VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

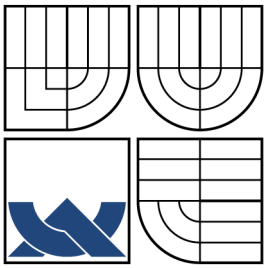
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MĚŘENÍ VZDÁLENOSTÍ MEZI STANICEMI V IP SÍTÍCH

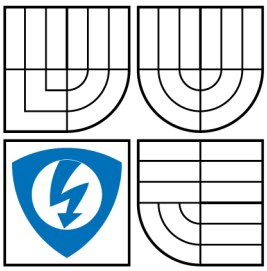
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. JAN ŠIMÁK
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

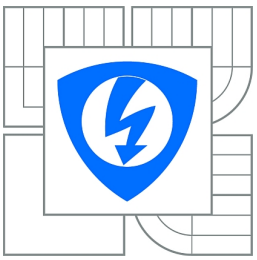
MĚŘENÍ VZDÁLENOSTÍ MEZI STANICEMI V IP SÍTÍCH
DISTANCE MEASUREMENT BETWEEN NODES IN IP NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. JAN ŠIMÁK
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. DAN KOMOSNÝ, PH.D.
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Jan Šimák	***ID:***	50473
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Měření vzdáleností mezi stanicemi v IP sítích

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s principy vyhodnocování polohy stanic v síti Internet. Zaměřte se na odhad pozice stanic pomocí systému Lighthouses. Proveďte simulace činnosti tohoto systému ve zvoleném simulačním prostředí. Dále proveďte porovnání systému Lighthouses s ostatními systémy pro určení pozice stanic v síti Internet. Při porovnání se zaměřte na dosahovanou přesnost, rychlost a náročnost odhadu pozice.

DOPORUČENÁ LITERATURA:

[1] PIAS, M. et al. Lighthouses for scalable distributed location [online]. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2003.
URL:<http://research.microsoft.com/~tharris/papers/2003-iptps.pdf> [cit. 13. 10 2009].
[2] EUGENE, T. S., ZHANG, H. Predicting Internet Network Distance with Coordinates-Based Approaches. Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2002.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Tato diplomová práce se zabývá problematikou predikce zpoždění mezi stanicemi v síti Internet. Přesná predikce zpoždění napomáhá při výběru nejbližšího internetového souseda a přispívá k efektivnějšímu využití síťových prostředků. Díky algoritmům predikujících zpoždění nedochází ke zbytečnému zvyšování zátěže v síti v důsledku měření zpoždění mezi všemi potřebnými uzly. Diplomová práce se teoreticky věnuje třem hlavním algoritmům mapujících reálnou síťovou topologii do souřadnicového systému - vektorového prostoru: GNP, Vivaldi, Lighthouses. Poslední jmenovaný je současně i hlavním tématem této práce. Algoritmus Lighthouses je v práci podrobně prozkoumán jak po stránce teoretické, tak i po stránce praktické. Za účelem ověření přesnosti predikce zpoždění algoritmu Lighthouses byl v rámci vypracování diplomové práce vyvinut simulační program, vypočítávající souřadnice stanic v umělé síťové topologii pomocí algoritmu Lighthouses. Popis simulačního pragramu a zhodnocení dosažených výsledků je součástí praktické části této diplomové práce.

KLÍČOVÁ SLOVA

zpoždění, vektorový prostor, překryvné sítě, predikce zpoždění, algoritmus Lighthouses, dimenze vektorového prostoru, protokol ICMP, poziční servery, souřadicový systém, simulace

ABSTRACT

This thesis deals with delay prediction issue between nodes on the Internet. Accurate delay prediction helps with choosing of the nearest internet neighbor and contributes to effective usage of network sources. Unnecessary network load is decreased due to algorithms of delay prediction (no need for many latency measuring). The thesis focuses theoretically on the three main algorithms using coordinate systems - GNP, Vivaldi, Lighthouses. Last one is at the same time the main subject of the thesis too. Algorithm Lighthouses is explored in detail theoretically and in practise too. In order to verify the accurate of delay prediction of Lighthouses algorithm the simulation application was developed. The application is able to compute node coordinates of synthetic network using Lighthouses algorithm. Description of simulation application and evaluation of simalution results are part of practice part of this thesis.

KEYWORDS

delay, vector space, overlay networks, delay predicting, Lighthouses algorithm, vector space dimension, ICMP protocol, well-known nodes, coordinate system, simulation

ŠIMÁK J. *Měření vzdáleností mezi stanicemi v IP sítích*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2010. 66 s. Diplomová práce. Vedoucí práce doc. Ing. Dan Komosný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Měření vzdáleností mezi stanicemi v IP sítích“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Danu Komosnému, Ph.D. z Ústavu telekomunikcí FEKT VUT v Brně za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce. Rovněž děkuji své manželce za trpělivost a podporu v průběhu psaní této práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Doba ve které dnes žijeme je, ač si to mnozí z nás ani neuvědomují, velmi úzce spjata s výpočetní technikou. Počítače v nejrůznějších podobách pronikly, od doby jejich vzniku[1], do všech oblastí lidské činnosti. Každý den je vyprodukováno obrovské množství dat, které je z velké části zapotřebí nějakým způsobem trvale uložit, abychom je mohli analyticky zpracovávat, vyhledávat v nich nebo jinak s nimi nakládat. K čemu by nám ale taková separovaná datová úložiště byla, kdybychom se k nim nemohli odkudkoli připojit. S touto problematikou přímo souvisí rozvoj celosvětové sítě Internet [1], který v posledních letech zaznamenává až exponenciální růst. Připojení k Internetu je dnes dostupné jak technologicky tak i finančně. Se stoupajícím počtem uživatelů Internetu úměrně roste množství dat, které je za den přeneseno. Rozvoj sítě Internet neprobíhá jen na úrovni fyzického propojení počítačů, ale také v oblasti síťových služeb. Mezi dnes nejpopulárnější a nejdynamičtěji se rozvíjející služby bezpochyby patří internetové televizní vysílá (IPTV), internetové rádiové vysílání, telefonování přes Internet (VoIP), internetové video půjčovny, výměnné datové sítě (P2P) [2] nebo servery nabízejí ke shlédnutí krátká videa[2]. Nesmíme také opomenout herní průmysl, který díky rozšiřování Internetu zaznamenává doslova boom v oblasti on-line her.

Všechny tyto internetové služby mají jedno společné. Tím je požadavek na dostatečnou šířku pásma a adekvátní hodnotu zpoždění v síti. Pokud uživatel využívající službu nesplňuje některý z těchto základních požadavků[3], stává se pro něj taková služba nekomfortní nebo až nepoužitelná. V zásadě existují dva způsoby, jak tento problém řešit. Prvním a ne vždy možným řešením je zkvalitnění datové linky od uživatele po poskytovatele Internetu (ISP – přístupové sítě). Tento krok je nevyhnutelný v případě, kdy datová linka mezi uživatelem a jeho poskytovatelem Internetu není dostatečně dimenzována na tento druh služeb. Druhý způsob řešení spočívá ve výběru vhodného serveru pro připojení. Když si chceme např. stáhnout instalační cd/dvd linuxové distribuce, jsme obvykle přesměrování na stránku, která obsahuje seznam serverů s námi žádaným obsahem. Tyto servery jsou seřazeny podle kontinentů a podle zemí ve kterých se fyzicky nacházejí. Zde záleží pouze na nás, který server si pro stahování vybereme. Podle mé zkušenosti ne vždy fyzicky nejbližší server rovná se nejvyšší rychlost stahování. Tato zkutečnost může být zapříčiněna dvěma faktory. Buď je nám fyzicky bližší server připojen linkou s užší šířkou pásma

[1] Historicky první funkční počítač byl Z1 navržený a zkonstruovaný německým inženýrem Konradem Zuse, dokončen roku 1936. Dnešní typy počítačů založené na technologii mikroprocesorů se začaly objevovat od roku 1981.

[2] Nejznámějším serverem tohoto typu v roce 2009 je <`http://www.youtube.com`>.

[3] Požadavky na šířku pásma a hodnotu odezvy se liší podle typu a vjemové kvality poskytované služby.

a nebo je vytížen velkým počtem uživatelů, kteří jsou k němu také připojeni.

Z výše uvedeného příkladu vyplývá význam správné volby serveru, od kterého bude klient odebírat službu. Ať už se jedná o pouhé stahování většího objemu dat, skupinové vysílání (multicast) nebo výměnné datové sítě typu P2P, vždy je zapotřebí nejprve nějakým vhodným způsobem vybrat nejlepší zdroj, ke kterému se klient má připojit. Správná volba zdrojového serveru nepřináší jen výhodu rychlejšího stahování dat, ale z principu také šetří volnou šířku pásma tím, že je celková zátěž rovnoměrněji rozprostřena mezi více zdroji[4]. Otázkou stále zůstává, jakým způsobem vybrat ten nejvhodnější server. Na příkladu se stahováním instalačního cd je vidět, že nechat tuto volbu na uživateli nepřináší žádané výsledky. Nehledě na to, že uživatel jako pouhý konzument služby by se touto otázkou zabývat neměl. Za tímto účelem se v současné době používá měření doby odezvy[5] mezi klientem a dostupnými zdroji s tím, že je vybrán zdroj s nejnižší dobou odezvy. Předpokládá se totiž, že zdroj s nejnižší dobou odezvy je z hlediska topologie sítě nejblíže. Mezi klientem a zdrojem je nejmenší možný počet síťových uzlů a data tak nejsou zbytečně přenášena přes více sítí než je skutečně zapotřebí.

Za účelem automatické volby vhodného zdroje bylo navrženo a prakticky zrealizováno několik algoritmů, které jsou schopny na základě několika málo měření dosti přesně určit, který z nabízených zdrojů je aktuálně nejblíže a tudíž i nejlepší volbou. Bohužel tyto algoritmy stále zůstávají spíše na úrovni laboratorního testování a jsou provozovány pouze v testovacích sítích jako je např. síť PlanetLab [3], [4]. Algoritmy používající k určení vzdálenosti mezi stanicemi souřadnicové systémy jsou např. Global Network Positioning (GNP) [5], Vivaldi [6], Practical Internet Coordinates (PIC) [7], Lighthouses [8], Internet Coordinate System (ICS). Druhá skupina algoritmů používá pro odhad vzdáleností mezi stanicemi model snažící se postihnout topologii reálné sítě tím, že celou síť člení do shluků uzlů. Těmito algoritmy jsou např. IDMaps nebo Internet Iso-bar.

Tato diplomová práce se podrobně věnuje algoritmu Lighthouses. V teoretické části podrobně popisuje principy fungování algoritmu. Jakým způsobem jsou vypočítávány umělé souřadnice reálné stanice a jak je následně z těchto souřadnic predikována vzdálenost mezi stanicemi, které jsou již v systému. Teoretická část práce také stručně popisuje některé z výše uvedených algoritmů. Pozornost je věnována algoritmům mapujícím reálnou síť do souřadnicového systému. Praktická část práce je zaměřena na simulaci algoritmu Lighthouses prostřednictvím vyvinutého simulačního programu, jenž provádí mapování imaginární síťové topologie do vícerozměrného souřadnicového systému. Závěr práce je věnován zhodnocení výsledků simulace a jejich rozboru.

[4]Pro více zdrojů poskytujících stejný obsah se používá pojem zrcadlo.

[5]Podrobněji je o této problematice pojednáno v kapitole 1.2.

1 POLOHA STANIC A JEJICH VZÁJEMNÁ VZDÁLENOST

1.1 Motivace určování polohy stanic

Výzkum v oblasti distribuovaných Internetových aplikací, které se začaly v hojné míře uplatňovat především v systémech pro sdílení obsahu (Napster, Gnutella), vedl ke vzniku sítí, které vytvářejí vlastní uspořádání aplikačních serverů (uzlů) nad uspořádáním (topologií) ve vrstvě síťové. Tyto sítě jsou obecně označovány jako překryvné sítě (Overlay Networks) [9]. Aplikační servery jsou v těchto sítích vzájemně propojeny virtuálními nebo logickými linkami. Hlavním přínosem překryvných sítí je možnost adresovat zprávu příjemci na základě jiného identifikátoru, než který je vyžadován v IP sítích – IP adresa. Prvními překryvnými sítěmi byly distribuované hašovací tabulky (DHT) [10] a distribuované stromy (DT). Distribuované hašovací tabulky jsou decentralizované systémy, které poskytují vyhledávací službu obdobného významu jakou mají hašovací tabulky v databázích. Vyhledávání a získávání uložených hodnot probíhá pomocí klíčů, které jsou přiřazeny jednotlivým uzlům. Odpovědnost za správně udržované vazby mezi klíčem a jeho hodnotou je rozprostřena mezi všemi uzly DHT. Z tohoto principu je takový systém velmi odolný vůči chybám na určité skupině uzlů a umožňuje dobré škálování při vzrůstajícím počtu uzlů. Systém, který v současné době aktivně využívá DHT, je protokol BitTorrent[1], vyvinutý za účelem výměny velkého množství dat mezi uživateli.

V překryvných sítích založených na původních verzích těchto protokolů je způsob rozhodování o výběru efektivní cesty mezi dvěma uzly velmi jednoduchý. Počítá se pouze počet uzlů v překryvné síti, přes které bude zpráva směrována. Trasa s nejmenším počtem uzlů je považována za nejoptimálnější. Tento způsob výběru trasy může být efektivní pouze ve velmi omezených síťových scénářích, v případě sítích sestávajících se z několika málo síťových prvků. Problém není ani tak v logice výběru trasy na základě nejmenšího počtu uzlů, ale ve skutečnosti, že topologie překryvné sítě většinou nezrcadlí topologii nižší vrstvy – vrstvy síťové. Z pohledu překryvné sítě mohou být dva uzly svými nejbližšími sousedy. Logická linka, která je spojuje, neobsahuje žádné další mezilehlé uzly. Ve světě reálných metalických nebo optických sítí by této topologii odpovídalo přímé propojení dvou stanic např. metalickým kabelem. Ve většině případů tomu tak ale není a co je pro překryvnou síť nejkratší trasa, může v reálném světě být až absurdní situace. Např. dva uzly, jeden v Praze a druhý Brně, spolu komunikují přes transatlantickou linku z výše uvedeného důvodu. Pokud by trasu nevolili na základě topologie překryvné sítě, s

[1]Specifikace protokolu BitTorrent <`http://www.bittorrent.org/beps/bep_0003.html`>.

největší pravděpodobností by spolu komunikovali přes optickou vnitrostátní linku. Dalším problémem mohou být vícenásobné přenosy po fyzických linkách.

Nízká efektivita těchto systémů vedla ke vzniku výzkumných projektů, které mají za cíl lépe využívat topologii síťové vrstvy, začlenit její existenci do algoritmu výběru blízkého serveru. Začalo se uvažovat o tom, jakým způsobem charakterizovat reálnou vzdálenost stanic ve škálovatelných modelech překryvných sítí. A zdali by takový model mohl vůbec napomoci při výběru vhodného blízkého serveru. Dále v této práci bude pojmem „blízký server“ nebo „síťová blízkost“ vždy myšleno jak blízko je stanice A od jiné stanice B s respektováním topologie síťové vrstvy. Síťová blízkost je zde charakterizována měřením parametru IP sítě, a to měřením hodnoty zpoždění (RTT). Kromě hodnoty zpoždění je také uvažováno o použití dalšího síťového parametru, dostupné šířky pásma. Otázce měření šířky pásma za použití deterministického modelu zpoždění paketů se věnuje studie [11]. Podrobněji bylo zatím prozkoumáno pouze použití parametru zpoždění. Nejvýznamnější z těchto projektů byly zmíněny v úvodu této práce.

1.2 Měření zpoždění v síti pomocí protokolu ICMP

Protokol ICMP [12] je součástí protokolové sady TCP/IP [1], která je nejrozšířenější protokolovou sadou současnosti. Komunikace v síti Internet probíhá výhradně prostřednictvím této protokolové sady. Protokol ICMP definuje sadu zpráv, pomocí kterých je možné informovat jiné stanice v síti o nejrůznějších událostech. Tyto zprávy jsou v praxi převážně používány v situacích, kdy dojde během zpracování IP datagramu k chybě a je žádoucí odeslat zpětnou vazbu o tom, že k chybě došlo, případně co bylo příčinou této chyby. Další oblastí, kde se tyto zprávy hojně využívají, je diagnostika sítě. Zprávy protokolu ICMP jsou posílány prostřednictvím hlavičky protokolu IP.

Pomocí protokolu ICMP můžeme měřit poměrně důležitý parametr sítě, a to dobu odezvy. Ekvivalentním názvem pro dobu odezvy je síťové zpoždění nebo jen zpoždění (latence). Měření zpoždění je vždy inicializováno stanicí, která má o tuto informaci zájem. Celý proces měření probíhá tímto způsobem. Inicializační stanice A odešle ICMP zprávu typu 8 – Echo Request (žádost). Pokud je zpráva správně doručena cílové stanici B a tato stanice na svém firewallu[2] neblokuje zpracování zpráv ICMP, je jako odpověď stanici A odeslána zpráva typu 0 – Echo Reply (odpověď). V tuto chvíli stanice A zná čas odeslání žádosti a také čas příjmu odpovědi. Takže si může velmi jednoduše spočítat síťové zpoždění jako rozdíl obou změřených časů. Pro čas uplynulý mezi odesláním žádosti a příjmem odpovědi se používá zkratka RTT

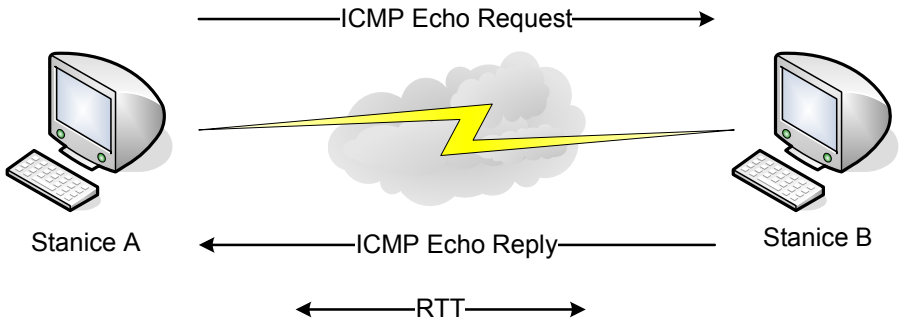
[2]Popis a přehled <http://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf>.

