. Měření s přístupem na obou stranách je tedy sice méně flexibilní, ale přesnější.

6.5.1 Disperze páru paketů

Měření pomocí párů paketů může být formálněji popsáno následovně. Uvažme síťovou cestu P definovanou jako posloupnost kapacit spojů $P = \{C_0, C_1, ..., C_H\}$. Dva pakety velikosti L jsou přenášeny mezi *SND* a *RCV*. Ve spoji i je pro paket velikosti L *serializační zpoždění* $\tau_i = L / C_i$. *Disperze* Δ_i páru paketů za spojem i je časový interval mezi kompletním přenosem těchto dvou paketů spojem i. Disperze páru paketů za *SND* je $\Delta_0 = \tau_0 = L / C_0$, tj. pakety jsou odeslány „back-to-back“. Když pár paketů dorazí na *RCV*, *RCV* změří disperzi Δ_H a následně spočítá odhad kapacity $b = L / \Delta_H$. Jelikož Δ_H se může měnit mezi měřeními různých párů paketů, b může být považováno za spojitou náhodnou veličinu, která následuje *rozdělení kapacity B.*

Pro začátek předpokládejme, že na uvažované síťové cestě se nevyskytuje žádný křížující provoz. V tomto případě disperze Δ_i nemůže být menší než disperze Δ_{i-1} na předchozím přeskoku a serializační zpoždění $\tau_i = L / C_i$ na přeskoku i, tedy $\Delta_i = \max\{\Delta_{i-1}, \tau_i\}$. Použijeme-li tento model rekurzivně pro každý spoj, zjistíme, že disperze na *RCV* je

$$\Delta_H = \max_{i=0,...,H}(\tau_i) = \frac{L}{\min_{i=0,...H}(C_i)} = \frac{L}{C} = \frac{L}{C_n} = \tau_n, \quad (45)$$

kde C_n a τ_n jsou kapacita a serializační zpoždění úzkého spoje. Tedy bez křížujícího provozu je kapacita změřená párem paketů vždy rovná kapacitě nezávisle na velikosti L.

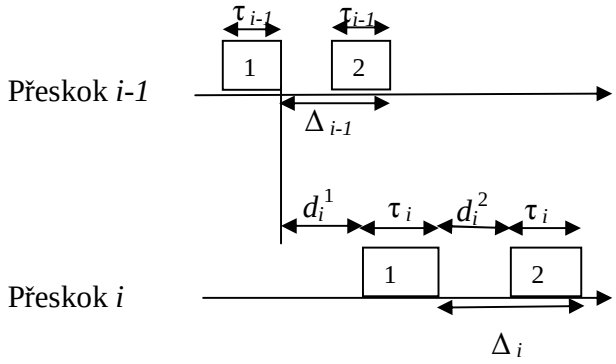
Když je přítomný křížující provoz, je potřeba počítat s dodatečným frontovým zpožděním. Označme frontové zpoždění prvního sondovacího paketu na přeskoku i jako d_i^1.

Frontové zpoždění druhého sondovacího paketu na přeskoku *i* po tom, co první paket opustil daný spoj, označíme jako d_i^2. Disperze za přeskokem *i* je

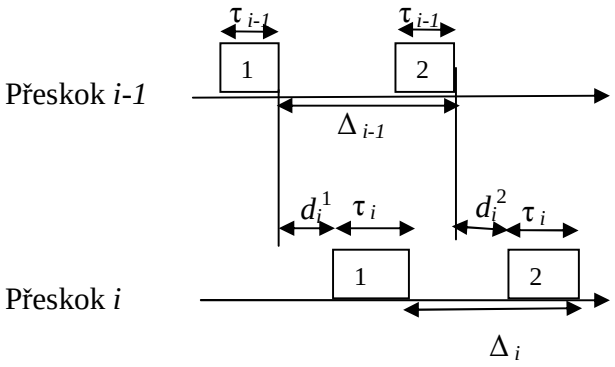
(46) $\Delta_i = \tau_i + d_i^2$ pokud $\tau_i + d_i^1 \geq \Delta_{i-1}$ - situace I

$\Delta_i = \Delta_{i-1} + (d_i^2 - d_i^1)$ v ostatních případech – situace II

Obě tyto situace jsou představeny na obr. 3 a 4.



Obr. 3: Definice Δ_i v situaci I



Obr. 4: Definice Δ_i v situaci II

Když $\tau_i + d_i^1 < \Delta_{i-1}$ a $d_i^2 < d_i^1$ disperze klesá od přeskoku $i - 1$ k přeskoku i, tj. $\Delta_i < \Delta_{i-1}$. Takto se tedy může stát, že disperze naměřená na *RCV* je menší než disperze na úzkém spoji. V tom případě však za úzkým spojem musí být nutně ještě další spoje. Tyto spoje budeme označovat jako *post-narrow spoje*. Je tedy zřejmé, že kapacitu cesty nelze určit z minimální naměřené disperze, protože ta mohla vzniknout v post-narrow spoji.

V [17] a [23] jsou popsány pokusy provedené pomocí aplikace *network simulator* za účelem prozkoumání hustoty rozdělení kapacity *B*. Dále bude popsány některé důležité výsledky, ke kterým autoři těchto pokusů dospěli.

B je odhadována z histogramu vzniklého z 1000 měření párů paketů. Pomocí statistických metod jsou detekovány *lokální módy* (*local modes*) v *B*. Lokální mód je interval kolem lokálního maxima v *B*. *Síla* (*Strength*) lokálního módu je maximální počet měření v intervalu délky ω tohoto módu. *Globální mód* (*global mode*) *B* je lokální mód s maximální sílou.

Na začátku byl zkoumán **vliv síťové zátěže na *B***. Když je na cestě nízká zátěž (u = 20%), vytvoří se *kapacitní mód* (*Capacity Mode – CM*), který je v tomto případě globálním módem rozdělení. CM je vytvořen páry paketů, které nebyly frontovány za pakety křížujícího provozu. Měření nalevo od CM jsou způsobena páry paketů, kterým překážely pakety křížujícího provozu, čímž se zvětšila disperze párů. Tato oblast nalevo od CM je nazývaná *Sub-Capacity Dispersion Range* (*SCDR*).

Měření napravo od CM jsou zapříčiněna tím, že v post-narrow spojích je první sondovací paket zpožděn více než druhý. Lokální módy v této oblasti jsou nazývány *post-narrow kapacitní módy* (*Post-Narrow Capacity Modes – PNCM*). Spoj s kapacitou C_i může vytvořit PNCM, pokud všechny spoje dále v cestě mají větší kapacitu, tj. pokud $C_i < C_j$ pro každý spoj j, kde $i < j \leq H$.

V případě vysoké zátěže na cestě (u = 80%) pravděpodobnost toho, že křížující pakety budou překážet sondovacím paketům, je mnohem vyšší a CM není globálním módem *B*. Místo toho se globální mód vytvoří v SCDR. V případě silně zahlcených cest se CM vůbec neobjevuje jako lokální mód, protože téměř každý pár paketů je ovlivněn křížujícím provozem.

Dále byl zkoumán **vliv velikosti křižujících paketů na *B***. Když všechny pakety mají stejnou velikost, vytvoří se několik lokálních módů v SCDR. Lokální mód v SCDR se vytvoří, když mnoho křížujících paketů překáží páru paketů na určitém spoji cesty. Tj. takový lokální mód odpovídá tomu, že se víckrát stane, že jeden křížující paket se dostane mezi pár paketů na cestě s určitou kapacitou.

Když velikost křížujících paketů rovnoměrně kolísá v určitém intervalu, SCDR nemá lokální módy. Výraznými módy jsou CM a některé PNCM, protože jsou způsobeny sondovacími pakety, které jsou obslouženy back-to-back na úzkém nebo post-narrow spojích.

Následně byl zkoumán **vliv velikosti *L* sondovacích paketů na *B***. Je zřejmé, že s větším L se zvětšuje pravděpodobnost toho, že mezi pár se dostane křížující provoz, protože čas mezi příchodem prvního a druhého paketu na spoji i je L / C_i, a tedy s větším L se zvětšuje časový interval, po který může přijít křížující paket.

