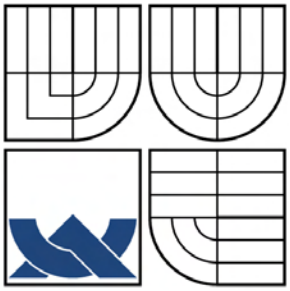
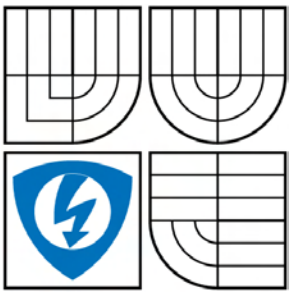


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE ZJIŠTĚNÍ POLOHY POČÍTAČE V INTERNETU

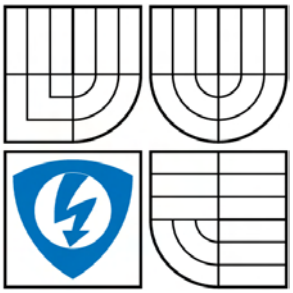
SIMULATION OF FINDING A COMPUTER POSITION IN INTERNET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ VRZAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAN KOMOSNÝ, Ph.D.
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student: Vrzal Tomáš ***ID:*** 77902
Ročník: 3 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

Simulace zjištění polohy počítače v Internetu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte algoritmus nazvaný „binning“ pro vyhodnocení fyzické polohy počítače v Internetu. Pro tento algoritmus vytvořte simulační prostředí, které bude běžet na jednom počítači. Toto prostředí realizujte tak, aby bylo možno vyhodnotit pozici libovolně zvoleného počítače z předem zadané množiny. Přehledně zobrazte také vypočtené vzdálenostní vektory.

DOPORUČENÁ LITERATURA:

[1] PUŽMANOVÁ, R. TCP/IP v kostce. 1. vyd. České Budějovice : Kopp, 2004. 607 s. ISBN 80-7232-236-2.

[2] RATNASAMY S, HANDLEY M, KARP R, SHENKER S. Topologically-Aware Overlay Construction and Server Selection, Proceedings of 21rd Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, 2002.

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 4.6.2008

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

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

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Tomáš Vrzal
Bytem: Spáčilova 3034/5, 76701, Kroměříž
Narozen/a (datum a místo): 16.4.1986, Kroměříž

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Simulace zjištění polohy počítače v Internetu
Vedoucí/školitel VŠKP: Ing. Dan Komosný, Ph.D.
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

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

Článek 2

Udělení licenčního oprávnění

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

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

Článek 3

Závěrečná ustanovení

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

V Brně dne: .....

....

Nabyvatel Autor

ABSTRAKT

V bakalářské práci je probírána problematika přenosu multimediálních dat pomocí protokolu RTP/RTCP s hierarchickou sumarizací. V dalších částích práce je vysvětlen princip a využití tohoto přenosu pomocí stromové struktury. Na závěr je uveden popis aplikace, která vznikla jako simulátor prostředí využívající stromovou strukturu. Pro nejlepší výsledky simulace je používán algoritmus nazvaný „binning“. Aplikace má grafický vzhled, takže není složité se s ní naučit pracovat. V přílohách jsou obrázky grafického prostředí a zdrojový kód aplikace.

KLÍČOVÁ SLOVA

Simulace, IP adresa, vzdálenost, stromová struktura, přenos multimédií, RTP/RTCP.

ABSTRACT

This bachelor’s thesis puts mind to problematic transmission multimedia data via RTP/RTCP protocol with hierarchical aggregation. In next parts of thesis are explaining design principle and use this transmission via tree structure. In last part is description application in which the simulation ambient uses tree structure. For best results of simulation is use algorithm called “binning”. Application has graphic interface, so that easy manipulation with it. In insertion are pictures and application code.

KEYWORDS

Simulation, IP address, range, tree structure, transmission multimedia, RTP/RTCP.

VRZAL, T. Simulace zjištění polohy počítače v Internetu. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 53 s. Vedoucí bakalářské práce Ing. Dan Komosný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma "Simulace zjištění polohy počítače v internetu" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a 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 především vedoucímu bakalářské práce Ing. Danu Komosnému, Ph.D., za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce. Dále bych chtěl poděkovat rodině, mým přátelům a kamarádovi Lubomíru Poláškovi, kteří mi vytvořili podmínky pro psaní bakalářské práce a podporovali mě při programování samotné aplikace.

V Brně dne …….. …...…….……..

(podpis autora)

Obsah

Seznam obrázků

Úvod

Bakalářská práce je součástí projektu, který se zabývá vývojem nových technologií zejména pro multicast, vysílání televize pomocí TCP/IP sítě a tedy i Internetu a vyžaduje více či méně spolupráci s ostatními řešiteli. Jedná se o skupinu Multicast IPTV Research Group, vzniklou na Ústavu telekomunikací Vysokého učení technického v Brně. Zabývá se zejména vývojem nových a využitím stávajících algoritmů, protokolů a software, který by zajistil možnost vysílání interaktivní televize pro velmi velké množství uživatelů. Jedná se převážně o využití protokolu RTP (Real Time Protocol), RTCP (Real Time Control Protocol) a principu hierarchické sumarizace.

Není třeba dlouho přemýšlet, abychom přišli na to, že se nacházíme v době, kdy se internet stal naší každodenní součástí. Nově i vysílání televize přes internet IPTV. Je to téma, které je novinkou v informačních technologiích a bude se zkoušet ještě mnoho možností, jak nejlépe tuto problematiku realizovat. Jednou z nich, je i náš projekt, jak je již zmíněno výše.

1 Seznámení s týmem pracujícím na projektu

Část této problematiky, dostal řešit tým čtyř lidí bakalářského studia, včetně mě (Obr. 1.1). Všechny události a programy se simulují na osobních počítačích studentů, kromě jednoho studenta, který pracuje se sítí PlanetLab. Tito studenti řeší, jak dosáhnout nejlepších výsledků bez nutnosti zkoušení na reálné síti. Další dílčí krok je odzkoušení, jak se bude chovat reálná síť, ale to mají na starosti studenti magisterského studia, proto je zde nebudu uvádět. V dalších kapitolách uvedu podrobněji práci studentů, kteří jsou součástí projektu.

Název projektu	Příjmení a jméno	E-mail
Simulace zjištění polohy počítače v Internetu	Vrzal Tomáš	xvrzal04@stud.feec.vutbr.cz
Vizualizace spojení mezi počítači	Kovář Jan	xkovar46@stud.feec.vutbr.cz
Výkon síťového serveru při komunikaci s velkým počtem klientů	Mašín Jan	xmasin00@stud.feec.vutbr.cz
Skripty pro instalaci a konfiguraci operačního systému Fedora	Czerner Lukáš	xczern00@stud.feec.vutbr.cz

Obr. 1.1: Souhrn členů týmu řešící danou problematiku.

1.1 Vizualizace spojení mezi počítači – Kovář Jan

Vizualizace spojení mezi počítači, má za úkol zobrazit graficky stromovou strukturu z dat, které bude generovat student, řešící Simulaci zjištění polohy počítače v internetu. Po dohodě se bude používat jednotný výstupní formát souboru, do kterého se budou posílat veškerá potřebná data.

1.2 Výkon síťového serveru při komunikaci s velkým počtem klientů – Mašín Jan

Výkon síťového serveru při komunikaci s velkým počtem uživatelů má za úkol počítat aktuální vytížení jednotlivých serverů, aby nedošlo k zahlcení, případně spadnutí určité větve

stromové struktury. Veškeré výpočty se budou simulovat na dvou počítačích, kde bude nainstalovaný příslušný software pro simulaci zatížení.

1.3 Skripty pro instalaci a konfiguraci operačního systému Fedora – Czerner Lukáš

Úkolem řešitele je tvorba skriptů pro instalaci a konfiguraci operačního systému Fedora, tedy systému, který má běžet na serverech v síti PlanetLab. Postupem času se ale ukázalo, že samotné uzly PlanetLabu, resp. virtuální servery na nich provozované, jsou již tímto softwarem vybaveny a nevyžadují žádné další větší zásahy z naší strany. Bylo tedy dohodnuto s vedoucím projektu Ing. Danem Komosným, Ph.D., že cílem snažení bude sledování „zdraví“ používané části sítě a tvorba nástrojů (skriptů), které mají dalším účastníkům projektu usnadnit umístění, nebo stáhnutí dalších potřebných souborů na virtuální servery.

2 Úvod do problematiky IPTV

Televize IPTV slouží k přenosu digitálního televizního vysílání k uživateli po síti založené na TCP/IP. Z toho plyne, že nemusí být již jednosměrné, jako je tomu u klasického broadcast vysílání, ale může uživatelům poskytovat zpětnou vazbu ve formě interaktivity a na druhou stranu i sledování statistik, na co se uživatel zrovna dívá. Nevýhodou ovšem je, že přenosový kanál není dostatečně dimenzován na to, aby například na jedné internetové přípojce ADSL bylo možné sledovat zároveň více kanálů. Určitým problémem je také doba přepínání kanálů, která je o poznání delší.

Princip vysílání spočívá v tom, že jeden program televize představuje multicastovou skupinu, do níž se uživatelé připojují. To umožňuje tarifikaci uživatelů podle doby připojení do skupiny a typu skupiny, možnost zvolit si jiný, než právě vysílaný pořad (videotéka) apod.[1]

2.1 Využívané protokoly RTP/RTCP

RTP (Real Time Protocol) je protokol sloužící pro přenos dat v reálném čase. Obvykle se používá pro přenos audia a videa. Protokol sám o sobě nezaručuje přenos dat v reálném

čase, disponuje jen procedurami, které umožňují rekonstrukci těchto vlastností na straně přijímače. V záhlaví protokolu jsou přenášeny informace o typu přenášených dat, způsobu kódování, zdroji synchronizace, jakož i sekvenční číslo paketu a časová značka pro obnovu synchronizace. Přijímač prostřednictvím sekvenčních čísel paketů zabezpečuje interpretaci dat z paketů RTP ve správném pořadí. Podle definovaného prostředku synchronizace umožní časová značka v každém paketu načtení dat z paketů v odpovídajících časových intervalech.

RTCP (Real Time Control Protocol) je řídící protokol spolupracující s protokolem RTP, používající periodické vysílání paketů od každého účastníka relace RTP všem ostatním účastníkům za účelem řízení výkonnosti a pro diagnostické účely. RTCP vykonává následující služby:

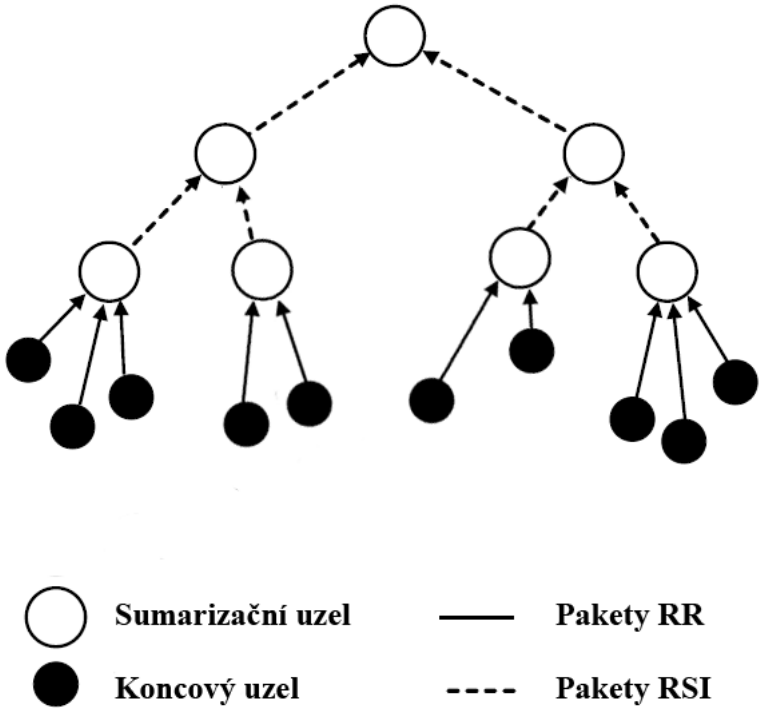
- poskytuje informace aplikaci týkající se kvality vysílaných dat
- identifikuje zdroj RTP
- provádí řízení intervalu vysílání RTCP
- přenos minimální informace o řízení relace

RTCP používá pro každý RTP mediální tok vyhrazený logický kanál, tak aby byl oddělen přenos uživatelských dat od signalizace. Protokol RTP/RTCP pracuje nad protokolem UDP (User Datagram Protocol), který je součástí protokolové sady TCP/IP. Jsou dva základní pakety, které protokol využívá pro signalizaci. Jeden z nich je paket SR (Sender Report), který se vysílá od zdroje k přijímači a druhý je RR (Receiver Report), ten je pro změnu vysílán od přijímače ke zdroji. Když je tento protokol použit ve velké relaci, jako je například IPTV, tak dochází k jednomu velkému problému. Při zvětšujícím se počtu dat při velkém počtu účastníků, se zároveň zvětšuje doba, mezi kterou se posílají pakety protokolu RTCP. Nicméně zpětná vazba produkována samotnými uživateli, může být zredukována na velikost, která je přípustná pro běh protokolu RTP. Toho se dosáhne tak, že nad klientem o úroveň výš, je tzv. sumarizační uzel (pomocný server), který z mnoha přijatých zpětných vazeb komprimací udělá jednu, kterou pošle ve stromové struktuře o úroveň výš.[2]

2.2 Jednotlivé části stromové struktury

Stromová struktura je speciální hierarchie, použita v multicastovém vysílání (Obr. 2.1). Je složena ze zdroje (vysílací server), pomocných serverů a klientů. Vysílací server je zdrojem multicastového vysílání, které vysílá na mnoha multicastových adresách a leží na nejvyšší

úrovni stromové struktury. Na dalších úrovních jsou pomocné servery a na nejnižší úrovni klienti, kteří přijímají multicastové vysílání. Pomocné servery, neboli sumarizační, mají za úkol shromažďovat požadavky klientů a vysílat je ke zdroji v určitých časových intervalech. Funguje to tak, že klienti pošlou nezávisle požadavek pomocí RR paketu do sumarizačního serveru na vyšší úrovni, ten je přidá do RSI (Receiver Summary Information) paketu a pošle ho o úroveň výš, kde ho přijme další sumarizační server nebo zdroj vysílání. Vysílací server, pak pošle odpověď všem klientům pomocí jedné z mnoha multicastových adres. Mechanismus pro kontrolu intervalů RTCP paketů je založen na tom, že je použito maximálně 5% z celkové šířky pásma pro přenos, které se smí využít. Dále se dělí na 3,75% pro RR pakety a 1,25% pro SR pakety.[2],[3]



Obr. 2.1: Stromová struktura.