(round-trip delay time). I když to nemusí být zřejmé, do celkové hodnoty RTT je započtena i doba potřebná na zpracování datagramu. Na obrázku 1.1 je znázorněn schématický postup při měření hodnoty zpoždění v síti - RTT.

Údaj získaný pouze z jednoho měření je pouze informativní a nelze na jeho základě vyvozovat žádné závěry. Zprávy protokolu ICMP jsou v IP sítích směrovány stejným způsobem jako ostatní komunikace a tudíž i zde platí, že cesta datagramu od odesílatele zprávy nemusí být shodná s cestou datagramu od dotazovaného. Proto je zapotřebí měření opakovat několikrát za sebou a z těchto hodnot vypočítat hodnotu mediánu. Statistická výhoda mediánu oproti aritmetickému průměru spočívá v tom, že není ovlivněn extrémními hodnotami, jenž mohou aritmetický průměr do značné míry zkreslit. Např. zdrojová data použitá při simulaci algoritmu Lighthouses ve studii [8], byla změřena v souvislosti s projektem GNP[3] a hodnoty RTT zde uvedené, jsou mediánem z 220 opakování měření hodnoty RTT mezi dvěma stanicemi.

Měření jediné hodnoty RTT vnáší do celkového síťového provozu jen velmi nepatrné zatížení. Sítí je přenášena malá datová jednotka o celkové velikosti 64 bajtů. Jelikož je pouze jedno měření naprosto nevyhovující a čím více změřených vzorků nasbíráme, tím přesnější hodnotu mediánu jsme schopni vypočítat, začíná být otázka zátěže sítě aktuálním tématem. Zatížení sítě měřením roste kvadraticky s počtem stanic v síti. Při N stanicích je zapotřebí provést měření. Praktická nemožnost provádět tato měření na vyžádání kdykoli je to potřeba byla a stále je motivací pro vznik projektů mající za cíl s vysokou přesností vypočítat hodnotu RTT mezi stanicemi, aniž by bylo zapotřebí provádět skutečná měření. Vstupními hodnotami jsou obvykle jen změřené vzdálenosti od stanice k několika málo referenčním bodům. Jiným přístupem k tomuto problému je měření protokolem ICMP neprovádět vůbec a pro odhad zpoždění využít existujících systémů. Takovým systémem je např. King [13], který pro odhad vzdálenosti mezi stanicemi využívá systému DNS. Další možnosti nabízejí výměnné datové sítě, ve kterých je možné pro měření RTT využít stávajícího systému zpráv a zpoždění vypočítat stejným způsobem jako při zasílání ICMP zpráv.

[3]Zdroj změřených dat <`http://www.cs.rice.edu/~eugeneng/research/gnp/`>.



Obrázek 1.1: Schéma měření zpoždění v síti protokolem ICMP.

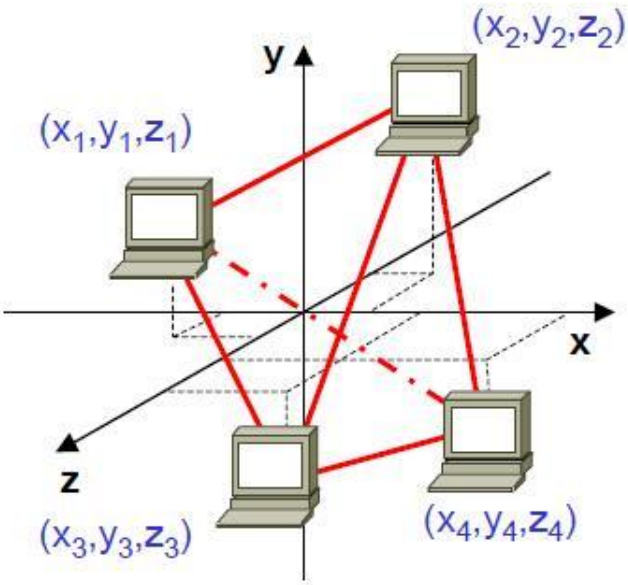
2 SOUŘADNICOVÉ SYSTÉMY

Za účelem vytvoření modelu síťové topologie, pomocí kterého by bylo možné predikovat s vysokou přesností zpoždění mezi dvěma stanicemi bez potřeby provádět skutečná měření, byly souřadnicové systémy shledány vhodným nástrojem pro tento úkol. Síťovou topologii můžeme vyjádřit pomocí souřadnicového systému na základě měření zpoždění mezi stanicemi. Každá stanice, jenž se chce stát součástí takového síťového modelu, změří svoji síťovou vzdálenost (realizováno měřením RTT) ke skupině uzlů, které jsou již součástí takového modelu. Počet měření a výběr uzlů pro které bude měření provedeno závisí na použitém algoritmu. Na základě změřených hodnot jsou vypočteny souřadnice definující pozici stanice v souřadnicovém systému. Obecný příklad 3D souřadnicového systému modelu sítě je uveden na obrázku 2.1.

2.1 Požadavky kladené na souřadnicové systémy

Požadavkem na algoritmy zabývající se touto problematikou je, aby vypočtené souřadnice správně reflektovaly naměřené hodnoty zpoždění. Pro výpočet predikovaného zpoždění mezi uzly v souřadnicovém modelu sítě se používají funkce vypočítávající vzdálenost – tzv. funkce vzdálenosti (např. Euklidovská vzdálenost). Druhým významným požadavkem na algoritmy vytvářející souřadnicový systém je, aby výsledný systém byl dobře škálovatelný. Vlastnost škálovatelnosti takového systému je přínosná zejména v situacích, kdy souřadnicový model reprezentuje topologii rozsáhlé sítě sestávající se ze stovek nebo tisíců uzlů. Pokud by pak nějaká aplikace chtěla pracovat pouze s menší skupinou logicky uspořádaných uzlů, bude pro ni určitě vhodnější pracovat pouze se souřadnicemi vztahujícími se k této skupině stanic, než s globálními souřadnice pro celý systém. Druhou vlastností vyplývající ze škálovatelnosti je schopnost takového systému lépe odolávat proti chybám (např. dočasná nedostupnost) na dílčí skupině uzlů. Tato druhá vlastnost nemusí být obecně platná pro všechny algoritmy. Pokud by např. došlo k výpadku referenčního uzlu u algoritmu GNP [5], byla by funkčnost takové sítě značně omezena.

Libovolný souřadnicový systém nazveme metrickým prostorem, pokud je v něm definovaná uspořádaná dvojice (X, d), kde X je nenulová množina bodů a d je funkce vzdálenosti, neboli metrika. Pro každou z uspořádaných dvojic náležející do množiny X tak, že platí $a, b \in X$, můžeme vypočítat jejich vzdálenost promocí vzdálenostní funkce $d(a, b)$. Číslo d nazveme vzdáleností prvků a, b v prostoru (X, d) - metrickém prostoru. Pro každou neprázdnou množinu prvků X můžeme definovat celou řadu různých metrik. Získáme tak různé metrické prostory, které budou tvořeny stejnou základní množinou prvků (nosičů), ale navzájem se budou lišit způsobem výpočtu, neboli měřením vzdálenosti.



Obrázek 2.1: Obecný příklad 3D souřadnicového systému modelu sítě.
Převzato z publikace [5].

Pokud je X libovolná nenulová množina $X \neq 0$ a d je metrika této množiny, pak pro tuto metriku platí následující axiomy:

1. nezápornost $d(a, b) \geq 0$,
2. definitnost $d(a, b) = 0 \Leftrightarrow a = b$, identické prvky z množiny X mají vždy nulovou metriku,
3. symetrie $d(a, b) = b(b, a)$,
4. trojúhelníková nerovnost $d(a, b) \leq d(a, c) + d(c, d)$.

Souřadnicové systémy budované na základě změřeného zpoždění mezi stanicemi často trpí porušením trojúhelníkové nerovnosti [14]. Porušení trojúhelníkové nerovnosti představuje libovolná trojice bodů (a, b, c), která na základě vzájemných měření zpoždění nesplňuje podmínku, že jedna hrana trojúhelníku musí být vždy menší než součet zbylých dvou hran. Porušení má vliv na přesnost predikce zpoždění. Algoritmy reprezentující reálnou síť v Euklidovském prostoru proto nemohou třem uzlům porušujícím trojúhelníkovou nerovnost přiřadit souřadnice v tomto prostoru, aniž by tím nebyla snížena celková přesnost modelu. Způsobů, jak se s tímto problémem vyrovnávají různé projekty v této oblasti, existuje několik. Často je s porušením nakládáno jako s artefaktem měření nebo je existence porušení úplně ignorována, případně za považována za nedůležitou. V publikaci [14] je zpracována studie, která

se zaměřuje na příčiny porušení trojúhelníkové nerovnosti. Prozkoumává vliv pravidel směrování jak v případě směrování vnitro-doménového, tak vliv směrování mezi jinými doménami. Dále je zde diskutována zkutečnost, že se planeta Země svým tvarem blíží tvaru tělesa koule, zatímco výpočty spoléhající na trojúhelníkovou nerovnost jsou vázány na rovinu, což nemalou měrou zvyšuje četnost porušení trojúhelníkové nerovnosti.

2.2 Volba dimenze souřadnicového systému

Volba vhodného počtu rozměrů má přímý vliv na přesnost predikce zpoždění mezi stanicemi. Při volbě dimenze budoucího souřadnicového modelu sítě nejsme nikterak omezeni maximálním počtem rozměrů. Je tedy možné vytvářet souřadnicové systémy o dvou, třech, čtyřech, ale i devíti, jedenácti atd. rozměrech. Neplatí zde ale přímá úměra, že se zvyšujícím se počtem rozměrů dochází ke zvyšování přesnosti systému. Více rozměrů znamená také vyšší výpočetní nároky, takže je zapotřebí najít kompromis mezi akceptovatelnou chybou a složitostí výpočtu. Za účelem dalšího zvýšení přesnosti predikce, byla zavedena speciální rozměrová složka označovaná jako výškový vektor. Odlišnost výškového vektoru od klasických rozměrových složek v *n-rozměrném* prostoru spočívá v tom, že osa definující směr výškového vektoru je definována pouze v oblasti nezáporných čísel. Důvodem pro jeho zavedení bylo nějakým vhodným způsobem zohlednit tu část zpoždění, která vzniká v přístupových sítích. Přičemž zpoždění vznikající v přístupových sítích bývá většinou dominantní složkou. Obrázek 2.2 znázorňuje použití výškového vektoru v 2D prostoru.

Reprezentace síťové topologie v 2D prostoru vykazuje nejvyšší míru nepřesnosti. Důvodem je nedostatečný počet možností, jak body v tomto prostoru umístit. Z tohoto důvodu se 2D prostory používají pouze pro názornou ukázku principu fungování algoritmu. Prostory vykazující nízkou chybu predikce se ukázaly být 4D a 5D prostory a 4D prostor s výškovým vektorem.

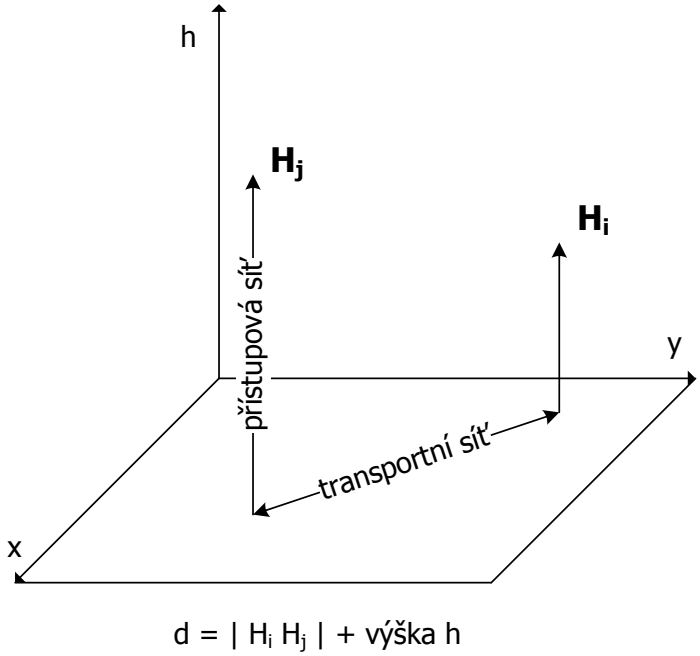
Výpočet predikce zpoždění mezi dvěma stanicemi v souřadnicovém modelu sítě spočívá ve výpočtu vzdálenosti dvou bodů téhož souřadnicového systému. Příkladem vzdálenostní funkce je Euklidovská vzdálenost

$$d(H_i, H_j) = |H_i\ H_j| = \sqrt{\sum_{k=1}^{d} (H_{ik} - H_{jk})^2}\ , \tag{2.1}$$

kde H_i a H_j jsou souřadnice dvou stanic v *n-rozměrném* souřadnicovím systému. Při zavedení výškového vektoru, jakožto speciální rozměrové složky, je vzdálenostní funkce modifikována následujícím způsobem

$$d(H_i, H_j) = |H_i\ H_j| + h\ , \tag{2.2}$$

kde parametr h značí výškový vektor.



Obrázek 2.2: Použití výškového vektoru v 2D prostoru.

3 ALGORITMY POUŽÍVAJÍCÍ SOUŘADNICOVÉ SYSTÉMY

Algoritmů postavených na myšlence mapování reálné síťové topologie do umělého souřadnicového systému existuje celá řada. Některé z nich používají k výpočtu souřadnic fixní množinu orientačních serverů, jiné naopak každý server připojený do takové sítě automaticky používají jako další orientační server s možností libovolného výběru ze všech těchto serverů. V následujících podkapitolách jsem se zaměřil na podle mého názoru tři hlavní představitele algoritmů v této oblasti. Každý z níže uvedených algoritmů svým vývojem přinesl zajímavou a někdy až zásadní myšlenku na zvýšení přesnosti při mapování reálné síťové topologie do souřadnicového systému.

3.1 Global Network Positioning (GNP)

Algoritmus GNP byl navržen za účelem predikce metriky zpoždění. Reálná síťová topologie je modelována do *n-rozměrného* vektorového prostoru (např. Euklidovský prostor). Pozice každého prvku v tomto prostoru je jednoznačně určena na základě vypočtených souřadnic. Hlavní myšlenou algoritmu GNP je existence skupiny obecně známých pozičních serverů, které tvoří základ souřadnicového systému. Ostatní stanice odvozují své souřadnice pomocí souřadnic těchto referenčních serverů. Architektura algoritmu GNP se skládá ze dvou základních částí.

V první části, která vytváří páteř celého systému, je vybrána relativně malá skupina serverů. Servery v této skupině se označují jako orientační body (landmarks) a slouží k orientaci ostatních stanic ve vektorovém prostoru. Předpokladem pro dobrou přesnost algoritmu je volba vhodného počtu orientačních bodů a jejich rozmístění. Minimální počet je určen dimenzí vektorového prostoru. Tzn. že musíme pro geometrický prostor o dimenzi d použít alespoň $d+1$ orientačních bodů. V opačném případě je totiž nemožné jednoznačně vypočítat souřadnice libovolné stanice. Více je o této problematice pojednáno v kapitole 3.3.1. V prvním kroku algoritmu jsou změřeny vzájemné vzdálenosti (zpoždění RTT) mezi všemi orientačními body. Měření vzdálenosti je několikrát opakováno a ze získaných dat je statistickou funkcí medián vypočítána průměrná hodnota zpoždění mezi serverem H_i a H_j. Na základě těchto hodnot je následně sestavena matice vzájemných vzdáleností mezi orientačními body o rozměru $N \times N$. Z matice změřených vzdáleností vypočte jeden z orientačních bodů (většinou hlavní řídící server) souřadnice pro všechny orientační body v prostoru D. Úkolem při hledání souřadnic pro textitN orientačních bodů je nalézt takové souřadnice, aby celková chyba mezi změřenými a vypočtenými vzdále-

nostmi byla co nejvíce minimalizována. Souřadnice jsou vypočteny minimalizováním hodnoty následující funkce

$$f_{srv}\left(c_1^S, ..., c_M^S\right) = \sum_{i,j} \varepsilon\left(d_{H_i,H_j}, \hat{d}^S_{H_i,H_j}\right) \ , \tag{3.1}$$

kde d_{H_i,H_j} je změřená a $\hat{d}^S_{H_i,H_j}$ vypočtená vzdálenost mezi orientačními body H_i a H_j. $\varepsilon(\cdot)$ je funkce chyby měření, kterou může být některá z chybových funkcí, např. kvadratická odchylka,

$$\varepsilon\left(d_{H_i,H_j}, \hat{d}^S_{H_i,H_j}\right) = \left(d_{H_i,H_j} - \hat{d}^S_{H_i,H_j}\right)^2 \ . \tag{3.2}$$

Druhá část algoritmu se vztahuje k procesu výpočtu souřadnic běžné stanice (např. klientského počítače). S využitím souřadnic orientačních bodů si nyní může každá stanice odvodit své vlastní souřadnice. Stanice, která se chce stát součástí souřadnicového systému dimenze N, si nejprve zvolí N orientačních bodů, ke kterým následně změří svoji síťovou vzdálenost (měření je opět realizováno pomocí RTT). Počet volených orientačních bodů je určen dimenzí vektorového prostoru. Ze vzorků několika měření si stanice, obdobně jako v případě orientačních bodů, funkcí medián vypočítá průměrnou hodnotu zpoždění. Změřené vzdálenosti mezi stanicí a vybranými orientačními body jsou použity pro výpočet souřadnic této stanice. Výpočet se provádí stejným způsobem jako v případě souřadnic orientačních bodů, tzn. vypočtené souřadnice jsou získány na základě minimalizace chyby mezi změřenými a vypočtenými vzdálenostmi. V publikaci [17] je pro řešení obecného problému minimalizace chyby v *n-rozměrném* prostoru použit algoritmus Simplex Downhill.

Odhad zpoždění mezi stanicí H_i a H_j lze jednoduše vypočítat jako vzdálenost mezi body H_i^N a H_j^N v *n-rozměrném* prostoru, např. pomocí Euklidovské vzdálenosti $\left|H_i^N \ H_j^N\right|$.