Při použití malého L vzhledem k velikosti křížujících paketů je SCDR mnohem slabší než při použití velkého L. Avšak se zmenšováním L se úměrně zmenšuje disperze a měření jsou

více náchylná na deformace na post-narrow spojích, což se prokázalo tím, že při zmenšování L bylo pravděpodobnější vytváření PNCM.

Pokud L je stejné ve všech párech paketů, vytvoří se lokální módy v SCDR, když křížující pakety určitých velikostí, které jsou běžné v Internetu, se dostanou mezi pár paketů. Kdežto použití proměnné velikosti L sondovacích paketů učiní SCDR širším a slabším v porovnání s CM.

Jako velikost L je tedy vhodné používat řadu hodnot mezi $L_{\min}$ a $L_{\max}$, kde $L_{\min}$ není příliš malé a $L_{\max}$ není příliš velké v porovnání s křížujícími pakety.

Je potřeba vzít v úvahu ještě další omezení $L_{\min}$. Hlavičky přidávané nižšími vrstvami sítě v rámci zapouzdřování paketů (detailněji – viz kapitola 3.3.1) mohou být příčinou příliš nízkých odhadů kapacity v případě použití příliš malých paketů. Proto je potřeba, aby sondovací pakety byly mnohem větší než typické hlavičky vrstvy síťového rozhraní (5 – 50 bytů).

Dále je potřeba počítat s časem zpracování paketů na *RCV*. *RCV* totiž může měřit disperzi páru paketů pouze když je větší než určitá dolní hranice Δ_m. Disperze Δ_m je určená časem potřebným pro příjem paketu ze síťového rozhraní, zpracování paketu v TCP/IP stacku, atd. Maximální kapacita, kterou je možné změřit pro sondovací pakety velikosti L je tedy L / Δ_m. Je-li znám hrubý odhad kapacity $\overline{C}$, pak minimální velikost sondovacích paketů by měla být $L_{\min} > \overline{C}\Delta_m$.

6.5.2 Disperze řetězců paketů

Techniku párů paketů lze zobecnit – *SND* může odesílat N paketů velikosti L do *RCV* stylem back-to-back, kde $N > 2$ – jedná se o techniku řetězců paketů. *RCV* měří celkovou disperzi řetězce $\Delta_R(N) = \sum_{k=1}^{N-1} \Delta^k$, kde Δ^k je disperze mezi pakety k a $k + 1$. *RCV* spočítá naměřený odhad kapacity $b(N)$ jako

$$b(N) = \frac{L}{\overline{\Delta}(N)} \quad (47)$$

kde

$$\overline{\Delta}(N) = \frac{\sum_{k=1}^{N-1} \Delta^k}{N-1} \quad (48)$$

je průměrná disperze mezi následujícími páry paketů v řetězci. Bez křížujícího provozu by naměřený odhad kapacity $b(N)$ byl rovný kapacitě. Nicméně v neprázdných síťových cestách bude $\Delta_R(N)$ kolísat mezi jednotlivými měřeními. V tomto případě $b(N)$ může být považováno za spojitou náhodnou veličinu, která následuje *rozdělení kapacity* $B(N)$.

Dále budou popsány některá důležitá pozorování, ke kterým dospěli autoři pokusů v [17] a [23], když zkoumali řetězce paketů.

První pozorování je, že s rostoucím N se CM a PNCM stávají slabšími a SCDR převládá v $B(N)$. Děje se tak, protože s rostoucím N roste pravděpodobnost, že řetězec paketů narazí na další křížující provoz. Když je délka řetězce dostatečně velká, téměř každé měření dává výsledek, který je menší než kapacita cesty kvůli rušení křížujícím provozem. Tedy optimální délka řetězce, která vytvoří silný kapacitní mód CM je $N = 2$. Delší řetězce s větší pravděpodobností vedou k příliš nízkým odhadům kapacity.

Další pozorování říká, že s rostoucím N klesá rozptyl $b(N)$. Toto chování lze vysvětlit pomocí vztahu (48). Pokud σ_Δ^2 je rozptyl disperze páru paketů Δ a pokud disperze párů paketů Δ^k ($k = 1,\ldots,N-1$) jsou navzájem nezávislé, rozptyl $\overline{\Delta}(N)$ je $\frac{\sigma_\Delta^2}{N-1}$. Kapacita $b(N)$ je určena pomocí $\overline{\Delta}(N)$, a tedy s rostoucím N klesá rozptyl $b(N)$.

Poslední pozorování se týká toho, že pokud N je dostatečně velké, výsledná měření kapacity konvergují k určité hodnotě – k *průměrné míře disperze*.

6.5.3 Průměrná míra disperze(Average (Asymptotic) Dispersion Rate – ADR)

Uvažme cestu $P = \{C_0, C_1, ..., C_H\}$ ze *SND* na *RCV*. Na spoji i je průměrná rychlost křížujícího provozu S_i, kde $S_i < C_i$. Dostupná kapacita je $A_i = C_i - S_i$, využití je $u_i = S_i / C_i$. Předpokládáme, že spoje používají FIFO buffery. *SND* odesílá řetězec paketů délky N na *RCV* rychlostí C_0, velikost sondovacích paketů je L bytů. Každý řetězec je odeslán poté, co předchozí byl přijat, proto nedochází ke konfliktům různých řetězců.

Disperze řetězce $\Delta_i(N)$ za spojem i je časový interval mezi kompletním přenosem prvního a posledního paketu řetězce na spoji i. Počáteční disperze řetězce za zdrojem je $\Delta_0(N) = L(N-1)/C_0$, tj. pakety jsou odeslány „back-to-back“. Řetězec dorazí na *RCV* s disperzí $\Delta(N) = \Delta_H(N)$, z čehož se odvodí naměřená kapacita $b(N) = (N-1)L/\Delta(N)$.

Průměrná míra disperze ADR souvisí se střední hodnotou disperze řetězců paketů E[Δ(*N*)]:

$$ADR = \frac{(N-1)L}{E[\Delta(N)]}. \tag{49}$$

Uvažme cestu s jedním přeskokem $P = \{C_0, C_1\}$, kde $C_0 \geq A_1$. Budeme předpokládat, že množství křížujícího provozu, který bude na spoji *i* v jakémkoliv časovém intervalu *T*, bude $S_i T$. Tedy křížující provoz, který dorazí do spoje během počáteční disperze řetězce $\Delta_0(N)$ je $X_1 = S_1\Delta_0(N)$. Provoz X_1 je prokládán řetězcem paketů ve frontě spoje. Disperze řetězce paketů za spojem je

$$\Delta_1(N) = \frac{(N-1)L + X_1}{C_1} = \frac{(N-1)L}{C_1}(1 + u_1 \frac{C_1}{C_0}), \tag{50}$$

a tedy *ADR* je rovné

$$ADR = \frac{(N-1)L}{E[\Delta_1(N)]} = \frac{C_1}{1 + u_1 \frac{C_1}{C_0}} \leq C_1. \tag{51}$$

ADR tedy nezávisí na délce řetězce *N*.

Ze vztahu (51) lze odvodit, že pro účely vyhodnocování kapacity je lepší když spoj s malou kapacitou následuje za spojem s velkou kapacitou, protože člen reprezentující chybu způsobenou křížujícím provozem (u_1C_1 / C_0) je pak redukován. Pro představu předpokládejme, že C_0 = 100 Mbps a C_1 = 100 Kbps. I kdyby byl úzký spoj téměř nasycen ($u \approx 1$), člen u_1C_1 / C_0 bude znamenat chybu menší než 0,1 %. Proto je složitější vyhodnocovat kapacitu cest, ve kterých je kapacita úzkého spoje stejného řádu jako rychlost odesílání paketů ze zdroje.

Nyní uvažujme cestu s více přeskoky $P = \{C_0, C_1, ..., C_H\}$. Opět budeme předpokládat, že množství křížujícího provozu, který bude na spoji *i* v jakémkoliv časovém intervalu *T*, bude $S_i T$ a dále, že množství křížujícího provozu na spoji *i* nezávisí na množství křížujícího provozu na předcházejících spojích, takový křížující provoz je označován jako *one-hop persistent*. ADR_i bude označovat průměrnou míru disperze na výstupu spoje *i*, $ADR_0 = C_0$.

