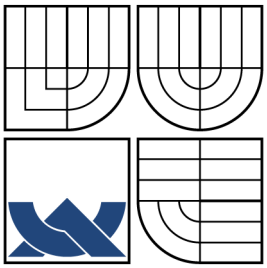
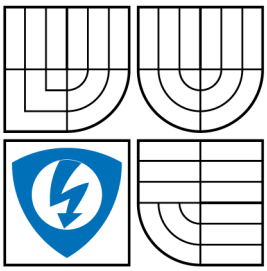


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

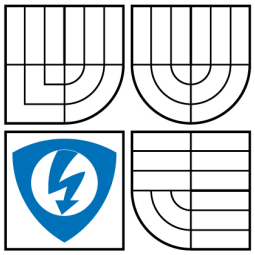
INTERNETOVÉ SOUŘADNICOVÉ SYSTÉMY
INTERNET COORDINATE SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. MARTIN KRAJČÍR
AUTHOR

VEDOUCÍ PRÁCE — ING. RADIM BURGET
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Krajčír	***ID:***	83592
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Internetové souřadnicové systémy

POKYNY PRO VYPRACOVÁNÍ:

Senamte se se centralizovanými a distribuovanými souřadnicovými systémy a stručně shrňte jejich vlastnosti. Seznamte se s problematikou hierarchické agregace. Vytvořte simulaci, která ověří přesnost předpovědi pozice stanice v síti. Výsledky vyneste přehledně do grafu. Vytvořte simulaci agoritmu Vivaldi a vystavte jej s pomocí technologie Java Web Start na webové stránky. Navrhněte nový souřadnicový systém pro potřeby hierarchické agregace. Simulaci opět vystavte na webové stránky. Popište způsob integrace navrženého řešení do metody hierarchické agregace.

DOPORUČENÁ LITERATURA:

[1] NOVOTNY, V., KOMOSNY, D. Optimization of Large-Scale RTCP Feedback Reporting in ICWMC 2007. ICWMC 2007 - The Third International Conference on Wireless and Mobile Communications. Guadeloupe, 2007, ISBN: 0-7695-2796-5

[2] T. S. Eugene Ng and Hui Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches", INFOCOM'02, New York, NY, June 2002

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Radim Burget

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

UPOZORNĚNÍ:

ABSTRAKT

Síťový souřadnicový systém je účinný nástroj pro předpovědi internetových vzdáleností s omezeným počtem měření. Práce se zabývá distribuovaným souřadnicovým systémem hodnoceným relativní chybou. Na základě získaných znalostí ze simulačního prostředí byl navržen algoritmus výpočtu vlastního souřadnicového systému. K testování tohoto algoritmu byly využity simulační data a RTT hodnoty ze sítě PlanetLab. Měření dokázala, že shlukové uzly dosahovaly pozitívních výsledků umělých souřadnic s omezeným počtem spojení mezi uzly. Práce poskytuje rozbor praktického nasazení vlastního souřadnicového systému v síti s hierarchickou agregací. Výsledná simulační aplikace byla vystavená na webových stránkách výzkumných projektů Ústavu telekomunikací FEKT VUT v Brně.

KLÍČOVÁ SLOVA

Vivaldi, Shlukování, K-means, Síťové souřadnice, Simulace

ABSTRACT

Network coordinates (NC) system is an efficient mechanism for prediction of Internet distance with limited number of measurement. This work focus on distributed coordinates system which is evaluated by relative error. According to experimental results from simulated application, was created own algorithm to compute network coordinates. Algorithm was tested by using simulated network as well as RTT values from network PlanetLab. Experiments show that clustered nodes achieve positive results of synthetic coordinates with limited connection between nodes. This work propose implementation of own NC system in network with hierarchical aggregation. Created application was placed on research projects web page of the Department of Telecommunications.

KEYWORDS

Vivaldi, Clustering, K-means, Netvork Coordinates, Simulation

KRAJČÍR, M. *Internetové souřadnicové systémy. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 63 s. Vedoucí diplomové práce Ing. Radim Burget.*

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Internetové souřadnicové systémy 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 Ing. Radimovi Burgetovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Sieťový súradnicový (NC) systém je účinný nástroj pre rozsiahle Internetové aplikácie, ktoré môžu mať prospech zo schopnosti predpovedania RTT vzdialenosti k uzlom, bez nutnosti priameho kontaktu s nimi a s minimálnym počtom meraní. V NC systéme má každý uzol pridelenú skupinu čísiel, ktoré predstavujú súradnice a sieťová vzdialenosť medzi dvomi uzlami môže byť vypočítaná z ich vlastných súradníc. NC systém tak významne redukuje sieťové preťaženie vzniknuté réžiou a ponúka najmä úžitok pre rozsiahle Peer-to-Peer aplikácie zahrňujúce distribúciu alebo zdielanie súborov, počítačové hry v režime multiplayer a pri výbere najvhodnejšieho serveru.

V tejto práci bolo zamerané na jeden konkrétny súradnicový systém, ktorý patrí do skupiny decentralizovaných a to je jedna z jeho hlavných výhod. Naproti tomu tento návrh algoritmu Vivaldi prináša veľkú nevýhodu v podobe času potrebného na správnu konvergenciu súradníc a preto sa práca zaoberá aj metódami upravujúcimi pôvodný algoritmus. Pri simuláciach tak bola odhalená jeho funkčnosť pri rôznych situáciach a prípadné komplikácie, ktoré by mohli nastať pri praktickom nasadení takéhoto systému.

V prvej a teoretickej časti je popísaná funkčnosť algoritmu Vivaldi aj s možnými úpravami, ktoré zlepšujú jeho slabé stránky. V druhej časti je naznačené praktické využitie NC systému pre zostavenie stromu hierarchickej agregácie použitej pri signalizácii navrhovaného IPTV systému. V tretej kapitole boli navrhnuté tri simulačné scenáre pre Vivaldi s adaptívnym časovým krokom. Na základe zistených vlastností bol zostrojený výpočet vlastného súradnicového systému, ktorý využíva metódu vytvárania zhlukov. Navrhnutý algoritmus bol testovaný pomocou dát zo siete PlanetLab a výsledky boli prezentované v príslušných grafoch. Záver práce sa zaoberá rozborom návrhu implementácie do systému zberu signalizačných správ v rámci internetovej televízie.

1 DECENTRALIZOVANÝ SIEŤOVÝ POZIČNÝ SYSTÉM

Viaceré Internetové aplikácie, najmä tie založené na Peer-to-Peer (P2P) technológii, môžu vyťažiť zo znalosti RTT (round-trip time) k ostatným uzlom bez nutnosti priameho kontaktu s nimi. Priame meranie RTT je nevhodné, pretože náročnosť merania môže prevyšovať nad výhodami získanými z tohto merania. Vivaldi je jednoduchý algoritmus, ktorý prideľuje umelé súradnice uzlom tak, že vzdialenosť medzi súradnicami dvoch uzlov predstavuje komunikačnú vzdialenosť medzi nimi. Vivaldi je plne distribuovaný systém, ktorý nepotrebuje žiadnu pevnú sieťovú infraštruktúru, ani zvláštne uzly [7].

1.1 Úvod do Vivaldi

Umelé súradnicové systémy umožňujú Internetovým uzlom predvídať RTT vzdialenosti. Uzly počítajú súradnice v niektorom súradnicovom priestore tak, že vzdialenosť medzi dvomi uzlami predpovedá RTT vzdialenosť medzi nimi. Táto vzdialenosť je v súradnicovom systéme označovaná ako predpovedané RTT [7].

Výsledné skreslenie latencie paketov znemožňuje použiť dvojdimenzionálne (2-D) súradnice uzlov, ktoré by predpovedali latenciu precízne. Preto musí mať umelý súradnicový systém ošetrený spôsob výberu súradníc tak, aby sa minimalizovala predikčná chyba. Súradnice nemusia byť limitované len na 2-D rozmer. Vivaldi umožňuje čiastočne zmenšiť chyby rozšírením súradnic o výšku [7].

Súradnice sú obzvlášť nápomocné pri veľkom počte potencionálnych serverov alebo malom množstve dát. Distribučné systémy ako KaZaA, BitTorrent a CoDeeN sú príkladom systémov, ktoré ponúkajú veľké množstvo duplicitných serverov. CFS a DNS majú menší počet duplicitných severov s menším dátovým objemom [7]. Všetky tieto distribučné systémy môžu vyťažiť zo sieťových súradníc.

Vivaldi počíta súradnice, ktoré predpokladajú latenciu s nízkou chybovosťou a bolo vyvinuté pre Chord peer-to-peer vyhľadávací systém, ktorý používa súradnice, aby sa predišlo zbytočnému kontaktovaniu vzdialených uzlov pri vyhľadávaní [8]. Vivaldi dosahuje rovnakú chybovosť ako GNP [7][17], systém založený na orientačných bodoch v súradnicovom systéme, a navyše Vivaldi neuvažuje s použitím ďalších sieťových prvkov.

1.2 Popis algoritmu pre Vivaldi

Vivaldi priraďuje každému uzlu umelé súradnice tak, že vzdialenosti v súradnicovom systéme medzi dvomi uzlami predpovedajú prenos paketu RTT medzi nimi. Žiaden nízkodimenzionálny súradnicový priestor nedokáže s použitím Vivaldi predpovedať RTT úplne presne, pretože napríklad Internetová latencia porušuje trojuholníkovú nerovnosť [7]. Algoritmus sa preto snaží nájsť súradnice s minimálnou predikčnou chybou.

1.2.1 Predikčná chyba

Uvažujme L_{ij} ako aktuálne RTT medzi uzlami i, j a x_i ako súradnice priradené uzlu i. Môžeme tak definovať chybu v súradniciach použitím funkcie celkovej kvadratickej chyby [7]:

$$E = \sum_i \sum_j (L_{ij} - \|x_i - x_j\|)^2 \tag{1.1}$$

Kde $\|x_i - x_j\|$ je vzdialenosť medzi súradnicami uzlov i a j vo vybranom súradnicovom priestore.

1.2.2 Centralizovaný algoritmus

Najprv naznačíme jednoduchý a centralizovaný algoritmus, ktorý zmenšuje chybu z rovnice 1.1. Vivaldi je distribuovaná verzia tohto algoritmu. Celkovú kvadratickú chybu môžme prirovnať k potenciálnej energii pružnosti. Túto chybu môžme minimalizovať posúvaním uzlov podľa pôsobiacej sily. Minimálna energia pružnosti sa zhoduje s minimálnou chybou pridelených súradníc, kde nie je zaručené, že simuácia nájde globálne minimum a tak systém môže zostať v lokálnom minime[7].

Teraz znázorníme centralizovaný algoritmus trochu presnejšie. Definujme vektor sily F_{ij} tak, že pružina medzi uzlami i a j pôsobí na uzol i. Z Hookovho zákona môžme dokázať, že pre silu F platí nasledovné [7]:

$$F_{ij} = (L_{ij} - \|x_i - x_j\|) \times u(x_i - x_j) \tag{1.2}$$

Skalárna veličina $(L_{ij} - \|x_i - x_j\|)$ určuje posun pružiny od ostatních uzlov. Táto veličina je významná pre silové uplatnenie pružiny medzi i a j. Jednotkový vektor $u(x_i - x_j)$ určuje smer sily od uzlu i.

$$F_i = \sum_{i \neq j} F_{ij} \tag{1.3}$$

V každom intervale algoritmus posúva uzol x_i o kúsok v súradnicovom priestore v smere vektoru F_i a potom zase prepočíta všetky silové vektory [7]. Súradnice na konci časového intervalu sú nasledovné:

$$x_i = x_i + F_i \times t \tag{1.4}$$

Kde t je dĺžka časového intervalu. Veľkosť t určuje ako ďaleko sa uzol posunie v jednotlivom časovom intervalom. Pri návrhu Vivaldi je dôležité nájdenie vhodného t.