3.2 Algoritmus Vivaldi

Vznik a vývoj algoritmu Vivaldi [6] je také motivován cílem přesné predikce zpoždění s minimálními nároky na počet měření. Algoritmus není postaven na skupině fixních, ostatním serverům dobře známých, orientačních bodů. Pro výpočet souřadnic stanice stačí pouze několik měření k ostatním stanicím, které jsou již současností modelu sítě. Z tohoto důvodu je možné se od měření zpoždění protokolem ICMP úplně oprostit a k měření využít stávající komunikaci aplikací, které vytváření překryvné sítě s využitím výpočtů algoritmu (např. projekt Azureus [18]). Algoritmus, stejně jako v případě GNP, hledá souřadnice pomocí minimalizace chyby predikce zpoždění, v tomto případě je minimalizována hodnota funkce kvadratické odchylky.

Inovací algoritmu Vivaldi je použití speciálního prvku – pružiny. Minimalizace chyby je dosažena simulací sítě fyzických pružin, které na sebe vzájemně působí. V umělém souřadnicovém systému je princip pružin realizován následujícím způsobem. Mezi každé dva servery v souřadnicovém systému je umístěn prvek pružiny. Působením pružiny dochází k posunu (upřesnění) souřadnic vypočtených ze změřených hodnot zpoždění a tím k minimalizaci chyby. Velikost síly působení pružiny je určena ze dvou vstupních hodnot. Klidová délka pružiny je nastavena na změřenou hodnotu zpoždění. Druhým parametrem je aktuální délka pružiny, neboli natažení, která je rovna vypočtené vzdálenosti mezi body v souřadnicovém systému.

Potenciální energie pružiny je rovna druhé mocnině posunutí z klidové délky pružiny. Z tohoto důvodu algoritmus Vivaldi jako chybovou funkci zvolil funkci kvadratické odchylky. K minimalizaci kvadratické odchylky dochází na základě pohybu stanic podle síly a směru působení pružiny. Princip pružin je postaven na Hookově zákoně, podle kterého je vratná síla pružiny rovna délce d prodloužení pružiny a vratnou silou F, která se snaží uvést pružinu zpět do klidového stavu. Pokud označíme $\overline{F_{H_i,H_j}}$ za vektor působení pružiny mezi dvěma servery H_i a H_j v souřadnicovém modelu, se směrem působení k serveru H_i, můžeme sílu pružiny vypočítat následujícím způsobem

$$\overline{F_{H_i,H_j}} = \left(d_{H_i,H_j} - |H_i\ H_j| \times u\left(H_i - H_j\right)\right). \tag{3.3}$$

Přičemž výraz $\left(d_{H_i,H_j} - |H_i\ H_j|\right)$ udává velikost posunutí pružiny z klidového stavu. Velikost tohoto posunutí je rovna síle, kterou pružina působí na servery H_i a H_j. Směr síly působení pružiny na server H_i je dán jednotkovým vektorem $u\left(H_i - H_j\right)$. Celková síla působící na každý uzel v modelu je pak určena součtem vektorů všech sil, působících z ostatních uzlů.

Aby bylo možné simulovat vývoj souřadnic serverů v modelu s působením pružin, působí vždy síla virtuální pružiny na konkrétní stanici pouze po krátký interval času. V každém intervalu dojde k posunutí stanice H_i o malou vzdálenost v souřadnicovém modelu sítě ve směru působení síly. Následně jsou všechny síly působící na stanici H_i přepočítány, aby reflektovaly přesnost nových souřadnic. Souřadnice na konci každého časového intervalu jsou určeny rovnicí

$$H_i = H_i + \overline{F_i} \times t\ , \tag{3.4}$$

kde t je délka časového intervalu. Správné nastavení hodnoty intervalu t hraje v algoritmu Vivaldi velmi důležitou roli. Jeho nevhodným nastavením lze podstatným způsobem degradovat přesnost predikce zpoždění.

Cílem algoritmu Vivaldi je tedy minimalizovat energii natažení pružiny, neboli jde o snahu uvést pružinu do klidového stavu. Minimální energie pružiny odpovídá

minimální chybě přiřazených souřadnic. Tento postup vede k nalezení lokálního minima systému. Nalezení globálního minima touto technikou nicméně není zaručeno.

Algoritmus Vivaldi je možné používat ve dvou odlišných módech. Prvním módem je algoritmus s konstantním krokem. Každá stanice si sama vypočítává a následně přizpůsobuje své souřadnice na základě změřených hodnot RTT ke skupině známých serverů. S každou další sadou změřených hodnot proběhne vždy jedna iterace algoritmu. V první iteraci algoritmu je stanice umístěna do počátku souřadnicového systému. Pohyb stanice v souřadnicovém systému je zde také řízen směrem a silou působení pružiny. S každým pohybem stanice (změnou souřadnic) dochází k minimalizaci chyby vzhledem k ostatním stanicím v systému, ke kterým měří hodnotu zpoždění. Druhým módem je algoritmus s adaptivním krokem. Jak již bylo naznačeno výše, klíčovým prvkem u algoritmu Vivaldi je vhodné stanovení velikosti časového kroku. Příliš velká hodnota způsobuje při každé iteraci posun stanice o příliš velkou vzdálenost a vede k oscilaci pohybu namísto postupného zpřesňování. Řešením je použití adaptivního časového kroku, jehož velikost je přizpůsobována podle přesnosti stávajících souřadnic. Pokud je přesnost vypočtených souřadnic ve fázi hrubého hledání pozice stanice v souřadnicovém systému, je použit časový krok s vyšší hodnotou. Díky vyšší hodnotě časového kroku se stanice dostane rychleji do předpokládané oblasti, kde bude své souřadnice dále zpřesňovat. S postupným zpřesňováním souřadnic dochází k pozvolnému snižování velikosti časového kroku.

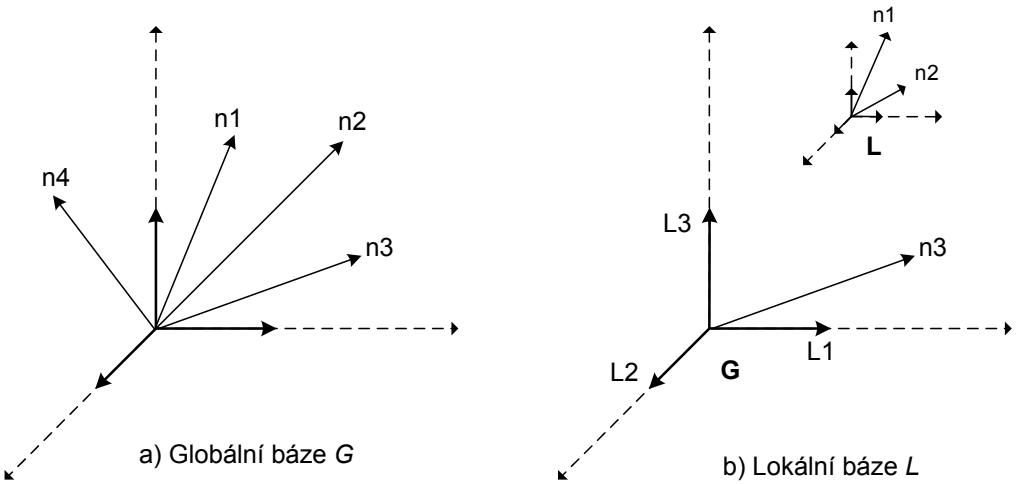
3.3 Algoritmus Lighthouses

Algoritmus Lighthouses [8] je koncepčně nejvíce podobný systému GNP. Motivací pro vznik tohoto algoritmu je skutečnost, že množina pevně daných orientačním bodů (landmarks) vytváří slabá místa ve fungování překryvné sítě založené na algoritmu GNP. Otázka, jak by měl systém zareagovat a vhodně se přeorganizovat, aby vyplnil prázdná místa v případě nedostupnosti části orientačních bodů, není v původním algoritmu GNP vyřešena. Při návrhu algoritmu byl tento nedostatek zohledněn a pozičními servery může být libovolná skupina serverů v systému. Každá stanice si proto může zvolit skupinu pro ní referenčních serverů zcela odlišnou od volby ostatních stanic. V publikaci [8] se o náhodně zvolených orientačních uzlech hovoří jako pivotech (pivots) a o technice výběru pivotů jako o technice pivotování (pivoting technique). Mechanismus náhodného výběru referenčních serverů vytváří mnohonásobné lokální souřadnicové systémy, tzv. lokální báze. Tímto způsobem se algoritmus Lighthouses brání vzniku slabých míst v systému, které by v případě chyby globálně omezily nebo narušili jeho fungování. Vznik lokálních bází dále umožňuje neomezenou škálovatelnost systému.

Princip fungování algoritmu je postaven na koexistenci mnohonásobné lokální báze spolu s tzv. přechodovou maticí P. Přechodovou matici P je povinen správně udržovat každý server v systému po celou dobu své existence. Na základě výběru skupiny referenčních serverů, tzv. majáků (lighthouses), si stanice pomocí základních aritmetických a vektorových operací dokáže sama vypočítat vlastní souřadnice v souřadnicovém modelu sítě. Takto získané souřadnice vždy určují pozici stanice relativně k souřadnicím použitých majáků. Jinými slovy, jde o souřadnice vztažené ke konkrétní lokální bázi. Abychom byli schopni predikovat zpoždění mezi servery které jsou umístěny v odlišných lokálních bázích, byl navržen způsob výpočtu globálních souřadnic z již známých souřadnic lokální báze pomocí přechodové matice P. Tím, že přechodová matice P tvoří vazbu mezi souřadnicemi vztaženými k některé z lokálních bází a společnou globální bází souřadnicového systému, je její bezchybné udržování klíčovým prvkem. Obrázek 3.1 znázorňuje rozdíl mezi globální bází G a lokální bází L téhož souřadnicového systému.

3.3.1 Postup při hledání majáků

Při procesu připojování nové stanice do překryvné sítě organizované algoritmem Lighthouses, si musí tato stanice nejprve vytvořit seznam majáků (referenčních serverů), které jí poslouží při výpočtu souřadnic lokální báze. Za účelem získání seznamu serverů, potencionálních majáků, nová stanice H_i kontaktuje libovolný server H_j ze všech serverů v souřadnicovém modelu sítě. Způsobů, jak stanice mimo



Obrázek 3.1: a) Globální báze G b) Lokální báze L. Převzato z publikace [8].

souřadnicový model sítě zjistí, které stanice jsou již připojeny, existuje několik. Pro překryvnou síť, kde není předpoklad expanze připojených stanic v řádů stovek až tisíců, může existovat v pravidelných intervalech aktualizovaný seznam připojených stanic. Nová stanice by si pak tento seznam stáhla a vhodným způsobem vybrala server, který bude kontaktovat. Pro rozsáhlejší překryvné sítě výměnných datových P2P sítí, by seznam potenciálních majáků mohla poskytovat služba tracker[1] na základě požadovaných dat a serverů nabízejících tato data ke stažení.

Jakmile stanice H_i některým z výše naznačených způsobů zvolí referenční server H_j, zašle mu žádost o zaslání seznamu serverů v systému. Stanice H_i si ze zaslaného seznamu následně vybere potřebný počet serverů, které budou v následujících krocích fungovat jako její majáky. Způsob výběru serverů je vždy ponechán na přistupující stanici. V původním návrhu algoritmu nejsou stanovena žádná kritéria či omezení, jako např. výběr tří po sobě jdoucích serverů v seznamu a podob. Počet vybíraných majáků je určen výrazem $d + 1$, kde proměnná d udává dimenzi vektorového prostoru tvořící model překryvné sítě. Především na počátku budování vektorového prostoru modelu sítě dochází k situaci, kdy počet serverů v systému neumožňuje splnit podmínku potřebného počtu majáků pro výpočet souřadnic lokální báze. Pokud je nová stanice *m-tou* stanicí v systému, tak že platí $m \leq d + 1$, je takové stanici přiřazen atribut „první stanice“ a souřadnice lokální báze musí být vypočítány pomocí serverů, které jsou již do systému zapojeny.

Ve chvíli kdy si stanice H_i vytvoří seznam vlastních majáků, může začít měřit zpoždění mezi ní a referenčním pozičním serverem. Referenčním pozičním serverem bývá většinou ten poziční server, který byl kontaktován stanicí za účelem získání seznamu pozičním serverů v systému. Tento server je zároveň také vždy obsažen v

[1]Služba udržuje aktualizovaný seznam připojených klientů v kontextu na sdílený obsah.

seznamu volených pozičních serverů. Měření zpoždění je i zde realizováno protokolem ICMP a měří se obousměrné zpoždění RTT. Jak již bylo vysvětleno v kapitole 1.2, počet měření je zapotřebí několikrát opakovat a ze vzorků vypočítat statistickou hodnotu mediánu. Změřená a statisticky zpracovaná hodnota zpoždění mezi přistupující stanicí H_i a jejím referenčním serverem je použita ve třetím kroku algoritmu Lighthouses při výpočtu souřadnic stanice H_i vztažených k lokání bázi.

3.3.2 Výpočet souřadnic lokální báze

Na cestě za výpočtem souřadnic stanice H_i relativních k bázi tvořenou vybranými majáky, je v druhém kroku algoritmu zapotřebí vypočítat souřadnice lokální báze. Lokální bázi L o stejné dimenzi jakou má základní vektorový prostor V^d, kde d značí stupeň dimenze, tvoří množina vektorů $l_1, l_2, ..., l_d$

$$L = \{l_1, l_2, ...l_d\} \ , \tag{3.5}$$

kde každý z vektorů $l_1, l_2, ..., l_d$ je tvořen dvojicí majáků stanice H_i. Každá dvojice majáků se skládá z počátečního pozičního serveru lokální báze a některého z dalších pozičních serverů téže báze. Obecně můžeme za počáteční poziční server označit ten server, který byl kontaktován přistupující stanicí H_i v předchozím kroku algoritmu a ke kterému bylo měřeno zpoždění. Např. vektor l_1 vypočítáme pomocí základního vzorce pro výpočet vektoru ze známých souřadnic dvou bodů v témže prostoru, kterými v tomto případě jsou počáteční poziční server lokální báze p_l a některý z dalších pozičních serverů o_l

$$l_1 = \overline{p_l o_l} \Rightarrow (p_{lx} - o_{lx}, p_{ly} - o_{ly}, ..., p_{ld} - o_{ld}) \ . \tag{3.6}$$

Algoritmus Lighthouses pro výpočet souřadnic lokální báze z vektorů $l_1, l_2, ..., l_d$ používá Gram-Schmidtův algoritmus [19] definovaný následujícím způsobem

$$\begin{aligned} l_1 &= proj_{W_0} l_1 + proj^{\perp}_{W_0} l_1 \ , \\ l_2 &= proj_{W_1} l_2 + proj^{\perp}_{W_1} l_2 \ , \\ &\vdots \\ l_d &= proj_{W_{d-1}} l_d + proj^{\perp}_{W_{d-1}} l_d \ , \end{aligned} \tag{3.7}$$

kde vektor $proj_{W_{d-1}} l_d$ je ortogonální promítnutí vektoru l_d na podprostor W_{d-1}. Přičemž musí platit, že W_{d-1} je podprostor základního prostoru V^d se skalárním součinem. Vektor $proj^{\perp}_{W_{d-1}} l_d$ je složka vektoru $proj_{W_{d-1}} l_d$ ortogonální na podprostor W_{d-1}.

Jelikož algoritmus Lighthouses umožňuje naprostou volnost při výběru majáků (s výjimkou počtu majáků, které je zapotřebí vybrat), je třeba zajistit, aby vypočítané

vektory lokální báze formovaly vektorový prostor o stejné dimenzi d jako je dimenze základního prostoru V^d. Pro splnění této podmínky musí být vypočítané vektory lokální báze lineárně nezávislé, tzn. každý vektor ve vektorovém prostoru V dimenze d musí být vyjádřitelný jako lineární kombinace vektorů lokální báze L. Ověření lineární nezávislosti vektorů lokální báze je realizováno prostřednictvím matice M_L, kde sloupce matice jsou vektory lokální báze,

$$\det(M_L) = \begin{vmatrix} l_1 \\ l_2 \\ \vdots \\ l_d \end{vmatrix} \neq 0 \tag{3.8}$$

Pokud je determinant matice M_L různý od nuly, je ověřena lineární nezávislost vektorů lokální báze. V opačném případě, kdy jsou vektory lineárně závislé, musí dojít v této fázi k ukončení procesu výpočtu souřadnic. Vypočítané souřadnice lokální báze z vektorů, jenž formují vektorový prostor o nižší dimenzi než je dimenze V^d, by totiž byly naprosto nesmyslné. Stanice H_i si tak musí vybrat jinou kombinaci majáků a celý proces výpočtu začíná od začátku.

3.3.3 Výpočet souřadnic stanice

Volnost při výběru majáků na jedné straně přináší zjednodušení algoritmu ve fázi výběru referenčních serverů. Na straně druhé je nezbytně nutné ošetřit algoritmus tak, aby byl schopen správně reagovat na nejrůznější situace, které z této volnosti výběru plynou. V případě, že by výběr majáků byl nějakým způsobem cíleně řízen, mohl by algoritmus výpočet souřadnic v této fázi ukončit pouhým součtem vektorů, jenž formují lokální bázi stanice. Tento postup by platil pouze za předpokladu, že jsou tyto vektory vzájemně ortogonální.

Aby byl algoritmus Lighthouses schopen pokračovat i za situace, kdy vektory lokální báze nejsou vzájemně ortogonální ale kosé, jsou souřadnice h_i stanice H_i vypočítány jako lineární kombinace vektorů lokální báze L

$$h_i = c_1 l_1 + c_2 l_2 + \cdots + c_d l_d \ , \tag{3.9}$$

kde čísla $c_1, c_2, ..., d_d$ jsou řešením soustavy lineárních rovnic 3.10.

$$\begin{aligned} c_1|l_1| + ... + c_d|l_d| \cos(\widehat{n_i, l_d}) &= |n_i| \cos(\widehat{n_i, l_1}) \\ &\vdots \\ c_1|l_1| \cos(\widehat{l_1, l_d}) + ... + c_d|l_d| &= |n_i| \cos(\widehat{n_i, l_d}) \end{aligned} \tag{3.10}$$

Výraz $\widehat{x_d, y_d}$ značí skalární součin vektorů definovaných pod znakem stříšky. Délky vektorů l_n jsou vypočítané vektory lokální báze z navrženého počátečního pozičního serveru (počátek lokální báze) ke všem ostatním pozičním serverům. Výraz $|n_i|$ je změřené zpoždění mezi počátečním pozičním serverem a přistupující stanicí. Souřadnice n_i zde značí aktuální umístění stanice, které právě počítáme.