Pokud $ADR_{i-1} < A_i$, platí $ADR_{i-1} + S_i < C_i$, tedy křížující provoz nezvětšuje disperzi sondovacích paketů. Tedy *ADR* na výstupu spoje *i* je $ADR_i = ADR_{i-1}$.

Pokud $ADR_{i-1} \geq A_i$ lze odvodit podobně jako v případě vztahu (51), že

$$ADR_i = \frac{C_i}{1 + u_i \frac{C_i}{ADR_{i-1}}} = \frac{ADR_{i-1}C_i}{S_i + ADR_{i-1}} = \frac{ADR_{i-1}C_i}{(C_i - A_i) + R_{i-1}}. \tag{52}$$

Dohromady tedy dostaneme:

(53) $ADR_i = ADR_{i-1} \frac{C_i}{S_i + ADR_{i-1}}$ pokud $ADR_{i-1} \geq A_i$

$ADR_i = ADR_{i-1}$ v ostatních případech

Tedy pokud známe kapacitu a dostupnou kapacitu každého spoje cesty, je možné odvodit *ADR* pomocí (53) rekurzivně pro i = 1 až i = *H*, *ADR* je totiž obecně určeno kapacitou a dostupnou kapacitou každého spoje cesty. Jedná se tedy o odlišnou situaci než v případě end-to-end dostupné kapacity $A = \min A_i$, která závisí na využití a kapacitě pouze těsného spoje.

Podle (52) pokud $ADR_{i-1} \geq A_i$, pak $ADR_i \leq C_i$. Tedy podle (53) *ADR* za každým spojem *i* je buďto menší než dostupná kapacita A_i, a tedy je menší než kapacita C_i, nebo je menší než kapacita C_i podle předchozí věty. Pokud úzkým spojem je spoj *n*, platí $ADR_n \leq C_n = C$. Jelikož *ADR* za spojem *i* není nikdy větší než míra disperze za spojem *i* -1, platí $ADR \leq C$.

6.5.4 Implementace odhadu kapacity

Měření kapacity v aplikaci *meas* je založeno na předpokladech podložených teorií z předchozích částí kapitoly 6.5.

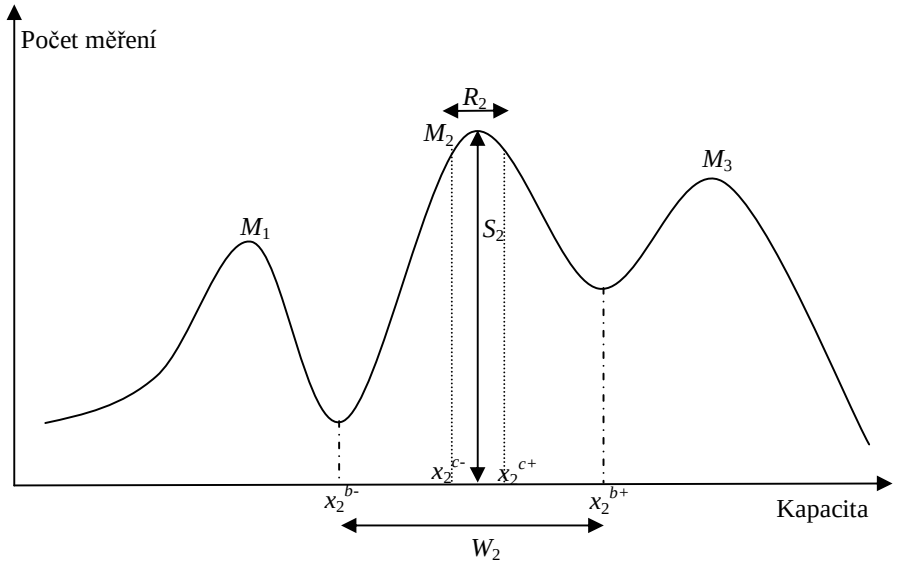
Optimální délka řetězce paketů je $N = 2$, delší řetězce totiž vedou k příliš nízkým odhadům kapacity kvůli interferencím sondovacích paketů s křížujícím provozem. Použití proměnné velikosti *L* sondovacích paketů učiní SCDR širším a slabším v porovnání s CM. Použití spíše větších sondovacích paketů učiní PNCM slabšími v porovnání s CM. Průměrná míra disperze ADR je dolním odhadem kapacity cesty a může být měřena dlouhými řetězci paketů.

Pro přenos sondovacích paketů se používá UDP. Dodatečně je otevřeno TCP spojení jako řídicí kanál. Páry nebo řetězce paketů, ve kterých došlo ke ztrátám v průběhu měřícího procesu, jsou ignorovány. Měřící proces je předčasně ukončen, pokud se vyskytne větší počet za sebou následujících ztrát.

Doba obrátky

Časový interval mezi následujícími páry nebo řetězci paketů je větší než doba obrátky. Toho je dosaženo tím, že na začátku měřícího procesu je změřená doba obrátky. *RCV* pošle zprávu po řídicím kanále, na kterou *SND* okamžitě odpoví. *RCV* změří čas od odeslání zprávy do obdržení odpovědi. Tento postup se opakuje desetkrát, výsledný odhad doby obrátky je pak

aritmetickým průměrem přes naměřené časy. Toto měření doby obrátky v rámci měření kapacity je v *meas* použito i pro samostatný typ měření doby obrátky.



Obr. 4: Vlastnosti lokálního módu

Detekce lokálních módů

Pro detekci lokálních módů je v měřícím procesu použit následující algoritmus:

Vstupem je K měření kapacity. Tato měření se setřídí do rostoucí posloupnosti $x_1 \leq x_2 \leq \ldots \leq x_K$. Důležitým parametrem algoritmu je *šířka okénka* ω.

Algoritmus iterativně určuje posloupnost lokálních módů $\{M_1, M_2, \ldots, M_P\}$, kde každá iterace přidá jeden mód. U každého módu jsou sledovány následující vlastnosti (viz Obr. 4):

- Rozsah centrálního okénka $R_l = [x_l^{c-}, x_l^{c+}]$. Tento rozsah není větší než ω.
- Počet měření v R_l – označíme ho S_l.
- Rozsah módu $W_l = [x_l^{b-}, x_l^{b+}]$.
- Počet měření v W_l – označíme ho B_l.

Hodnoty x_l^{c-}, x_l^{c+}, x_l^{b-} a x_l^{b+} se určí následujícím postupem:

- Na začátku jsou všechna měření neoznačená a $l = 1$.
- Nejdříve se určí rozsah R_l a počet měření S_l v centrálním okénku módu M_l. Centrální okénko je definováno tak, že obsahuje maximální počet následujících, neoznačených

měření v intervalu délky maximálně ω. x_l^{c-}, x_l^{c+} jsou hraniční hodnoty kapacity, které určují R_l.

- Dále je potřeba najít pravou hranici módu M_l , tj. určit x_l^{b+} . Tato část algoritmu je iterativní. Na začátku bude *nejpravější* okénko totožné s centrálním okénkem M_l . V každém kroku se určí okénko napravo od současného nejpravějšího okénka M_l, které se překrývá s nejpravějším okénkem a které obsahuje maximální počet měření v intervalu délky maximálně ω. Pokud toto okénko obsahuje více měření než nejpravější okénko M_l, patří k jinému lokálnímu módu. V opačném případě se stane toto okénko nejpravějším okénkem M_l a pokračuje se dalším krokem.
- Stejným postupem jako v předchozím kroku se určí *nejlevější* okénko módu M_l a x_l^{b-} .
- Všechna měření v módu M_l od x_l^{b-} do x_l^{b+} se označí. Nastaví se $l = l + 1$ a pokračuje se další iterací až do doby, kdy budou všechna měření označená.

Samotný proces měření kapacity se skládá ze tří fází.

Fáze I

Na začátku je zjištěna maximální délka řetězce $N_{\max}$, pro kterou na cestě nedochází ke ztrátám paketů. Následně v etapě III jsou použity řetězce této délky k odhadu ADR.