Algoritmus 1: *Centralizovaný algoritmus*

Vstup : Matica latencie L a inicializačné súradnice uzlu x

Výstup: Upresnené súradnice v uzle x.

```
1  while error(L, x) > tolerancia do
2      foreach i do
3          F = 0;
4          foreach j do
5              e = L_ij − ‖x_i − x_j‖;
6              F = F + e × u(x_i − x_j);
7          end
8          x_i = x_i + t × F;
9      end
10 end
```

1.2.3 Jednoduchý algoritmus pre Vivaldi

Každý uzol, ktorý je súčasťou Vivaldi, simuluje svoj vlastný pohyb v pružinovom systéme. Uzly si urdžujú svoje vlastné súradnice začínajúce s nejakými inicializačnými hodnotami [7]. Ľubovolný uzol komunikujúci s iným meria RTT k tomuto uzlu a rovnako sa oboznamuje s aktuálnymi súradnicami.

Vstupom distribuovaného Vivaldi algoritmu je postupnosť prvkov (samples). Výsledkom je posunutie uzlu o malý časový krok (time step) prislúchajúcou pružinou. Kažý tento pohyb znižuje chybovosť uzla s ohľadom na ostatné uzly v systéme [7]. Uzly pokračujú v tejto komunikácii s ostatnými a tak sa približujú k súradniciam, ktoré pomerne presne predpovedajú skutočné vzdialenosti RTT.

Ak uzol i so súradnicami x_i získava informácie o uzle j, ktorý má súradnice x_j a meria sa RTT rtt, tak aktualizácia súradníc sa bude počítať podľa tohto pravidla:

$$x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times u(x_i - x_j) \tag{1.5}$$

Pretože všetky uzly začínajú na rovnakej pozícii, tak ich je treba vhodným spôsobom rozlíšiť. Vivaldi to rieši definovaním $u(0)$ vektoru jednotkovej dĺžky s náhodne vybraným smerom [7]. Preto dva uzly začínajúce na rovnakej pozícii budú od seba oddialené v ľubovolom smere.

Nasledujúci algoritmus 2 znázorňuje pseudokód jednoduchého Vivaldi. Táto procedúra je volaná vždy, keď je dostupná nová nameraná hodnota RTT. Uvažuje sa s konštantnou hodnotou časového kroku δ.

Algoritmus 2: *Jednoduché Vivaldi*

Vstup : Nameraná vzdialenosť rtt a súradnice x_j vzdialeného uzlu j
Výstup: Aktualizované súradnice pre uzol x_i

1 // výpočet chyby pre tento prvok
2 $e = rtt - \|x_i - x_j\|$;
3 // nájdenie smeru sily spôsobenej chybou
4 $dir = u(x_i - x_j)$;
5 // vektor sily je úmerný chybe
6 $f = dir \times e$;
7 // posunutie súradníc o malý krok v smere sily
8 $x_i = x_i + \delta \times dir$;

Algoritmus 3 zobrazuje výpočet súradníc s použitím optimalizácie časového kroku. Najprv je vypočítaná váha príslušného prvku založená na lokálnej a vzdialenej chybe. Ďalej je potrebné vypočítať novú relatívnu chybu a určiť lokálnu chybu, ktorá zase bude použitá v ďalšom cykle pri konvergencii súradníc. Zvyšok algoritmu je už zhodný s predchádzajúcim jednoduchým Vivaldi.

1.2.4 Výškový vektor

Výškový vektor je založený na Euklidovských súradniciach rozšírených o výšku. Ak majú dva uzly rovnakú výšku, tak vzdialenosť medzi nimi je Euklidovská vzdialenosť plus dve výšky [7]. Toto je podstatný rozdiel medzi výškovým vektorom a pridaním ďalšieho rozmeru do Euklidovského priestoru. Výškový vektor je implementovaný predefinovaním zvyčajných vektorových operácií:

$$[x, x_h] - [y, y_h] = [(x - y), x_h + y_h] \tag{1.6}$$

$$\|[x, x_h]\| = \|x\| + x_h \tag{1.7}$$

$$\alpha \times [x, x_h] = [\alpha x, \alpha x_h] \tag{1.8}$$

Algoritmus 3: *Vivaldi s adaptívnym časovým krokom*

Vstup : Nameraná vzdialenosť rtt, súradnice x_j a odhadovaná chyba e_j vzdialeného uzlu j

Výstup: Aktualizované súradnice x_i a odhad chyby e_i pre uzol i

1 // konštanty c_e a c_c sú ladiace parametre výpočtu
2 // vyváženosť lokálnej a vzdialenej chyby
3 $w = e_i/(e_i + e_j)$;
4 // výpočet relatívnej chyby prvku
5 $e_s = |\|x_i - x_j\| - rtt|/rtt$;
6 // aktualizácia váhoveho posuvu lokálnej chyby
7 $e_i = e_s \times c_e \times w + e_i \times (1 - c_e \times w)$;
8 // aktualizácia lokálnych súradníc
9 $\delta = c_c \times w$;
10 $x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times u(x_i - x_j)$;

Uzol, ktorý nájde svoje súradnice príliš blízko k ostatným uzlom v okolí, sa už nemá kam posúvať. Toto platí pre Euklidovský priestor. Pri vektorovom systéme sú sily prerušené z Euklidovskej roviny nahradené silami z výšok a tým je uzol posunutý od tejto roviny [7].

1.3 Problém krátkych vzdialeností

V tejto kapitole sa budeme zaoberať presnosťou určenia súradníc v závislosti na rôznych vzdialeností uzlov v rámci systému Vivaldi. Na základe experimentov bolo dokázané, že obmedzený rozsah vzdialeností neovplivňuje presnosť predpokladaných súradníc vo Vivaldi, ale krátke vzdialenosti trpia väčšou predikčnou chybou ako dlhé vzdialenosti medzi uzlami [4].

Problém s krátkymi vzdialenosťami zaniká pri pozičných systémoch založených na centrálnych uzloch [4]. Príkladom takéhoto systému je GNP [17]. Tento systém má oproti Vivaldi nevýhodu pridávaním centrálneho uzlu pre určitú skupinu prvkov v sieti a tak v sytéme vzniká centralizované riadenie. Naproti tomu je Vivaldi decentralizovaný systém bez závislosti na uzloch v sieti. Preto sa v tejto kapitole budeme zaoberať problematikou dosiahnutia rovnakých výsledkov relatívnej chyby pre Vivaldi v porovnaní s GNP bez nutnosti použitia centálnych uzlov.

Hlavná myšlienka tejto problematiky je združovanie uzlov v rozličných skupinách s odlišným rozsahom vzdialeností a návrhu množiny súradníc, ktoré sú určené pre krátke vzdialenosti a skupiny súradníc určené pre dlhé vzdialenosti [4]. Počet množín súradníc može byť ľubovolný v závislosti na škálovitosti rozsahov.

Na základe týchto predpokladov bol navrhnutý systém Pharos [4], decentralizovaný a hierarchycký sieťový súradnicový systém.

1.3.1 Dopad rozsahu vzdialeností na Vivaldi

Predpovedanie presnosti sieťových súradnicových systémov je často označované pojmom relatívna chyba (RE) predpovedanej vzdialenosti k reálnej latencii nameranej v rámci Internetu. Táto chyba je definovaná podľa nasledovného vzorca [7][17]:

$$R_{\mathrm{ERROR}} = \frac{P_{\mathrm{DIST}} - M_{\mathrm{DIST}}}{\min(P_{\mathrm{DIST}}, M_{\mathrm{DIST}})} \tag{1.9}$$

Kde P_{DIST} udáva predpokladanú vzdialenosť a M_{DIST} predstavuje nameranú vzdialenosť.

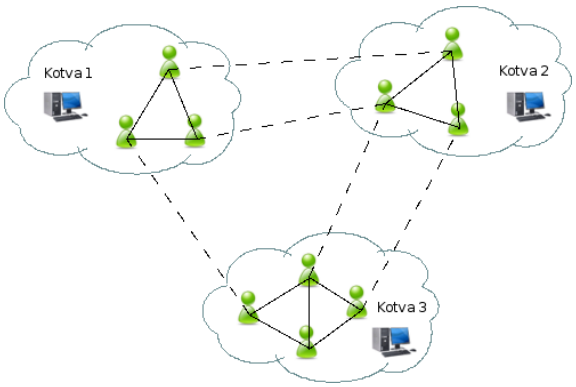
1.3.2 Systém Pharos

V tejto časti sa budeme zaoberať prístupom nazvaným Pharos, ktorý využíva rozličné množiny súradníc pre rovnaký uzol. Každý uzol má priradené viacnásobné súradnice, jedni pre globálne merítko a tie ostatné pre kratšie vzdialenosti. Pre zjednosušenie uvažuje metóda Pharos s dvomi množinami súradníc [4]. Máme tak dva

druhy spojenia. Základné vrstvové spojenie, ktoré je vytvorené náhodne medzi vrstvami, kde každá vrstva predstavuje množinu uzlov, ktoré nie sú od seba príliš vzdialené. Druhým typom je lokálne spojenie množiny uzlov, ktoré je vytvorené náhodne medzi uzlami rovnakej vrstvy.

Uzly sa musia spojiť medzi jednotlivými vrstvami a aj v rámci lokálnej množiny kvôli získaniu dvoch typov súradníc s rozdielnym rozsahom predpokladov. Pre pripojenie novej základovej vrstvy sa vytvorý k náhodných spojení k susedným vrstvám. Pre vytvorenie lokálnej množiny sa vyberú uzly pomáhajúce k zoskupovaniu uzlov, ktoré nazývame kotvy [4]. Každý stabilný uzol, ktorý je schopný odpovedať na ICMP ping správy, môže zastávať funkcie kotvy tak, ako skutočný Internetový server. Uzly sú potom riadené týmito kotvami a zoskupované do rôznych množín.

Každý nový uzol meria vzdialenosti medzi všetkými kotvami a po obdržaní týchto hodnôt nájde najlbižšiu kotvu a pripojí sa k množine [4]. Pre toto pripojenie sa spojí s k náhodnými susedými uzlami z rovnakej množiny prvkov. Hierarchická štruktúra modelu Pharos je naznačená na obrázku 1.1.



Obrázok 1.1: Vzájomné prepojenie uzlov v systéme Pharos

Algoritmus Vivaldi je použitý medzi jednotlivými vrstvami ako aj v lokálnej množine uzlov v rámci vrstvy. Súradnice vypočítané medzi vrstvami nazývame *globálne sieťové súradnice*. Súradnice odpovedajúce lokálnej množine označujeme ako *lokálne sieťové súradnice* [4] obsahujúce kratší rozsah vzdialeností.

Nalsedujúci algoritmus zobrazuje postup pripojenia nového uzlu do systému Pharos. Uzol najprv kontaktuje Rendezvous uzol (RP) systému, tak ako v každom P2P systéme [4]. Po obdržaní zoznamu kotevných uzlov z sa začnú vymeriavať vzdialenosti k týmto kotvám a vyberie sa najbližšia množina pre pripojenie. Nový uzol sa pripojí k obom typom množín, vrstvovej aj lokálnej, použitím gossip protokolu [6]. Celý proces tohto pripojenia je znázornený v algoritme 4.