3.3.4 Přechodová matice

Souřadnice stanice H_i, vypočítané v předchozích krocích algoritmu, určují pozici vždy vztaženou k lokální bázi, jenž je tvořena zvolenými pozičními servery. Aby bylo možné určit globální pozici stanice H_i, je zapotřebí vypočítané souřadnice vhodným způsobem doplnit o informaci globální báze. Pokud bychom chtěli predikovat zpoždění mezi dvěma stanicemi v souřadnicovém modelu sítě, přičemž obě stanice jsou umístěny v odlišných lokálních bázích, nebylo by to bez znalosti jejich globálních souřadnic možné. Pro potřeby odvozování globálních souřadnic stanic v souřadnicovém modelu sítě byla navržena přechodová matice P.

Poslední informaci, kterou každá nová stanice musí ještě vypočítat, je přechodová matice P. Znalost přechodové matice P a její správné udržování (aktualizace v případě změny souřadnic stanice způsobené změnou v síťové topologii) po celou dobu existence stanice v souřadnicovém systému, je podmínkou správné funkčnosti algoritmu Lighthouses. Sloupce přechodové matice P jsou tvořeny souřadnicemi nové báze vektorů, jenž je vztažena k původní bázi vektorů - lokální bázi stanice H_i,

$$P = \begin{vmatrix} [u'_1]_B \\ [u'_2]_B \\ \cdots \\ [u'_d]_B \end{vmatrix} . \tag{3.11}$$

Stanice H_i vypočítá přechodovou matici P vyjadřující vazbu mezi její lokální bází L a globální bází G souřadnicového systému. K tomu aby mohla stanice přechodovou matici spočítat, musí kromě vlastních souřadnic lokální báze znát ještě buď souřadnice globální báze nebo přechodovou matici referenčního majáku, kterého kontaktovala za účelem získání seznamu pozičních serverů v systému. Přičemž souřadnice globální báze G potřebné pro výpočet přechodové matice ve skutečnosti tvoří globální souřadnice majáků, které byly použity při výpočtu lokální báze a souřadnic stanice v této lokální bázi. Toto tvrzení je dalším z řady důkazů, proč počet vybíraných majáků musí být $d + 1$, kde d je dimenze souřadnicového systému. Pro výpočet přechodové matice P algoritmus Lighthouses používá následující rovnici

$$[v]_{B'} = P^{-1} \, [v]_B \ , \tag{3.12}$$

popisující změnu libovolné báze vektorů $B = \{u_1, ..., u_d\}$ vektorového prostoru V^d na jinou bázi vektorů $B' = \{u'_1, ..., u'_d\}$ v témže vektorovém prostoru. V publikaci [8] je proces změny vektorové báze označován pojmem „měnící se báze“ (basis changing). V rovnici 3.12 je výrazem $[v]_B$ označena matice původních souřadnic báze a výrazem $[v]_{B'}$ matice souřadnic nové báze. Jednoduchou úpravou rovnice 3.12 dostaneme vztah pro výpočet přechodové matice P ze znalostí souřadnic lokální báze L a globální báze G.

4 CHYBOVÉ FUNKCE

Přesnost algoritmů mapujících síťovou topologii na body v *n-rozměrném* souřadnicovém modelu je měřitelná pomocí tzv. chybových funkcí. Jediným kvalitativním parametrem, který je společný pro oba typy prostorů (reálný fyzický prostor a umělý souřadnicový), je hodnota zpoždění mezi stanicemi/body. Jak již bylo řečeno v předchozích kapitolách, zpoždění RTT změřené např. nástrojem ping[1] a mající jednotku času *ms*, by mělo přibližně odpovídat hodnotě vzdálenosti d vypočtené pomocí vzdálenostní funkce ze souřadnic bodů. I když o vypočtené vzdálenosti d říkáme, že představuje velikost zpoždění mezi dvěma uzly ve zkoumané síti, jedná se ve své podstatě o bezrozměrnou jednotkou.

4.1 Funkce chyby měření

Z výše uvedeného vyplývá, že přesnost algoritmů je určována výhradně na základě výsledku porovnání hodnoty změřeného a predikovaného zpoždění. Funkce, které jsou k tomuto účelu používány, označujeme obecným názvem chybové funkce nebo funkce chyby měření. Chybových funkcí existuje několik a liší se způsobem interpretace poměru změřeného a predikovaného zpoždění. Např. obyčejná funkce poměru dvou hodnot

$$f_R(pz, zz) = \frac{pz}{zz} \ , \tag{4.1}$$

kde pz je predikované zpoždění a zz je reálné změřené zpoždění v ms, vyjadřuje poměr vypočteného zpoždění k reálně změřenému. V ideálním případě by měla hodnota funkce oscilovat kolem hodnoty 1 nebo se jí i přímo rovnat. Jak již z poměru dvou hodnot vyplývá, pokud jsou si vstupní proměnné rovny, je hodnota funkce rovna jedné. Jinými slovy, pro tento konkrétní případ se podařilo ze souřadnic dvou uzlů predikovat reálné zpoždění se 100 % přesností. Hodnota funkce menší než jedna $f_R(pz, zz) < 1$ říká, že odhadnuté zpoždění je podhodnoceno. Naopak výsledná hodnota funkce větší než jedna $f_R(pz, zz) > 1$ naznačuje, že odhadnuté zpoždění je nadhodnoceno. V některých oblastech výzkumu může být skutečnost, jestli se jedná o hodnotu nadhodnocenou nebo naopak podhodnocenou důležitá a zásadně ovlivňující chování takového systému. V oblasti umělých souřadnicových systémů má tato informace nulový význam. Nezáleží na tom, jestli jsme predikované zpoždění nadhodnotili nebo naopak podhodnotili, v obou případech se jedná o odchylku od reálně existujícího zpoždění, které chceme co s největší přesností postihnout.

[1] Nástroj ping v prostředí OS Unix <`http://linux.die.net/man/8/ping`>.

Další skupinou funkcí vyhodnocující přesnost algoritmů jsou tzv. funkce relativní chyby. Výpočet přesnosti je zde realizován rozdílem změřeného a vypočteného zpoždění a to celé poděleno změřeným zpožděním.

$$f_{RE}(pz, zz) = \frac{|zz - pz|}{zz} \ . \tag{4.2}$$

Proměnné pz a zz zde mají stejný význam jako u funkce 4.1. V tomto případě by měla hodnota funkce oscilovat v blízkosti hodnoty 0. Hodnota funkce rovna nule značí 100 % přesnost predikce. Tyto dvě základní funkce nám poskytují odlišný pohled na přesnost námi zkoumaného algoritmu. Obecně můžeme říci, že dobře navržený algoritmus by měl vykazovat dobrou míru přesnosti predikce zpoždění bez ohledu na použitou chybovou funkci. Další z řady chybových funkcí, které jsou také použity v praktické části diplomové práce při zkoumání algoritmu Lighthouses, je funkce směrované relativní chyby

$$f_{DE}(pz, zz) = \frac{pz - zz}{min\,(zz, pz)} \ . \tag{4.3}$$

Jak již z letmého porovnání této funkce s funkcí relativní chyby vyplývá, jde o funkci vycházející z myšlenky funkce relativní chyby. Také v tomto případě platí, že hodnota funkce v nule značí maximální možnou přesnost predikce. Rozdíl oproti předchozí funkci nastává v situaci, kdy hodnota funkce je rovna jedné. Pak můžeme říci, že predikované zpoždění je dvojnásobně větší oproti reálnému zpoždění. Obdobně můžeme podobný výrok pronést i pro situaci, kdy je hodnota funkce rovna mínus jedné. V tomto případě je predikované zpoždění dvojnásobně menší oproti reálnému zpoždění. Poslední funkcí, kterou zde zmíním, je funkce normalizované chyby měření

$$f_{NE}(pz, zz) = \left(\frac{zz - pz}{zz}\right)^2 \ . \tag{4.4}$$

Tato chybová funkce je nejčastěji používána při vyhodnocování celkové přesnosti mapování do souřadnicového systému. Je to z toho důvodu, že z výše uvedených funkcí poskytuje nejobjektivnější posouzení přesnosti predikce. Při výpočtu chyby predikce zpoždění bere v úvahu skutečnost, že jedna jednotka chyby v případě velmi malé vzdálenosti není úměrná jedné jednotce chyby u velké vzdálenosti. Díky tomuto přístupu nedochází ke zkreslování, ať už ve prospěch či neprospěch, přesnosti fungování algoritmu.

4.2 Funkce minimalizované algoritmem Simplex Downhill

Abych mohl algoritmus Lighthouses použít při výpočtu souřadnic uzlů, musím nejprve vhodnou metodou inicializovat počáteční souřadnicový systém. Ten bude sloužit jako výchozí stav, od kterého se budou odvíjet všechny následující výpočty a operace související s transformací reálné síťové topologie do umělého souřadnicového systému. V kapitole 3.3.1 je uvedena obecně platná podmínka, jenž platí pro většinu algoritmů zabývajících se touto problematikou. A to sice minimální počet orientačních uzlů v závislosti na dimenzi souřadnicového systému, potřebných pro spolehlivé fungování použitého algoritmu.

Bylo navrženo několik postupů, které dokáží z matice vzájemných měření zpoždění mezi budoucími počátečními uzly vypočítat jejich souřadnice. Mezi nejposlednější studie patří metoda Simulace velkého třesku [15], simulující explozi částic/uzlů systému zapříčiněnou silou odvozenou od celkové chyby souřadnicového systému. Algoritmy vývojově nejstarší GNP [5], PIC [7], Lighthouses [8], které prakticky stály u zrodu této oblasti zkoumání, používají pro inicializaci souřadnicového systému metodu minimalizace některé z chybových funkcí např. algoritmem Simplex Downhill [16]. Inicializace počátečního souřadnicového systému v souvislosti s algoritmem Lighthouses je diskutována v kapitole 5.1.1. Funkce minimalizované algoritmem Simplex Downhill jsou obecně označovány jako účelové funkce. Základní operací, jenž je v těchto funkcích prováděna, je matematické porovnání zpoždění reálného ku zpoždění vypočítaného ze současných souřadnic uzlu v *n-rozměrném* prostoru. Jelikož je celá metoda postavena na minimalizaci chyby účelové funkce, neboli na hodnotě této funkce v *k-té* iteraci, je způsob jakým funkce vypočítává chybu velmi důležitý. Přesnost umístění počátečních pozičních serverů a s tím související přesnost celého systému je tak přímo závislá na těchto funkcích. Ve studiích zabývajících se touto problematikou byly prozkoumány následující tři typy účelových funkcí. Základní a svým principem nejjednodušší funkcí je funkce kvadratické odchylky

$$\varepsilon\left(d_{H_1H_2}, \hat{d}^S_{H_1H_2}\right) = \left(d_{H_1H_2} - \hat{d}^S_{H_1H_2}\right)^2 \ , \tag{4.5}$$

kde $d_{H_1H_2}$ je hodnota reálně změřeného zpoždění a $\hat{d}^S_{H_1H_2}$ je hodnota predikovaného zpoždění v aktuální iteraci procesu minimalizace mezi stanicemi H_1 a H_2. Druhou funkcí je funkce normalizované chyby měření. Tato funkce 4.4 byla již představena v předchozí kapitole 4.1 a v nezměněné podobě je i používána ve spojení s algoritmem Simplex Downhill. Třetí a poslední funkce vychází z funkce kvadratické odchylky, ale před rozdílem predikovaného a změřeného zpoždění provádí výpočet hodnoty

dekadického logaritmu s parametrem zpoždění. Pro tuto funkci bylo zavedeno pojmenování funkce logaritmického měření,

$$\varepsilon\left(d_{H_1H_2}, \hat{d}^S_{H_1H_2}\right) = \left(\log(d_{H_1H_2}) - \log(\hat{d}^S_{H_1H_2})\right)^2 \ . \tag{4.6}$$

V publikaci [17] byly výše uvedené účelové funkce podrobeny podrobnému zkoumání. Cílem zkoumání bylo ověřit, jak přesně a jak objektivně je ta která funkce schopna vyjádřit chybu měření. Z výzkumu dopadla nejhůře funkce kvadratické odchylky 4.5. Důvodem je neschopnost funkce rozlišit jednotku chyby v případě velmi krátké vzdálenosti a od jednotky chyby v případě naopak velmi velké vzdálenosti. Může tak docházet ke zkreslení chyby měření a vynucení si dalšího cyklu při hledání minima funkce nebo naopak k předčasnému ukončení minimalizační funkce. V obou případech nejsou nalezené souřadnice pozičních serverů dostatečně přesné a mají za následek další kumulaci chyby v souřadnicovém systému. V praktické části této diplomové práce jsem provedl vlastní prozkoumání prvních dvou účelových funkcí. Výsledek vlastního zkoumání se nijak neodchyluje od závěrů uvedených v publikaci [17]. Při použití funkce normalizované chyby měření je dosahováno vyšší přesnosti při pozicování stanic do souřadnicového systému a tedy i vyšší přesnosti při predikci zpoždění. Při simulaci algoritmu Lighthouses byla vždy výhradně použita účelová funkce normalizované chyby měření 4.4.

5 SIMULACE ČINNOSTI ALGORITMU LIGHTHOUSES

Za účelem praktického ověření vlastností algoritmu Lighthouses, jsem v rámci vypracování diplomové práce vyvinul CLI[1] aplikaci, prostřednictvím které simuluji procesy algoritmu na umělém modelu sítě. Aplikace provádí mapování síťových uzlů do *n-dimenzionálního* vektorového prostoru a vypočítává vzdálenosti mezi jednotlivými uzly. Následně tato data porovnává se změřenými hodnotami zpoždění. Neomezené zvyšování dimenze vektorového prostoru není možné. Teoreticky by sice aplikace byla schopna vypočítat souřadnice v prostoru o 10 a více rozměrech, ale se zvyšujícím se počtem rozměrů exponenciálně roste výpočetní náročnost a po překročení určitého počtu dimenzí především nepřesnost při výpočtu souřadnic uzlů. Tento vývoj přesnosti algoritmu může být zapříčiněn buď nevhodnou implementací, případně nedostatečně efektivní, některého z pomocných algoritmů a/nebo exponenciálně se kumulující chybou dílčích nepřesností při výpočtu souřadnic uzlů. Při zvyšujícím se počtu rozměrů pochopitelně roste i počet orientačních serverů, které je zapotřebí použít při výpočtu souřadnic.

Obecně při praktickém používání algoritmů predikujících zpoždění platí pravidlo o vyváženém nastavení poměru mezi přesností výpočtu zpoždění a výpočetní náročností. Nastavení, při kterém jsme sice schopni vypočítat zpoždění s vysokou přesností, ale za cenu podstatně delšího výpočetního času, není v praxi příliš vhodné. Navíc s rostoucí výpočetní složitostí rostou také nároky na hardwarové vybavení počítače. Z výše uvedených důvodů jsem v aplikaci omezil maximální počet dimenzí na 10. Z informací získaných v průběhu bádání vyplývá, že pro dostatečně přesné ověření přesnosti algoritmu plně postačuje simulace jeho procesů v rozsahu $2 - 10$ rozměrů souřadnicového systému.

Celý simulační program je řízen konfiguračním souborem, kde je možné definovat řadu parametrů a ovlivňovat tak chování jednotlivých algoritmů nebo např. určit, pomocí které chybové funkce, viz kapitola 4, bude zjišťována přesnost simulace. Uživateli je samozřejmě umožněna volba počtu rozměrů použitých při simulaci a také počet stanic, jenž budou během simulace transformovány na body v souřadnicovém modelu sítě. Simulační program je vnitřně rozdělen do dvou logických částí. V první části programu je nejprve inicializován základní souřadnicový systém potřebný pro vlastní simulaci algoritmu Lighthouses. V druhé části programu jsou již vypočítávány souřadnice stanic za použití tohoto algoritmu. Procesy při výpočtu souřadnic nejsou nijak modifikovány (zjednodušovány) nebo nějakým způsobem optimalizovány. Veškeré výpočty jsou přesnou algoritmizací postupů uvedených ve studii [8] a

[1] Command Line Interface (aplikace komunikující s uživatelem přes příkazovou řádku)

v dokumentu [20].

Vstupní data potřebná pro realizaci jednoho cyklu simulace jsou vždy při spuštění programu náhodně vygenerována z omezené množiny prvků. Generovanými daty jsou vzájemné hodnoty zpoždění a počáteční odhad souřadnic v případě části inicializující souřadnicový systém a hodnoty zpoždění mezi přistupující stanicí a jí zvoleným referenčním serverem. Krajní hodnoty omezující množinu výběru jsou rovněž konfigurovatelné prostřednictvím hlavního konfiguračního souboru programu.

Výstup z programu je dvojího typu. Na příkazovou řádku, odkud byl program spuštěn, jsou průběžně vypisovány nejrůznější informace z běhu programu. Uživatel má tak možnost po skončení simulace tato data projít a přesvědčit se, jestli nedošlo za běhu programu k nějakému nestandardnímu chování nebo k neopravitelné chybě v některém z algoritmů. Výstupní data také názorně zobrazují vývoj simulace, její jednotlivé kroky, jakým způsobem na sebe jednotlivé části algoritmu navazují, která data jsou pro ten který proces důležitá a co je s konkrétními daty v konkrétním procesu prováděno. Druhým typem výstupu je grafický výstup. Program na základě konfiguračního souboru po skončení simulace automaticky vypočte chyby predikce zpoždění z celého vzorku nasimulovaných stanic. Výsledky jsou poté vytištěny do přehledných grafů, křivkou znázorňujících dosaženou přesnost simulace. Protože je simulační program rozdělen do dvou logických celků, jsou i grafy vyjadřující dosaženou přesnost vytvářeny zvlášť pro každou z částí. Po skončení simulace najde uživatel v adresáři pro ukládání výstupů simulace dva grafy (opět je možno toto chování ovlivnit nastavením programu). Jeden vyjadřující dosaženou přesnost simulace pro počáteční souřadnicový systém a druhý vyjadřující dosaženou přesnost algoritmu Lighthouses.