Poté je generováno několik řetězců paketů s postupně rostoucí délkou. Z těchto přípravných měření se odvodí vhodná šířka okénka ω a je nastavena na 10% kvartilového rozpětí (definici kvartilového rozpětí lze najít v [24]) přípravných měření. Takto je zajištěno, že širší rozdělení vede k většímu ω.

Když je během přípravných měření zjištěno velmi malé kolísání naměřené kapacity, měřící proces skončí ihned po provedení přípravných měření. V tomto případě je výsledná kapacita vypočítaná jako průměrná hodnota přípravných měření po odstranění 10% nejmenších a největších hodnot. K této situaci dojde, pokud na měřené cestě je velmi řídký křížující provoz.

Fáze II

V této fázi je použito mnoho párů paketů k objevení lokálních módů rozdělení kapacity B, jeden z těchto módů bude kapacitní mód CM. Konkrétně je v této fázi provedeno 1000 měření párů paketů s proměnnou velikostí paketů L, která kolísá mezi $L_{\min}$ a $L_{\max}$, kde $L_{\min}$ je voleno s ohledem na omezení zmíněná v závěru kapitoly 6.5.1.

Na konci fáze II jsou nalezeny lokální módy rozdělení kapacity *B*. K tomu je použit algoritmus popsán výše v této kapitole v části **Detekce lokálních módů**. Pro každý mód M_i jsou určeny centrální okénko R_i, počet měření S_i v R_i, rozsah módu W_i a počet měření B_i v W_i. Průměrná kapacita v centrálním okénku R_i je označena jako H_i . Posloupnost všech lokálních módů $M = \{M_1, M_2, \ldots, M_P\}$ je setříděna tak, aby $H_i < H_{i+1}$.

Jeden z těchto lokálních módů bude kapacitním módem CM – označme jej M_k, tj. $H_k \approx C$. Módy M_i pro $i > k$ jsou PNCM, zatímco módy M_i pro $i < k$ jsou SCDR. Dále je potřeba vybrat správný lokální mód M_k, k tomu se použije odhad *ADR* ve fázi III.

Fáze III

V této fázi je pomocí dlouhých řetězců paketů odhadnuto ADR. Konkrétně je provedeno 500 měření řetězců paketů délky $N_{\max}$, s pakety maximální velikosti $L_{\max}$. $N_{\max}$ je typicky několik desítek paketů, proto výsledné rozdělení kapacity $B(N)$ má malý rozptyl a jeden převažující mód. Použitím stejného algoritmu pro detekci lokálních módů jako ve fázi II je nalezeno *ADR* jako globální mód. Jelikož $ADR \leq C$ (viz závěr kapitoly 6.5.3), jsou ignorovány všechny módy M_i z fáze II, pro které $H_i < ADR$, protože se s největší pravděpodobností jedná o SCDR módy.

Pokud existuje více módů z fáze II, které jsou větší než *ADR*, je potřeba vybrat jeden z nich, který je s největší pravděpodobností CM.

Ve fázi II jsou používány sondovací pakety s proměnnou velikostí, což způsobí, že SCDR módy budou širší a slabší. Používané sondovací pakety jsou relativně velké, což by mělo oslabit PNCM. CM by měl být tedy relativně silným a úzkým lokálním módem.

Za účelem vyhodnocení síly a úzkosti módů z fáze II zavedeme *míru kvality* F_i pro každý mód jako

(54) $$F_i = S_i \Psi_i ,$$

kde S_i je počet měření v centrálním okénku módu M_i - tedy jakýsi odhad síly módu M_i, Ψ_i je špičatost módu M_i – tedy určuje, jak úzký je tento mód.

Finální odhad kapacity $\hat{C}$ je tedy mód M_k a fáze II, který je větší než *ADR* a který má maximální míru kvality *F*.

6.6 Měření propustnosti TCP

Měření propustnosti TCP v *meas* je implementováno jednoduše. *RCV* vytvoří TCP soket a čeká na spojení. *SND* se připojí a začne posílat data. Data jsou posílána bez přestávek za sebou pomocí volání funkce *write* na vytvořeném TCP soketu. Každým voláním je odesláno 8 KB dat (tj. používají se buffery o velikosti 8 KB), funkce je volána 2048 krát za sebou. Výsledná propustnost TCP je vypočtena jako podíl velikosti odeslaných dat a času spotřebovaného na všechna volání funkce *write*.

6.7 Měření ztráty paketů

Měření ztráty paketů v *meas* je implementováno pomocí posílání ICMP zpráv Echo. Tohoto měření se neúčastní *RCV*, využívá se toho, že ICMP zprávy se automaticky vracejí na *SND*. Když vzdálená stanice přijme zprávu ICMP Echo Request, odpoví *SND* zprávou ICMP Echo Reply.

Konkrétně *SND* odešle 1000 zpráv ICMP Echo Request v poměrně krátkých intervalech (10 ms) a následně počítá, kolik dorazí zpráv ICMP Echo Reply, z čehož vypočítá procentuální ztrátu paketů.

K vytváření a posílání ICMP zpráv je použit „raw“ soket, proto pro spuštění měření ztráty paketů je potřeba mít administrátorská práva (konkrétně by mělo stačit SUID uživatele root).

6.8 Tvar výstupů naměřených hodnot

Kromě vypisování výsledků měření na konzoli v případě jednorázových měření *meas* zapisuje naměřené hodnoty do souborů. Jsou vytvářeny soubory dvojího typu:

- XML soubor, který je možné zobrazit jako tabulku v OpenOffice Spreadsheet nebo v MS Excel
- textový soubor přehledně zobrazující naměřené hodnoty

6.8.1 Výstupní XML soubor

V aktuálním adresáři je po každém spuštění hlavního programu vytvořen jeden XML soubor, který je pojmenován následovně: `[typ_měření][časová_známka].xml`, kde:

- `typ_měření` je `bandwidth_chirp, bandwidth, capacity,`

`throughput, rtt` nebo `loss`

- `časová_známka` je identifikace času, kdy byl *meas* spuštěn ve tvaru: RRRRMMDDHHMMSS, tj. například pro 16:10:17 dne 15.7.2007 bude ve tvaru: `20070715161017`

Tento XML soubor má takový obsah, že je možné ho bez jakýchkoliv úprav otevřít v programu OpenOffice Spreadsheet nebo Microsoft Excel. Zde se soubor otevře jako tabulka se třemi nebo čtyřmi sloupci (podle toho, zda příslušné měření vrátilo jednu hodnotu nebo meze intervalu hodnot). V prvním sloupci je adresa stanice s agentem, který prováděl měření, v druhém je přesný čas konce měření, v třetím (resp. třetím a čtvrtém) je naměřená hodnota (resp. dolní a horní mez intervalu naměřených hodnot). Hodnoty dostupné kapacity, kapacity a propustnosti TCP jsou uvedeny v Mbps, doba obrátky je v ms, ztráta paketů v procentech. Každý řádek odpovídá jednomu měření – tj. tabulka je zajímavá hlavně v případě periodických a nepřetržitých měření. Pomocí nástrojů OpenOffice Spreadsheet nebo Microsoft Excel je možné jednoduše vygenerovat graf zobrazující vývoj naměřených hodnot v čase.

Tento soubor je validní XML soubor, tj. je možné ho podrobit jakékoliv XSL transformaci a vygenerovat z něj jakákoliv potřebná data (např. HTML stránku…).

6.8.2 Výstupní textový soubor

V aktuálním adresáři je umístěn pro každý typ měření maximálně jeden textový soubor, který je pojmenován `bandwidth_chirp.out, bandwidth.out, capacity.out, throughput.out, rtt.out` nebo `loss.out.`

Na rozdíl od XML souboru, tento soubor nevzniká při každém spuštění *meas*. Pokud už příslušný textový soubor existuje, je výsledek aktuálního měření připsán na konec. Je zde takto možné sledovat výsledky měření z různých spuštění hlavního programu a také měření síťových cest od různých agentů.