Po získaní globálnych a lokálnych sieťových súradníc môžeme určiť vzdialenosť medzi ľubovolnými dvomi uzlami. Ak sú dva uzly z rovnakej množiny, tak vzdi-

alenosť medzi nimi sa predpokladá z lokálnych súradníc. Naopak, ak sú dva uzly z rozdielnych množín, tak vzdialenosť medzi nimi je predpokladaná z globálnych súradníc [4]. Tento hierarchický prístup zlepšuje presnosť predpokladaných vzdialeností, ktoré môžeme definovať nasledovne:

$$d = \begin{cases} \|x_{\text{A.local}} - x_{\text{B.local}}\| & skupina_A = skupina_B \\ \|x_{\text{A.global}} - x_{B.global}\| & skupina_A \neq skupina_B \end{cases} \tag{1.10}$$

Metóda Pharos výrazne zlepšuje presnosť predpokladaných súradíc pre krátke vzdialenosti a zanecháva rovnakú presnosť predpokladov ako Vivaldi pre stredné a dlhé vzdialenosti [4].

Algoritmus 4: *Pharos*

```
1  Connect_to_Rendezvous_Point(rp);
2  Get_Anchors_List(rp);
3  Nearest_Anchor_Distance = ∞;
4  foreach i in Anchors do
5  |   d(i) = Measure Distance to i;
6  |   if Nearest_Anchor_Distance > d(i) then
7  |   |   Nearest_Anchor_Distance = d(i);
8  |   |   Nearest_Anchor = i;
9  |   end
10 end
11 Join_Cluster(Nearest_Anchor);
12 while forever do
13 |   j = random(local neighbors of i);
14 |   x_i.local = vivaldi(rtt, x_j.local, e_j.local );
15 |   j = random(global neighbors of i);
16 |   x_i.global = vivaldi(rtt, x_j.global, e_j.global );
17 |   Wait(Update_Interval);
18 end
```

1.4 Náhodné odpájanie uzlov

V tejto kapitole sa budeme zaoberať problematikou náhodného odpájania uzlov v sieťových súradnicových systémoch. Experimenty so systémom Vivaldi odhalili, že pri častom odpájaní uzlov bude narušená presnosť predpovedaných súradníc [5], čo zapríčiní nestabilitu systému pre mnohé aplikácie. Ako opatrenie k tomuto častému a hlavne náhodnému odpájaniu uzlov bol navrhnutý sieťový súradnicový systém Myth. Exprimenty ukázali, že Myth prekonáva Vivaldi v porovnaní relatívnej chyby pri vysokom náhodnom odpájaní uzlov bez obmedzenia výkonu pre stabilné prostredie, v ktorom sa predpokladá dlhotrvajúce pripojenie uzlov bez náhlych výpadkov [5].

Čas konvergencie vo Vivaldi zaberie desiatky sekúnd než sa uzol dostane do stabilného stavu v systéme. Je veľmi pravdepodobné, že proces konvergencie uzlu bude poznačený susednými odpájajúcimi a práve pripájajúcimi sa uzlami, tak ako je to bežné v P2P systémoch. Novo pripájaný uzol s ešte nepresnými súradnicami bude referenčným uzlom pre ostatné v nádeji zlepšenia ich súradníc. Tento scenár samozrejme zväčší nestabilitu celého systému. A tak doba konvergencie spolu s nepresnými inicializačnými súradnicami uzlov môže podstatne narušiť presnosť predpokladaných súradníc [5] v prostredí s vysokým náhodným odpojovaním uzlov.

Použitím algoritmu Myth sú inicializačné súradnice blízko konečnej ideálnej pozícii a preto sa výrazne skráti doba konvergencie. Pri porovnaní výkonnosti systémov Vivaldi a Myth bolo zistené, že Myth prekonáva Vivaldi pre každú metódu merania chyby [5]. Navyše v oboch prostrediach, v stabilnom aj s vysokým počtom odpojovaním a pripojovaním uzlov, dosahuje Myth rovnkú presnosť predpokladaných súradníc.

1.4.1 Systém Myth

Použitie rovnakých inicializačných súradníc pre všetky uzly nie je príliš vhodný heuristický postup pri zavádzaní systému, aj keď uzly budú rozptýlené v nasledujúcich krokoch [5]. Preto v prostredí s častým odpojovaním uzlov je veľký podieľ uzlov v stave zmätku s nepresnými sieťovými súradnicami. Ako opatrenie tohto problému Myth riadi proces zavádzania každého uzlu so súradnicami, ktoré sú blízko ideálnych [5]. Je použitá schéma, ktorá využíva súradnice uzlov už zapojených do systému. Táto schéma pre inicializáciu uzla je použitá ešte pred samotným nasadením Vivaldi. Každý nový uzol už potom potrebuje len niekoľko krokov ku konvergencii. Takto je celková predikčná chyba redukovaná, čo spôsobí väčšiu stabilitu systému.

Keď sa nový uzol A pripája do vrstveného systému, tak sa najprv vytvorí spojenie pomocou protokolu Gossip [6] [5] so susednými uzlami, ktoré sa už v systéme

nachádzajú dlhšie. Po obdržaní dostatočného počtu susedných uzlov meria uzol A svoje RTT vzdialenosti k L susedným ($L >$ N, kde N je dimenzia priestoru). Použitím L nameraných vzdialeností môže uzol A vypočítať svoje inicializačné súradnice I_{CA}, ktoré minimalizujú celkovú chybu medzi nameranými a vypočítanými vzdialenosťami [5] od uzlu A k týmto L susedným.

Myšlienka inicializačných súradníc je podobná k výpočtu súradníc pri GNP uzloch alebo podobných LBA(landmark-based algorithm) systémoch [17]. Je tu samozrejme podstatný rozdiel a to ten, že Myth nevyžaduje rozmiestnenie špeciálnych centrálnych uzlov. Nasledujúci algoritmus 5 zobrazuje proces pripájania nového uzlu k celému systému [5].

Algoritmus 5: *Myth*

```
1  Connect_to_Rendezvous_Point(rp);
2  Get_Neighbor_Candidates_List(rp);
3  Join_Overlay();
4  Make_Connection_to_Neighbors();
5  foreach i in Neighbors do
6  |   d(i) = Measure Distance to i;
7  end
8  x_i = Initial_Coordinates_Prediction(d(.),NCs(.));
9  Wait(Update_Interval);
10 while forever do
11 |   j = random(neighbors of i);
12 |   x_i = vivaldi(rtt, x_j, e_j );
13 |   Wait(Update_Interval);
14 end
```

Výpočtové zaťaženie tohto systému je zanedbateľné, pretože čas spotrebovaný na výpočet inicializačných súradníc I_{C} je oveľa kratší ako interval medzi jednotlivými aktualizáciami v každom kroku [5], ktoré sú zvyčajne rádovo v sekundách pri štandardnom Vivaldi.

1.5 Zhlukovanie entít

Zhlukovanie je tradičný pohľad bez dohľadovej metódy analýzy dát. Tieto algoritmy sú reprezentované sadou dátových entít, ktoré musia byť zoskupené podľa určitej predstavy o podobnosti. Algoritmus má prístup len k vlastnostiam popisujúcim každú entitu. Nie je predaná žiadna informácia o tom, kde by sa mali nachádzať jednotlivé entity v priebehu rozdelenia do zhlukov. [19].

Proces zaraďovania konkrétnych, či abstraktných entít do tried podľa rozličných vlastností sa nazýva zhlukovanie. Zhlukom sa nazýva skupina entít(objektov), ktoré sú navzájom podobné a zároveň sa čo najviac líšia od entít v iných zhlukoch. Cieľom zhlukovania je možnosť posudzovať jednotlivé entity ako časti zhluku a teda možnosť práce so zhlukom namiesto práce s množstvom entít [10][15].

Problém zhlukovania sa vyskytuje v rozdielnych aplikáciach ako data mining, získavania znalostí z databáz (Knowledge Discovery in Databases), kompresie dát, rozpoznávanie a klasifikovanie obrazcov. Dojem dosiahnutia dobrých zhlukov záleží od aplikácie a v tomto smere existuje mnoho metód na nájdenie zhlukov na základe rôznych kritérií, ktoré môžme rozdeliť na náhodné a systematické [11].

Pre dosiahnutie maximálnej užitočnosti a zrozumiteľnosti zhlukovania, je nevyhnutné vedieť, ktoré algoritmy zhlukovania sú vhodné pre rôzne typy úloh. Pri rozhodovaní sa pre konkrétny algoritmus sa berie ohľad aj na spôsob zobrazenia výsledkov a najmä ich zrozumiteľnosť [15]. Pre náš účel bude úplne postačovať zhlukovací algoritmus založený na metóde k priemetov označovaný tiež ako K-means.

1.5.1 Zhlukovanie podľa algoritmu K-means

Metóda K-means je jedným z veľkého počtu zhlukovacích algoritmov. Každý hodnotený objekt je reprezentovaný práve jedným bodom a každý sledovaný atribút jednou súradnicou. Táto metóda je vhodná hlavne na veľké množiny dát u ktorých očakávame rozloženie do malého počtu zhlukov.

Základnou charakteristikou tejto metódy je, že podobnosť jednotlivých objektov a zhlukov sa meria ako ich vzdialenosť vzhľadom na priemernú hodnotu zhluku. Cieľom metódy je minimalizácia zhlukovacieho kritéria, ktorým je najčastejšie suma štvorcov chýb (rozptylov) všetkých objektov vzhľadom na priemerný objekt [15].

Metóda K-means je bežne použitá pre automatické rozdelenie dátovej sady do k skupín. Najprv sa vyberie k inicializačných stredov zhlukov, ktoré označujeme centroidy a potom sa iteratívne opakujú nasledujúce dva kroky [19]:

- Každá entita d_i je priradená k najbližšiemu centroidu
- Každý centroid C_j je aktualizovaný a posunutý na stred určený z entít, ktoré vytvorili daný zhluk.

Algoritmus konverguje vo chvíli keď nenastávajú ďalšie zmeny v priradení entít do zhlukov. Je dokázané, že algoritmus je konečný [19]. Pre numerické atribúty entít sa používa metrika Euklidovskej vzdialenosti a pre entity popísané symbolmi sa používa Hammingova vzdialenosť [19].

Algoritmus nezaručuje výber najlepšieho riešenia a ani nevedie vždy k rovnakému výsledku. Preto sa vo väčšine prípadoch počíta viacnásobne s rôznym inicializačným nastavením centroidov a uchováva sa najlepší výsledok. Pre výber najlepšie dosiahnutého riešenia je použitie vhodne zvolenej metriky, napr. súčet všetkých vzdialeností od jednotlivých bodov k ich centroidom. Riešenie s najmenšou takto vypočítanou hodnotou je najlepšie dosiahnuté v lokálnom optime.

Hlavným problémom tejto metódy je správne zvolenie počtu zhlukov k, ktorý je podrobnejšie popísaný v [9]. Avšak pre naše účely sme použili oveľa jednoduchšiu metódu, ktorá plynie z našich požiadaviek a je podrobnejšie popísaná v kapitole 3.2.1.