5.1 Popis simulačního programu

Jak jsem již uvedl v předchozí kapitole, vyvinutý simulační program je program spouštěný z příkazové řádky a neobsahující grafické uživatelské rozhraní - GUI. Pro vlastní simulaci zkoumaného algoritmu není pochopitelně grafické rozhraní důležité. Výstupy v podobě grafů jsou ukládány do definovaného adresáře, případně vlastní data z průběhu simulace se zobrazují přímo na příkazovou řádku. Po skončení simulačního procesu je tak okamžitě možné se z nabízených výstupů přesvědčit o výsledku simulace. Ovládání programu je podle mého názoru jednoduché a to prostřednictvím modifikace předpřipraveného konfiguračního souboru, doplněného o vysvětlující komentáře u jednotlivých parametrů. Z tohoto důvodu jsem původně zamýšlenou grafickou nadstavbu programu nerealizoval a raději jsem věnoval ušetřený čas kompletní implementaci algoritmu Lighthouses a implementaci chybových

funkcí. Při vývoji programu byl použit programovací jazyk python[2], konkrétně jeho dvojková řada ve verzi 2.5.5[3]. Proč jsem si zvolil právě tento programovací jazyk? Python má několik předností, které usnadňují používání aplikací v něm napsaných i samotný vývoj aplikací. Především je multiplatformní. Také je možné kód v pythonu jednoduše spojit s webovou prezentací. Ve všech těchto případech je tak teoreticky možné používat vyvinutý simulační program. V neposlední řadě bylo pochopitelně mé rozhodnutí ovlivněno vlastní zkušeností s tímto jazykem.

5.1.1 Inicializace počátečních pozičních serverů

Simulační program je vnitřně rozdělen do dvou logických částí. První část programu je zaměřena na proces inicializace počátečního souřadnicového modelu sítě. V této části programu jsou z vygenerované matice vzájemných zpoždění mezi počátečními pozičními servery a z matice odhadu jejich souřadnic (souřadnice serverů jsou náhodně volené) pomocí algoritmu Simplex Downhill vypočítány souřadnice počátečních pozičních serverů. Algoritmus Simplex Downhill [16] představuje metodu minimalizace chyby *n-dimenzionálního* prostoru. Myšlenka použití tohoto algoritmu pro výpočet souřadnic a postup jakým jsou souřadnice vypočítány je převzat z projektu GNP. Proto pro podrobnější popis odkazuji na kapitolu 3.1, kde se problematice blíže věnuji v souvislosti s již zmíněným projektem GNP.

Význam inicializace počátečního souřadnicového systému spočívá v přípravě prostředí, vhodného pro běh vlastního algoritmu Lighthouses. Přípravou prostředí mám na mysli umístění potřebného počtu pozičních serverů do systému. Minimální počet pozičních serverů je určen, jak již bylo v této práci mnohokrát zmíněno, výrazem $d+1$, kde d je velikost souřadnicového systému použitého při běhu simulace. Pokud by při spuštění simulace nebyl k dispozici potřebný počet pozičních serverů, nebyl by implementovaný algoritmus v takovém případě schopen vypočítat souřadnice serverů s akceptovatelnou chybou. Ve studii [8] je sice situace, kdy počet pozičních serverů v systému je nižší nebo roven dimenzi souřadnicového systému, popsána krátkým odstavcem hovořícím o pojmu „první stanice“. Následována větou o nutnosti v takovém případě vypočítat souřadnice pomocí pozičních serverů, které jsou již připojeny, viz kapitola 3.3.1. Při implementaci algoritmu Lighthouses se však ukázalo toto zdánlivě jednoduché tvrzení bez doplňujících informací neřešitelné. Z tohoto důvodu jsem se nechal inspirovat publikací [20], kde je použita metoda minimalizace celkové chyby *n-dimenzionálního* prostoru v souvislosti s algoritmem Lighthouses. Tímto způsobem jsem schopen připravit vhodné simulační prostředí pro libovolný souřadnicový systém.

[2] `http://www.python.org/`

[3] `http://docs.python.org/release/2.5.4/`

V konfiguračním souboru se této části týkají následující parametry:

```
## Initial Basis Coordinates
ib_do_error = 0 # vypocitat chybu predikce [0|1] [true|false]
iberr = 'error_relative' # reletivni chyba
# alternativy:
# iberr = 'error_normalized_measure' # normalizovana chyba
# iberr = 'both' # pouzit obe chybove funkce
## Simplex Downhill parameters
rtt_min = 1 # min hodnota rtt
rtt_max = 1000 # max hodnota rtt
cmin = 0 # minimalni hodnota pocatecni souradnice
cmax = 10000 # maximalni hodnota pocatecni souradnice
maxiter = 20000 # max pocet iteraci / Simplex Downhill
maxfun = 20000 # max pocet vykonani chybove funkce / Simplex
    Downhill
disp = True # Simplex Downhill - zprava o vysledku minimalizace
    funkce
# chybova funkce pouzita v algoritmu Simplex Downhill
errorf = 'error_simple_squared' # chyba druhe mocniny
    vzdalenosti
# alternativy:
# errorf = 'error_normalized_measure' # normalizovana chyba
    mereni
```

Parametrem `ib_do_error` je možné aktivovat nebo naopak vypnout možnost vytváření grafického výstupu ze simulace počátečního souřadnicového systému. U většiny simulací je doporučeno nechat tuto volbu zapnutou. Představa o dosažené přesnosti umístění počátečních pozičních serverů dobře napomáhá při interpretaci grafů týkajících se pouze algoritmu Lighthouses. Pokud se nám podařilo dobře umístit poziční servery v této části simulace, je dobrý předpoklad pro dosažení vysoké přesnosti u algoritmu Lighthouses a naopak.

Parametr `iberr` určuje, která z chybových funkcí bude použita při vypočtu chyby predikce zpoždění mezi počátečními pozičními servery. V simulačním programu jsou implementovány dvě hlavní chyby měření, relativní chyba a normalizovaná chyba měření. Výchozí a doporučenou volbou je použití normalizované chyby měření, viz vysvětlení v kapitole 4.1.

Proměnné `rtt_min` a `rtt_max` nastavují číselný rozsah s plovoucí desetinnou čárkou, ze kterého budou pseudonáhodně generované hodnoty vzájemného zpoždění mezi počátečními pozičními servery.

`cmin` a `cmax` nastavují číselný rozsah s plovoucí desetinnou čárkou, ze kterého

jsou opět pseudonáhodně generovány počáteční souřadnice pozičních serverů. Takto získané souřadnice slouží pouze jako počáteční odhad (startovací bod), od kterého metoda Simplex Downhill zahájí hledání přesných souřadnic na základě minimalizace zvolené chybové funkce. Z tohoto důvodu je možné tyto souřadnice generovat zcela náhodně, bez jakýchkoli vzájemných vztahů mezi nimi.

Parametry `maxiter` a `maxfun` uvádím společně, protože i když každý z nich řídí trochu jinou část algoritmu Simplex Downhill, mají obecný význam společný. Parametr `maxiter` nastavuje maximální počet iterací algoritmu Simplex Downhill, kterého je možno dosáhnout při hledání minima funkce. Pokud minima není dosaženo do počtu iterací maximálně `maxiter`, je další hledání minima funkce ukončeno. Algoritmus vrátí poslední hodnotu souřadnic získaných při `maxiter` integraci. Pokud dojde k situaci vynuceného ukončení algoritmu, nelze takto získané souřadnice považovat za přesné souřadnice pozičních serverů. Výsledek simulace algoritmu Lighthouses může být za těchto okolností značně zkreslen a tudíž statisticky naprosto bezcenný. Pokud se s problémem opakovaného přerušení při hledání minima potkáme opakovaně, může být řešením nastavení většího počtu iterací. Zde musím upozornit na skutečnost, že větší počet iterací které je zapotřebí vykonat, je přímo úměrné výpočetní náročnosti při řešení takového problému. Parametr `maxfun` nastavuje maximální počet iterací vykonání chybové funkce. I v tomto případě překročení nastaveného počtu iterací vede k vynucenému ukončení algoritmu bez ohledu na to, jestli bylo minimum chybové funkce nalezeno či nikoli.

Typ použité chybové funkce při hledání globálního minima *n-dimenzionálního* prostoru je možné nastavit pomocí parametru `errorf`. Implementovanými chybovými funkcemi přizpůsobenými pro běh s algoritmem Simplex Downhill jsou chyba kvadratické odchylky a normalizovaná chyba měření (výchozí a doporučená). Pro detailní popis chybových funkcí a zdůvodnění vhodnosti použití normalizované chyby měření, opět odkážu na kapitolu 4.

Posledním uživatelsky nastavitelným parametrem v této části programu je parametr `disp`. Jeho význam je jednoduchý a sice určuje, jestli bude zobrazena zpráva o výsledku běhu algoritmu Simplex Downhill. Nejdůležitější informací z celé zprávy je údaj oznamující úspěch nebo naopak neúspěch při hledání minima funkce. Druhým důležitým údajem je hodnota chybové funkce v okamžiku ukončení algoritmu. Hodnotu funkce můžeme interpretovat jako celkovou chybu predikce zpoždění mezi všemi počátečními pozičními servery. Statisticky zajímavými údaji ve zprávě jsou počet iterací potřebných pro dosažení minima funkce a počet vykonání chybové funkce.

5.1.2 Výpočet souřadnic algoritmem Lighthouses

Druhá část programu je již plně zaměřena na algoritmus Lighthouses a realizuje jeho simulaci podle výpočetních postupů popsaných ve studii [8]. Teoretickému popisu algoritmu je věnována kapitola 3.3. Po inicializaci počátečních pozičních serverů v první části programu, již nic nebrání tomu, zahájit výpočet vlastních souřadnic požadovaného počtu stanic.

V konfiguračním souboru se algoritmu Lighthouses týkají následující tři parametry:

```
## Lighthouses Coordinates
number_of_lighthouses = 19 # pocet stanic, ktere budou pridany
    [0|libovolne cele cislo]
lh_do_error = 0 # vypocitat chybu predikce [0|1] [true|false]
lherr = 'error_normalized_measure' # normalizovana chyba
# alternativy:
# lherr = 'error_relative' # reletivni chyba
# lherr = 'both' # relativni + normalizovana chyba
# lherr = 'error_directional' # smerova relativni chyba
# lherr = 'ratio' # pomer odhadnute ku zmerene hodnote RTT
```

Nejdůležitějším parametrem je parametr `number_of_lighthouses`, pomocí něhož uživatel nastavuje požadovaný počet stanic, které budou do souřadnicového modelu sítě postupně přidány. Počet stanic je určen celým kladným číslem různým od nuly. Pokud bychom chtěli provést pouze simulaci výpočtu počáteční báze souřadnicového systému, musí být hodnota tohoto parametru 0. V takovém případě není do systému přidána žádná stanice. Maximální počet stanic není nikterak z hora omezen. Pochopitelně i zde platí pravidlo, že s rostoucím počtem stanic, se úměrně prodlužuje i čas potřebný k výpočtu takového modelu.

Parametr `lh_do_error`, stejně jako v případě konfigurace počátečního souřadnicového systému, zapíná nebo naopak vypíná funkci výpočtu chyby predikce zpoždění. Pokud je hodnota parametru nastavena na False, nejsou generovány žádné výstupy z této části programu.

Parametr `lherr` definuje chybovou funkci 4, pomocí které bude vypočtena chyba predikce mezi stanicí v souřadnicovém modelu sítě a jejím referenčním pozičním serverem.

5.1.3 Globální parametry simulačního programu

Vyjma parametrů zmíněných a vysvětlených v kapitolách 5.1.1 a 5.1.2, existují ještě další uživatelsky nastavitelné parametry, ovlivňující globální chování simulačního programu. Globální chování programu je tak možné ovlivnit pomocí následujících parametrů:

```
1 ## General
2 dimension = 3 # dimenze souradnicoveho systemu
3 rd_rtt = 2 # zaokrouhleni pri generovani rtt
4 rd_cd = 4 # zaokrouhleni pro vypocty souradnic
5 ## Graphs
6 gdir = '/home/honzas/NetBeansProjects/Lighthouses/grafy' #
      hlavni adresar pro umisteni grafy
7 gfile_iberr = 'iberr' # jmeno souboru grafu pro Initial Basis
8 gfile_lherr = 'lherr' # jmeno souboru grafu pro Lighthouses
    Coordinates
```

Nejdůležitějším parametrem z výše uvedených je parametr `dimension`. Hodnotou tohoto parametru je určen počet prostorů souřadnicového modelu sítě. Jelikož není prakticky ani teoreticky možné mít počáteční souřadnicový systém realizovaný v euklidovském prostoru o jiném počtu rozměrů než pro jaký jsou pak vypočítávány souřadnice stanic, je tento parametr společný pro obě části programu. Hodnotou parametru `dimension` je celé kladné číslo d, pro které platí, $2 \leq d \leq 10$. Důvod, proč je maximální počet prostorů omezen deseti, je vysvětlen v kapitole 5.

Parametry `rd_rtt` a `rd_cd` nastavují počet desetinných míst, použitých při zaokrouhlování výsledků. Konkrétně parametr `rd_rtt` řídí zaokrouhlování při generování náhodných hodnot RTT, a to jak v případě matice vzájemných hodnot RTT mezi všemi počátečními pozičními servery, tak při generování doby zpoždění mezi přistupující stanicí a jejím referenčním pozičním serverem. Parametr `rd_cd` řídí finální zaokrouhlení vypočítaných souřadnic. Opět se zaokrouhlení týká jak souřadnic počátečních pozičních serverů, tak i souřadnic stanic.

Následující parametry se už týkají pouze grafického výstupu z aplikace. Parametr `gdir` nastavuje hlavní adresář, do něhož budou ukládány grafy zobrazující křivku přesnosti výpočtů, popřípadě jiné statisticky zajímavé informace. Parametry `gfile_iberr` a `gfile_lherr` definují jméno souboru, do něhož bude uložen grafický výstup z části vypočítávající souřadnice počátečních pozičních serverů, respektive části vypočítávající souřadnice stanic algoritmem Lighthouses.

5.1.4 Implementace algoritmu pro výpočet lokální báze

Algoritmus pro výpočet lokální báze, teoreticky popsaný v kapitole 3.3.2, je v simulačním programu implementován ve funkci `local_basis_coordinates()`. Zvolený způsob implementace vysvětlím s pomocí ukázky ze zdrojového textu funkce (čísla řádků nejsou součástí zdrojového textu):

```
1  def local_basis_coordinates(lighthouses_coordinates):
2      """
3      Vypocet lokalni baze stanice.
4      """
5      local_basis = matrix.Matrix()
6      vector_basis = matrix.Matrix()
7      for i in range(1, len(lighthouses_coordinates)):
8          coordinates = souradnice_vektoru(lighthouses_coordinates
               [0], lighthouses_coordinates[i])
9          vector_basis.add_row(coordinates)
10
11     print 'baze vektoru:', vector_basis.get_rows()
12     determinant = round(vector_basis.determinant(), 4)
13     print 'determinant:', determinant
14
15     # pokud je determinant nulovy nema vyznam dale pokracovat
16     if not determinant: return
17
18     def null(x): return 0
19
20     local_basis.add_row(vector_basis.get_row(0))
21     for i in range(1, vector_basis.row_count()):
22         vproj_sum = [null(x) for x in range(vector_basis.
               row_count())]
23         for j in range(i):
24             vproj = ortogonalni_projekce(vector_basis.get_row(i)
                   , local_basis.get_row(j))
25             vproj_sum = soucet_vektoru(vproj, vproj_sum)
26
27         pvector = perpendicular_vector(vproj_sum)
28         sum = soucet_vektoru(vproj_sum, pvector)
29         local_basis.add_row(sum)
30
31     return local_basis
```

Funkce `local_basis_coordinates()` přijímá jediný parametr, a to dvojrozměrné pole obsahující souřadnice zvolených pozičních serverů stanice, pro kterou právě vypočítáváme její souřadnice. Na řádku 3 a 4 jsou inicializovány datové struktury `local_basis` a `vector_basis` nutné pro vlastní výpočet lokální báze. Obě datové struktury jsou instancemi třídy `Matrix`. Třídu `Matrix` jsem implementoval za účelem unifikované práce s matematickým objektem matice. Cyklus na řádcích 7 až 9 vypočítává vektory ze souřadnic zvolených pozičních serverů, pomocí nichž budou vypočítány souřadnice lokální báze L. Kontrola, zda vypočítané vektory jsou vzájemně ortogonální, viz kapitola 3.3.2, a má tak smysl dále pokračovat ve výpočtu souřadnic lokální báze, je realizována na řádku 12 s následným podmínečným ukončením funkce na řádku 16. Kontrola ortogonality spočívá ve výpočtu determinantu matice, jejíž řádky jsou tvořeny z výše vypočtených vektorů. Ve výpočtu souřadnic lokální báze je pokračováno pouze za předpokladu, že vypočtený determinant je různý od nuly.

Operace dosud popisované mají společně za úkol předpřipravit požadované datové struktury před vlastním výpočtem souřadnic lokální báze. Výpočet souřadnic lokální báze je realizován na řádcích 20 až 29. Na těchto řádcích je implementován Gram-Schmidtův algoritmus popsaný v kapitole 3.3.2 a v dokumentu [19]. Postup výpočtu se skládá z několika dílčích částí. Vektor v prvním řádku maticové datové struktury `vector_basis` je také prvním vektorem budoucích souřadnic lokální báze. S ostatními vektory v `vector_basis` jsou vždy provedeny následující operace. Nejprve je vypočtena ortogonální projekce vektoru l_i na podprostor W_i. Následně je k ortogonálně projektovanému vektoru vypočten jeho doplněk kolmý na podprostor W_i. Součtem těchto dvou dílčích složek dostáváme výsledný vektor lokální báze l_i.