2.3 Intervaly přenosu signalizace ve stromové struktuře

Záměrem omezení šířky pásma je předejít zahlcení sítě pakety, když se začne struktura chovat neočekávaně. Například, pokud v relaci selže část sítě, tak současný počet uživatelů může extrémně zpomalit intervaly procházejících paketů v síti. To může mít nežádoucí vliv

na zatížení sítě, jestliže je interval zpětné vazby počítán z aktuálního počtu uživatelů. Účelem stromové struktury je zajistit co nejmenší možné intervaly, proto je nutné, znát počet úrovní ve stromové struktuře. Konečný počet uživatelů na spodní úrovni a počet sumarizačních serverů na ostatních úrovních, může být důležitý pro funkčnost celé stromové struktury.[2]

Interval RR paketu se vypočítá jako

$$PR_{rr} = \frac{0.75 \times BW_{\mathrm{rtcp}}}{PS_{\mathrm{rr}} \times n_{\mathrm{r}}}, \tag{2.1}$$

kde BWrtcp je celková šířka pásma, PS_{rr} průměrná velikost RR paketu, n_{r} celkový počet přijímačů.

Interval SR paketu se vypočítá jako

$$PR_{sr} = \frac{0.25 \times BW_{\mathrm{rtcp}}}{PS_{\mathrm{sr}} \times n_{\mathrm{r}}}, \tag{2.2}$$

kde BWrtcp je celková šířka pásma, PS_{sr} průměrná velikost SR paketu, n_{s} celkový počet vysílačů.

2.4 Sumarizace paketů ve stromové struktuře

Sumarizace je vhodná pro velké relace jakým IPTV bezpochyby je. Princip je takový, že se určitý počet klientů rozdělí do skupin, kde se zvolí vždy jeden, často nazván sumarizační server. Nevýhoda použití tohoto mechanizmu je, že zdroj vysílání z takto spojených paketů nepozná, od kterého uživatele přicházejí. Proto nám zaniká možnost rozpoznat signalizaci specifického uživatele, který má třeba problém s příjmem. Dalším problémem je, že potřebujeme, aby po aplikování sumarizace, byla velikost RR nebo RSI paketu stejná po celou dobu průchodu stromovou strukturou. Nicméně i celkový odhad velikosti paketů, je poněkud obtížný. Vyčíslením klasické hlavičky v paketu (IP, UDP, RR, RSI) a konstantní velikosti zpětné vazby přidávané do paketu RR, můžeme lokalizovat pevnou část dat v RR a RSI paketech. Pak už jen zbývá odhadnout proměnlivou velikost dat. Bohužel, kromě RR paketů sumarizačních serverů v určité multicastové aplikaci, mohou být posílány i jiné RTCP pakety, které nemohou projít agregací. Tyto pakety jsou například APP a BYE pakety, které vysílací zdroj ignoruje, protože nepatří mezi jeho signalizační pakety.[2],[3]

2.5 Stromová struktura pro IPTV vysílání

V této kapitole se zabývám IPTV vysíláním postaveném na hierarchické sumarizaci. Formát videa, je většinou SD-DVB s rozlišením 720x576 pixelů s maximálním obnovovacím kmitočtem 30Hz. Video podléhá kompresi používající MPEG-2 standard MP (Main Profile). Požadovaná šířka pásma pro video je 15Mb/s, ale také předpokládáme, že určitá šířka pásma je používána pro přenos audia a zpětné vazby. Proto odhadujeme celkovou šířku pásma na 16Mb/s, včetně 5% pro signalizaci (tj. 800kb/s). Teoretický možný počet příjemců IPTV vysílání je jeden milion. Výsledek nejmenované statistiky udává, že tento počet IPTV uživatelů jedné mezinárodní TV stanice je reálný v blízké budoucnosti. Ve stromové struktuře nemůže zdroj posílat RSI pakety, obsahující povinné bloky pro různé uživatele, pokud je ve struktuře mnoho skupin s rozdílným počtem uživatelů. Místo toho zkusí poskytnout tyto informace ve vyhrazeném paketu, společně s vypočítanou strukturou stromu, který by se mohl poslat každému novému uživateli připojenému do struktury. Tento vyhrazený paket je posílán všem uživatelům přes SSM (Specific Source Multicast) kanál. Pro tento záměr jsme doplnili do RSI paketu nové bloky, nazvané „Group Block“. Nový blok, kromě dalších, obsahuje celkový počet členů každé skupiny v relaci. Použitím tohoto čísla, můžeme vypočítat interval zpětné vazby, který nepřekračuje šířku pásma poskytovanou uvnitř skupiny. V ostatních případech stromové struktury, musíme znát dva klíčové prvky, a to počet konečných uživatelů a počet sumarizačních serverů. Použitím známé velikosti RR paketu, můžeme určit počet konečných uživatelů ve skupině. Se standardním RTP/RTCP protokolem, je zpětná vazba od příjemce posílána pomocí RR paketů, které jsou rozděleny mezi všechny uživatele, bez jakéhokoliv vysílaného požadavku. Hierarchická sumarizace se specifikací pro stromovou strukturu, nabízí výsledky 25 krát lepší v intervalu zpětné vazby pro jeden milion uživatelů, než standardní RTP/RTCP protokol.[1],[2]

3 Řešení vlastní části projektu

Mým úkolem je navrhnout aplikaci, která bude využívat algoritmus pro zajištění nejlepších vzdálenostních poměrů, mezi serverem a klientem. Vše bude probíhat jako simulace, která bude mít podobu aplikace vytvořené v jazyce C++.

3.1 Použitý programovací jazyk

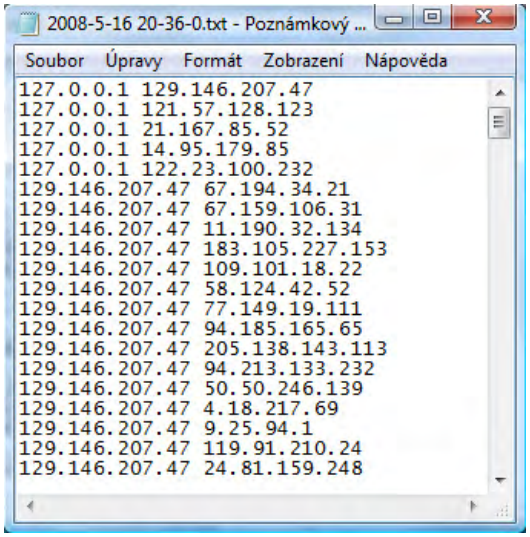
Stál jsem před rozhodnutím, jaký programovací jazyk použít, ale zjistil jsem, že ne všechny programovací jazyky se hodí pro tuto aplikaci. Ze široké nabídky programovacích jazyků, jsem nakonec vybral jazyk C/C++.

Bezpochyby elegantní řešení, co se týče rychlosti, algoritmus bude ve zkompilované aplikaci bezpochyby velmi rychlý. Bohužel, problémem jsou dvě zásadní věci. Přenositelnost aplikace mezi operačními systémy a její dostupnost. Aplikace by se zřejmě musela vytvořit pro různá prostředí zvlášť. Další velká nevýhoda je po grafické stránce, kde dosavadní rozhraní neposkytuje veškeré inovace, kterými prošly jiné jazyky nebo dokonce vznikly nové. Jelikož s jazykem C/C++ pracuji více než 2 roky, tak znám všechny potřebné funkce a vlastnosti jazyka, které budou pro tuto aplikaci dostačující.

3.2 Datové úložiště

Pokud budeme brát v úvahu, že budou zobrazeny informace o uzlech, budeme je muset nějakým způsobem ukládat k pozdějšímu čtení a informace už nebudou pouze proměnné v programu. Má to však poměrně značnou nevýhodu. Celkové množství dat bude velmi rychle narůstat. Musíme také počítat s tím, že informace o uzlech je soubor s několika hodnotami.

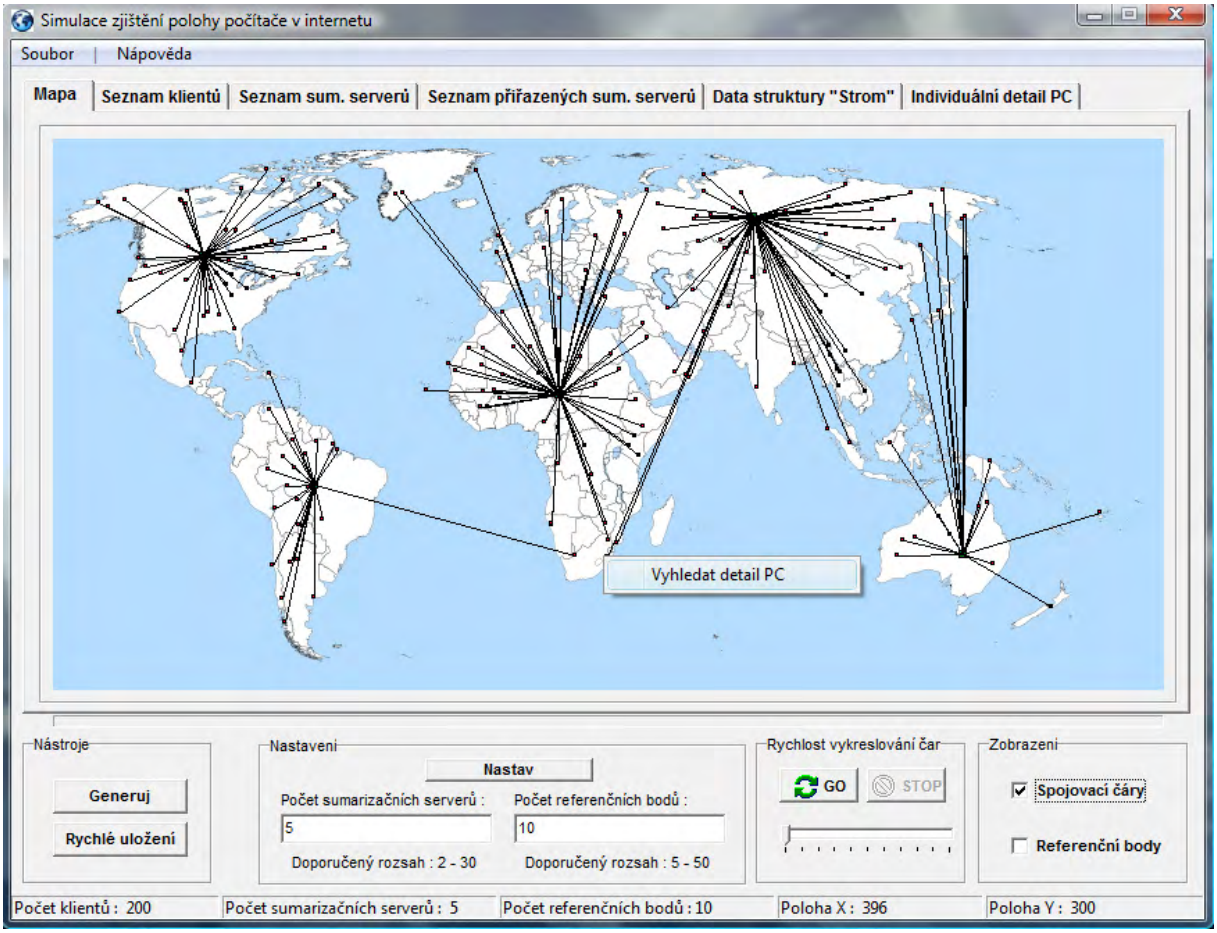
Zvolíme-li jednoduchý způsob ukládání dat do souboru, například v obyčejné textové podobě na řádky textového souboru, je poměrně problematické, jak data dodatečně vyhledávat. Procházení textových řetězců v tomto souboru není složité, ale vzhledem k počtu záznamů, bude velmi časově náročné a velmi špatně čitelné. Velkou nevýhodou může být počet takovýchto souborů, pokud jej chceme aktualizovat v krátkých časových intervalech. Proto je lepší zvolit uložení jen na vyžádání uživatele. V průběhu realizace se nakonec dospělo k závěru, že stačí použít obyčejný textový soubor. V souboru je uložena na každém řádku dvojice IP adres oddělená mezerou (Obr. 3.1), která představuje nadřazený uzel a klienta. Další dodatečné informace o uzlech k dalšímu zpracování nejsou potřeba, jak je tomu u kolegů pracujících v PlanetLabu.