Tento textový soubor obsahuje tři (resp. čtyři) sloupce. V prvním je čas konce měření, v druhém adresa stanice s agentem, v třetím (resp. v třetím a čtvrtém) je naměřená hodnota (resp. dolní a horní mez intervalu naměřených hodnot). Hodnoty dostupné kapacity, kapacity a propustnosti TCP jsou uvedeny v Mbps, doba obrátky je v ms, ztráta paketů v procentech.

Kapitola 7 – *meas* – struktura zdrojových kódů

meas se skládá ze dvou programů – z hlavního programu a z *meas_agent*. Oba tyto programy byly vyvinuty v C++ pro operační systém Linux. K jejich vývoji jsem použil vývojová prostředí Netbeans (www.netbeans.org) a Eclipse (www.eclipse.org).

Nad rámec základního zadání práce jsem dále vytvořil verzi *meas_agent* pro operační systémy Windows. Tato verze však postrádá implementaci částí pro měření dostupné kapacity. *meas_agent* pro Windows byl vyvinut v jazyce C# nad platformou .NET. *meas_agent* totiž přistupuje k sítí na úrovni soketů, nebylo tedy potřeba použit nativní kód. Pro vývoj v jazyce C# jsem se rozhodl především s ohledem na jednoduchost jeho používání. K vývoji jsem použil nástroj Visual Studio.

Verze *meas_agent* pro Linux i Windows obsahují totožně pojmenované hlavní moduly i funkce, proto dále nebudou rozlišovány.

7.1 Hlavní program meas

Hlavní program *meas* se skládá z následujících modulů:

- *main*
- *bandwidth_chirp*
- *bandwidth*
- *capacity*
- *throughput*
- *loss*

7.1.1 *main*

Tento modul obsahuje funkci `int main(int argc, char * argv[])`, která je hlavní funkcí celého programu. Tato funkce nejdříve parsuje příkazovou řádku nebo řeší komunikaci s uživatelem pomocí konzole. Následně je agentovi pomocí funkce `int chooseMeasurement(char choice, char * hostname)` oznámeno, jaké měření má spustit a je předáno řízení modulu pro příslušné měření. Pokud se jedná o periodické či nepřetržité měření, je toto prováděno v cyklu.

Před začátkem cyklu měření je zde také vytvořen začátek výstupního xml souboru.

7.1.2 *bandwidth_chirp*

Modul *bandwidth_chirp* slouží k měření dostupné kapacity metodou CPT – jedná se o *RCV* v tomto měření. Měření se zahájí po spuštění funkce `int runChirpRcv(char * h, int tcp, int print, char * xml_name).` Na začátku je vytvořen synovský proces, který bude sbírat výsledky a řídit běh tohoto měření – v tomto synovském procesu je spuštěna funkce `int runChirp(char * h, int debug, char * xml_name).` Otcovský proces pokračuje zavoláním funkce `int create_listen_socket(int tcp)`, která inicializuje řídicí TCP soket (pokud už nebyl inicializován dříve v tomto běhu hlavního programu). Dále je zavolána funkce `int initiate_connection()`, která inicializuje UDP soket. Nakonec je spuštěna funkce `int receive_chirp_pkts(),` která začne přijímat chirpy a počítat odhady dostupné kapacity tak, jak je to popsáno v kapitole 6.3. Po příjmu paketu patřícího do nového chirpu je spuštěn časovač. Když časovač pošle signál, je pomocí funkce `void chirp_sig_alrm(int signo)` vypočítán odhad dostupné kapacity. Pro výpočet dostupné kapacity chirpu se používá algoritmus uvedený v kapitole 6.3.5 a implementovaný ve funkci `double compute_inst_bw_excursion().` Nakonec je funkcí `void write_instant_bw (double inst_av_bw_excursion, double time_stamp, int nc)` poslán výsledný odhad dostupné kapacity synovskému procesu.

7.1.3 *bandwidth*

Modul *bandwidth* slouží k měření dostupné kapacity metodou SLoPS – jedná se o *RCV* v tomto měření. Měření se zahájí po spuštění funkce `int runBandwidthRcv(char * h, int udp, int print, char * xml_name)`. Na začátku jsou inicializovány UDP soket pro příjem sondovacích paketů (pokud už nebyl inicializován dříve v tomto běhu hlavního programu – v tomto případě je předán v parametru `udp`) a řídicí TCP soket, který se následně pomocí volání *connect* spojí s agentem. Následně je s agentem dohodnuta maximální velikost paketu. Nakonec je spuštěn algoritmus pro přizpůsobení rychlosti tak, jak je popsán v kapitole 6.4.2.

Funkce `int send_ctr_mesg(int sock_tcp, char *ctr_buff, int ctr_code)` a `int recv_ctr_mesg(int ctr_strm, char *ctr_buff)`

slouží k posílání a příjmu zpráv po řídicím TCP kanálu. Existuje 10 typů zpráv, které jsou po tomto kanálu posílány. Jsou to následující zprávy: SEND_FLEET, RECV_FLEET, CONTINUE_STREAM, FINISHED_STREAM, TERMINATE, ABORT_FLEET, SEND_TRAIN, FINISHED_TRAIN, BAD_TRAIN a GOOD_TRAIN. Tyto zprávy slouží ke komunikaci hlavního programu s agentem – k určení, jaká operace bude následovat a k oznamování výsledků operací.

Algoritmus pro přizpůsobení rychlosti je na začátku spuštěn s rychlostí *R*, která je odvozena od průměrné míry disperze (ADR) – viz kapitola 6.5.3. ADR je vypočteno funkcí `double get_adr(int sock_tcp),` která používá funkci `int recv_train(int sock_tcp, int exp_train_id, struct timeval *time, int train_len)` k příjmu řetězců paketů. Během provádění algoritmu pro přizpůsobení rychlosti je volána funkce `int recv_fleet(int sock_tcp)`, která slouží k příjmu *fleet*, což je soubor N proudů.

Ve funkcích `int recv_train(int sock_tcp, int exp_train_id, struct timeval *time, int train_len)` a `int recv_fleet(int sock_tcp)` je vytvořeno nové vlákno, které čeká na zprávu o ukončení řetězce (resp. fleet) na řídicím kanálu. Po obdržení této zprávy pošle toto vlákno signál hlavnímu vláknu, které je v tomto čase zablokováno čekáním na data z UDP soketu. Poté je nové vlákno i funkce ukončena.

7.1.4 *capacity*

Modul *capacity* slouží k měření kapacity - jedná se o *RCV* v tomto měření. Měření se zahájí po spuštění funkce `int runCapacityRcv(char * h, bool is_rtt, int udp, int print, char * xml_name)` s parametrem `is_rtt` nastaveným na *false*. Pokud je tato funkce spuštěna s parametrem `is_rtt` nastaveným na *true* (tj. je požadováno pouze měření doby obrátky), měření je ukončeno po odhadnutí doby obrátky. Na začátku jsou inicializovány UDP soket pro příjem sondovacích paketů (pokud už nebyl inicializován dříve v tomto běhu hlavního programu – v tomto případě je předán v parametru `udp`) a řídicí TCP soket, který se následně pomocí volání *connect* spojí s agentem. Na začátku je vypočtena doba obrátky postupem popsaným v kapitole 6.5.4. Na základě doby obrátky je vypočítán minimální rozestup paketů v řetězci. Dále jsou spuštěny 3 fáze algoritmu měření kapacity tak, jak jsou popsány v kapitole 6.5.4.

Funkce `int send_ctr_mesg (long ctr_code)` a `long recv_ctr_mesg(int ctr_strm, char *ctr_buff)` slouží k posílání a příjmu zpráv po řídicím TCP kanálu. Existuje 11 typů zpráv, které jsou po tomto kanálu posílány. Jsou to následující zprávy: TRAIN_LEN, NO_TRAINS, GAME_OVER, ACK_TRAIN, PCK_LEN, MAX_PCK_LEN, SEND, CONTINUE, TRAIN_SPACING, SENT_TRAIN, NEG_ACK_TRAIN. Tyto zprávy slouží ke komunikaci hlavního programu s agentem – k určení, jaká operace bude následovat a k oznamování výsledků operací.

K detekci lokálních módů slouží funkce `double get_mode(double ord_data[], short vld_data[], double bin_wd, int no_values, mode_struct *curr_mode)`.