5.1.5 Implementace algoritmu pro výpočet souřadnic stanice

Vypočtené souřadnice lokální báze pomocí funkce `local_basis_coordinates()` přijímá na svém vstupu jako jeden ze dvou povinných parametrů druhá klíčová funkce simulačního programu `host_coordinates()`. Jak už je z názvu funkce patrné, úkolem této funkce je vypočítat souřadnice stanice v rámci její lokální báze. Druhým povinným parametrem funkce je změřená hodnota RTT mezi připojující se stanicí a jejím referenčním pozičním serverem.

```
1  def host_coordinates(local_basis, rtt):
2      """
3      Vypocet souradnic stanice v ramci lokalni bazi.
4      """
5      dimension = local_basis.row_count() # dimenze = pocet
           vektoru lokalni baze
6      ni = local_basis.get_row(0)
7      for vector in range(1, dimension):
8          ni = soucet_vektoru(ni, local_basis.get_row(vector))
9
10     # sestaveni systemu linearnich rovnic
11     linear_equations = matrix.Matrix()
12     dot_product = skalarni_soucin(local_basis.get_row(0),
           local_basis.get_row(-1))
13     for i in range(dimension):
14         linear_equation = []
15         for j in range(dimension):
16             if i+j != dimension -1:
17                 linear_equation.append(euklidovska_norma(
                       local_basis.get_row(j)))
18             else:
19                 en = euklidovska_norma(local_basis.get_row(j))
20                 linear_equation.append(en * cos(dot_product))
21         else:
22             ni_lk = skalarni_soucin(ni, local_basis.get_row(i))
23             linear_equation.append(rtt * cos(ni_lk))
24         linear_equations.add_row(linear_equation)
25
26     system_solution = rref(linear_equations) # reseni systemu
           linearnich rovnic
27     # pokud se nepodarilo system vyresit, nema smysl pokracovat
```

```
28      if not system_solution: return
29
30      c_coefficients = system_solution.get_column(-1)
31
32      ni = []
33      for i in range(dimension):
34           ni.append(nasobeni_vektoru_skalarem(local_basis.get_row
                (i), c_coefficients[i]))
35      host_coordinates = ni[0]
36      for v in range(1, dimension):
37          host_coordinates = soucet_vektoru(host_coordinates, ni[v
                ])
38
39      return host_coordinates
```

Souřadnice určující pozici stanice v lokální bázi jsou vypočteny jako lineární kombinace vektorů lokální báze skalárně vynásobených proměnnými c_i. Proměnné c_i získáme řešením následující soustavy lineárních rovnic. Na řádcích 14 až 16 je vypočítán vektor n_i představující počáteční odhad budoucích souřadnic stanice. Na řádku 19 je inicializována datová struktura reprezentující soustavu lineárních rovnic jako instance třídy `Matrix`. Vlastní sestavení soustavy lineárních rovnic se ve zdrojovém textu nachází na řádcích 13 až 24. Pro řešení soustavy rovnic používám v simulačním programu Gaussovu eliminační metodu aplikovanou na maticový zápis soustavy. Gaussova eliminační metoda je implementovaná ve funkci `reduced_row_echelon_form()`. Jelikož téma řešení soustav lineárních rovnic přímo nesouvisí se zadáním diplomové práce, nebudu dále tuto metodu rozebírat. Na řádku 26 řeším soustavu rovnic pomocí výše zmíněného algoritmu ve funkci `reduced_row_echelon_form()`. Podmínka na řádku 28 ověřuje, jestli bylo řešení soustavy rovnic nalezeno. V opačném případě není možné ve výpočtu souřadnic pokračovat a funkce je nuceně přerušena. Řešením soustavy rovnic jsou proměnné c_i, které získám z diagonálně upravené matice soustavy na řádku 30. Poslední část výpočtu souřadnic spočívá ve vynásobení každého vektoru l_i lokální báze L s příslušnou skalární proměnnou c_i a výpočet lineární kombinace vektorů l_i. Tato část algoritmu se nachází na řádcích 32 až 37. Podrobný teoretický popis algoritmu je opět možné nalézt v teoretické části diplomové práce, konkrétně v kapitole 3.3.3.

5.1.6 Popis hlavních tříd aplikace

Rozdělení simulačního programu do dvou logických částí (inicializace počátečního souřadnicového systému a vlastní výpočet souřadnic algoritmem Lighthouses) a potřeba současně udržovat řadu informací (atributů) u každého objektu v souřadnicovém modelu sítě, vedly k použití objektově orientovaného způsobu programování a návrhu základních stříd programu.

Inicializace počátečního souřadnicového systému a všechny potřebné metody s touto částí související jsou součástí třídy `InitialBasisCalculation`. Objekt odvozený z této třídy představuje počáteční souřadnicový systém a poskytuje celkem šest základních metod. Pro potřeby generování matice vzájemných hodnot zpoždění RTT mezi budoucími počátečními servery slouží metoda `gen_rtt_matrix()`. Druhou metodou, jenž předpřipravuje potřebná vstupní data před vlastním výpočtem souřadnic, je metoda `gen_initial_points()`. Úkolem této metody je generování náhodných souřadnic pozičních serverů, které jsou použity jako počáteční odhad pro algoritmus Simplex Downhill. Pokud jsou k dispozici požadovaná vstupní data (získaná prostřednictvím výše popsaných metod) může objekt třídy `InitialBasisCalculation` zavolat svoji nejdůležitější metodu `calculate_basis()`. Metoda `calculate_basis()` má za úkol pomocí algoritmu Simplex Downhill vypočítat souřadnice počátečních pozičních serverů. Výpočet chyby predikce zpoždění v rámci počátečního souřadnicového systému je implementován v metodě `compute_errib()`, kterou je možné zavolat kdykoli po výpočtu souřadnic. Poslední metodou, kterou je dle mého názoru dobré zmínit je `save_stats()`, umožňující uložení vypočítaných hodnot chyb predikce zpoždění a vytvoření grafického zpracování těchto dat.

Každý objekt v souřadnicovém modelu sítě (s výjimkou počátečních pozičních serverů) je instancí třídy `NewPointCalculation`. Objekt této třídy poskytuje všechny metody potřebné pro výpočet souřadnic stanice. Základní metodou, kterou je zapotřebí volat jako první, je metoda `choose_lighthouses()`. Jejím úkolem je náhodně vybrat požadovaný počet pozičních serverů z již existujících v souřadnicovém modelu. Nastavení hodnoty zpoždění RTT mezi připojující se stanicí a jejím referenčním pozičním serverem zajišťuje metoda `set_rtt()`. Po té co byly úspěšně volány výše popsané metody, může objekt zavolat hlavní metodu pro výpočet vlastních souřadnic `calculate_coordinate()`. V principu tato metoda postupně volá funkci pro výpočet souřadnic lokální báze 5.1.4 a funkci pro výpočet souřadnic stanice vztažených k její lokální bázi 5.1.5. Pokud stanice úspěšným voláním metody `calculate_coordinate()` získala souřadnice určující její pozici v souřadnicovém modelu sítě, může nyní pomocí metody `calculate_distance()` vypočítat zpoždění (predikovat zpoždění) mezi ní a jejím referenčním pozičním serverem. Predikce zpoždění je vypočtena jako vzdálenost dvou bodů v souřadnicovém systému.

Výpočet chyby mezi predikovaným zpožděním a reálně změřeným (nebo např. náhodně generovaným, jako v případě parametrů simulace v této diplomové práci) je realizován voláním metody `error_prediction()`.

Pro potřeby vhodné organizace instancí třídy `NewPointCalculation`, jenž představují reálné stanice v simulované síťové topologii, bylo zapotřebí efektivně implementovat datovou strukturu, ve které by byly uloženy odkazy na jednotlivé instance. K tomuto účelu slouží třída `Lighthouses`. Instance odvozená od této třídy udržuje informace o všech objektech typu `NewPointCalculation` a objektu typu `InitialBasisCalculation` počátečního souřadnicového systému. Díky tomu je možné voláním metody `get_lighthouse()` případně metody `add_lighthouse` získat odkaz na požadovaný poziční server nebo nově vytvořený poziční server do souřadnicového systému přidat. Pomocí metod objektu typu `Lighthouses` je tak možné pohodlně a bezpečně ovládat celý souřadnicový systém. Mezi metody zvyšující komfort při práci s takto reprezentovaným souřadnicovým systémem patří metoda `lighthouses_count()` vracející počet pozičních serverů v systému a metoda `get_lighthouses()` vracející všechny poziční servery včetně počátečních pozičních serverů jako jednorozměrné pole odkazů. Výpočet chyby predikce zpoždění mezi každou stanicí a jejím referenčním pozičním serverem je docíleno zavoláním metody `compute_errlh()`. O uložení statistických dat o simulovaném souřadnicovém modelu sítě se stará metoda `save_stats()`, která podobně jako v případě třídy `InitialBasisCalculation`, řídí výstup ze simulačního programu.

Na závěr se velmi krátce zmíním o třídě `Matrix`. Třídu `Matrix` jsem implementoval pro potřeby reprezentace maticové datové struktury a unifikovaného rozhraní pro práci s ní. Např. počáteční poziční servery jsou v objektu typu `InitialBasisCalculation` uloženy v této datové struktuře. Také některé výpočty prováděné při simulaci algoritmu Lighthouses jsou realizovány nad maticovou datovou strukturou.

5.2 Výsledek simulace algoritmu Lighthouses

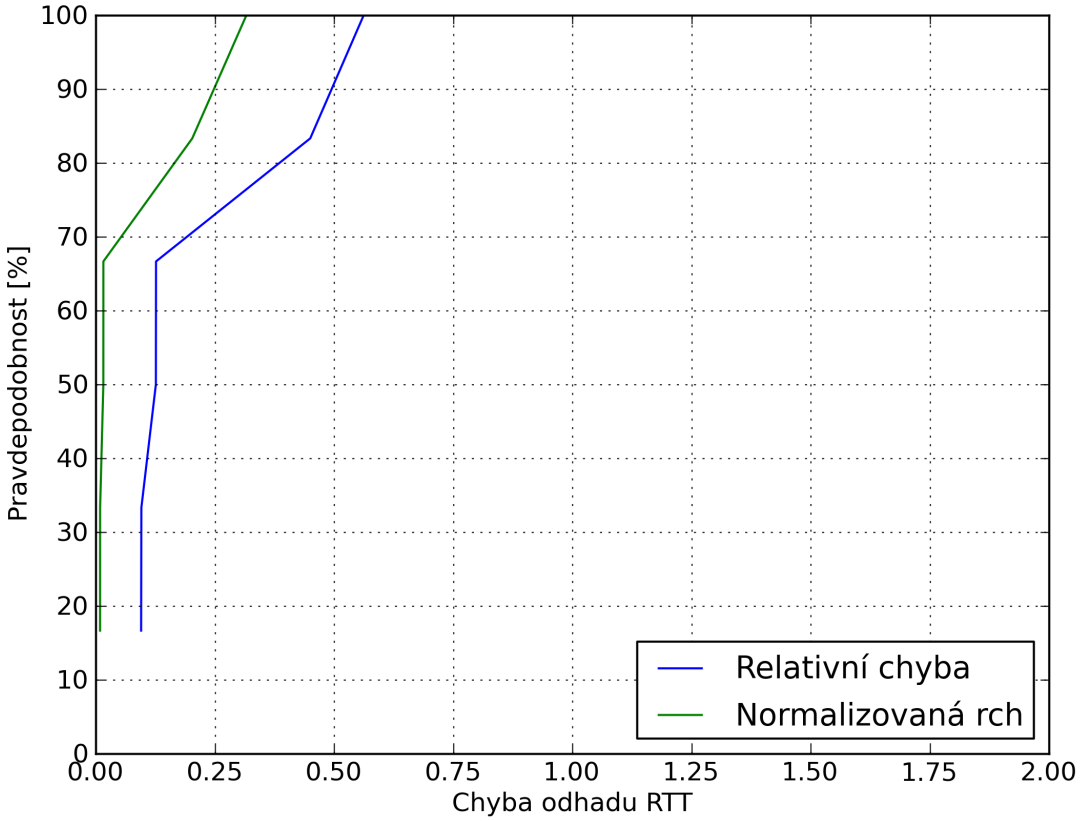
Při simulacích algoritmu Lighthouses provedených za účelem ověření přesnosti predikce zpoždění byla vždy použita data z uměle vytvořených sítí. Hodnoty zpoždění RTT mezi připojující se stanicí a jejím referenčním pozičním serverem nejsou tak hodnotami reálně změřenými ve zkoumané fyzické síťové topologii, ale jsou náhodně generována z předem definovaného rozsahu hodnot. Musím poznamenat, že zvolený postup při získávání hodnot zpoždění nijak nesnižuje přesnost nebo důvěryhodnost výsledků zde uvedených.

Přesnost algoritmu Lighthouses by se měla podle teoretických předpokladů zvyšovat v závislosti na zvyšujícím se počtu rozměrů souřadnicového modelu sítě, do kterého jsou mapovány fyzické stanice zkoumané sítě. Toto je důležité si uvědomit při porovnávání výsledků dosažených algoritmem Lighthouses a jiným algoritmem, taktéž mapujícím fyzickou síťovou topologii do souřadnicového systému, např. GNP [5]. Konkrétně u algoritmu GNP je teoretický předpoklad zvyšování přesnosti predikce zpoždění postaven na základě zvyšujícího se počtu fixních pozičních serverů, pomocí kterých jsou souřadnice vypočítávány. Zvyšováním počtu fixních pozičních serverů roste počet iterací, které je zapotřebí provést při minimalizace chyby některé z chybových funkcí algoritmem Simplex Downhill. Naopak zvyšováním počtu rozměrů souřadnicového systému úměrně roste výpočetní náročnost při výpočtu souřadnic a také zde můžeme pozorovat vliv kumulace chyby dílčích výpočtů.

Výsledky simulace algoritmu Lighthouses jsou rozděleny do dvou hlavních částí. První část je zaměřena na přesnost predikce zpoždění mezi všemi pozičními servery v počátečním souřadnicovém systému. V druhé části je analyzována přesnost predikce zpoždění mezi každou stanicí a jí zvoleným referenčním pozičním serverem. Rozdělení výsledků simulace tak odpovídá logickému rozdělení simulačního programu.

Na obrázku 5.1 je znázorněna přesnost predikce zpoždění mezi počátečními pozičními servery, jejichž souřadnice byly vypočítány algoritmem Simplex Downhill z matice vzájemných hodnot zpoždění RTT. Chyba predikce zpoždění je zde vypočítána pomocí dvou chybových funkcí, a to funkce relativní chyby a funkce normalizované relativní chyby. Pohledem na obrázek 5.1 můžeme porovnat vliv použité chybové funkce na výslednou přesnost predikce zpoždění. Teoretický předpoklad z kapitoly 4 je potvrzen výsledkem provedené simulace. Osa y v grafu vyjadřuje pravděpodobnost, s jakou je možné docílit chyby predikce zpoždění v rozsahu $< 0, E >$, kde E značí maximální hodnotu chyby v definovaném intervalu. Ve většině studií uvedených v seznamu použité literatury této diplomové práce, je maximálně přípustnou chybou predikce zpoždění akceptována hodnota 0,5. Chyby nad touto hodnotou jsou už příliš velké na to, aby mohly být použity v reálných podmínkách. Z obrázku 5.1 vidíme, že simulace počátečního souřadnicového systému dosáhla velmi dobrých vý-

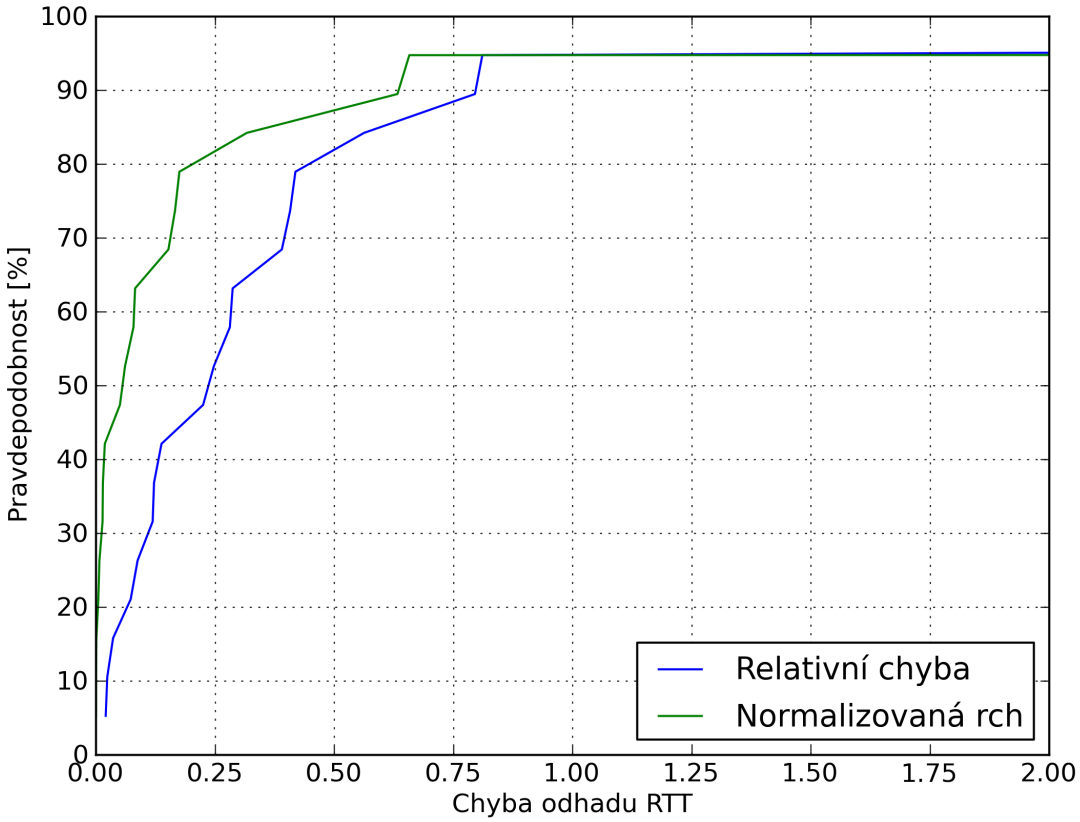
sledků, protože se podařilo predikovat zpoždění do maximální hodnoty chyby 0,5 v 90 % ze všech provedených výpočtů. Jak již bylo uvedeno, funkce normalizované relativní chyby vykazuje lepší hodnoty predikce zpoždění. Je to způsobeno tím, jak tato chybová funkce dokáže rozlišit mezi chybou v rámci velmi krátké vzdálenosti mezi stanicemi a chybou v rámci naopak velké vzdálenosti.



Obrázek 5.1: Chyba predikce zpoždění mezi počátečními pozičními servery v 3D prostoru.