Obr. 3.1: Ukázka uložených dat stromové struktury v souboru.

3.3 Základní informace o programu

Program je vytvořen v aplikaci CodeGear™ C++Builder® 2007, má grafické prostředí (Obr. 3.2), takže ovládání je pro uživatele přátelské (user-friendly). Program byl odzkoušen v operačním systému Microsoft Windows XP a Vista, jak v 32 bitovém prostředí, tak v 64 bitovém prostředí, takže by neměli nastat žádné problémy při běhu programu v jednom z těchto systémů. Hlavní část okna programu, zabírá přepínač se záložkami, ve kterých jsou zobrazeny jak tabulky s IP adresami, tak mapa s jejich polohou. V horní části samozřejmě nemůže chybět menu, ve kterém by měl uživatel nalézt veškeré pracovní nástroje, které jsou potřeba pro řízení programu, ale bohužel tomu tak není. U této aplikace všechny dostupné a použitelné nástroje jsou zobrazeny ve spodní části okna viditelné po celou dobu běhu aplikace. Úplně na spodní liště je komponenta StatusBar (Stavový řádek), která zobrazuje aktuální hodnoty různých proměnných použitých při simulování. Okno aplikace je pevné velikosti, takže jej nelze maximalizovat, aby nedošlo k chybnému zobrazení bodů na mapě, které jsou závislé na počtu pixelů k samotnému oknu aplikace. Po minimalizaci a opětovném obnovení aplikace, nebudou zobrazena žádná data v mapě, proto je nutné kliknout do mapy nebo na záložku.



Obr. 3.2: Grafický vzhled aplikace.

3.4 Popis jednotlivých komponent

Jak už bylo výše řečeno, největší část zabírá mapa a tabulky s IP adresami. Je to z toho důvodu, že mapa slouží jako vstupní portál pro sběr dat důležitých pro bezchybný běh algoritmu. Na mapě (Obr. 3.2) se zobrazuje poloha IP klientů (červené čtverečky), poloha sumarizačních serverů (zelené čtverečky) a poloha referenčních bodů (modré čtverečky). Přes mapu lze po kliknutí pravým tlačítkem myši v blízkosti klienta (červeného čtverečku) zvolit vyhledání a zobrazení jeho detailů. Tento detail se nachází na záložce „Individuální detail PC“, do které aplikace automatický skočí při výběru daného klienta a zobrazí detaily.

V záložce „Individuální detail PC“ je zobrazena IP adresa, souřadnice umístění bodu na mapě, vzdálenostní vektor a ID sumarizačního serveru. Ve spodní části záložky jsou pak zobrazeny stejné informace, ale tentokrát o sumarizačním severu, ke kterému je klient připojen.

Tabulky se seznamem jak IP adres klientů, tak IP adres sumarizačních serverů jsou řešeny podobně. V prvním sloupci je zobrazena IP adresa, v dalších dvou pak x-ové a y-ové souřadnice umístění bodu na mapě a v posledním sloupci vzdálenostní vektor. Pokud je zadáno více, jak deset referenčních bodů, tak nejsou vidět všechny hodnoty vzdálenostního vektoru v buňce tabulky. Z toho důvodu je zavedena bublinová nápověda, která po kliknutí levým tlačítkem myši na buňku s vektorem, zobrazí všechny hodnoty vzdálenostního vektoru. Trochu rozdílné uspořádání je na záložce „Seznam přiřazených sum. serverů“, kde vystřídal sloupec vzdálenostního vektoru, sloupec s hodnotou úrovně shody vzdálenostního vektoru a přibyl ještě jeden sloupec a to ID přiřazeného sumarizačního serveru. V poslední nezmíněné záložce „Data stromové struktury“, jsou zobrazena konečná data pro vizualizaci stromové struktury.

V menu máme dvě položky „Soubor“ a „Nápověda“. Každá funkce má přiřazenou klávesovou zkratku, podle známého Windows schématu. V položce „Soubor“ máme tři funkce:

- **Nový (Ctrl+N) –** Tato funkce vymaže veškeré předchozí data v tabulkách a vynuluje pomocné proměnné, důležité pro chod aplikace. Uživatel tak začne kompletně znovu simulovat, jako by znovu otevřel aplikaci. Touto volbou se změní i IP adresy klientu a sumarizačních serverů, které jsou jinak při simulování stejné, jen se mění poloha klientů.
- **Ulož (Ctrl+S) –** Touto volbou uživatel uloží data potřebná pro vizualizaci stromové struktury. Zde je možnost zvolit umístění a název souboru, což u volby přes tlačítko „Rychlé uložení“ nemáme.
- **Exit (Ctrl+Q) –** Tímto se regulérně ukončuje celý program. Dojde k vymazání vytvořených dynamických míst v paměti, takže nejsou dále blokována.

V položce „Nápověda“ se nachází odkaz na formulář s údaji o programu, kde je uveden název a verze programu.

Ve spodní části aplikace je stavový řádek, ve kterém jsou zobrazeny hodnoty proměnných, s nimiž se právě pracuje. Když to vemem z levé stany, tak pořadí je takové:

- **Počet klientů –** Tato hodnota nám udává pevný počet klientů (červené čtverečky), který uživatel nemůže nijak změnit, je pevně dán v jádru aplikace.

- **Počet sumarizačních serverů –** Tuto hodnotu už získáme zadáním od uživatele a jedná se o počet zelených čtverečků, které budou rozmístěny po mapě, dle libosti uživatele.
- **Počet referenčních bodů –** Hodnota tohoto čísla je opět proměnlivá a závisí pouze na uživateli, jaké si zvolí číslo z doporučeného rozsahu. Jedná se o automatické rozmístění bodů v zadaném počtu po mapě, díky kterým jsou počítány vzdálenostní vektory.
- **Poloha X –** Udává nám souřadnici x-ové osy v mapě, kde uživatel klikl levým tlačítkem myši.
- **Poloha Y –** To samé jako v předešlém případě, jen s tou změnou že tady se nám zobrazuje souřadnice y-ové osy v mapě.

Pod oknem se záložkami je ProgressBar (Průběhový pás), který nám zobrazuje aktuální stav průběhu vykreslování čar. Tato možnost je zpřístupněna v panelu nazvaném „Rychlost vykreslování čar“. Zde může uživatel zvolit pomalé vykreslování spojovacích čar, aby bylo lépe znatelné, ke kterým sumarizačním serverům, patří daný klient. Nacházejí se zde dvě tlačítka. Jedno je „GO“, tím se vykreslování spouští a druhé „STOP“, tím se zastavuje. Při zastavení a opětovném spuštění se vykreslování rozběhne v místě, kde skončilo, ale pouze za podmínky, že uživatel v době mezi zastavením a znovu spuštěním nekline na mapu nebo nějakou záložku. V tom případě se nedokončené vykreslování kompletně dokončí a aktuální pozice ve vykreslování se vymaže. Po stisknutí „GO“ vše poběží od začátku. V panelu máme ještě další komponentu TrackBar (posuvník), kterou můžeme měnit rychlost vykreslování čar i za běhu vykreslování. V levé pozici je nejrychlejší vykreslování, které je rovno 15ms na jednu čáru a v pravé pozici je rovno 500ms, takže v nejrychlejším režimu se všechny čáry vykreslí za necelé tři sekundy.

Vpravo od vykreslování čar je panel „Zobrazení“, ve kterém jak už napovídá název, lze pomocí zaškrtávacích políček určit, co se má zobrazit a co skrýt. Položka „Spojovací čáry“, nám zaškrtnutím zobrazí spojové čáry mezi klientem a sumarizačním serverem a druhé políčko „Referenční body“, nám zobrazí na mapě referenční body (modré čtverečky). Jedná se pouze o zobrazení, takže jejich viditelnost nemá na běh aplikace a algoritmů žádný vliv. Tento panel řeší jen vizuální stránku programu, v případě zadání mnoha sumarizačních serverů, by byla mapa se spojovacími čárami nepřehledná.

Úplně vlevo programu máme panel „Nástroje“. V tomto panelu jsou dvě tlačítka, která nám obsluhují celý program. Pro běh programu nám tedy stačí pouze jedno tlačítko a kliknutí myší na mapu. Tlačítko „Generuj“ nám spustí generování IP adres a pozic klientů, které se v zápětí ukážou na mapě. Při prvním stisku tlačítka se vygenerují jak IP adresy, tak pozice, ale při dalším stisku tlačítka se generuje jen pozice, aby se při uložení dat mohla simulovat měnící se stromová struktura v čase. Pokud bychom měnili pokaždé i IP adresy, tak by tato možnost sice byla k dispozici, ale nebyla by věrohodná s reálným chováním sítě. Druhým tlačítkem je „Rychlé uložení“, to nám ukládá data do souboru, bez nutnosti získání umístění nebo názvu souboru od uživatele. Soubor se nám ukládá do složky „Save“, v místě otevření aplikace s názvem obsahující aktuální datum, čas a s příponou „txt“. Toto uložení je vhodné pro simulování měnící se stromové struktury v čase, protože můžeme lehce vysledovat, v jakém pořadí byly soubory uloženy.

V posledním panelu „Nastavení“, má uživatel možnost nastavit jak počet referenčních bodů, tak i sumarizačních serverů. Jediné omezení je, že to musí být celé číslo, zadáno v doporučeném rozsahu uvedeném pod každým políčkem. Pokud uživatel zadá jiné číslo nebo nějaký znak, bude varovným oknem upozorněn na správný formát. Nastavené hodnoty se potvrzují tlačítkem „Nastav“.

3.5 Ovládání aplikace

Pro správnou simulaci je nejen nutné znát všechny funkce komponent, ale i jejich správné pořadí při použití. V tom by měla pomoci tato kapitola, věnovaná této problematice. Hned jako první po spuštění aplikace je nutné, aby uživatel věděl, kolik chce použít sumarizačních severů a kolik referenčních bodů. Tato volba je důležitá a volí se jako kompromis těchto dvou hodnot. Například, při zadání velkého počtu sumarizačních serverů, může nastat situace, kdy nebudou mít některé sumarizační servery pod sebou žádné klienty. Tuto situaci lze vykompenzovat zvětšením počtu referenčních bodů. Tím bude detailněji pokryta mapa referenčními body a tak se budou s větší přesností počítat i vzdálenostní vektory podle, kterých se přiřazují klienti k nadřazeným serverům. Změnou jedné, tak můžeme částečně ovlivnit chování druhé. Pokud má uživatel jasno s počtem, tak nastaví číslo do polí pro ně určené a to na panelu „Nastavení“. Pokud by chtěl zadat něco jiného (např.: znak, písmeno apod.), než celé číslo typu integer, bude upozorněn vyskakovacím oknem. Z toho plyne, že program částečně drží uživatele ve správných kolejích, aby nedošlo ke zhroucení celého programu uprostřed práce.

Když máme nastaveno, můžeme dále kliknout na tlačítko „Generuj“, a tím nám program vygeneruje pozice a IP adresy klientů, a zároveň je zobrazí na mapě. Poté může uživatel již kliknutím levého tlačítka myši v mapě rozmístit sumarizační servery. Po umístění posledního serveru se automaticky zobrazí propojovací čáry, mezi klienty a servery, a ve stejném okamžiku se v tabulkách (Příloha A) zobrazí výsledné hodnoty. Teď jsou dvě možnosti, jak dále postupovat. První možnost je, že se vygenerují pouze nové pozice klientů a IP adresy zůstanou stejné. To provedeme opětovným kliknutím na tlačítko „Generuj“. V této volbě už nemáme nadále možnost měnit pozici sumarizačních serverů. Druhá možnost je kliknout na položku „Soubor → Nový“, a tím spustit kompletně novou úlohu, jako by se aplikace otevřela poprvé. Dává nám to možnost zadat nové hodnoty v panelu „Nastavení“, a také určení nových pozic sumarizačních serverů.

Když jsme provedli všechny výše uvedené kroky, můžeme si nechat zpomaleně zobrazit spojovací čáry, mezi serverem a klientem v panelu „Vykreslovaní spojovacích čar“, nebo si zobrazit referenční body. Když chceme využít funkci pro pozdější časové zobrazení stromové struktury, klikáme střídavě na tlačítko „Generuj“ a „Rychlé uložení“. Tím se nám ukládají soubory, bez nutnosti zadání cesty uložení a názvu souboru uživatelem. Pro korektní ukončení programu, doporučuji použít položku „Soubor → Exit“.