Funkce `int recv_train(int train_len, int * round, struct timeval *time[])` slouží k příjmu řetězce paketů. V této funkci je vytvořeno nové vlákno, které čeká na zprávu o ukončení řetězce na řídicím kanálu. Po obdržení této zprávy pošle toto vlákno signál hlavnímu vláknu, které je v tomto čase zablokováno čekáním na data z UDP soketu. Poté je nové vlákno i funkce ukončena.

7.1.5 *throughput*

Modul *throughput* slouží k měření propustnosti TCP - jedná se o *RCV* v tomto měření. Měření se zahájí po spuštění funkce `int runThroughput(char * h, int tcp, int print, char * xml_name)`. Na začátku je inicializován TCP soket (pokud už nebyl inicializován dříve v tomto běhu hlavního programu – v tomto případě je předán v parametru `tcp`). Následně tento soket přijímá sondovací data a nakonec vypočítá výslednou propustnost TCP tak, jak je naznačeno v kapitole 6.6.

7.1.6 *loss*

Modul *loss* slouží k měření ztráty paketů. Měření se zahájí po spuštění funkce `int runLoss(char * h, int print, char * xml_name)`. Na začátku je inicializován RAW soket pro přenos ICMP zpráv. Následně je po tomto soketu posíláno 1000 ICMP Echo Request paketů s rozestupem 10 ms. Navíc po každém odeslaném paketu je zkontrolováno, zda nedorazila ICMP Echo Reply odpověď na některou z dříve odeslaných žádostí. Nakonec je vypočítána procentuální ztráta paketů.

7.2 meas_agent

meas_agent se skládá z následujících modulů:

- *main_agent*
- *bandwidth_chirp_agent*
- *bandwidth_agent*
- *capacity_agent*
- *throughput_agent*

7.2.1 main_agent

Tento modul obsahuje funkci `int main(int argc, char * argv[])`, která je hlavní funkcí celého agenta. Tato funkce obsahuje nekonečnou smyčku, ve které je nejdříve zavolaná funkce `char chooseMeasurement(int sock_tcp, int print)`, která vrátí identifikaci typu měření, které bylo zvoleno v hlavním programu. Následně je předáno řízení modulu pro příslušné měření.

Funkce `char chooseMeasurement(int sock_tcp, int print)` se zablokuje do doby než po řídicím TCP kanále dorazí od hlavního programu požadavek na určité měření. Tento požadavek je přijat a je vrácena jeho identifikace.

7.2.2 bandwidth_chirp_agent

Modul *bandwidth_chirp_agent* slouží k měření dostupné kapacity metodou CPT - jedná se o SND v tomto měření. Měření se zahájí po spuštění funkce `int runChirpSnd(int sock, int print)`. Na začátku je inicializován UDP soket pro odesílání sondovacích paketů (pokud už nebyl inicializován dříve v tomto běhu agenta – v tomto případě je předán v parametru `sock`). Agent čeká až hlavní program pošle po tomto soketu paket a následně začne posílat chirpy pomocí funkce `int chirps_snd()`.

7.2.3 bandwidth_agent

Modul *bandwidth_agent* slouží k měření dostupné kapacity metodou SLoPS - jedná se o *SND* v tomto měření. Měření se zahájí po spuštění funkce `int runBandwidthSnd(int sock, int print)`. Na začátku jsou inicializovány UDP soket pro odesílání sondovacích paketů a řídicí TCP soket (pokud už nebyl inicializován dříve

v tomto běhu agenta – v tomto případě je předán v parametru `sock`), který se spojí s hlavním programem. Následně je s agentem dohodnuta maximální velikost paketu. Nakonec je spuštěn algoritmus pro přizpůsobení rychlosti tak, jak je popsán v kapitole 6.4.2.

Funkce `int send_ctr_mesg(char *ctr_buff, int ctr_code)` a `int recv_ctr_mesg(char *ctr_buff)` slouží k posílání a příjmu zpráv po řídicím TCP kanálu. Existuje 10 typů zpráv, které jsou po tomto kanálu posílány. Jsou to následující zprávy: SEND_FLEET, RECV_FLEET, CONTINUE_STREAM, FINISHED_STREAM, TERMINATE, ABORT_FLEET, SEND_TRAIN, FINISHED_TRAIN, BAD_TRAIN a GOOD_TRAIN. Tyto zprávy slouží ke komunikaci hlavního programu s agentem – k určení, jaká operace bude následovat a k oznamování výsledků operací.

K odeslání řetězce paketů a fleet slouží funkce `int send_train()` a `int send_fleet()`.

7.2.4 *capacity_agent*

Modul *capacity_agent* slouží k měření kapacity - jedná se o *SND* v tomto měření. Měření se zahájí po spuštění funkce `int runCapacitySnd(bool is_rtt, int sock, int print)` s parametrem `is_rtt` nastaveným na *false*. Na začátku jsou inicializovány UDP soket pro odesílání sondovacích paketů a řídicí TCP soket (pokud už nebyl inicializován dříve v tomto běhu agenta – v tomto případě je předán v parametru `sock`), který se spojí s hlavním programem. Na začátku je vypočtena doba obrátky postupem popsaným v kapitole 6.5.4. Pokud je tato funkce spuštěna s parametrem `is_rtt` nastaveným na *true* (tj. je požadováno pouze měření doby obrátky), měření je ukončeno po odhadnutí doby obrátky. Dále je spuštěn fáze algoritmus pro měření kapacity tak, jak je popsány v kapitole 6.5.4. Agent v cyklu přijímá zprávy posílané hlavním programem a spouští požadované operace.

Funkce `int send_ctr_mesg(int ctr_strm, long ctr_code)` a `long recv_ctr_mesg(int ctr_strm, char *ctr_buff)` slouží k posílání a příjmu zpráv po řídicím TCP kanálu. Existuje 11 typů zpráv, které jsou po tomto kanálu posílány. Jsou to následující zprávy: TRAIN_LEN, NO_TRAINS, GAME_OVER, ACK_TRAIN, PCK_LEN, MAX_PCK_LEN, SEND, CONTINUE, TRAIN_SPACING, SENT_TRAIN, NEG_ACK_TRAIN. Tyto zprávy slouží ke komunikaci hlavního programu s agentem – k určení, jaká operace bude následovat a k oznamování výsledků operací.

7.2.5 *throughput_agent*

Modul *throughput_agent* slouží k měření propustnosti TCP - jedná se o *SND* v tomto měření. Měření se zahájí po spuštění funkce `int runThroughput(int sock, int print, char * xml_name)`. Na začátku je inicializován TCP soket (pokud už nebyl inicializován dříve v tomto běhu agenta – v tomto případě je předán v parametru `sock`). Následně tento soket posílá sondovací data.

Kapitola 8 – Zhodnocení

8.1 Srovnání s jinými projekty řešícími měření datových přenosů

Známým nástrojem pro měření výkonu datových přenosů v počítačových sítích je *Ixia Qcheck* (viz [30]). *Qcheck* se obdobně jako *meas* zaměřuje pouze na měření end-to-end. Kromě protokolů TCP/IP však *Qcheck* zkoumá i protokoly IPX/SPX. *Qcheck* umožňuje čtyři druhy end-to-end testů. Jedná se o *response time test* – měření latence, *throughput test* – měření propustnosti, *streaming test* – měření síťové cesty s použitím toků imitujících multimediální aplikace a *traceroute test* – zjistí počet přeskoků, průměrnou latenci přeskoků a adresy strojů na každém přeskoku. Na rozdíl od *meas* však *Qcheck* nedokáže měřit kapacitu síťové cesty. Hlavní program *Qcheck console* je pouze pro operační systém Windows, *Ixia Performance Endpoint*, což je obdoba programu *meas_agent* má verze pro více platforem.