Přesnost predikce zpoždění mezi stanicemi v souřadnicovém modelu sítě a jejich referenčními pozičními servery je zobrazena na obrázku 5.2. Při výpočtu souřadnic stanic byl použit algoritmus Lighthouses. Opět jsem pro výpočet chyby predikce zpoždění použil dvě základní chybové funkce a vzájemně je porovnal. Podle očekávání funkce normalizované relativní chyby vykazuje lepší výsledky algoritmu Lighthouses při výpočtu souřadnic. Výpočet souřadnic stanic byl v tomto případě realizován ve tří rozměrném souřadnicovém systému. Pokud vezmeme jako mezní akceptovatelnou chybu predikce hodnotu chyby 0,5, tak z grafu vidíme, že v případě výsledku normalizované relativní chyby dokázal algoritmus Lighthouses vypočítat

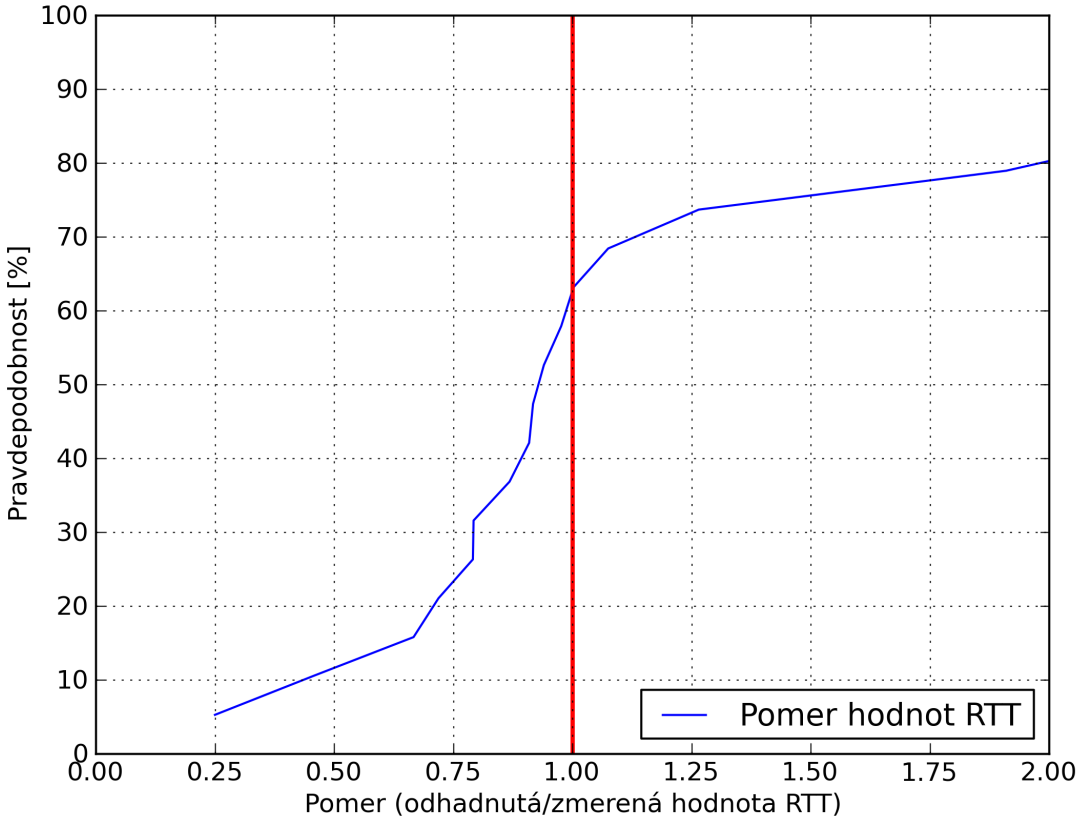
zpoždění mezi stanicemi do maximální chyby 0,5 v přibližně 87 % ze všech provedených výpočtů. Ani výsledek získaný na základě výpočtu relativní chyby nedopadl nejhůře. V 82 % ze všech provedených výpočtů zpoždění bylo dosaženo přesnosti do hodnoty chyby 0,5.



Obrázek 5.2: Chyba predikce zpoždění mezi stanicemi v 3D prostoru.

Chybová funkce vypočítávající pouze poměr mezi predikovaným a změřeným zpožděním je zobrazena na obrázku 5.3. Výpočet souřadnic stanic byl opět realizován v 3D souřadnicovém systému. Jak matematicky vyplývá z poměru dvou hodnot, pokud jsou si obě hodnoty rovny nebo se sobě velikostně blíží, je výsledek jejich poměru roven jedné nebo je blízký jedné. Na obrázku 5.3 je červenou čarou vyznačena nejideálnější situace, kdy se hodnota vypočteného zpoždění rovná hodnotě zpoždění změřeného. Kromě jiného vyjádření chyby predikce zpoždění můžeme z grafu vyčíst, jaký směr má chybně vypočtená hodnota zpoždění. Jinými slovy, jestli je chybně vypočtené zpoždění nadhodnoceno nebo naopak podhodnoceno. V obou těchto případech se jedná o stejně velkou chybu výpočtu zpoždění a nelze tedy říci, že vypočítat zpoždění menší než jaké je ve skutečnosti je v praxi lepším výsledkem

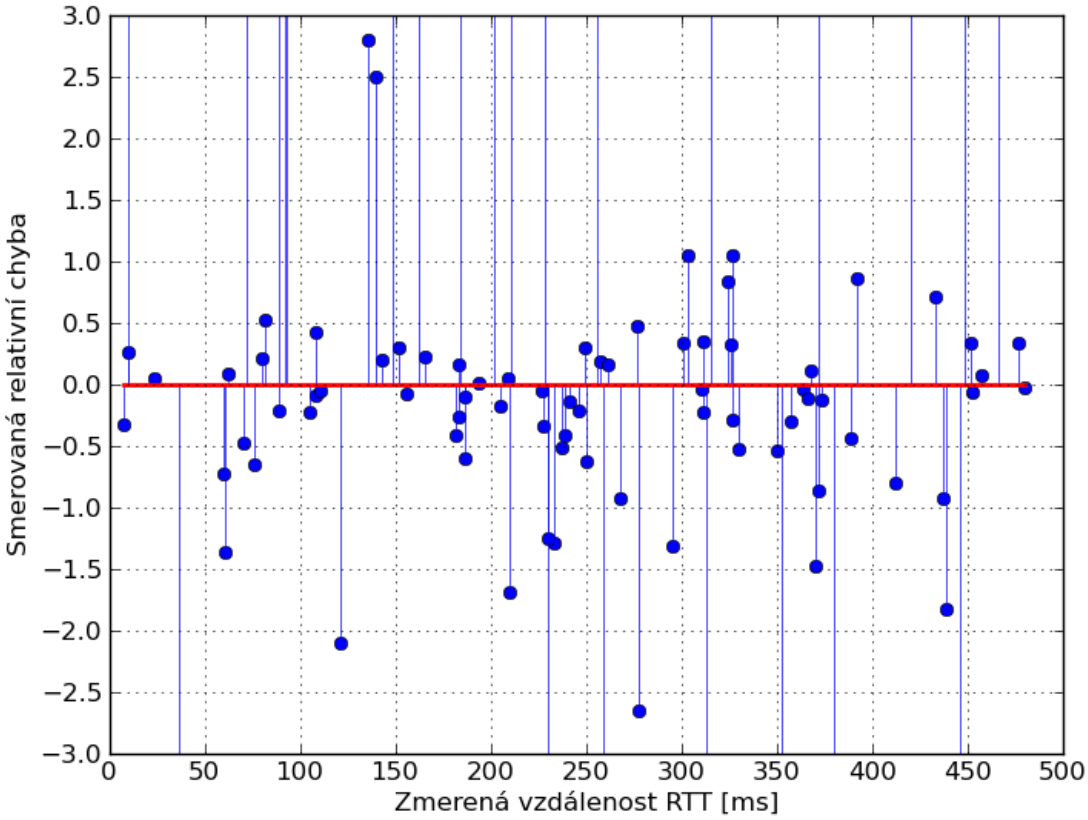
než vypočítané zpoždění větší a naopak. Jak můžeme z grafu vidět, případů kdy se podařilo vypočítat zpoždění se 100 % přesností příliš mnoho není. Cílem algoritmu Lighthouses a ostatních v této oblasti výzkumu je především snaha výpočty souřadnic a potažmo výpočty/predikce zpoždění co nejvíce přiblížit reálnému zpoždění a predikovat tak zpoždění v intervalu akceptovatelné chyby.



Obrázek 5.3: Chybová funkce poměru predikované a změřené hodnoty zpoždění.

Na obrázku 5.4 je zobrazena chyba predikce zpoždění simulované umělé sítě pomocí další z chybových funkcí, funkce směrované relativní chyby. Směrovaná relativní chyba, podobně jako chyba vypočítávající poměr dvou hodnot, vyjadřuje směr, jaký má predikované zpoždění. Simulace byla v tomto případě provedena pro 100 stanic (z důvodu lepší přehlednosti grafu) v souřadnicovém prostoru o třech rozměrech. Funkce směrované relativní chyby je odvozena od funkce relativní chyby a tudíž je osa ideálního výpočtu zpoždění rovna hlavní ose x. Z grafu vidíme, že nedošlo k výrazné převaze chyb menších než nula nebo naopak větších nula. Opět zde musím připomenout, že teoreticky ideální algoritmus by predikoval zpoždění se 100 % přesností a jakákoli odchylka od této přesnosti je nežádoucí bez ohledu na směr chyby. Z

grafu 5.4 můžeme také přibližně odhadnout počet případů, kdy byla hodnota chyby vypočítaného zpoždění v přijatelném intervalu $< -0,5; 0,5 >$. Letmým pohledem na graf vidíme, že více jak polovina všech predikcí zpoždění se nachází uvnitř tohoto intervalu.



Obrázek 5.4: Směrovaná relativní chyba pro 100 stanic v 3D prostoru.

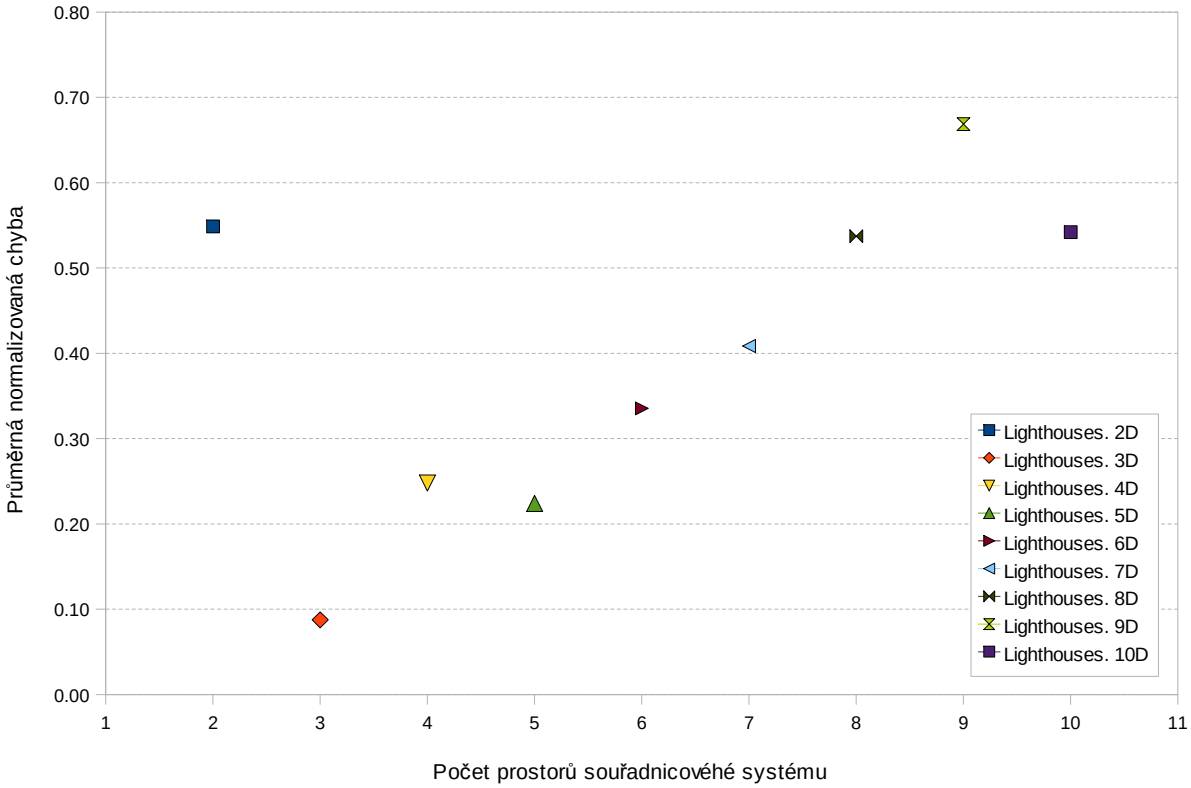
Vliv počtu rozměrů souřadnicového modelu sítě na přesnost výpočtu souřadnic stanic je uveden na obrázku 5.5. Jsou zde porovnány čtyři hlavní prostorové rozměry, které jsou nejčastěji používány. Podle teoretických předpokladů nejméně vhodným rozměrem při reprezentaci reálné síťové topologie je dvojrozměrný prostor. Důvodem je nedostatečný počet rozměrů, pomocí kterých je možné postihnout topologii reálné sítě. Ostatní zkoumané rozměry umožnily o mnoho lépe vypočítat souřadnice stanice a tak i predikce zpoždění v těchto prostorech je výrazně přesnější. Nejlépe ze zkoumaných prostorů dopadl trojrozměrný prostor, který dosáhl 90 % pravděpodobnosti predikce zpoždění do maximální chyby 0,5. Trochu překvapivě čtyřrozměrný ani pěti-rozměrný souřadnicový systém neumožnily přesnější výpočet souřadnic a tak je i chyba přesnosti predikce u těchto prostorů vyšší než u tolik úspěšného

trojrozměrného prostoru. Konkrétně je normalizované relativní chyby do maximální hodnoty 0,5 dosaženo v případě čtyřrozměrného souřadnicového systému v 78 % ze všech realizovaných výpočtů zpoždění a v případě pěti-rozměrného systému je pravděpodobnost výpočtu na úrovni 75 % pravděpodobnosti.

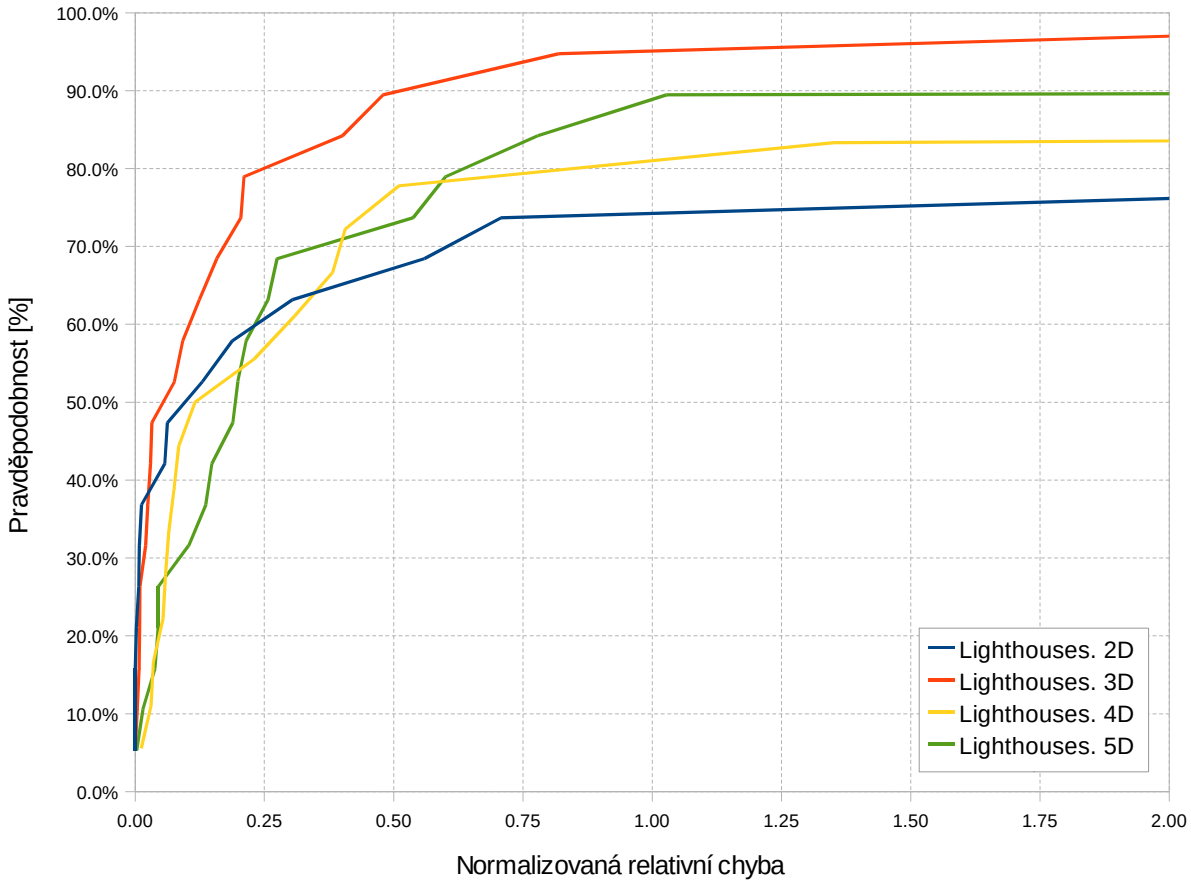
Důvodů pro tento vývoj přesnosti algoritmu Lighthouses může existovat několik. Za prvé může být za snižující se přesností predikce zpoždění u systémů s vyšším počtem rozměrů neefektivní nebo málo přesná implementace pomocných algoritmů v algoritmu Lighthouses. Chyba v dílčím výpočtu v důsledku špatné implementace pomocného algoritmu by se s rostoucí výpočetní náročností (počtem rozměrů) v poměru k počtu rozměrů lineárně zvyšovala a docházelo by tak k umělému snižování přesnosti algoritmu Lighthouses.

Dále je také možné, že vliv na klesající přesnost predikce zpoždění může mít celková nepřesnost implementovaného algoritmu, popsaného ve studii [8]. Při výpočtu souřadnic stanice je vždy zapotřebí ze všech stanic již zapojených do souřadnicového systému vybrat $d+1$, kde proměnná d určuje dimenzi souřadnicového systému. Pro vyšší počet rozměrů lineárně roste i počet pozičních serverů, které jsou zapotřebí k výpočtu souřadnic. Pokud by implementovaný algoritmus nedokázal s dostatečnou přesností vypočítat souřadnice v prostorech o čtyřech a více dimenzích, znamenalo by to, že i souřadnice dalších stanic, vypočítaných na základě chybně vypočítaných souřadnic zvolených pozičních serverů, budou vykazovat určitou míru nepřesnosti. Tento předpoklad je víceméně potvrzen na obrázku 5.5, kde čtyřrozměrný souřadnicový systém dosahuje lepších výsledků predikce zpoždění než souřadnicový systém o pěti rozměrech.