Algoritmus 6: *K-means*

Vstup : počet požadovaných zhlukov k a vektor inštancií, ktoré chceme rozdeliť

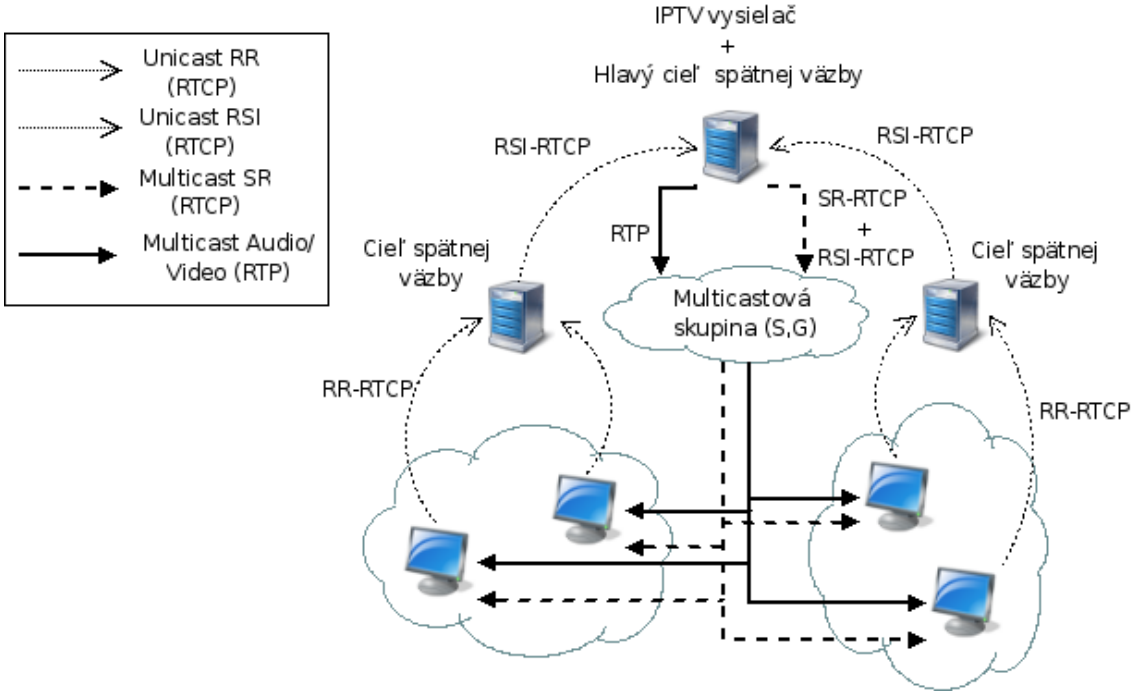
Výstup: k zhlukov (disjunktné množiny vstupných entít)

```
1 inicializuj k centroidov náhodnými súradnicami
2 while zmena najbližšieho centroidu ľubovolnej entity do
3 |   foreach entita do
4 |   |   priraď entite najbližší centroid
5 |   end
6 |   zmena polohy centroidov podľa priemeru priradených entít
7 end
8 entity priradené rovnakému centroidu tvoria jeden zhluk, získame k zhlukov
```

2 VYUŽITIE SÚRADNICOVÉHO SYSTÉMU PRE IPTV

Súčasné RTP a RTCP protokoly boli navrhnuté za účelom distribúcie multimediálnych dát a pre spätný zber informácií od prijímačov. Žiaľ, pre rozsiahle IPTV spojenia môže tento spätný zber zaberať pomerne značný čas. Preto bola navrhnutá metóda založená na hierarchickej agregácii, ktorá značne redukuje tento čas a ponúka veľmi rýchlu metódu pre monitorovanie kvality od prijímačov [2][3][12][16].

Hierarchická agregácia, ktorá zlepšuje celkovú štruktúru uzlov v porovnaní so systémom SSM(Source-specificMulticast) [14][13][18], prináša do siete nový prvok - takzvaný cieľ signalizácie "feedback target" (FT) [12][18]. FT ďalej rozdeľujeme na pasívne, ktoré nemajú žiadnu úlohu a čakajú na pokyn k aktivácii, a aktívne, ktoré tvoria stromovú štruktúru. Koreň tohto stromu je tvorený koreňovým cieľom signalizácie označovaným ako RFT (root feedback target) [12]. Štruktúra zapojenia uzlov pri hierarchickej agregácii je znázornený na obrázku 2.1.



Obrázok 2.1: Štruktúra hierarchickej agregácie

2.1 Použitie FT staníc

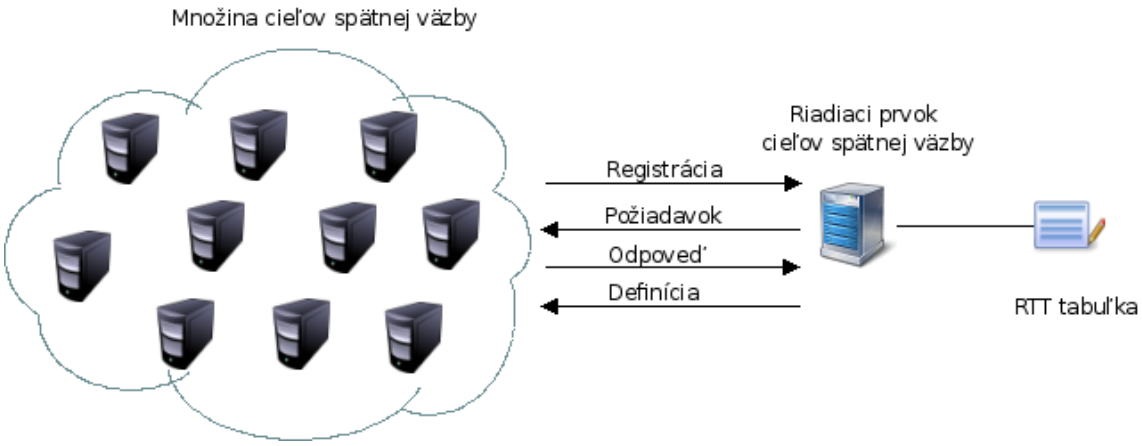
Cieľ spätnej väzby FT je prvok, ktorý v štandarde RTP/RTCP vôbec neexistuje. V sieti ich je niekoľko a ich úlohou je zber správ RR (receiver report) prijímačov

o kvalite príjmu, odstrániť redundantné a irelevantné časti informácie a v podobe správy RSI (receiver summary information) doručiť vysielaču štatistiky o kvalite príjmu [1][12].

Tieto FT stanice potom tvoria hierarchickú stromovú štruktúru, kde FT najnižšej vrstvy stromu zbierajú správy od prijímačov. Ostatné FT potom majú nastarosť agregáciu informácie z predchádzajúcich FT a predávajú ju ďalej do vyššej vrstvy až ku koreňovému uzlu RFT [1][12].

Aby bolo možné rovnomerne rozmiestniť FT stanice v oblasti vysielania, bol navrhnutý optimalizovaný algoritmus TTF (Tree TransmissionFeedback) [1], ktorý je schopný nájsť množinu aktívnych FT staníc tak, že rovnomerne pokrýva množinu prijímačov a určuje výšku jednotlivých FT staníc v stromovej štruktúre [12].

Tento algoritmus požaduje na vstupe polohu jednotlivých FT staníc a funguje správne len za predpokladu, že vieme čo možno najpresnejšie určiť súradnice aktívnych FT. Jeden z hlavných problémov takéhoto systému môže byť nájdenie počiatočnej pozície zapojených uzlov pri jeho spustení. Od týchto súradníc sa potom bude odvíjať vytvorená stromová štruktúra.



Obrázok 2.2: Schéma počiatočného nastavenia pozíc FT staníc

3 NÁVRH A REALIZÁCIA VLASTNÉHO SÚRADNICOVÉHO SYSTÉMU

Pre zavedenie vlastného súradnicového systému je potrebné poznať správanie celého systému v rôznych situáciach, ktoré sa môžu vyskytnúť počas prevádzky. V tejto kapitole som sa preto zaoberal rozmiestnením a počtom jednotlivých uzlov v závislosti na celkovej presnosti súradnicového systému.

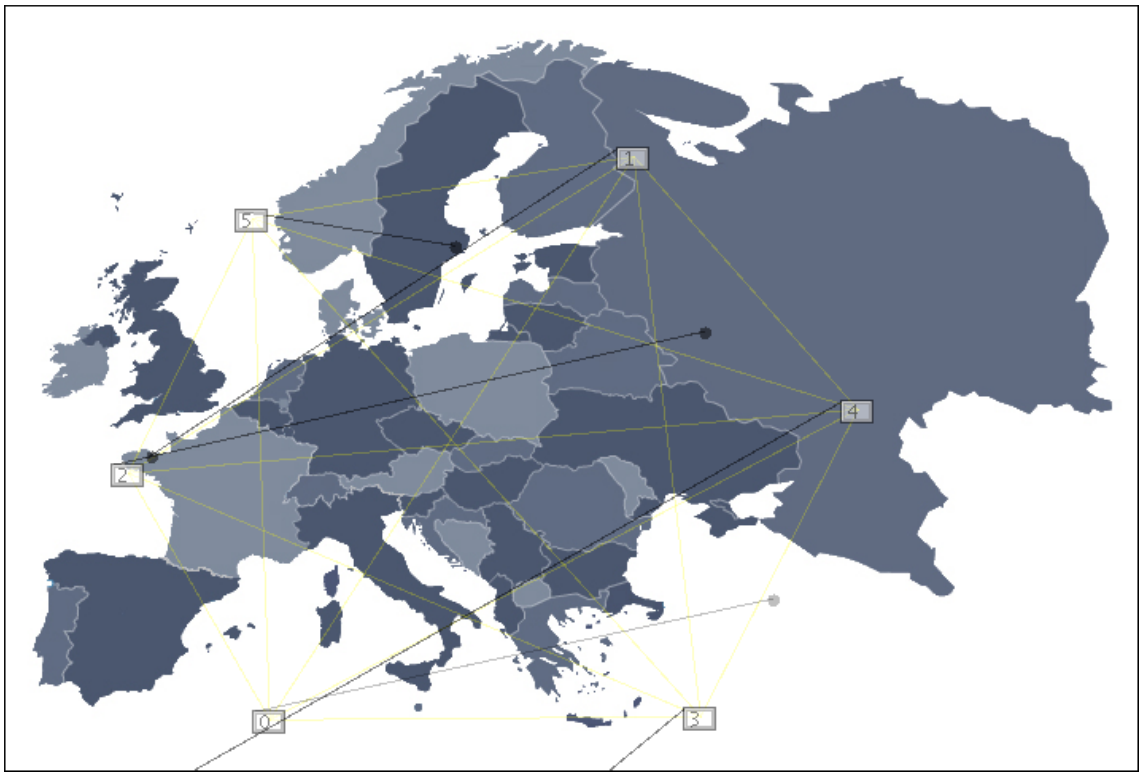
3.1 Výber vhodného simulačného scenária

Pre naše účely je potrebné, aby celý súradnicový systém konvergoval rýchlo a pritom čo možno najpresnejšie. Preto je potrebné zvoliť konpromis medzi náročnosťou algoritmu na výpočet a prenosťou vypočítaných súradníc. Rozhodol som sa postupne navrhovať a skúmať rôzne scenáre rozmiestnenia uzlov v systéme. Ako základ som zvolil upravený algoritmus Vivaldi s adaptívnym časovým krokom, ktorý oproti jednoduchej verzii zlepšuje rýchlosť a presnosť konvergencie.