3.6 Funkce aplikace z pohledu kódu

Z pohledu kódu je chod aplikace poněkud složitější, než se může navenek zdát. Po stisknutí tlačítka „Nastav“, aplikace dynamicky alokuje paměť o velikosti potřebné pro zadané hodnoty. Proto je velmi důležité před samotnou simulací nastavit hodnoty, aby pak nedocházelo ke kolizi, mezi daty při čtení z paměti. Dále se vypíší informace do stavového řádku, protože už známe jejich hodnoty. Po stisku tlačítka „Generuj“, se rozběhnou funkce, které mají za úkol vygenerovat IP adresy a polohy klientů na mapě. Jedná se o cyklus, který generuje čísla ne vyšší jak 223 v prvním bytu a 255 v dalších třech bytech IP adresy, aby byla dosažena autentičnost s reálnými IP adresami. Hned po vygenerování IP adresy se generuje pozice na mapě opět náhodně, ale tentokrát s čísly do 800 pro x-ovou osu a 396 pro y-ovou osu. Navíc se musí porovnávat, aby klient (červený čtvereček) neležel v oceánu a to provedeme pomocí vyhodnocení barvy podkladu původních vygenerovaných souřadnic. Pokud odpovídá bílé, je vše pořádku, ale jestli bod leží na modrém podkladu, tak se celý proces generování polohy opakuje, dokud se nedosáhne správné polohy. Tlačítko obsahuje podmínku, aby se před stiskem napřed nastavily požadované hodnoty v panelu „Nastavení“.

Dalším krokem je zadávání pozic sumarizačních serverů. Aplikace odchytává kliknutí levého tlačítka myši a přitom čte pozici kurzoru na mapě. Pokud bylo tlačítko „Generuj“ stisknuto, tak ukládá do pole hodnoty souřadnic a zároveň počítá počet kliknutí. Jestliže tlačítko „Generuj“ nebylo stisknuto, nebo už byly zadány všechny body, tak se pouze zobrazují aktuální souřadnice kurzoru do stavového řádku. Po naklikání všech bodů do mapy, aplikace spustí jednu s mnoha funkcí, která má na starost algoritmus pro výpočet stromové struktury.

Algoritmus musí v první řadě zjistit, který klient a sumarizační sever se kde nachází. To se provede pomocí funkce „ping“ (Příloha B), která ho vypočítává pomocí vzorce pro vzdálenost dvou bodů v rovině. Všechny takto získané data se ukládají do patřičných polí. Po skončení této funkce se automaticky skočí na další funkci. Ta má za úkol vyhledat nejbližší sumarizační server. Vezme postupně každý prvek vektoru klienta a sumarizačního serveru a provede se jejich rozdíl v absolutní hodnotě, aby výsledná hodnota nebyla záporná. Poté takto zpracované prvky ze dvou vektorů sečte a výsledné číslo nám udává shodu těchto dvou vektorů. Čím menší číslo, tím jsou si vektory podobnější. Pro tuto operaci se musela zavést proměnná, o velikosti trojrozměrného pole, kde první rozměr je počet sumarizačních serverů, druhý je počet klientů a třetí obsahuje různá data (např.: IP adresu, pozici, vzdálenostní vektor, apod.). Po projití všech vektorů, aplikace přiřadí na základě předchozích hodnot sumarizační servery. Přiřazení se provádí střídavě pro každý server tzn., vezme se první server, projede se celé pole klientů a vybere se klient s nejmenším číslem. To samé se provede pro druhý server, dokud nejsou přiřazeni všichni klienti. Tato metoda je důležitá ze dvou důvodů, aby každý server měl pod sebou opravdu nejbližší klienty a při naklikání sumarizačních serverů blízko sebe, aby jeden z nich neobsahoval všechny klienty a ostatní servery, by zůstaly nevyužity.

V další sérii funkcí se provádí seřazení klientů podle ID nadřazeného serveru a sestavení struktury stromu pro uložení pomocí takto seřazených dat. Takto fungují základní funkce, které jsou nezbytné pro správný běh programu. Další funkce a cykly jsou už jen vizuálního a informativního charakteru, které zde už nebudu podrobně uvádět.

4 Závěr

Výsledkem bakalářské práce je aplikace, pomocí které lze simulovat algoritmus nazvaný „binning“. Veškeré funkce a vzhled programu byly konzultovány s Ing. Danem Komosným, Ph.D. Vzdálenostní vektory jsou přehledně zobrazeny, jak je požadováno v zadání. Veškeré komplikace, na které jsem narazil v průběhu realizace, se mi bez obtíží podařilo vyřešit. Program je jednoduché ovládat i bez předchozí znalosti všech funkcí a komponent. Uživatel za chvíli přijde na všechna úskalí, které se v aplikaci vyskytují. Stromová struktura s hierarchickou sumarizací má opravdu kladné využití v přenosu multimédií po síti internet. Její výsledky jsou bezpochyby lepší než dosavadní protokoly využívané na přenos multimediálních dat.

Přehrání IPTV vyžaduje buď osobní počítač nebo set-top box připojený k TV. Video obsah je většinou komprimovaný použitím buď MPEG-2 nebo MPEG-4 kodeků a potom posíláno přes IP Multicast. Toto je metoda, ve které mohou být informace vysílány jen jednou a přijímány zároveň mnoha počítači najednou. Nově vydaný MPEG-4 kodek stále častěji nahrazuje starší MPEG-2 kodek. Projekt IPTV má v blízké budoucnosti kladné využití s nárůstem lidí, kteří budou připojeni k internetu. Přece jen sledování televize kdekoliv, s přístupem k internetu je velký komfort, který si bude moci dopřát každý. Nebude potřeba žádná televizní karta, ale jen obslužný software a připojení k internetu. Výsledná kvalita celého vysílání, se bude zlepšovat se zvyšujícími se rychlostmi připojení k internetu.

Literatura

[1] **Peterka, Jiří.** Jak funguje IPTV? Lupa.cz. [Online] 24. Srpen 2006. [Citace: 12. Listopad 2007.] http://www.lupa.cz/clanky/jak-funguje-iptv/. ISSN 1213-0702.

[2] **Komosný, Dan a Novotný, Vít.** Tree Structure for Source-Specific Multicast with feedback Aggregatio. In The Sixth International Conferenc on Networinkg. Marinuque : Computer Society, 2007. ISBN 0-7695-2805-8.

[3] **Komosný, Dan a Novotný, Vít.** Optimization of Large-Scale RTCP Feedback Reporting. In The Third International Conferenc on Wireless and Mobile Communications. Guadeloupe : Computer Society, 2007. ISBN 0-7695-2796-5.

[4] **Alexandrescu, Andrei.** Moderní programování v C++ : Šablony, generické komponenty a návrhové vzory. Praha : Computer Press, 2004. str. 344. ISBN 80-251-0370-6.

[5] **Koenig, Andrew, MOO a Barbara, E.** Rozumíme C++. Praha : Computer Press, 2003. str. 412. ISBN 80-7226-656-X.

[6] **Prata, Stephen.** Mistrovství v C++. 2. aktualizované vydání. Praha : Computer Press, 2004. str. 1028. ISBN 80-251-0098-7.

[7] **Prata, Stephen.** Mistrovství v C++. 3. aktualizované vydání. Praha : Computer Press, 2007. str. 1120. ISBN 978-80-251-1749-1.

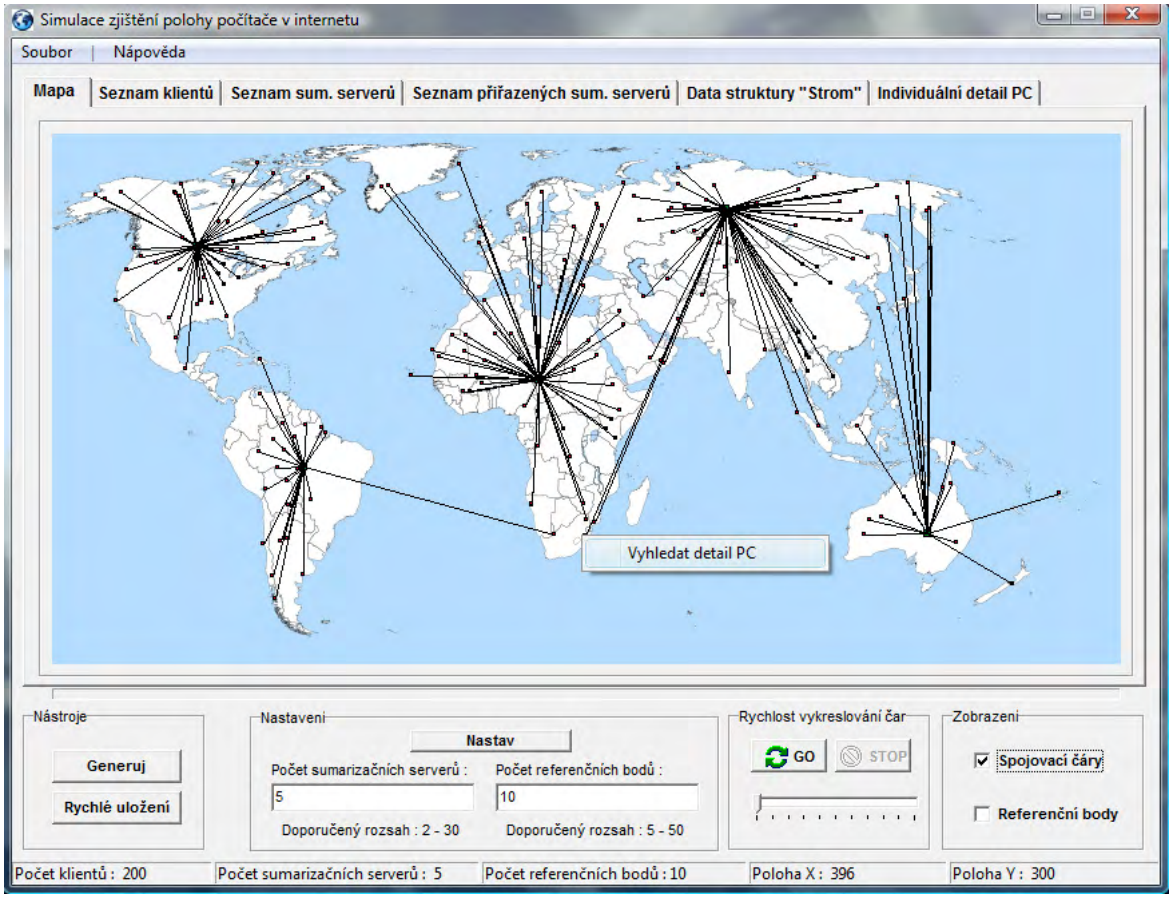
[8] **Pužmanová, Rita.** TCP/IP v kostce. 1. vydání. České Budějovice : Kopp, 2004. str. 607. ISBN 80-7232-236-2.

[9] **Virius, Miroslav.** Pasti a propasti jazyka C++. 2. aktualizované vydání. Praha : Computer Press, 2005. str. 376. ISBN 80-251-0509-1.

[10] **Kadlec, Václav.** Učíme se programovat v Borland C++ Builder a jazyce C++. Praha : Computer Press, 2002. str. 402. ISBN 80-7226-550-4.