Dalším zajímavým nástrojem je *NetDoppler* (viz [31]). Tento program umí měřit latenci, propustnost a trasování (traceroute). Tento program se liší od ostatních tím, že nepotřebuje agenta spuštěného na vzdálené stanici – všechna měření jsou totiž prováděná pomocí ICMP paketů. Konkrétně měření propustnosti (throughput) je implementováno tak, že je poslán ICMP paket větší než MTU v příslušné síťové cestě, čímž je zajištěno to, že se tento paket rozpadne na více malých paketů, nad kterými pak *NetDoppler* může implementovat algoritmy pro měření pomocí řetězců paketů. Nevýhodou přístupu pomocí ICMP je to, že některé uzly v síti filtrují ICMP pakety. *NetDoppler* je vyvíjen pouze pro operační systém Windows.

PC Pitstop Internet Connection Center (viz [32]) nabízí možnost měření rychlosti downloadu a uploadu přímo z internetového prohlížeče

Dále existuje množství programů určených k měření jednotlivých veličin. Tyto programy jsou většinou akademického původu a jsou určeny pro operační systémy UNIX. Některé jsou již zastaralé a nedají se bez úprav ve zdrojových kódech zprovoznit na moderních operačních systémech, nicméně stojí za zmínku, neboť hrály významnou roli v historii síťových měření.

K měření end-to-end kapacity slouží nástroje *bprobe* [33], *nettimer* [34], *pathrate* [35] a *sprobe* [36]. Všechny tyto nástroje používají k měření kapacity techniku párů či řetězců paketů.

End-to-end dostupnou kapacitu měří nástroje *cprobe* [33], *pathload* [37], *IGI/PTR* [38] a *pathChirp* [39]. *cprobe* používá techniku řetězců paketů, *pathload* a *IGI/PTR* používají SLoPS, *pathChirp* používá CPT.

Propustnost TCP měří nástroje *ttcp* [40], *Iperf* [41] a *Netperf* [42].

NÁSTROJ	**TVŮRCE**	**MĚŘENÉ VELIČINY**
Qcheck	Ixia	latence, propustnost, …
NetDoppler	WildPackets	propustnost, latence, …
Internet Connection Center	PC Pitstop	propustnost downloadu, uploadu, …
bprobe	Carter	kapacita
nettimer	Lai	kapacita
pathrate	Dovrolis-Prasad	kapacita
sprobe	Saroiu	kapacita
cprobe	Carter	dostupná kapacita
pathload	Jain-Dovrolis	dostupná kapacita
IGI/PTR	Hu-Steenkiste	dostupná kapacita
pathChirp	Ribeiro	dostupná kapacita
ttcp	Muuss	propustnost TCP
Iperf	NLANR	propustnost TCP
Netperf	NLANR	propustnost TCP
meas	Matteoni	dostupná kapacita, kapacita, propustnost TCP, doba obrátky, ztráta paketů

Tabulka 1: Srovnání známých nástrojů pro měření datových nástrojů

8.2 Testování

V rámci práce jsem provedl sérii testů všech měření implementovaných v *meas*. Tato měření jsem prováděl na stanici *labts.troja.mff.cuni.cz* umístěné v budově Matematicko-fyzikální fakulty Univerzity Karlovy v Praze Troji, dále na stanicích *u-pl1.ms.mff.cuni.cz* a *u1-1.ms.mff.cuni.cz* umístěných v budově Matematicko-fyzikální fakulty Univerzity Karlovy v Praze na Malé Straně a na mém domácím počítači v Praze Vysočanech. Fakultní počítače jsou připojeny k Internetu spoji s kapacitou 100 Mbps, můj domácí počítač je připojen spojem s kapacitou downloadu 256 Kbps, uploadu 64 Kbps.

Dále uváděná měření byla vždy provedena 4krát po sobě – zde uvedené výsledky jsou průměry těchto 4 měření.

8.2.1 Testování měření dostupné kapacity

Měření s agentem na *labts.troja.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* CPT – 37.36 Mbps
- *meas* SLoPS – 17.54 Mbps
- *pathchirp* – 38.58 Mbps

Měření s agentem na *u-pl1.ms.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* CPT – 93.08 Mbps
- *meas* SLoPS – 95.55 Mbps
- *pathchirp* – 91.16 Mbps

Síťová cesta z *u-pl1.ms.mff.cuni.cz* na *u1-1.ms.mff.cuni.cz* obsahovala zřejmě mnohem méně křížujícího provozu, proto jsou výsledky získané všemi způsoby měření podobné a přesné. Odchylky v řádu Mbps mohly být způsobeny aktuální intenzitou křížujícího provozu.

V případě cesty z *labts.troja.mff.cuni.cz* na *u1-1.ms.mff.cuni.cz* se jedná o cestu s mnohem větší intenzitou křížujícího provozu. Měření pomocí metody CPT programem *meas* a *pathchirp* vydala podobné výsledky, avšak měření metodou SLoPS odhadlo dostupnou kapacitu daleko nižší. V [25] bylo pozorováno, že metoda SLoPS dává mnohem nižší odhady dostupné kapacity pokud se na zkoumané síťové cestě vyskytuje křížující provoz s konstantní rychlostí (*Constant Bit Rate – CBR).*

8.2.2 Testování měření kapacity

Měření s agentem na *labts.troja.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* – 95.96 Mbps
- *pathrate* – 95.50 Mbps

Měření s agentem na *u-pl1.ms.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* – 97.58 Mbps
- *pathrate* – 98.00 Mbps

Zřejmě kapacita obou uvažovaných síťových cest je 100 Mbps. V druhém případě, kdy se jedná o cestu s menší intenzitou křížujícího provozu, jsou výsledky měření přesnější.

8.2.3 Testování měření propustnosti TCP

Měření s agentem na *labts.troja.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* – 58.61 Mbps
- *Iperf* – 62.75 Mbps

Měření s agentem na *u-pl1.ms.mff.cuni.cz* a s hlavním programem na *u1-1.ms.mff.cuni.cz* dopadla následovně (měření byla prováděna přibližně ve stejnou dobu):

- *meas* – 89.51 Mbps
- *Iperf* – 95.80 Mbps

Kapitola 9 – Shrnutí dosažených výsledků a doporučení dalšího směru práce

Cílem práce bylo seznámit se s možnostmi měření datových přenosů v počítačových sítích, identifikovat veličiny vhodné k měření a vyvinout systém pro měření kvality a rychlosti datových přenosů se zaměřením na protokoly TCP/IP a výstupem naměřených hodnot vhodným pro prezentaci i další zpracování.

Po důkladné analýze jsem vybral 5 nejdůležitějších aktivních veličin a implementoval jejich měření pomocí metod, které byly již dříve důkladně prozkoumány. Výstupem těchto měření je buďto strukturovaný textový soubor nebo XML soubor, který je možné otevřít ve známých tabulkových procesorech a dále ho zpracovávat.

Výsledná aplikace *meas* je určená primárně pro operační systémy Linux, kde v oblasti nástrojů pro měření síťového provozu jedinečná širokou škálou nabízených měření. Nad rámec základního zadání práce jsem vytvořil verzi *meas_agent* pro operační systémy Windows. Verze pro Windows však postrádá implementaci částí pro měření dostupné kapacity.

Za účelem dosažení jednoduchosti a jednotného ovládání všech typů měření není možné měnit rozličné parametry měřících metod. Jeden možný směr dalšího vývoje bych tedy viděl ve zlepšení uživatelského rozhraní aplikace. Uživatelské rozhraní by mohlo mít dva módy:

- jednodušší – ve kterém by bylo zachováno spouštění měření s výchozími hodnotami parametrů
- složitější – ve kterém by zkušenější uživatel měl možnost nastavit parametry měřících metod

Dalším možným rozšířením aplikace *meas* by mohla být možnost určit, ve kterém směru budou posílány sondovací pakety – tj. určit, zda agent bude *SND* nebo *RCV*.

V případě další práce na aplikaci *meas* by bylo zajímavé dokončit verzi *meas_agent* pro Windows, tj. přidat tam možnost měření dostupné kapacity alespoň jednou z metod implementovaných v hlavním programu *meas*.

Nakonec by bylo vhodné navrhnout rozsáhlejší testovací scénáře než ty, které jsem použil, a otestovat, jak se různá měření chovají na různých síťových cestách.