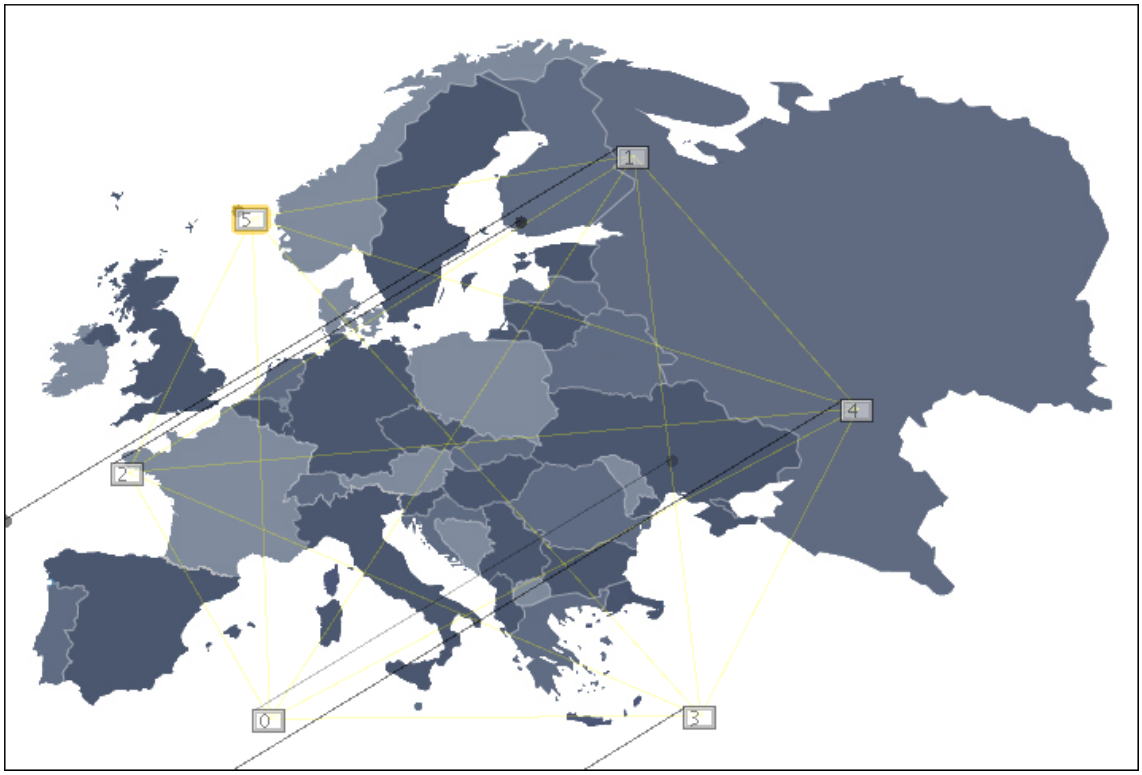
Pre zhodnotenie presnosti výpočtov som uvažoval s relatívnou 1.9 a absolútnou chybou z rovnice 3.1, kde R_{POS} predstavuje reálnu a P_{POS} predpovedanú pozíciu. Relatívna chyba poukazuje na presnosť predpovedanej RTT vzdialenosti a absolútna chyba zase na presnosť vypočítaných súradníc a ich vzdialenosť od skutočnej pozície.

$$A_{\mathrm{ERROR}} = R_{\mathrm{POS}} - P_{\mathrm{POS}} \tag{3.1}$$

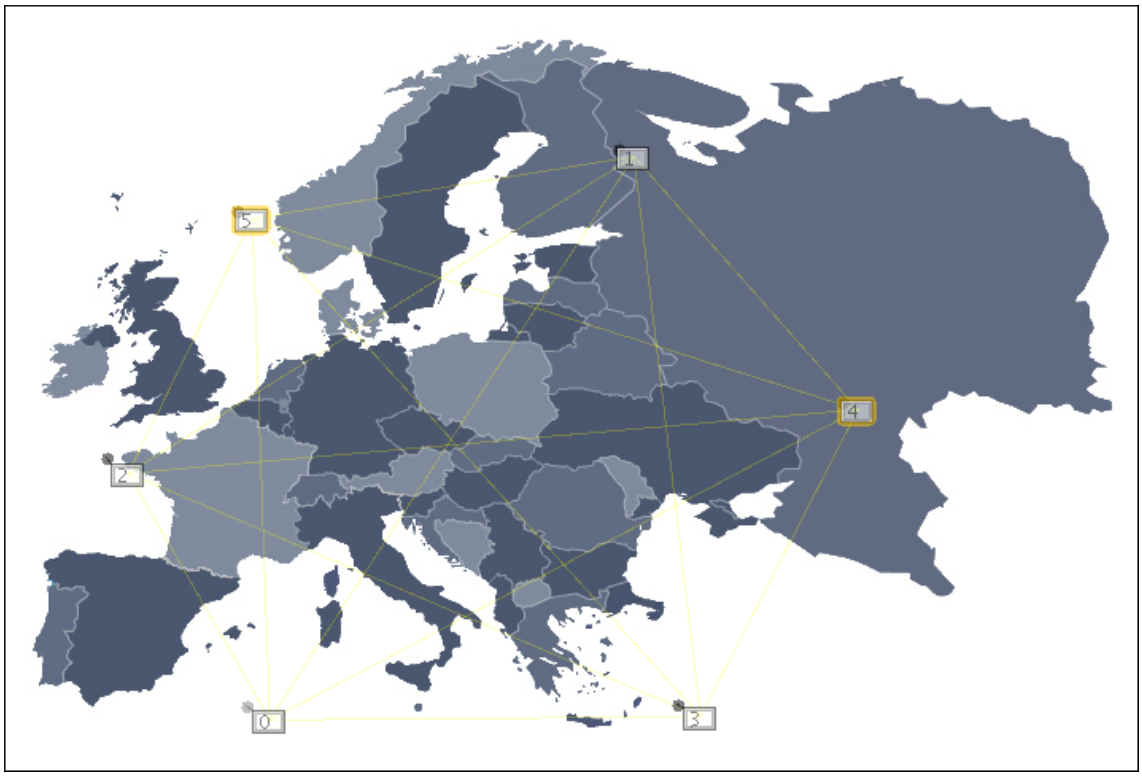
Pri tomto experimente som najprv ponechal uzlom dostatočný čas, aby zamerali svoje súradnice v systéme. Keď už bolo zjavné, že sa súradnice ustálili a viac sa nepribližujú k svojim reálnym pozíciam, vybral som náhodný uzol a nastavil ho do fixného režimu. Tak už uzol nehľadal svoje súradnice, pretože mu boli pridelené podľa skutočných a takýto uzol pomohol ku konvergencii ostatných. Ďalej som zase ponechal nevyhnutný čas spustenú simuláciu, až kým sa uzly nezastavili v približovaní ku svojim skutočným súradniciam. Ak bola zreteľná ešte pomerne vysoká absolútna chyba, tak som pokračoval v nastavovaní uzlov do fixného režimu. Z tohto experimentu som sa ďalej pokúsil zistiť a percentuélne vyjadriť počet potrebných fixných uzlov alebo takých, ktoré sú presne zamerané, aby bola celková absolútna chyba systému minimálna.



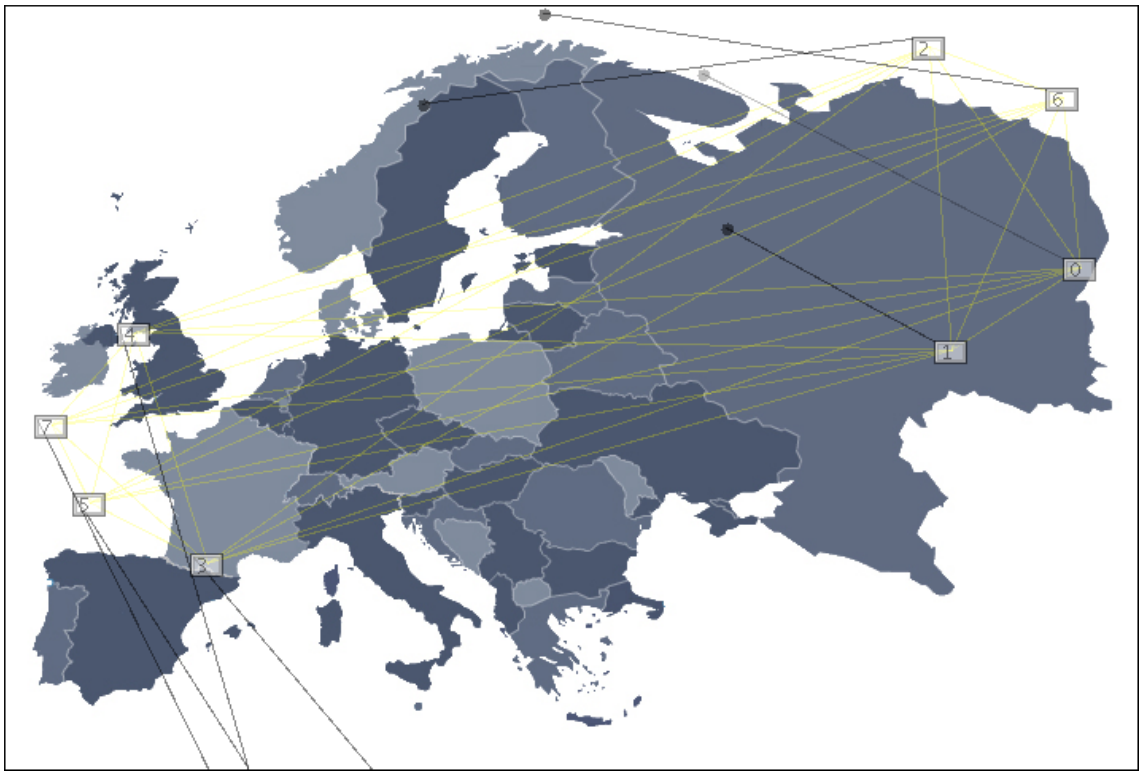
Obrázok 3.1: Simulácia č.1: Ustálenie uzlov po spustení simulácie



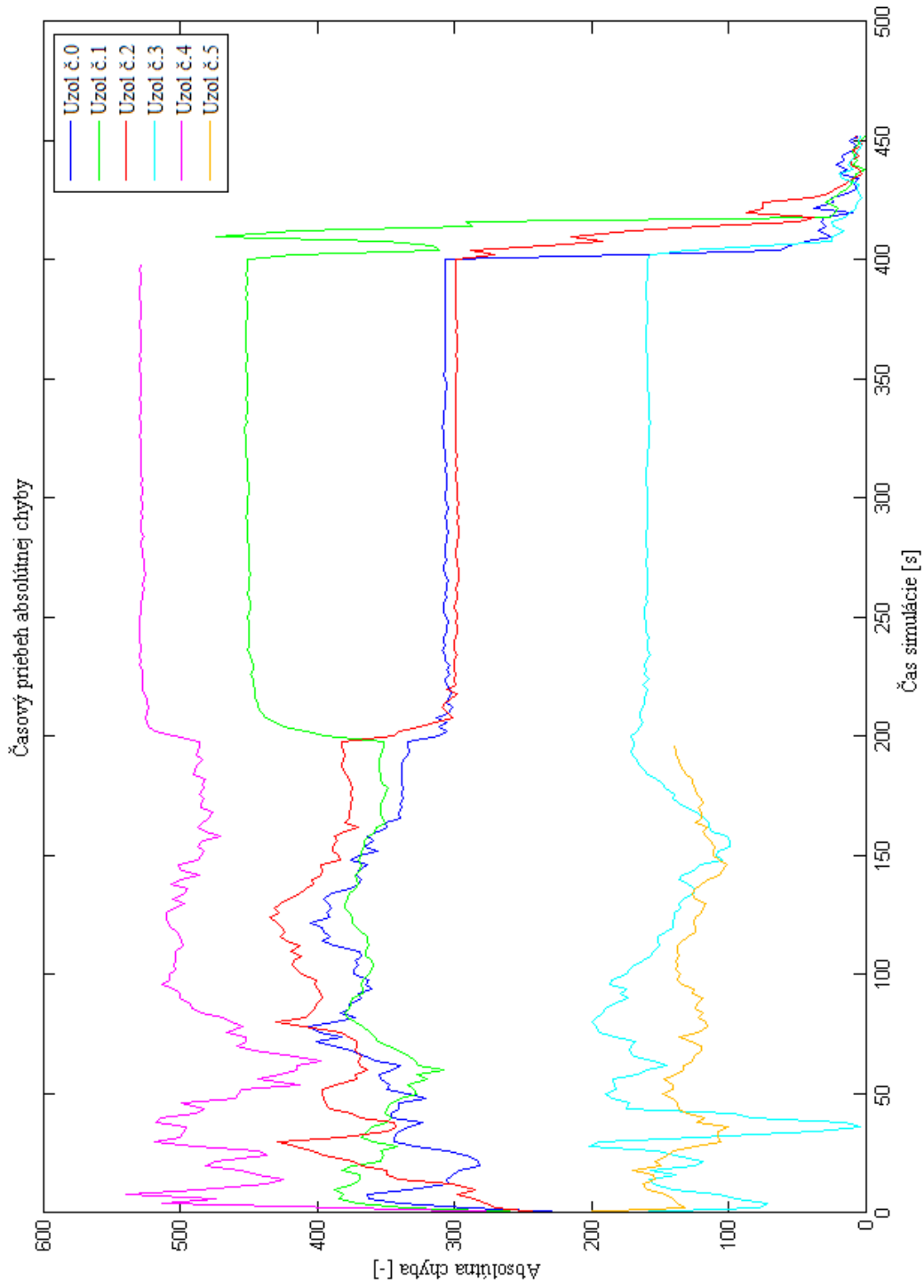
Obrázok 3.2: Simulácia č.1: Nastavenie uzlu č.5 do fixného režimu