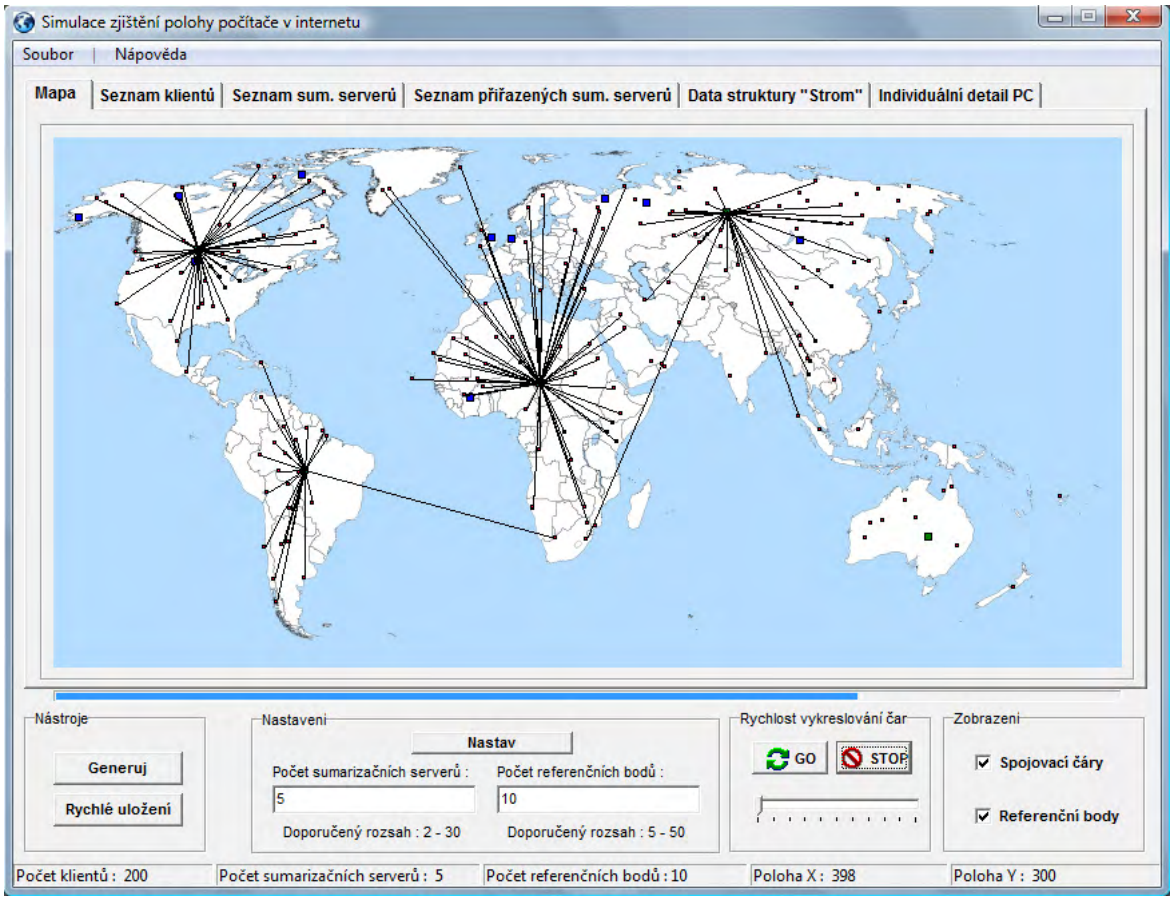
Seznam příloh

Příloha A: Obrazovky aplikace



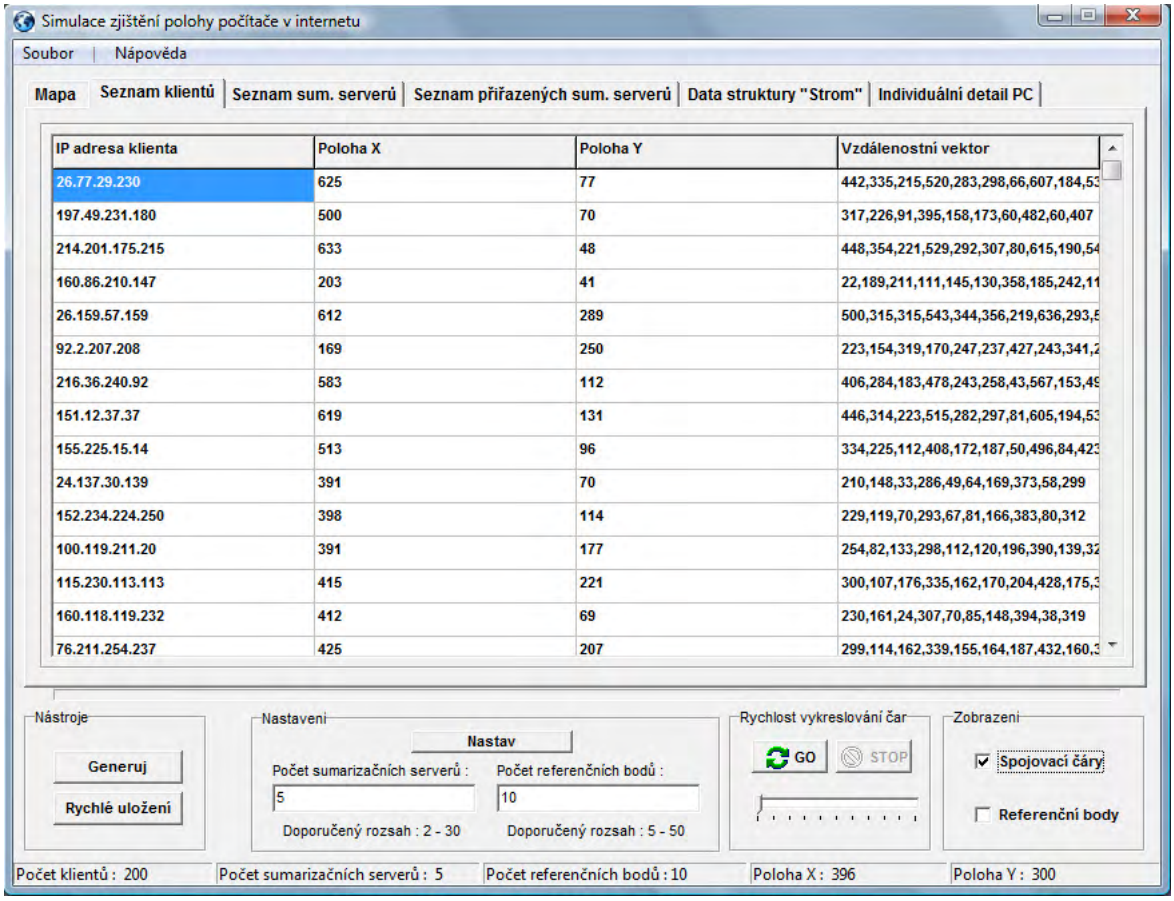
Obr. A.1: Mapa s rozmístěnými servery.

Na tomto obrázku můžeme vidět, jak se nám ne úplně vhodně rozmístili klienti pod servery. Je to částečně způsobeno malým počtem referenčních bodů. Dále zde máme zobrazenou kontextovou nabídku „Vyhledat detail PC“ po stisknutí pravého tlačítka myši. Tento detail je zobrazen v poslední záložce (Obr. A.7).



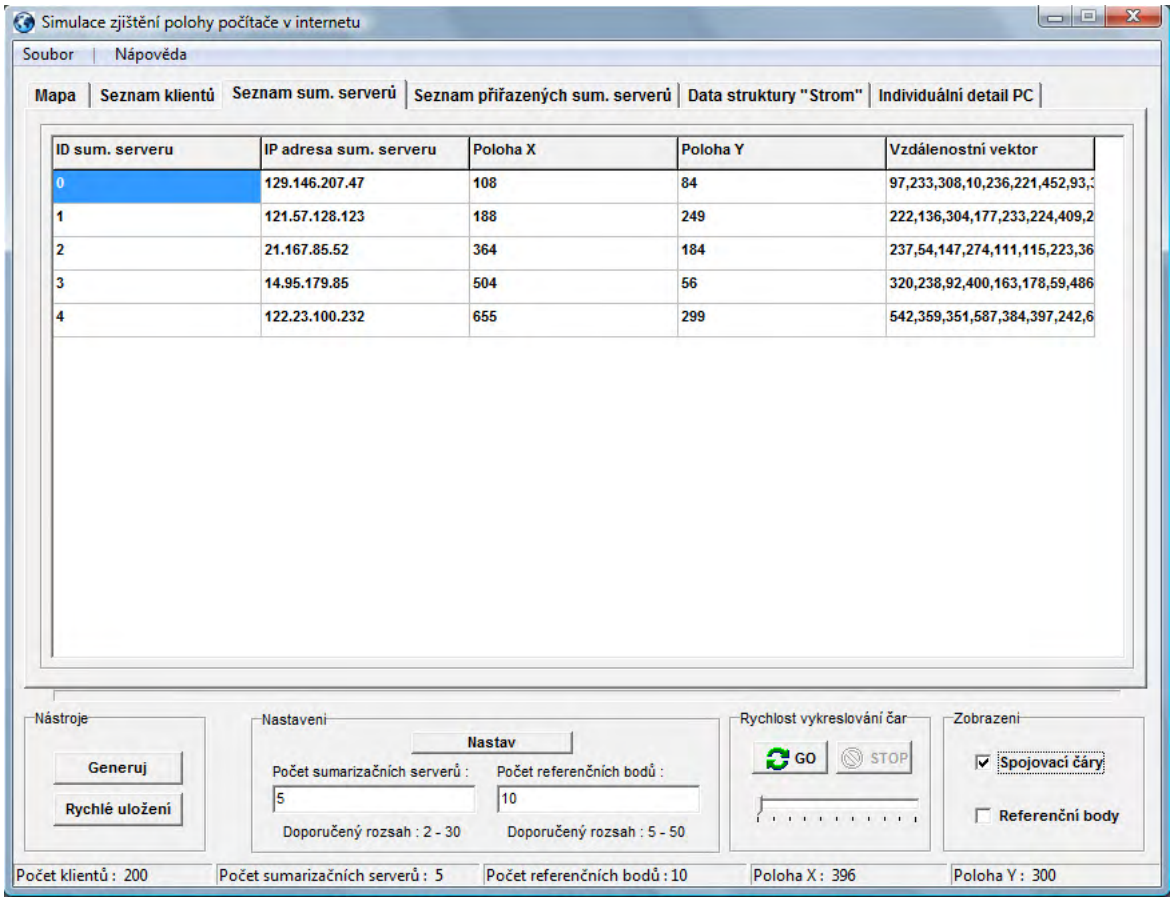
Obr. A.2: Postupné zobrazení zastavené uprostřed vykreslování.

Tento obrázek se od předchozího (Obr. A.1) moc neliší. Ale zde máme navíc zobrazeny referenční body (modré čtverečky) a využívá se funkce zpomaleného vykreslování spojovacích čar. Takže jsou propojeni jen někteří klienti a průběhový pás pod mapou nám ukazuje v jaké fázi je celkové vykreslování.



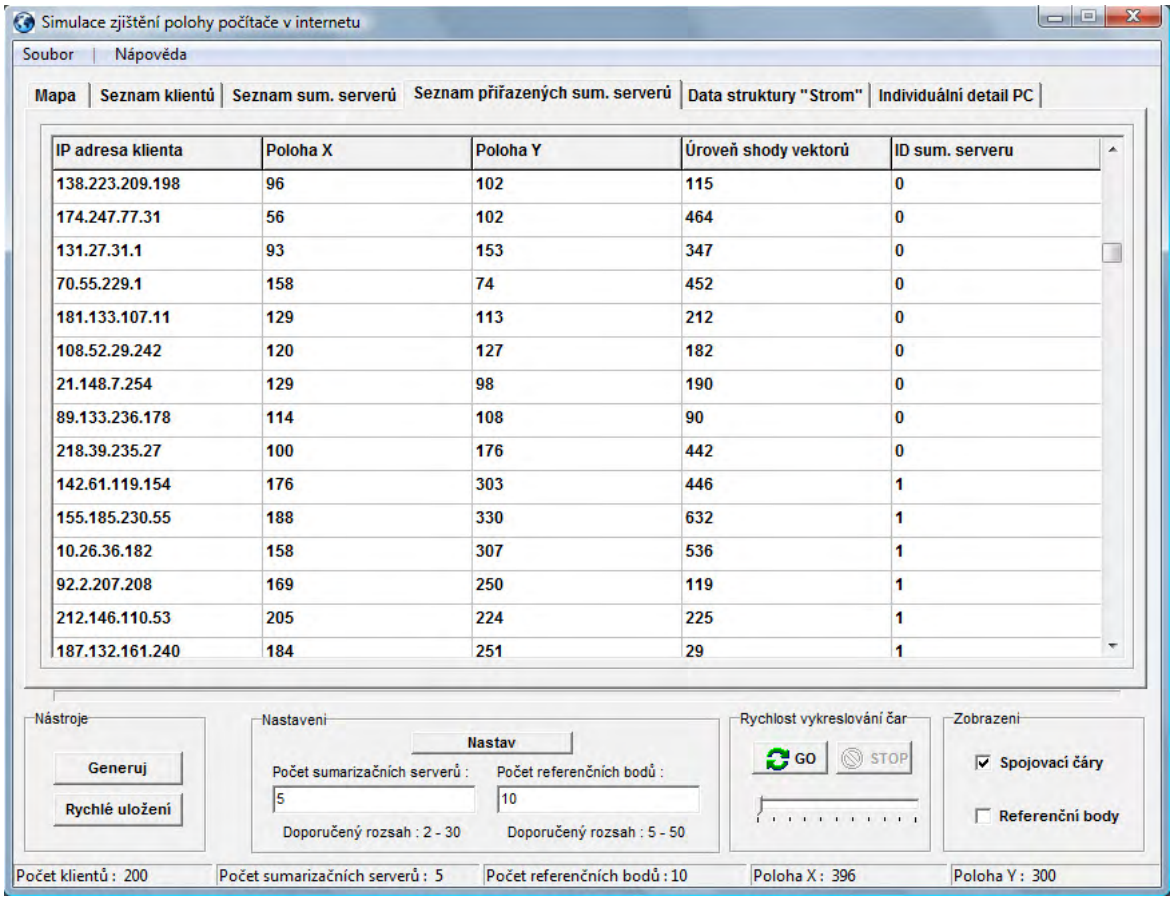
Obr. A.3: Seznam s IP adresami, souřadnicemi a vzdálenostními vektory klientů.

V prvním seznamu jsou vypsány detaily o všech klientech. Jejich počet je 200, který nelze měnit. V prvním sloupci je IP adresa v dalších dvou souřadnice umístění na mapě. V poslední sloupci je vypsán vzdálenostní vektor, kde v jednom políčku lze vidět maximálně 10 hodnot. Pro zobrazení všech hodnot se musí kliknout levým tlačítkem myši na dané políčko, a tím se vyvolá bublinová nápověda zobrazující všechny hodnoty daného vektoru.