Příloha - Ovládání aplikace *meas*

meas_agent – Linux

Program je potřeba rozpakovat a přeložit. Tj. je potřeba spustit následující příkazy:

```
$gunzip ma.tar.gz
$tar xvf ma.tar
$cd meas_agent
$make
```

Spustitelný soubor *meas_agent* bude vytvořen v podadresáři *dist*, tj. program spustíme zadáním příkazů:

```
$cd dist
$./meas_agent
```

meas_agent – Windows

Program není potřeba instalovat. Stačí spustit soubor *meas_agent.exe*. Při použití *meas_agent* pro Windows není možné spustit měření dostupné kapacity, tedy měření označená čísly 0 a 1 v hlavním programu *meas*.

meas – Linux

Program je potřeba rozpakovat a přeložit. Tj. je potřeba spustit následující příkazy:

```
$gunzip m.tar.gz
$tar xvf m.tar
$cd meas
$make
```

Spustitelný soubor *meas* bude vytvořen v podadresáři *dist*, tj. program spustíme zadáním příkazů:

```
$cd dist
$./meas [-k <p|c>] [-m <0|1|2|3|4|5>] [-t <minutes_count>]
[-h <hostname>]
```

Význam parametrů je následující:

- `-k p` – spustí periodické měření; `-k c` – spustí nepřetržité měření (viz kapitola 6.1)

- `-m` – výběr typu měření:
  - 0 – dostupná kapacita metodou CPT
  - 1 – dostupná kapacita metodou SLoPS
  - 2 – kapacita
  - 3 – propustnost TCP
  - 4 – doba obrátky
  - 5 – ztráta paketů
- `-t` – v případě periodického měření se jedná o délku periody v minutách, v případě nepřetržitého měření se jedná o celkový čas měření
- `-h` – adresa stanice, na které je spuštěn *meas_agent*

Po spuštění hlavního programu *meas* bez zadání parametrů se vypíše: „Choose a type of a measurement:“ a je potřeba vybrat požadovaný typ měření. Výběr se provede zadáním číslice z intervalu 0 – 5 a potvrzením klávesou „Enter“. Význam jednotlivých voleb je následující:

- 0 – měření dostupné kapacity metodou CPT
- 1 – měření dostupné kapacity metodou SLoPS
- 2 – měření kapacity
- 3 – měření propustnosti TCP
- 4 – měření doby obrátky
- 5 – měření ztráty paketů

Následně se objeví výzva: „Type a hostname where meas_agent runs:“ - je potřeba zadat adresu stanice, na které je spuštěn *meas_agent*. Poté se spustí jednorázové měření. Délka měření závisí na zvoleném typu měření a také na charakteristikách síťové cesty. Obecně platí, že nejdéle trvá měření kapacity – toto měření může trvat až desítky minut.

Tvar výstupů měření je popsán v kapitole 6.8.

Seznam použité literatury

[1] Comer D. E. (1991): *Internetworking with TCP/IP Volume I*. Prentice-Hall.

[2] Comer D. E., Stevens D. L. (1991): *Internetworking with TCP/IP Volume II*. Prentice-Hall.

[3] Cottrell L. (1999): *Comparison of some Internet Active End-to-end Performance Measurement projects.* SLAC.

[4] Claffy K. (1999): *Internet measurement and data analysis: topology, workload, performance and routing statistics.* NAE ’99 workshop, CAIDA.

[5] Kalidindi S., Zekauskas M. J. (1999): *Surveyor: An Infrastructure for Internet Performance Measurements.* INET ’99.

[6] Cardwell N., Savage S., Anderson T. (2000): *Modeling TCP Latency.* INFOCOM 2000, 1742-1751.

[7] Padhye J., Floyd S. (2001): *On Inferring TCP Behavior.* ACM SIGCOMM 2001, 287-298.

[8] Padhye J., Firoiu V., Towsley D., Krusoe J. (1998): *Modeling TCP Throughput: A Simple Model and its Empirical Validation.* ACM SIGCOMM ’98, 303-314.

[9] Horneffer M. (2000): *Assessing Internet Performance Metrics Using Large-Scale TCP-syn Based Measurements.* Passive & Active Measurement Workshop 2000.

[10] Jacobson V. (1997): *pathchar – a tool to infer characteristics of Internet paths.* MSRI 1997.

[11] Downey A. B. (1999): *Using pathchar to estimate Internet link characteristics.* ACM SIGCOMM ’99, 241-250.

[12] Mathis M., Allman M. (2001): *A Framework for Defining Empirical Bulk Transfer Capacity Metrics.* Network Working Group, RFC 3148.

[13] Allman M., Paxson V., Stevens W. (1999): *TCP Congestion Control.* Network Working Group, RFC 2581.

[14] Bellovin S. M. (1992): *A Best-Case Network Performance Model.* ATT Research.

[15] Jacobson V. (1988): *Congestion Avoidance and Control.* ACM SIGCOMM ’88, 314-329.

[16] Bolot J. C. (1993): *Characterizing End-to-End Packet Delay and Loss in the Internet.* ACM SIGCOMM ’93, 289-298.

[17] Dovrolis C., Ramanathan P., Moore D. (2001): *What do packet dispersion techniques measure?* INFOCOM 2001, 905-914.

[18] Pásztor A., Veitch D. (2002): *The Packet Size Dependence of Packet Pair Like Methods.* IEEE/IFIP International Workshop on Quality in Service 2002.
[19] Jain M., Dovrolis C. (2002): *End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput.* University of Delaware.
[20] Melander B., Bjorkman M., Gunningberg P. (2000): *A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks.* IEEE Global Internet Symposium 2000.
[21] Melander B., Bjorkman M., Gunningberg P. (2002): *Regression-Based Available Badwidth Measurements.* International Symposium on Performance Evaluation of Computer and Telecommunications 2002.
[22] Carter R. L., Crovella M. E. (1996): *Measuring Bottleneck Link Speed in Packet-Switched Networks.* Performance Evaluation, Vol. 27-8, 297-318.
[23] Dovrolis C., Ramanathan P., Moore D. (2001): *Packet dispersion techniques and a capacity estimation methodology.* Transactions on Networking.
[24] Zvára K., Štěpán J. (2002): *Pravděpodobnost a matematická statistika.* MATFYZPRESS.
[25] Castellanos C. Ú., Villa D. L., Teyeb O. M., Elling J., Wigard J. (2006): *Comparison of available bandwidth estimation techniques in packet-switched mobile networks.* IEEE Internation Symposium on Personal, Indoor and Mobile Radio Communications 2006.
[26] Pásztor A. (2003): *Accurate Active Measurement in the Internet and its Applications.* University of Melbourne.
[27] Prasad R. S., Murray M., Dovrolis C., Claffy K. (2003): *Bandwidth estimation: metrics, measurement techniques, and tools.* IEEE Network 2003.
[28] *The Ping Page* - http://www.ping127001.com/pingpage.htm
[29] Paxson V. (1997): *Measurements and Analysis of End-to-End Internet Dynamics.* University of California, Berkeley.
[30] *Ixia Qcheck* - http://www.ixiacom.com/products/performance_applications/pa_display.php?skey=qcheck
[31] *NetDoppler* - http://www.wildpackets.com/products/free_utilities/netdoppler/overview
[32] *PC Pitstop Internet Connection Center* - http://www.pcpitstop.com/internet/default.asp
[33] *BPROBE and CPROBE* - http://cs-people.bu.edu/carter/tools/Tools.html
[34] *Nettimer: A Tool for Measuring Bottleneck Link Bandwidth* - http://www.hpl.hp.com/personal/Kevin_Lai/projects/nettimer/publications/usits2001/main.html
[35] *Pathrate* - http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathrate.html

[36] *SProbe* - http://sprobe.cs.washington.edu/

[37] *Pathload* - http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/pathload.html

[38] *IGI/PTR* - http://www.cs.cmu.edu/~hnn/igi/

[39] *pathChirp* - http://www.spin.rice.edu/Software/pathChirp/

[40] *TTCP Utility* - http://www.pcausa.com/Utilities/pcattcp.htm

[41] *NLANR/DAST: Iperf* - http://dast.nlanr.net/Projects/Iperf/

[42] *Netperf Homepage* - http://www.netperf.org/netperf/NetperfPage.html