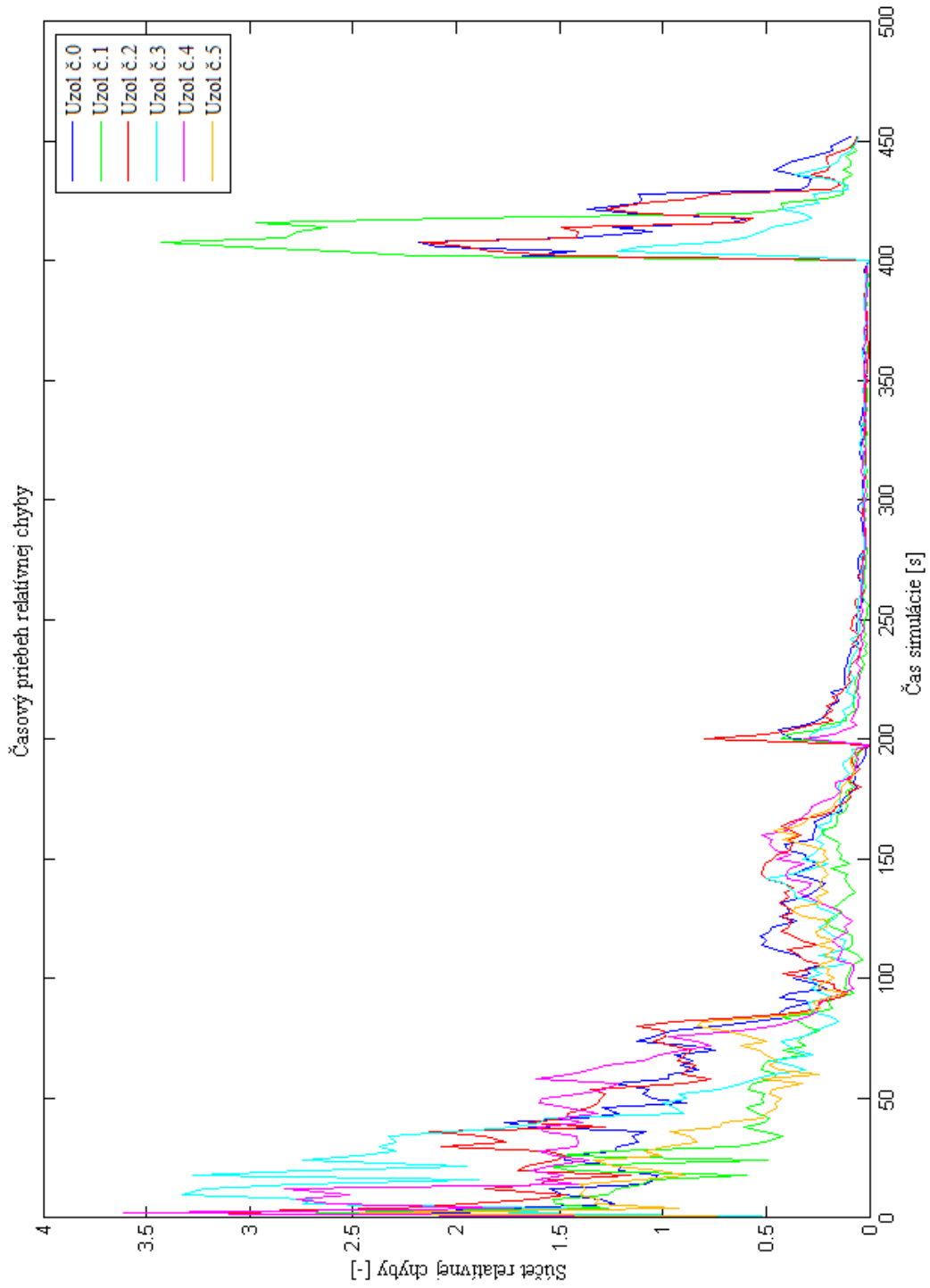
Obrázok 3.3: Simulácia č.1: Nastavenie uzlu č.4 do fixného režimu



Obrázok 3.4: Simulácia č.2: Ustálenie uzlov po spustení simulácie



Obrázok 3.5: Simulácia č.1: Časový priebeh absolútnej chyby



Obrázok 3.6: Simulácia č.1: Časový priebeh relatívnej chyby

Na obrázkoch sú vypočítané súradnice uzlov znázornené čiernymi bodmi a k nim prislúchajúce reálne pozície zase šedými štvorčekmi s poradovým číslom. Fixný uzol je označený žltým okrajom okolo šedého štvorčeka.

Skúmal som rôzne rozmiestnenia uzlov, či už náhodné alebo cielené, a výsledok týchto poznatkov by som zosumarizoval do nasledovných troch skupín možných scenárov. Pri tomto skúmaní som zväčša používal menší počet zapojených uzlov z dôvodu dosiahnutia väčšej rýchlosti simulácie a hodnotené tri kategórie som odskúšal aj na väčšom počte uzlov. Pre prehľadnosť v tomto dokumente uvádzam simulácie vykonané s menším počtom uzlov.

Prvým scenárom je zapojenie uzlov tak, že sú od seba rozmiestnené v pravidelných vzdialenostiach ako to vidíme na nasledujúcich obrázkoch.

Z obrázku 3.1 môžene vidieť, na akých súradniciach sa simulácia zastavila a uzly už ďalej nekonvergovali k svojim reálnym hodnotám. Bolo to spôsobené tým, že relatívna chyba bola minimálna a z priloženehého grafu 3.6 je viditeľné, že súčet relatívnej chyby uzlov klesol až na hodnotu 0,1 v čase 180 sekúnd od spustenia simulácie. Takto už nemohla mať relatívna chybva významný vplyv na výpočet aktualizovaných súradníc. Protikladom je v tomto prípade veľká absolútna chyba viditeľná z grafu 3.5, ktorú z pochopitelných dôvodov nemôžme zahrnúť do výpočtu súradníc.

Pre účely nášho súradnicového systému bude uspokujúci aj takýto výsledok s veľkou absolútnou chybou, pretože je rovnomerne rozmiestnená na všetky uzly a tak pri hľadaní najbližšieho sa nám vždy podarí nájsť jeden z najbližších, čo demonštruje aj obrázok 3.1.

Ak by napríklad uzol č.5 hľadal najbližší susedný, tak podľa aktuálne zostavených súradníc by sa rozhodoval medzi uzlami č.1 a č.2, čo odpovedá aj v skutočnosti najbližším.

Ďalej som v pokuse nastavil uzol č.5 do fixného režimu, čo sa prejavilo na výpočte nových súradníc celého systému. Výsledok je možný vidieť na obrázku 3.2 a v grafoch 3.6, 3.5 v časovom intervale 200 - 400 sekúnd. Z toho sa dá usúdiť, že predpovedané súradnice uzlov sa výrazne zmenili, avšak stále zostáva pomerne veľká absolútna chyba, ktorá sa zlepšila len v uzloch č.0 a č.2, zatiaľ čo relatívna chyba sa minimalizovala. Rovnako ako aj pri obrázku 3.1, tak aj tu môžme považovať vypočítané súradnice za dostatočné pre náš účel, aj keď pre aplikácie závislé na veľkosti absolútnej chyby, by boli nevyhovujúce.

Na obázku 3.3 je možné vidieť, že po nastavení ďalšieho uzlu, v tomto prípade č. 4, do fixného stavu sa zlepší presnosť vypočítaných súradníc a relatívna aj absolútna chyba výrazne klesnú. Z tejto simulácie môžme usúdiť, že pre minimalizovanie chýb výpočtu je potrebné mať v takomto systéme aspoň dva uzly správne zamerané, aby sa aj zvyšné čtyri zamerali presne z pohľadu absolútnej chyby. Simulácie potvrdili, že

sa toto číslo zväčšuje s pribúdajúcim počtom uzlov pripojených do systému, ale vždy je postačujúce, ak sa v takomto systéme nachádza 20% až 30% uzlov s minimálnou absolútnou chybou.

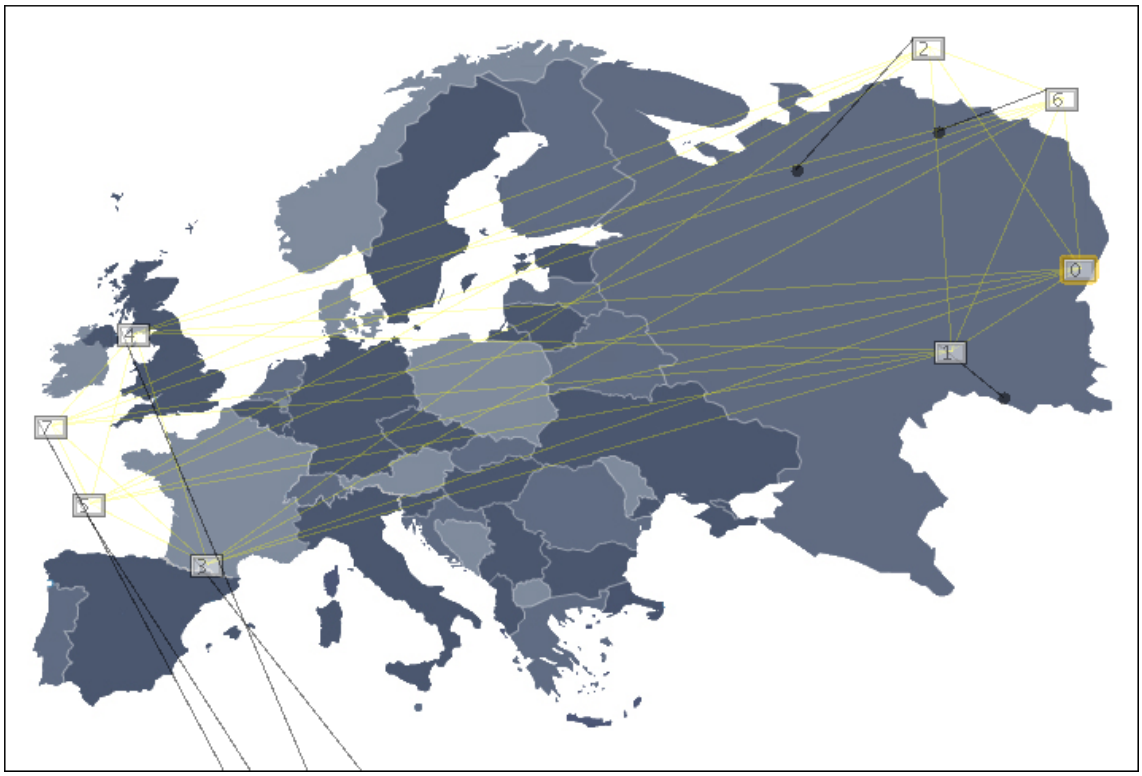
Pri hľadaní druhého simulačného scenára som rozdelil použité uzly do dvoch skupín, ktoré sú od seba pomerne vzdialené v porovnaní so vzdialenosťou uzlov v týchto skupinách.