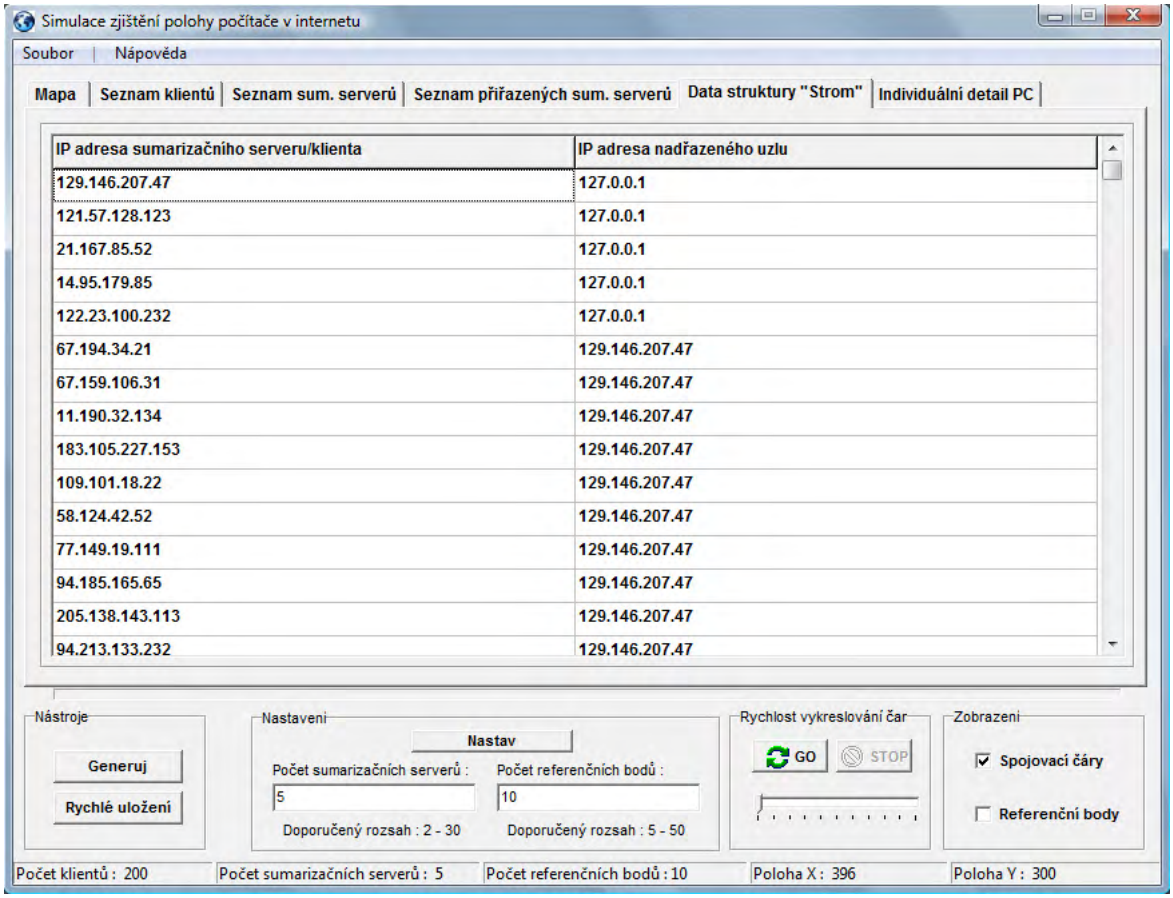
Obr. A.4: Seznam sumarizačních serverů.

Ve třetí záložce je seznam sumarizačních serverů. Jejich počet je proměnlivý a záleží na uživateli, kolik si jich zvolí. V prvním sloupci je ID sumarizačního serveru, podle kterého se pozná, kteří klienti jsou k serveru připojeni. Ve zbylých čtyřech sloupích je obsah shodný se seznamem ve druhé záložce (Obr. A.3).



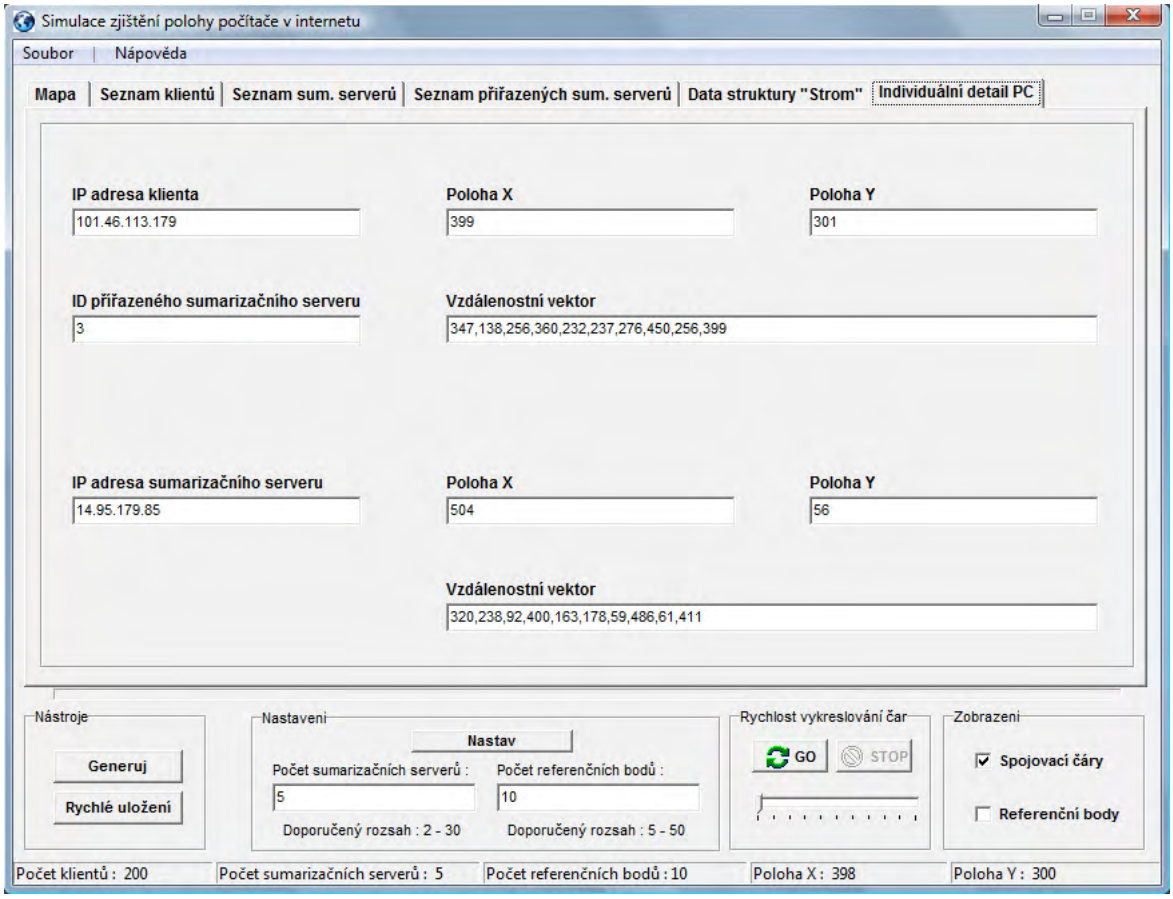
Obr. A.5: Seznam přiřazených sumarizačních serverů ke klientům.

Tento seznam nám zobrazuje v prvních třech sloupcích hodnoty klientů. V třetím sloupci je zobrazena úroveň shody vektorů, která symbolizuje rovnost vektoru klienta a sumarizačního serveru. V posledním sloupci je ID přiřazeného sumarizačního serveru. Celý seznam je vzestupně seřazen právě podle tohoto ID sumarizačního serveru.



Obr. A.6: Výsledná data pro stromovou strukturu.

V posledním seznamu jsou zobrazeny výsledná data, která jsou potřeba pro grafické vykreslení stromové struktury. V prvním sloupci jsou IP adresy sumarizačních serverů a klientů a ve druhém jim nadřazené servery.



Obr. A.7: Záložka s individuálními detaily daného klienta.

V poslední záložce jsou zobrazeny detaily určitého klienta a sumarizačního serveru jemu nadřazeného. V horní polovině je IP adresa, souřadnice, vzdálenostní vektor a ID sumarizačního serveru, který je nadřazen tomuto klientovi. Ve spodní polovině se vypisují detaily sumarizačního serveru, kterému náleží patřičné ID.