Posledním z grafů zde uvedených je graf 5.6, zobrazující průměrnou hodnotu normalizované relativní chyby v závislosti na použitém počtu rozměrů souřadnicového systému. Oproti předchozímu grafu 5.5 je zde počet zkoumaných rozměrů rozšířen na 10. Můžeme si opět všimnout trendu ve vývoji přesnosti algoritmu Lighthouses, kdy nejvyšší přesnosti je dosaženo při použití trojrozměrného prostoru. Žádný jiný ze zkoumaných prostorů nedosáhl lepších výsledků v přesnosti predikce zpoždění. Dokonce prostory o osmi, devíti a desíti rozměrech dosáhly srovnatelné přesnosti, jaké bylo dosaženo u dvojrozměrného systému. Což ve srovnání s výpočetní náročností těchto prostorů hovoří na základě výsledků simulací mnou provedených v neprospěch takto rozsáhlých prostorů.



Obrázek 5.5: Porovnání přesnosti predikce zpoždění v závislosti na počtu rozměrů.



Obrázek 5.6: Porovnání průměrné přesnosti predikce zpoždění v závislosti na počtu rozměrů.

6 ZÁVĚR

Zadáním diplomové práce bylo prozkoumat možnosti v oblasti predikce zpoždění mezi stanicemi v síti Internet se zaměřením na algoritmus Lighthouses. V teoretické části práce jsem stručně popsal hlavní představitele v oblasti predikce zpoždění využívající k tomuto účelu souřadnicové systémy. Převážná část teoretické přípravy je věnována algoritmu Lighthouses, který je stěžejním tématem diplomové práce. Teoretický popis dílčích kroků algoritmu Lighthouses při výpočtu souřadnic stanice v simulované síťové topologii, posloužil jako podklad při vývoji simulačního programu.

Pro potřeby praktického ověření teoretických poznatků jsem v rámci vypracování diplomové práce vyvinul simulační program, simulující činnost algoritmu Lighthouses. Vyvinutý simulační program je spolu s teoretickým popisem algoritmu Lighthouses hlavním pilířem této diplomové práce. Podkladem pro provedené simulace mi byly uměle vytvořené síťové topologie. Z výsledků provedených simulací byly částečně potvrzeny teoretické předpoklady o vhodném počtu rozměrů, při kterém je dosaženo požadované přesnosti predikce. Předpoklad o nevhodnosti použití 2D prostoru, jakožto prostředí pro mapování reálné síťové topologie, byl z výsledků simulací potvrzen. Predikce zpoždění na základě souřadnic stanic v 2D prostoru vykazuje největší hodnotu chyby. Naopak nejvhodnějším prostorem se v tomto ohledu ukázal 3D prostor, pomocí kterého jsem byl schopen predikovat zpoždění do maximální hodnoty chyby 0,5 u 90 % ze všech provedených výpočtů.

Velkým překvapením byly výsledky simulací u souřadnicových systémů o čtyřech a více prostorech. Teoretický předpoklad hovoří o postupném nárůstu přesnosti predikce zpoždění se zvyšujícím se počtem rozměrů. Na základě výsledků provedených simulací nebyla tato vlastnost potvrzena. Otázku, proč nedocházelo k postupnému zpřesňování vypočítaných souřadnic, ale naopak docházelo k snižování jejich přesnosti, diskutuji v kapitole 5.2.

Velká pozornost byla také věnována chybovým funkcím, pomocí kterých je určována přesnost použitého algoritmu. Použitím nevhodné chybové funkce může docházek ke zkreslování výsledků provedených simulací. V grafech zobrazujících přesnost predikce zpoždění za tímto účelem porovnávám dva hlavní představitele chybových funkcí, relativní chybu a normalizovanou relativní chybu. Funkce normalizované relativní chyby byla výsledky simulací potvrzena jako nejvhodnější k tomuto účelu.

REFERENCE

[1] NOVOTNÝ, V. *Architektura sítí.* Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2002, s. 135.

[2] SCHODER, D.; FISCHBACH, K.; SCHMITT, CH. *Core Concepts in Peer-to-Peer Networking* [online]. In Peer-to-peer Computing: The Evolution of a Disruptive Technology. University of Cologne, 2005. Dostupné z URL: <http://comp301.rice.edu/wiki/images/f/f2/Schroder-Core_Concepts_in_Peer-to-Peer_Networking-2005.pdf>.

[3] *PlanetLab An open platform for developing, deploying, and accessing planetary-scale services* [online]. Dostupné z URL: <http://www.planet-lab.org/>.

[4] PETERSON, L.; BAVIER, A.; FIUCZYNSKI, M., E.; MUIR, S. *Experiences Building PlanetLab* [online]. Proceedings of the Seventh Symposium on Operating System Design and Implementation (OSDI). Princeton University, 2006. Dostupné z URL: <http://nsg.cs.princeton.edu/publication/experiences_osdi_06.pdf>.

[5] EUGENE NG, N., S.; ZHANG, H. *A Network Positioning System for the Internet* [online]. Proceedings of the 4th Symposium on Internet Technologies and Systems. Pittsburgh: Carnegie Mellon University, 2004. Dostupné z URL: <http://www.cs.rice.edu/~eugeneng/papers/USENIX04.pdf>.

[6] DABEK, F.; COX, R.; KAASHOEK, F.; MORRIS, R. *Vivaldi: A Decentralized Network Coordinate System* [online]. In SIGCOMM. Massachusetts: Massachusetts Institute of Technology, 2004. Dostupné z URL: <http://www.sigcomm.org/sigcomm2004/papers/p426-dabek111111.pdf>.

[7] COSTA, M.; CASTRO, M.; ROWSTRON, A.; KEY, P. *PIC: Practical Internet Coordinates for Distance Estimation* [online]. 24th IEEE International Conference on Distributed Computing Systems (ICDCS'04), 2004. Dostupné z URL: <http://research.microsoft.com/en-us/um/people/mcastro/publications/PIC-ICDCS.pdf>.

[8] PIAS, M.; CROWCROFT, J.; WILBUR, S.; HARRIS, T.; BHATTI, S. *Lighthouses for Scalable Distributed Location* [online]. In Proc. of Second International Workshop on Peer-to-Peer Systems (IPTPS' 03). University of Cambridge, 2003. Dostupné z URL: <http://research.microsoft.com/users/Cambridge/tharris/papers/2003-iptps.pdf>.

[9] DOVAL, D.; O'MAHONY, D. *Overlay Networks A Scalable Alternative for P2P* [online]. In IEEE Internet Computing. Dublin: Trinity College Dublin, 2003, s. 2-5. Dostupné z URL: <http://www.dynamicobjects.com/papers/w4spot.pdf>.

[10] BALAKRISHNAM, H.; KAASHOEK, M., F.; KARGER, D.; MORRIS, R.; STOICA, I. *Looking Up Data In P2P Systems* [online]. Communications of the ACM, Special Issue: Technical and Social Components of Peer-to-peer Computing. Massachusetts: Massachusetts Institute of Technology, 2003. vol. 46, no. 2, s. 43-48. Dostupné z URL: <http://www.cs.berkeley.edu/~istoica/papers/2003/cacm03.pdf>.

[11] LAI, K.; BAKER, M. *Measuring Link Bandwidths Using a Deterministic* [online]. Department of Computer Science. Stanford University, 2000. Dostupné z URL: <http://www.hpl.hp.com/personal/Kevin_Lai/projects/nettimer/publications/sigcomm2000/deterministic.pdf>.

[12] *RFC792 - Internet Control Message Protocol* [online]. Dostupné z URL: <http://www.faqs.org/rfcs/rfc792.html>.

[13] GUMMADI, K., P.; SAROIU, S.; GRIBBLE, D., S. *King: Estimating Latency between Arbitrary Internet End Hosts* [online]. Seattle: University of Washington, 2002. Dostupné z URL: <http://www.mpi-sws.org/~gummadi/king/king.pdf>.

[14] ZHENG, H.; LUA, K., E.; PIAS, M.; GRIFFIN, G., T. *Internet Routing Policies and Round-Trip-Times* [online]. In PAM. University of Cambridge. Dostupné z URL: <http://www.cl.cam.ac.uk/~ekl25/p53CameraReady.pdf>.

[15] SHAVITT, Y.; TANKEL, T. *Big-bang simulation for embedding network distances in Euclidean space* [online]. Tel-Aviv University. Department of Electrical Engineering, 2004. Dostupné z URL: <http://www.ieee-infocom.org/2003/papers/47_02.PDF>.

[16] NELDER, J., A.; MEAD, R. *A simplex method for function minimization.* Computer Journal, 1965. vol. 7, s. 308–313.

[17] EUGENE NG, N., S.; ZHANG, H. *Predicting Internet Network Distance with Coordinates-Based Approaches* [online]. Pittsburgh: Carnegie Mellon University, 2001. Dostupné z URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.1445&rep=rep1&type=pdf>.

[18] *Azureus - now called Vuze - Bittorrent Client* [on-line]. URL: <http://azureus.sourceforge.net/>.

[19] *The Gram-Schmidt Algorithm* [online]. Harvey Mudd College, Math Tutorial. Dostupné z URL: <http://www.math.hmc.edu/calculus/tutorials/gramschmidt/gramschmidt.pdf>.

[20] SPENCE, D., R. *An implementation of a coordinate based location system* [online]. Cambridge: University Of Cambridge, Computer Laboratory. Technical Report, 2003. Dostupné z url: <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-576.pdf>.

[21] MADRUGA, F.; FILHO, N., D.; COELHO, R.; PIAS, M. *Estimando Atrasos em Redes de Computadores Através Sistemas de Coordenadas - A Ferramenta Euclideana* [online]. Rio Grande: Fundacao Universidade Federal do Rio Grande. Departamento de Matemática, 2005. Dostupné z url: <http://www.ncc.furg.br/euclideana/manual_instalacao.pdf>.

[22] LEDLIE, J.; GARDNER, P.; SELTZER, M. *Network Coordinates in the Wild* [online]. Harvard School of Engineering and Applied Sciences and Aelitis, 2007. Dostupné z url: <http://www.cs.ucla.edu/~kohler/class/08w-dsi/ledlie07network.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DT distribuovaný strom – Distributed tree

DHT distribuovaná hashovací tabulka – Distributed hash table

GUI grafické uživatelské rozhraní – Graphical user interface

ICMP signalizační protokol sady protokolů internetu – Internet Control Message Protocol

IP internetový protokol – IP adresa – Internet Protocol address

IPTV televizní vysílání šířeno prostřednictvím IP sítí – Internet Protocol television

ISP firma či organizace zprostředkovávající přístup do Internetu – Internet service provider

P2P označení síťové architektury typu klient-klient – Peer-to-Peer

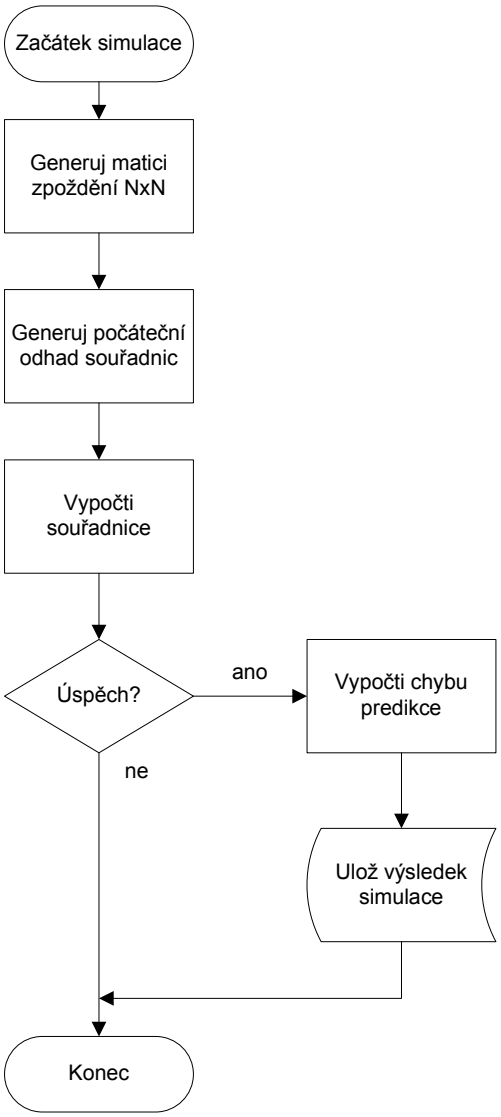
RTT obousměrné zpoždění zprávy ICMP – round-trip delay time

TCP/IP označení sady protokolů Internetu – Internet Protocol Suite

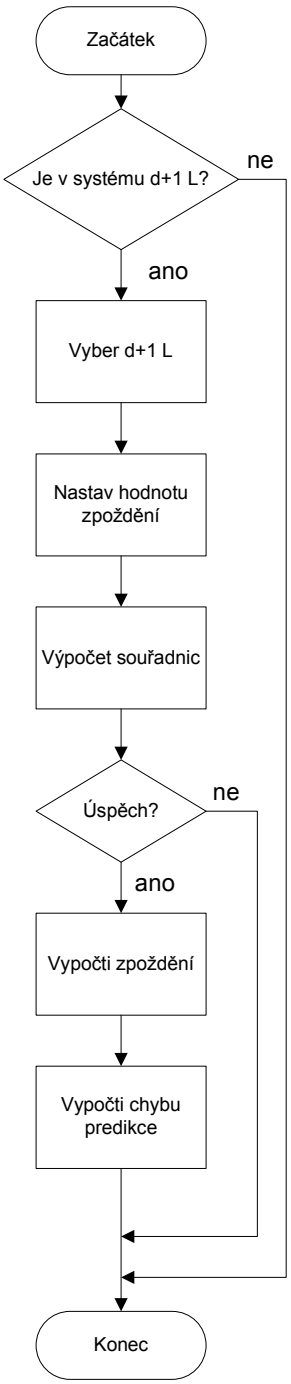
SEZNAM PŘÍLOH

A DOKUMENTACE SIMULAČNÍHO PROGRAMU

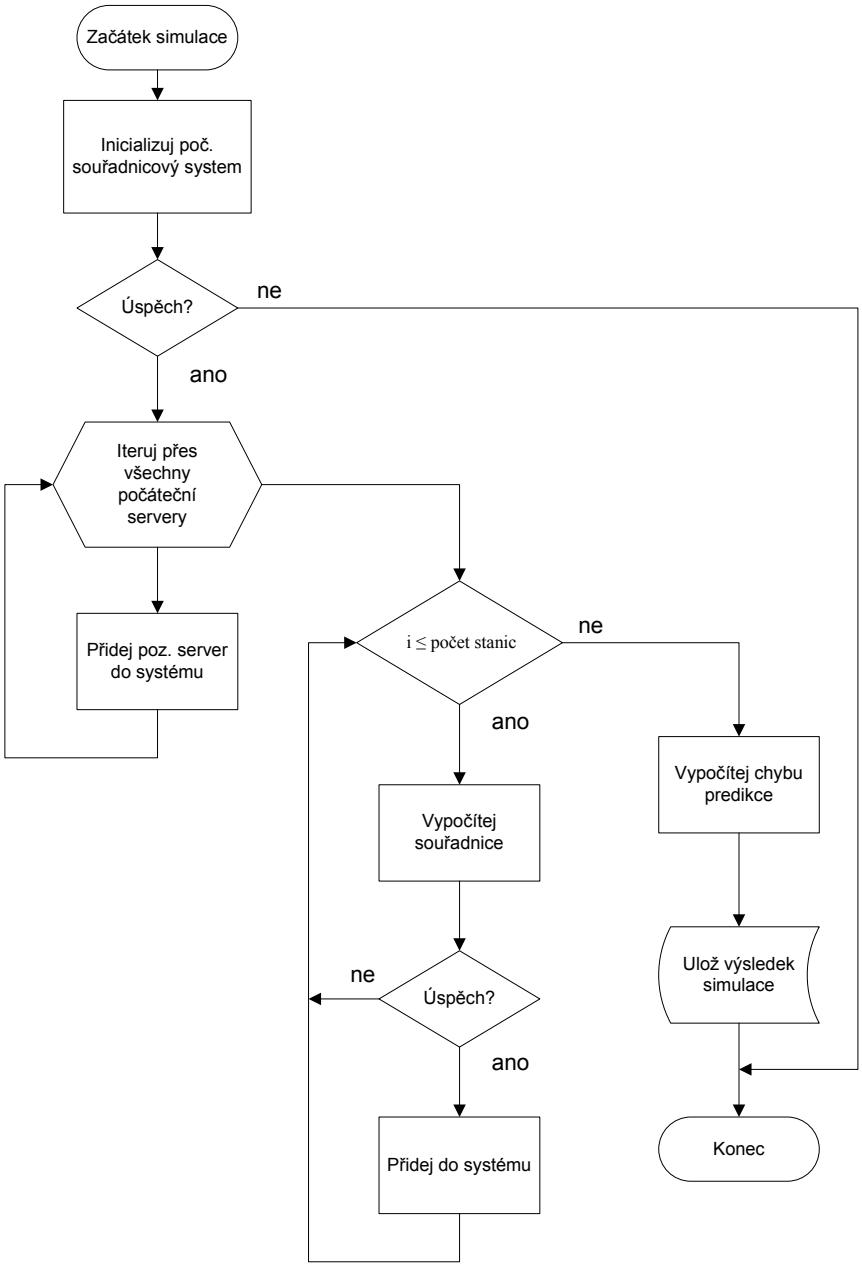
A.1 Inicializace počátečního souřadnicového systému



A.2 Výpočet souřadnic stanice



A.3 Třída Lighthouses



B OBSAH PŘILOŽENÉHO CD

Adresář	Obsah adresáře
Lighthouses	vyvinutý simulační program
src	zdrojové texty programu
conf	konfigurační soubor programu
grafy	grafické výstupy ze simulací a podklady ke grafickým výstupům
gf_dp	grafické výstupy použité v diplomové práci v plném rozlišení
Diplomova_prace	text diplomové práce a související dokumenty
vd	vývojové diagramy použité v diplomové práci