Na obrázku 3.4 sú znázornené uzly po ustálení výpočtu. To znamená, že relatívna chyba je minimálna a už nemá vplyv na výpočet súradníc. Z obrázku je zrejmé, že aj v tomto prípade simulácia trpí pomerne veľkou absolútnou chybou. Zaujímavosťou je, že súradnice každej množiny sa udržujú pri sebe a navzájom sa nepremiešavajú uzly z rôznych množín. Toto podáva dobý základ pre praktické využitie súradníc, pretože ak by akýkoľvek uzol hľadal svoj susedný najbližší, tak by bol vybraný uzol z rovnakej množiny a nie zo vzdialenej a súčasne aj v rámci množiny by bol vybraný jeden z najbližších.

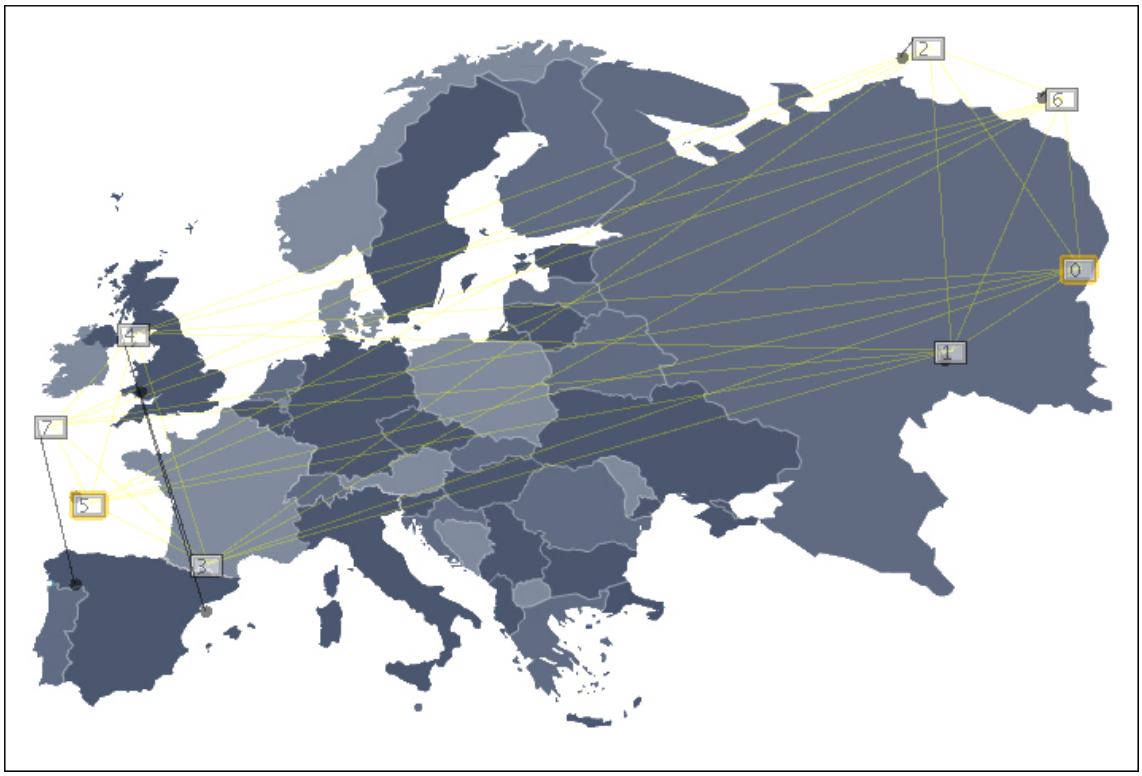
Z grafu časovej závislosti relatívnej chyby 3.10 pre tento scenár môžme sledovať, že reltívna chyba klesla pomerne rýchlo a už po dvoch minútach od spustenia simulácia mala hodnotu rádovo niekoľko desatín. Z grafu časovej závislosti absolútnej chyby 3.9 môžme pozorovať, že hodnoty jednotlivých uzlov sa urdžujú v skupinách, tak ako ich uzly tvorili v priebehu simulácie.

Pri ďalšom obrázku 3.7 je vidieť ako sa zmenili súradnice po tom, čo som nastavil uzol č.0 do fixného stavu. Z priebehu je patrné, že sa výrazne zmenšila absolútna chyba súradníc v množine, v ktorej je jeden uzol zameraný presne. Tento výsledok môžme sledovať aj v grafe 3.9 na časovom intervale 160 - 280 sekúnd. Táto zmena sa takmer neprejavila na zmene absolútnej chyby v druhej množine prvkov.

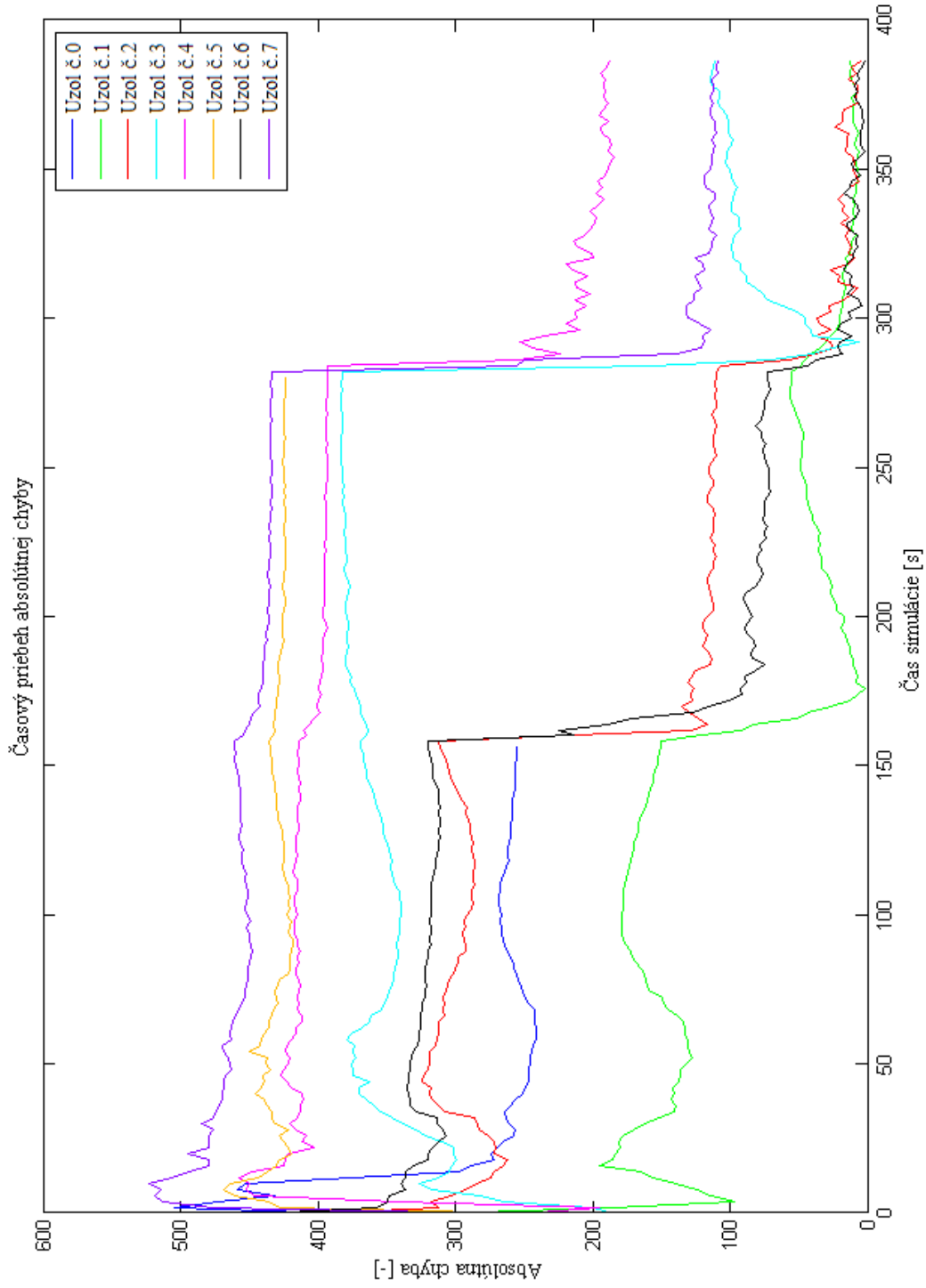
Preto som nastavil uzol č. 7 do fixného stavu a tým sa celková absolútna chyba systému minimalizovala, čo môžeme pozorovať z grafu 3.9 na časovom intervale 280 - 380 sekúnd. Z tohto priebehu je viditeľné, že súradnice pre množinu umiestnenú na ľavej strane neskonvergovali k svojím reálym hodnotám a tým sa neminimalizovala absolútna chyba. Tento stav bol zapríčinený vyberom fixného uzlu, ktorý sa nachádzal v dolnej polovici v smere vypočítaných súradníc. Rovnaký výsledok by sme získali aj pri nastavení fixného stavu pre uzol č.3. Naopak presné zameranie súradníc by sme dosiahli pri výbere uzlu z opačnej polovičky množiny v smere zamerania súradníc, tak ako tomu bolo v pravej množine uzlov. Pri väčších množinách je možné tento problém odstrániť tak, že sa bude nachádzať aspoň jeden uzol správne zameraný v každej polmnožine.



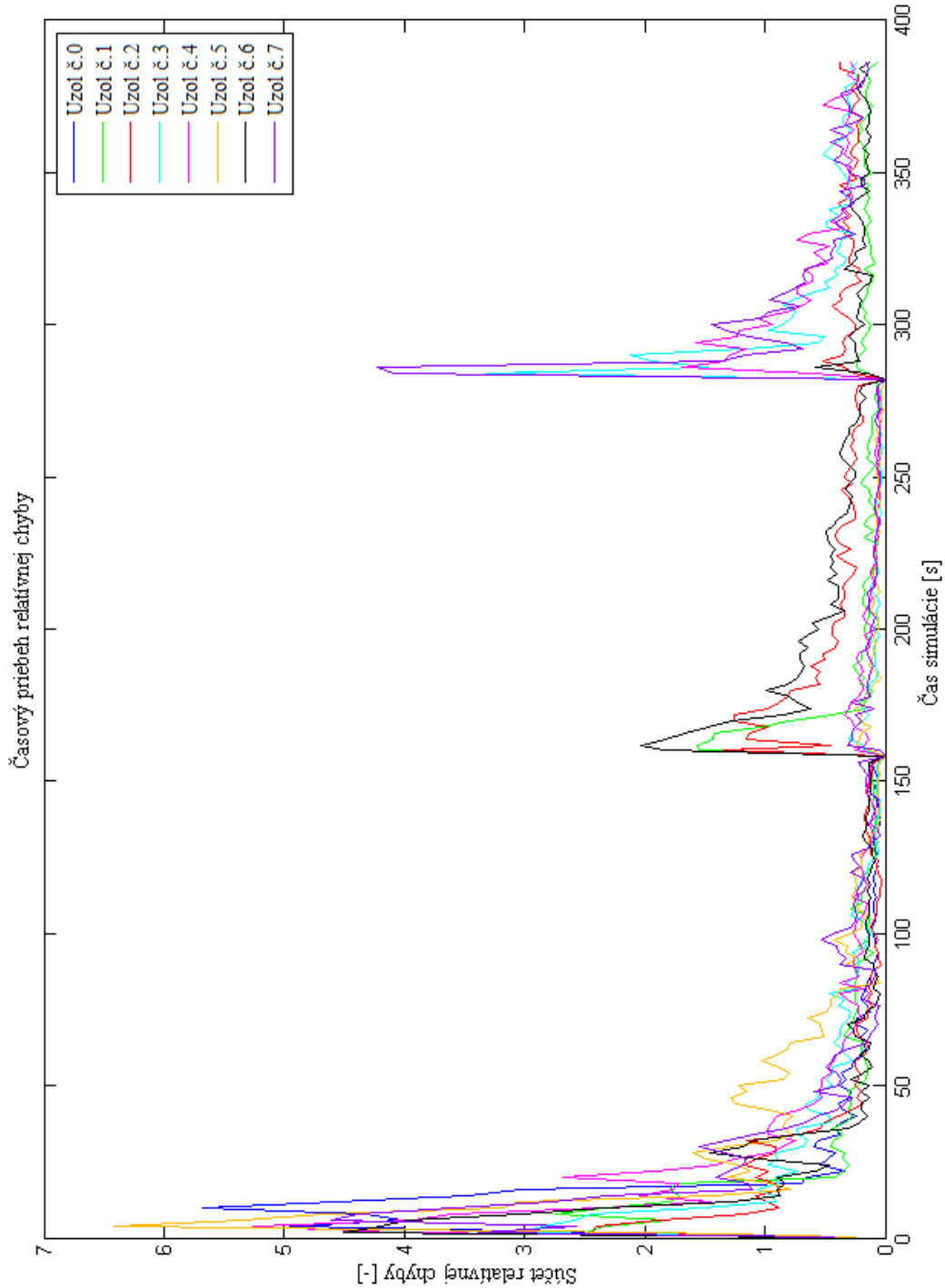
Obrázok 3.7: Simulácia č.2: Nastavenie uzlu č.0 do fixného režimu