Příloha B: Zdrojový kód aplikace

```
//---------------------- Proměnné -------------------------------
int klik = 0, pocet = 2, refpocet = 5, vzd, start = 0,novy = 0,novy1 = 0, buffer, opakovani = 0, stop = 0,
gen = 0, cary = 1, rbody=0, xko, yko;
int datastrom[200][8], poleIP[200][56];
int (*feedback)[200][7],(*Target)[56], (*strom)[8], (*refbody)[2];
//---------- Počítá vzdálenost dvou bodu ---------
int vzdalenost(int Xa,int Xb,int Ya,int Yb)                    \\deklarování funkce
{
double buffer;
buffer = sqrt(pow(Xa-Xb,2)+pow(Ya-Yb,2));                      \\rovnice vzdálenosti přímky v rovině
vzd = ceil(buffer);                                            \\zaokrouhlení nahoru
if (vzd>750)                                                   \\pokud je vzdálenost větší než 750
{
vzd = random(700) + 750;                                       \\vygeneruje číslo do 1450
}
return (vzd);
}
//----------------- Generuje IP klientů ---------
void generujIP()
{
for (int m=0; m<200; m++)                                      \\cyklus pro 200 klientů
{
if (novy == 0)                                                 \\podmínky pro tlačítko soubor->nový
{
for (int n=1;n<=3;n++)
{
poleIP[m][0]=random(223)+1;                                    \\generuje 1.byte IP adresy
poleIP[m][n]=random(255)+1;                                    \\generuje zbytek IP adresy
}}
//----- Generuje polohu, aby byla na kontinentech -------
poleIP[m][4]=random(800);                                      \\generuje x-ovou souřadnici
poleIP[m][5]=random(396);                                      \\generuje y-ovou souřadnici
skok:
if ((Form1->Image1->Canvas->Pixels[poleIP[m][4]][poleIP[m][5]] == 16571572))
{
poleIP[m][4]=random(800);                                      \\porovnává barvu se souřadnicemi,
```

```
poleIP[m][5]=random(396);                    \\popř. generuje nové souřadnice
goto skok;                                   \\zacyklování generování souřadnic
}}}
//---------- Vyhledá aktuální pozici pingem na referenční body ----------
void ping()
{
for (int m=0; m<200; m++)                    \\cyklus pro 200 klientů
{
for (int n=0; n < refpocet; n++)             \\cyklus pro zadaný počet referenčních bodů
{
poleIP[m][n+6] = vzdalenost(refbody[n][0],poleIP[m][4],refbody[n][1],poleIP[m][5]);
}}                                           \\volá funkci vzdalenost
for(int r=0;r<pocet;r++)                     \\cyklus pro zadaný počet sum. serverů
{
for (int s=0; s<refpocet; s++)               \\cyklus pro zadaný počet referenčních bodů
{
Target[r][s+6] = vzdalenost(refbody[s][0],Target[r][4],refbody[s][1],Target[r][5]);
}}}                                          \\volá funkci vzdalenost
//----------- Hledá nejbližší body podle vektoru (algoritmus) --------
void najdi()
{
ping();                                      \\volá funkci ping
int shoda;
for(int x=0;x<pocet;x++)                     \\cyklus pro zadaný počet sum. serverů
{
for (int m=0;m<200;m++)                      \\cyklus pro 200 klientů
{
shoda = 0;                                   \\nulování proměnné shoda
for (int n=6;n<(refpocet+6);n++)             \\cyklus v poli obsahující vzdálenostní vektor
{
shoda += abs(Target[x][n]-poleIP[m][n]);     \\odečítá prvky ve vektoru a v abs
}
feedback [x][m][6] = shoda;                  \\uloží výsledek výpočtu
for(int o=0;o<6;o++)
{
feedback [x][m][o] = poleIP[m][o];           \\zkopíruje zbylé informace dané IP adresy
}}}}
```

```
//---------- Přiřazuje nejbližší body k sum. serverům ----------
void hledej()
{
najdi();                                        \\volá funkci hledej
for(int m=0;m<200;m++)                          \\cyklus pro 200 klientů
{
for (int n=0;n<7;n++)
{
datastrom[m][n] = feedback[0][m][n];            \\kopíruej hodnoty z jednoho pole do
druhého
}
datastrom[m][7] = 0;
}
for (int x=1;x<pocet;x++)                       \\cyklus pro počet sum. serverů
{
for(int a=0;a<200;a++)                          \\cyklus pro 200 klientů
{
for(int b=0;b<200;b++)                          \\druhý cyklus pro 200 klientů
{
if ((datastrom[a][4] == feedback[x][b][4]) && (datastrom[a][5] == feedback[x][b][5]))
{                                               \\hledá IP adresu v hlavním poli, po nalezení
if(datastrom[a][6] > feedback[x][b][6])         \\kontroluje velikost vektorů
{
datastrom [a][7] = x;                   \\pokud je vektor větší, přiřadí mu sum. server
for (int s=0;s<7;s++)                   \\a zkopíruje všechny informace o IP adrese
{
datastrom[a][s] = feedback[x][b][s];
}}}}}}}
//----- Seřazuje prvky podle ID sum. serverů ---------
void serad()
{
hledej();                                       \\volá funkci hledej
for(int c=0;c<500;c++)                          \\cyklus pro seřazení
{
for (int m=0;m<200;m++)
{
if (datastrom[m+1][7] <= datastrom[m][7])       \\hledá menší číslo ID sum. serveru
{
for(int n=0;n<8;n++)
```

```
{
buffer = datastrom[m][n];
datastrom[m][n] = datastrom[m+1][n];                    \\řazení pomocí bubblesort
datastrom[m+1][n] = buffer;
} }}}}
//---------- Sestavuje pole IP pro stromovou strukturu --------
void stromecek()
{
int pomocna;
serad();                                                \\volá funkci serad
for (int m=0;m<pocet;m++)                               \\cyklus pro počet sum. serveru
{
for (int n=0;n<4;n++)
{
if (novy1 == 0)
{
Target[m][n] = random(250);                             \\generuje IP adresu sum. serveru
}
strom[m][n] = Target[m][n];                             \\kopíruje IP adresu jako nadřazený uzel
}
strom[m][4] = 127;                                      \\IP adresa zdroje vysílání
strom[m][5] = 0;
strom[m][6] = 0;
strom[m][7] = 1;
}
for(int r=0;r<200;r++)
{
for(int s=0;s<4;s++)
{
strom[r+pocet][s] = datastrom[r][s];                    \\sestavý stromovou strukturu
pomocna = datastrom[r][7];
strom[r+pocet][s+4] = Target[pomocna][s];
}}
novy1 = 1;
}
//---------- Kreslí referenční body -----------
void referbody()
{
if ((rbody == 1) && (start == 1))                       \\testu jestli je zaškrtnuto políčko zobrazení
```

```
{
Form1->PaintBox1->Canvas->Brush->Color = clBlue;        \\nastavuje barvu pro vykreslení
for (int m=0;m<refpocet;m++)                            \\cyklus pro zadaný počet ref. bodů
{
Form1->PaintBox1->Canvas->Rectangle(refbody[m][0]-3,refbody[m][1]-
3,refbody[m][0]+3,refbody[m][1]+3);                     \\vykresluje čtverečky
}}}
//---------- Vykresluje počítače a sum. servery ----------
void kresli()
{
Form1->PaintBox1->Repaint();
Form1->PaintBox1->Canvas->Brush->Color = clGreen;       \\mění barvu na zelenou
for (int x=0;x<pocet;x++)                               \\cyklus pro počet sum. serverů
{
Form1->PaintBox1->Canvas->Rectangle(Target[x][4]-3,Target[x][5]-3,Target[x][4]+3,Target[x][5]+3);
}
for (int m=0; m<200; m++)                               \\cyklus pro 200 klientů
{
Form1->PaintBox1->Canvas->Brush->Color = clRed;         \\červená barva
Form1->PaintBox1->Canvas->Rectangle(poleIP[m][4]-1.5,poleIP[m][5]-
1.5,poleIP[m][4]+1.5,poleIP[m][5]+1.5);
}
referbody();                                            \\volá funkci referbody
}
//----- Označí nejbližší body -------
void soused()
{
int pomocna;
if ((cary == 1) && (klik == pocet))                     \\testuje jestli je zaškrtnuté políčko pro čáry
{
for(int m=0;m<200;m++)
{
pomocna = datastrom[m][7];                              \\bere ID sum. serverů
Form1->PaintBox1->Canvas->MoveTo(Target[pomocna][4],Target[pomocna][5]);
Form1->PaintBox1->Canvas->LineTo(datastrom[m][4],datastrom[m][5]);
}}}                                                     \\vykresluje spojovací čáry
//---------- Vynuluje data sum. serverů ----------
void Targetzero()
{
```

```
for (int i=0; i < 200; i++)                    \\cyklus pro 200 klinetů
{
for (int k= 0; k < 8; k++)                     \\cyklus pro 8 sloupců
{
datastrom[i][k];                               \\pole s daty
}
for (int j = 0; j < 56; j++)                   \\caklus pro počet hodnot ve vektoru
{
poleIP[i][j];                                  \\umístění hodnyty vektorů
}}
for (int m=0;m<pocet;m++)                      \\cyklus pro pocer sum. serverů
{
for (int n=0;n<(refpocet+6);n++)               \\cyklus pro počet  referenčních bodů
{
Target[m][n] = 0;                              \\pole sum. serverů
}}}
//---------- Zobrazuje detaily v tabulkách----------
void detaily ()
{
Form1->StringGrid1->RowCount = 201;            \\počet řádků
Form1->StringGrid1->FixedRows = 1;
Form1->StringGrid2->RowCount = 201;
Form1->StringGrid2->FixedRows = 1;             \\počet sloupců
Form1->StringGrid3->RowCount = 201+pocet;
Form1->StringGrid3->FixedRows = 1;
Form1->StringGrid4->RowCount = pocet+1;
Form1->StringGrid4->FixedRows = 1;
String vektor,vektor1;
if (klik == pocet)
{
for (int k=0;k<pocet;k++)
{
vektor = Target[k][6]   ;                      \\skládá vektor pro vykreslení
Form1->StringGrid4->Cells[0][k+1] = k;
Form1->StringGrid4->Cells[1][k+1] = AnsiString (Target[k][0]) +'.'+ AnsiString(Target[k][1]) +'.'+
AnsiString(Target[k][2]) +'.'+ AnsiString(Target[k][3]);     \\dává tečky v IP mezi byty
Form1->StringGrid4->Cells[2][k+1] = AnsiString(Target[k][4]);
Form1->StringGrid4->Cells[3][k+1] = AnsiString(Target[k][5]);
for (int l=0;l<refpocet-1;l++)                 \\zobrazuje ostatní detaily sum. serveru
```

```
{
vektor = vektor +','+ Target[k][l+7];
}
Form1->StringGrid4->Cells[4][k+1] = vektor;                    \\vypíše vektor do políčka
}
for (int m=0;m<200;m++)
{
vektor1 = poleIP[m][6];                                        \\obdobně akorát pro klienty viz. výše
Form1->StringGrid1->Cells[1][m+1] = AnsiString(poleIP[m][4]);
Form1->StringGrid1->Cells[2][m+1] = AnsiString(poleIP[m][5]);
Form1->StringGrid1->Cells[0][m+1] = AnsiString (poleIP[m][0]) +'.'+ AnsiString(poleIP[m][1]) +'.'+
AnsiString(poleIP[m][2]) +'.'+ AnsiString(poleIP[m][3]);
Form1->StringGrid2->Cells[0][m+1] = AnsiString (datastrom[m][0]) +'.'+ AnsiString(datastrom[m][1])
+'.'+ AnsiString(datastrom[m][2]) +'.'+ AnsiString(datastrom[m][3]);
for(int n = 0 ; n<refpocet-1 ; n++)
{
vektor1 = vektor1 +','+ poleIP[m][n+7];
}
for (int o=1;o<5;o++)
{
Form1->StringGrid2->Cells[o][m+1] = datastrom[m][o+3];
}
Form1->StringGrid1->Cells[3][m+1] = vektor1  ;
}}
for (int p=0;p<200+pocet;p++)                                  \\cyklus pro 200 klientů
{
Form1->StringGrid3->Cells[0][p+1] = AnsiString (strom[p][0]) +'.'+ AnsiString(strom[p][1]) +'.'+
AnsiString(strom[p][2]) +'.'+ AnsiString(strom[p][3]);
Form1->StringGrid3->Cells[1][p+1] = AnsiString (strom[p][4]) +'.'+ AnsiString(strom[p][5]) +'.'+
AnsiString(strom[p][6]) +'.'+ AnsiString(strom[p][7]);
}}
//------------------------------------- Tlačítko GENERUJ ------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (start == 1)
{
gen = 1;
generujIP();                                                   \\volá funkci na generování IP adres
kresli();                                                      \\funkce a vykreslení bodů na mapě
Form1->Hledatdetail1->Enabled = true;
```

```
novy = 1;
}
else
{
MessageDlg("Nejdřív nastavte požadované hodnoty v sekci 'Nastavení'",mtInformation, TMsgDlgButtons()
<< mbOK, 0);
}
if(klik == pocet)                    \\porovná počet kliknutí na mapu se zadanou hodnotou
{
stromecek();
soused();                                    \\vykresluje propojovací čáry
detaily();                                   \\zobrazuje informace do tabulek
}}
//---------------------------------- Obrázek mapy ----------------------------------
void __fastcall TForm1::PaintBox1MouseDown(TObject *Sender,
TMouseButton Button, TShiftState Shift, int X, int Y)
{
//------- zobrazuje aktualni pozici a urcuje souradnice sum. serveru ----------------
if ((klik != pocet) && (gen == 1))           \\kontroluje klinutí na tlačítko generuj
{
Form1->StatusBar1->Panels->Items[3]->Text = Format("Poloha X :  %d",ARRAYOFCONST((X)));
                                             \\zobrazuje souřadnice ve stavovém řádku
Form1->StatusBar1->Panels->Items[4]->Text = Format("Poloha Y :  %d",ARRAYOFCONST((Y)));
          \\zobrazuje souřadnice ve stavovém řádku
if (Button == mbRight)                       \\odchytává kliknutí pravým tlačítkem
{
xko = X;
yko = Y;
}
if (Button == mbLeft)                        \\odchytává kliknutí levým tlačítkem
{
Form1->StatusBar1->Panels->Items[3]->Text = Format("Poloha X :  %d",ARRAYOFCONST((X)));
Form1->StatusBar1->Panels->Items[4]->Text = Format("Poloha Y :  %d",ARRAYOFCONST((Y)));
Form1->PaintBox1->Canvas->Brush->Color = clGreen;
Target[klik][4] = X;                         \\zapisuje souřadnice do pole
Target[klik][5] = Y;
Form1->PaintBox1->Canvas->Rectangle(X-3,Y-3,X+3,Y+3);
klik++;
kresli();                                    \\vola funkci kresli
```

```
}
Form1->ProgressBar1->Position = 0;                    \\nuluje průběhový pás
Form1->TrackBar1->Position = 1;
opakovani = 0;
stop = 0;
}
else
{
Form1->StatusBar1->Panels->Items[3]->Text = Format("Poloha X :  %d",ARRAYOFCONST((X)));
Form1->StatusBar1->Panels->Items[4]->Text = Format("Poloha Y :  %d",ARRAYOFCONST((Y)));
if (Button == mbRight)
{
xko = X;                                              \\ukládá souřadnice kurzoru do proměnné
yko = Y;
}
Form1->ProgressBar1->Position = 0;                    \\nastavení pozice průběhového pásu
Form1->TrackBar1->Position = 1;
opakovani = 0;                                        \\nulování pomocných proměnných
stop = 0;
}
if ((klik == pocet) && (gen == 1))                    \\kontroluje stisk tlačítka generuj
{
opakovani = 0;
stop = 0;
Form1->Timer1->Enabled = false;                       \\vypne timer
Form1->TrackBar1->Position = 0;
Form1->BitBtn1->Enabled = true;                       \\zviditelní tlačítko go
Form1->BitBtn2->Enabled = false;
stromecek();
stromecek();                                          \\volání funkcí
soused();
detaily();
}}
//----------------------------Tlacitko NASTAV ----------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
try                                                   \\odchytávání neplatných znaků
{
pocet = StrToInt(Form1->LabeledEdit11->Text);         \\čtení zadané hodnoty
```

```
refpocet = StrToInt(Form1->LabeledEdit10->Text);
}
catch (...)                                          \\zobrazení zprávy při špatném rozsahu
{
MessageDlg("Zadejte číslo",mtError, TMsgDlgButtons() << mbOK, 0);
}
if ((pocet >= 2) && (pocet <= 30) && (refpocet >= 5) && (refpocet <= 50))
{                                                    \\kontroluje doporučený rozsah
refbody = new int [refpocet][2];                     \\alokace dynamických pamětí
feedback = new int [pocet][200][7];
Target = new int [pocet][56];
strom = new int [200+pocet][8];
start = 1;
Targetzero();
klik = 0;
generujrefbody();                                    \\generování referenčních bodů
Form1->StatusBar1->Panels->Items[1]->Text = "Počet sumarizačních serverů :  " + IntToStr(pocet);
                                                     \\zobrazení hodnot do stavového řádku
Form1->StatusBar1->Panels->Items[2]->Text = "Počet referenčních bodů : " + IntToStr(refpocet);
}
else
{
MessageDlg("Zadejte číslo v doporučeném rozsahu",mtError, TMsgDlgButtons() << mbOK, 0);
Form1->LabeledEdit11->Text = 2;                      \\předdefinovaný minimální rozsah, který se
Form1->LabeledEdit10->Text = 5;                      \\zobrazí v editech
}}
//----------------------Tlačítko ULOZ-------------------------
void __fastcall TForm1::Ulo1Click(TObject *Sender)
{
if ((klik == pocet) && (gen == 1))
{
String buffer;
if (Form1->SaveDialog1->Execute())                   \\vyvolá dialog pro uložení
{
TStringList * to_file = new TStringList ();          \\ukazatel na seznam hodnot k zápisu
for (int i = 0; i < 200+pocet; i++)                  \\cyklus pro 200 klientů
{
```

```
to_file-> Add(buffer+IntToStr(strom[i][4]) +'.'+ IntToStr(strom[i][5]) +'.'+ IntToStr(strom[i][6]) +'.'+
IntToStr(strom[i][7]) +' '+ IntToStr(strom[i][0]) +'.'+ IntToStr(strom[i][1]) +'.'+ IntToStr(strom[i][2]) +'.'+
IntToStr(strom[i][3]));
}
to_file->SaveToFile(SaveDialog1->FileName);          \\uložení pod zadaným jménem
}}
else
MessageDlg("! Nejsou žádná data k uložení !",mtError, TMsgDlgButtons() << mbOK, 0);
}                                                    \\varovaní pro uložení
//---------------------- Menu soubor->exit ----------------------
void __fastcall TForm1::Exit1Click(TObject *Sender)
{
delete [] feedback;                                  \\ruší dynamickou paměť
delete [] Target;
delete [] strom;
delete [] refbody;
Application->Terminate();                            \\ukončuje aplikaci
}
//---------------------------------- Menu nápověda -> o programu --------------------
void __fastcall TForm1::Oprogramu1Click(TObject *Sender)
{
AboutBox->ShowModal();                               \\otvírá okno s informacemi o programu
}
//------------------------------ Záložky -----------------------------
void __fastcall TForm1::TabbedNotebook1Click(TObject *Sender)
{
//------- prekresluje paintbox "oprava nevýhody paintboxu" -------
if (start == 1)                                      \\testuje klinutí na nastav
{
kresli();                                            \\volá funkci kresli
if (klik == pocet)                                   \\pokud jsou zadány všechny sum. servery
{
Form1->ProgressBar1->Position = 0;
Form1->TrackBar1->Position = 1;
opakovani = 0;
stop = 0;
soused();                                            \\vykreslí spojovací čáry
detaily();                                           \\zobrazí detaily v tabulkách
}}}
```

```
//---------------------------------- Menu soubor -> nový ------------------
void __fastcall TForm1::Nov1Click(TObject *Sender)
{
if (start == 0)
{
MessageDlg("Již máte novou úlohu, můžete pokračovat v sekci 'Nastavení'",mtInformation,
TMsgDlgButtons() << mbOK, 0);                    \\hláška při stisknutí hned po otevření aplikace
}
if (start == 1)
{
Targetzero();                                    \\maže data o sum. serverech
Form1->CheckBox1->Checked = true;                \\vrací nastavení pro zobrazení
Form1->Hledatdetail1->Enabled = false;
Form1->CheckBox2->Checked = false;
delete [] feedback;                              \\maže dynamicky alokovanou paměť
delete [] strom;
delete [] refbody;
start = 0;                                       \\nuluje pomocné proměnné
klik = 0;
gen = 0;
stop = 0;
Form1->PaintBox1->Repaint();                     \\čistí mapu
Form1->TabbedNotebook1->PageIndex = 0;
Form1->LabeledEdit1->Text = "";                  \\maže zobrazené detaily
Form1->LabeledEdit2->Text = "";
Form1->LabeledEdit3->Text = "";
Form1->LabeledEdit4->Text = "";
Form1->LabeledEdit5->Text = "";
Form1->LabeledEdit6->Text = "";
Form1->LabeledEdit7->Text = "";
Form1->LabeledEdit8->Text = "";
Form1->LabeledEdit9->Text = "";
Form1->StringGrid1->RowCount = 1;                \\nastavuje počet sloupců a řádků
Form1->StringGrid2->RowCount = 1;
Form1->StringGrid3->RowCount = 1;
Form1->StringGrid4->RowCount = 1;
novy = 0;
novy1 = 0;
```

```
}}
//------------------------------ Časovač ----------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
int pomocna;
if ((klik == pocet) && (gen == 1))                    \\testujezmáčknuté tlačítka
{
Form1->Timer1->Interval = Form1->TrackBar1->Position;
Form1->ProgressBar1->Position ++;                     \\posouváprběh v Progressbaru
pomocna = datastrom[opakovani][7];
Form1->PaintBox1->Canvas->MoveTo(Target[pomocna][4],Target[pomocna][5]);
Form1->PaintBox1->Canvas->LineTo(datastrom[opakovani][4],datastrom[opakovani][5]);
}                                                     \\vykreslování spojovacích čar
if (opakovani == 199)                                 \\po dokončení poslední čáry
{
Form1->Timer1->Enabled = false;                       \\vypne časovač
Form1->ProgressBar1->Position = 0;                    \\vynuluje průběhový pás
opakovani = 0;                                        \\vynuluje pomocné proměnné
stop = 0;
Form1->BitBtn1->Enabled = true;                       \\zviditelní potřebná tlačítka
Form1->BitBtn2->Enabled = false;
}
else
{
Form1->BitBtn1->Enabled = false;
opakovani++;
}}
//--------------------------- Při otevření apikace --------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
// ---- Popíše tabulky -------
Form1->TabbedNotebook1->PageIndex = 0;
Form1->StringGrid1->Cells[0][0] = "IP adresa klienta";
Form1->StringGrid1->Cells[1][0] = "Poloha X";
Form1->StringGrid1->Cells[2][0] = "Poloha Y";
Form1->StringGrid1->Cells[3][0] = "Vzdálenostní vektor" ;
Form1->StringGrid2->Cells[0][0] = "IP adresa klienta" ;
Form1->StringGrid2->Cells[1][0] = "Poloha X";
Form1->StringGrid2->Cells[2][0] = "Poloha Y";
```

```
Form1->StringGrid2->Cells[3][0] = "Úroveň shody vektorů";
Form1->StringGrid2->Cells[4][0] = "ID sum. serveru";
Form1->StringGrid3->Cells[0][0] = "IP adresa sumarizačního serveru/klienta";
Form1->StringGrid3->Cells[1][0] = "IP adresa nadřazeného uzlu";
Form1->StringGrid4->Cells[0][0] = "ID sum. serveru";
Form1->StringGrid4->Cells[1][0] = "IP adresa sum. serveru";
Form1->StringGrid4->Cells[2][0] = "Poloha X";
Form1->StringGrid4->Cells[3][0] = "Poloha Y";
Form1->StringGrid4->Cells[4][0] = "Vzdálenostní vektor" ;
}
//---------- Tlačítko GO ----------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
if ((stop == 0) && (klik == pocet))                    \\testuje zmáčknutí tlačítek
{
Form1->Timer1->Interval = Form1->TrackBar1->Position;\\hlídá nastavenou rychlost vykreslování
Form1->Timer1->Enabled = true;
Form1->BitBtn2->Enabled = true;
}
else
{
Form1->Timer1->Enabled = true;                         \\začne vykreslovat kde skončil
}
if ((opakovani == 0) && (klik == pocet))
{
kresli();                                              \\smaže mapu před vykreslováním
}}
//---------- Tlačítko STOP ----------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Form1->Timer1->Enabled = false;                        \\zastavuje časovač
Form1->BitBtn1->Enabled = true;                        \\zviditelňuje tlačítko go
stop = 1;
}
//---------- Zaškrtávací políčko spojovací čáry ----------
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
if (Form1->CheckBox1->Checked == false)                \\kontroluje jestli je zaškrtnuté
{
```

```
cary = 0;
Form1->Timer1->Enabled = false;
Form1->GroupBox3->Enabled = false;
Form1->BitBtn1->Enabled = false;
Form1->BitBtn2->Enabled = false;
Form1->ProgressBar1->Position = 0;
Form1->TrackBar1->Position = 1;
opakovani = 0;
stop = 0;
if (klik == pocet)                                  \\jsou zadány všechny sum. servery
{
kresli();                                           \\překresluje mapu
}}
else
{
cary = 1;
Form1->GroupBox3->Enabled = true;
Form1->BitBtn1->Enabled = true;
Form1->BitBtn2->Enabled = true;
if (klik == pocet)
{
kresli();
soused();                                           \\vykresluje spojovací čáry
}}}
//------------------- Kontextové menu po stisknu pravého tlačítka myši---------
void __fastcall TForm1::Hledatdetail1Click(TObject *Sender)
{
String vektor2,vektor3;                             \\pomocné pro výpis vektorů
int find = 0,id;
for (int m=0;m<200;m++)
{
for (int x=(xko-3);x<(xko+3);x++)                   \\rozmezí kolem bodu na mapě
{
for (int y=(yko-3);y<(yko+3);y++)
{
if ((poleIP[m][4] == x) && (poleIP[m][5] == y))     \\hledá souřadnice klienta
{
find = 1;
vektor2 = poleIP[m][6];
```

```
Form1->LabeledEdit1->Text = AnsiString (poleIP[m][0]) +'.'+ AnsiString(poleIP[m][1]) +'.'+
AnsiString(poleIP[m][2]) +'.'+ AnsiString(poleIP[m][3]);   \\vypisuje IP adresu
Form1->LabeledEdit2->Text = poleIP[m][4];                  \\vypisuje souřadnice
Form1->LabeledEdit3->Text = poleIP[m][5];
for (int i=0 ; i < refpocet-1; i++)                        \\cyklus pro počet ref. bodů
{
vektor2 = vektor2 +','+ poleIP[m][i+7];                    \\vykresluje vzdálenostní vektory
}
Form1->LabeledEdit4->Text = vektor2;
}
if ((datastrom[m][4] == x) && (datastrom[m][5] == y))      \\hledá souřadnice klienta
{
id = datastrom[m][7];
}
vektor3 = Target[id][6];
Form1->LabeledEdit9->Text = id;
Form1->LabeledEdit5->Text = AnsiString (Target[id][0]) +'.'+ AnsiString(Target[id][1]) +'.'+
AnsiString(Target[id][2]) +'.'+ AnsiString(Target[id][3]);  \\vypisuje IP adresu
Form1->LabeledEdit6->Text = Target[id][4];
Form1->LabeledEdit7->Text = Target[id][5];
for (int f=0; f < refpocet-1; f++)
{
vektor3 = vektor3 +','+ Target[id][f+7] ;
}
Form1->LabeledEdit8->Text = vektor3;
}}}
if (find == 0)
{
MessageDlg("Detail PC nenalezen, upřesněte polohu ...",mtError, TMsgDlgButtons() << mbOK, 0);
}                                                          \\chybná zpráva
else
{
MessageDlg("Detail PC nalezen ...",mtInformation, TMsgDlgButtons() << mbOK, 0); \\potvrzení nálezu
Form1->TabbedNotebook1->PageIndex = 5;
}}
//------------------------- Zaškrtávací políčko referenční body----------------------------------
void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
if(Form1->CheckBox2->Checked == true)                      \\testuje jestli je zaškrtlé
```

```
{
rbody = 1;
referbody();                                    \\volá funkci referbody
}
else                                            \\pokud není zaškrtlé
{
rbody = 0;
Form1->PaintBox1->Repaint();                    \\vymaže mapu
if (start == 1)
{
kresli();                                       \\zobrazí body na mapě
soused();                                       \\zobrazí spojovací čáry
}}}
//---------------------------------- Tlačítko rychlé uložení -------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
 {
String datum;
SYSTEMTIME st;
GetLocalTime (&st);                             \\získává aktuální čas a datum
CreateDirectory("Save",NULL);
datum = AnsiString(IntToStr(st.wYear) +'-'+I ntToStr(st.wMonth) +'-'+ IntToStr(st.wDay) +' '+
IntToStr(st.wHour) +'-'+ IntToStr(st.wMinute) +'-'+ IntToStr(st.wSecond));      \\spojuje jako název
if ((klik == pocet) && (gen == 1))              \\testuje kliknutí na tlačítka
{
TStringList * to_file = new TStringList ();
for (int i = 0; i < 200+pocet; i++)             \\připravuje seznam pro uložení
{
to_file->Add(IntToStr(strom[i][4]) +'.'+ IntToStr(strom[i][5]) +'.'+ IntToStr(strom[i][6]) +'.'+
IntToStr(strom[i][7]) +' '+ IntToStr(strom[i][0]) +'.'+ IntToStr(strom[i][1]) +'.'+ IntToStr(strom[i][2]) +'.'+
IntToStr(strom[i][3]));
}
to_file->SaveToFile("Save/"+datum+".txt");      \\udává cestu pro uložrení
MessageDlg("Data uložena ve složce 'Save'", mtInformation, TMsgDlgButtons () << mbOK, 1);
}
else
MessageDlg("! Nejsou žádná data k uložení !",mtError, TMsgDlgButtons() << mbOK, 0);
}
```

```
//---------------------------- Zobrazuje bublinovou nápovědu -----------------------------------
void __fastcall TForm1::StringGrid1MouseUp(TObject *Sender, TMouseButton Button,
TShiftState Shift, int X, int Y)
{
int r,s;
Form1->StringGrid1->MouseToCell(X,Y,s,r);                    \\rozpoznává výběr políčka
Form1->StringGrid1->Hint = Form1->StringGrid1->Cells[s][r];      \\zobrazuje celý vektor
}
//---------------------------- Zobrazuje bublinovou nápovědu -----------------------------
void __fastcall TForm1::StringGrid4MouseUp(TObject *Sender, TMouseButton Button,
TShiftState Shift, int X, int Y)
{
int r,s;
Form1->StringGrid4->MouseToCell(X,Y,s,r);                    \\rozpoznává výběr políčka
Form1->StringGrid4->Hint = Form1->StringGrid4->Cells[s][r];      \\zobrazuje celý vektor
}
\\-------------------------------------------------------------------------------------------------
```