Obrázok 3.8: Simulácia č.2: Nastavenie uzlu č.5 do fixného režimu



Obrázok 3.9: Simulácia č.2: Časový priebeh absolútnej chyby



Obrázok 3.10: Simulácia č.2: Časový priebeh relatívnej chyby

Aj z tejto simulácie je zrejmé, že pre presné zameranie súradníc je potrebné, aby sa v systéme nachádzalo okolo 25% uzlov s nízkou absolútnou chybou. Avšak tieto uzly musia byť rovnomerne rozmiestnené vo všetkých množinách. V prílohe je séria obrázkov A.1, A.2, A.3 so simulácie pre väčší počet uzlov v množine. Aj z tejto simulácie získavame hodnotu okolo 25% uzlov s nízkou reálnou chybou pre presnú konvergenciu celého súradnicového systému.

V poslednej simulácii som sa zaoberal myšlienkou rozloženia uzlov do množín podobných tým v predchádzajúcom príklade. Ale tentokrát bude v každej množine nerovnomerne zastúpený počet uzlov. Pre zjednodušenie som oddialil jeden uzol od zvyšných, ktoré sú zoskupené pomerne blízko seba. Ustálený stav som získal po 100 sekundách od spustenia simulácie, čo je znázornené na obrázku 3.11. Z grafov 3.16 a 3.15 je viditeľné tak ako v predošlých príkladoch, že je príliš veľká absolútna chyba, zatiaľ čo relatívna chyba sa pomerne rýchlo minimalizovala. Chyba z množiny uzlov je rozmiestnená rovnomerne a tak pri hľadaní najbližšieho uzlu by bol vybraný správny.

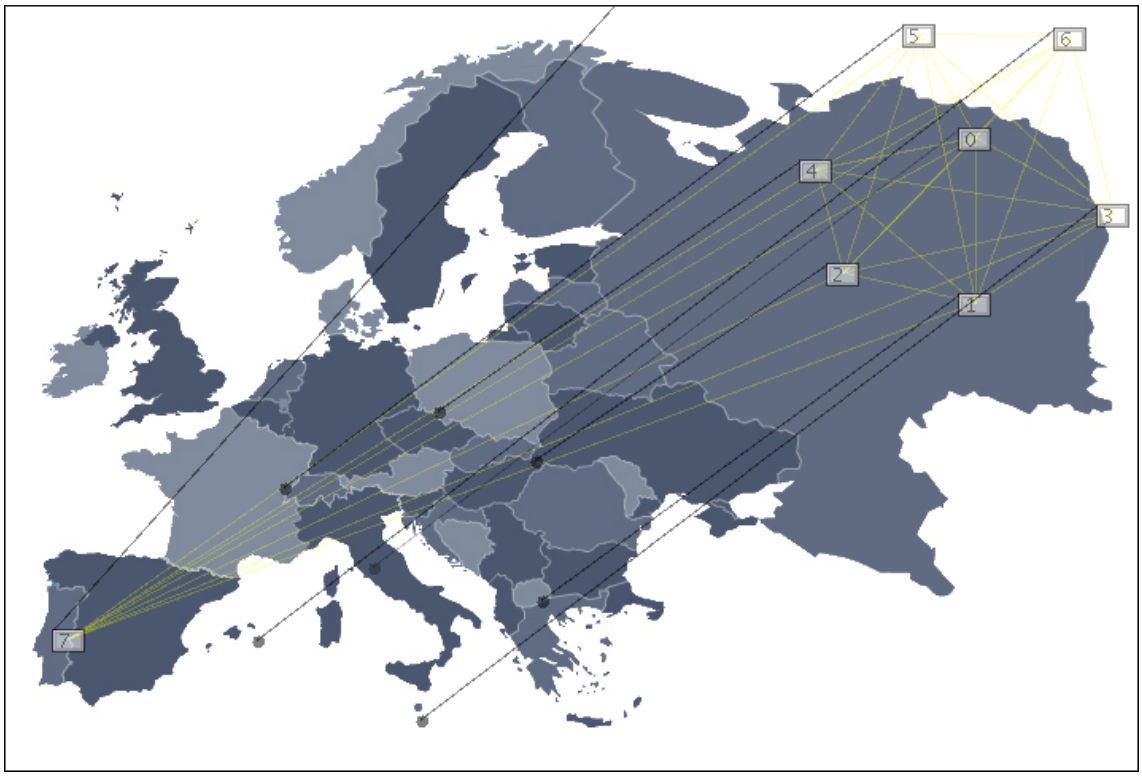
Pre zlepšenie presnosti predpovedaných súradníc som nastavil uzol č.3 do fixného režimu a na obrázku 3.12 môžme vidieť ako sa zmenili vypočítané súradnice v systéme. Najväčšiu absolútnu chybu tu predstavuje osamotený uzol č. 7 a podľa grafu 3.16 je vidieť, že tento uzol mal počas celej simulácie najmenšiu relatívnu chybu. Pôsobil tak na ostatné uzly ako uzol s dôverihodnými súradnicami a ovplivnil ich presnosť konvergencie. Z tohto grafu je rovnako vidieť, že uzly v množine mali podstatne väčšiu relatívnu chybu a minimalizovanie tejto chyby si vyžadovalo oveľa väčší časový interval ako v predošlých príkladoch.

Preto som do fixného režimu nastavil uzol č.4 s očakávaním zlepšenia absolútnej chyby. Výsledok tejto simulácie je zobrazený na obrázku 3.13. Z grafov je zrejmé, že absolútna chyba osamoteného uzlu sa stále prejavuje na konvergencii súradníc uzlov z osamotenej množiny. Ďalším problémom je, že v tejto množine vznikol zrkadlový efekt, kedy sa vypočítané súradnice zobrazujú presne na opačnej strane ako sú ich reálne pozície. Ak by bol uzol č.7 zameraný s minimálnou absolútnou chybou, tak by to pomohlo ku správnej konvergencii tejto množiny.

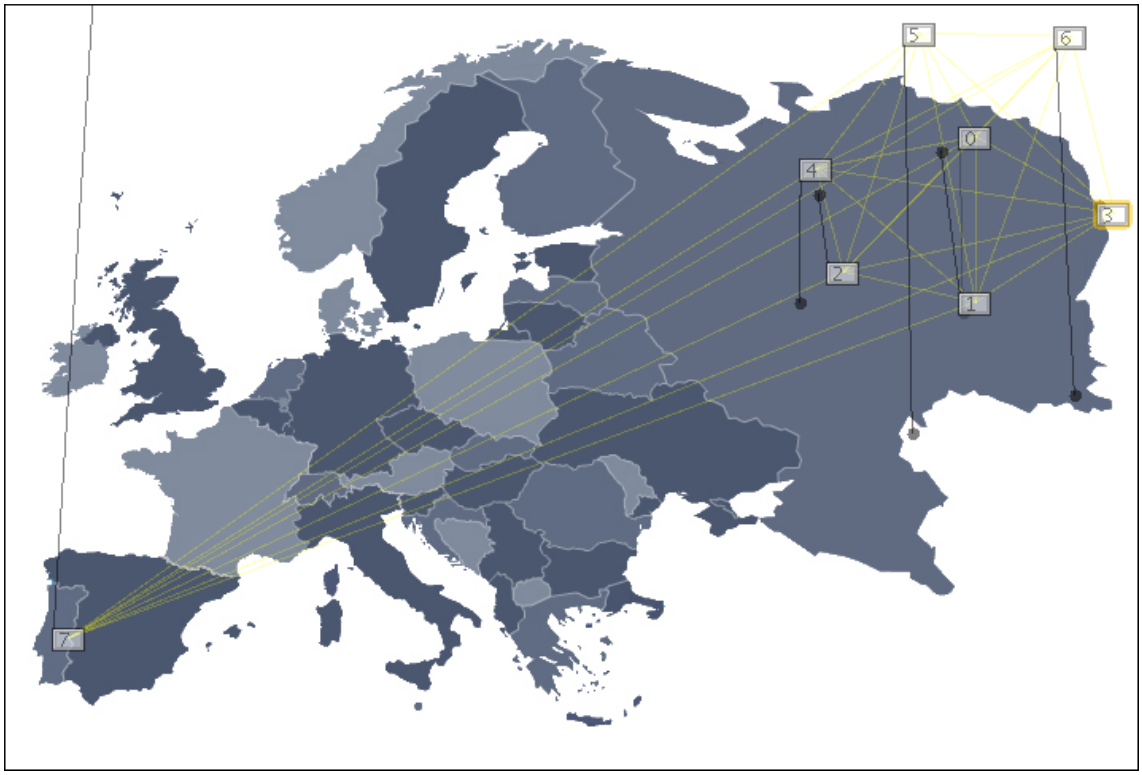
V poslednom kroku som nastavil ešte aj uzol č. 6 do fixného režimu, čo pomohlo k rýchlemu a presnému vypočítaniu súradníc v množine. Toto tvrdenie môžeme pozorovať z grafu 3.15 v poslednom časovom intervale simulácie, ktorý začína od hodnoty 700 sekúnd. Z grafu môžme pozorovať, že absolútna chyba množiny uzlov sa minimalizovala v priebehu niekoľkých sekúnd, zatiaľ čo súradnice osamoteného uzlu konvergovali k svojim reálnym hodnotám veľmi pomaly. Bolo to spôsobené tým, že v systéme prevládala komunikácia uzlov, ktoré už nepotrebovali znižovať svoju absolútnu chybu a aj ich relatívna chyba bola na minime zatiaľ čo relatívna chyba uzlu č.7 bola o čosi väčšia a konvergovala pomaly ako aj absolútna chyba. Preto by bolo

v tomto prípade vhodné, aby bola v simulácii adaptívna komunikácia medzi uzlami a algoritmus Vivaldi sa použil len na uzloch, ktoré majú väčšiu relatívnu chybu.

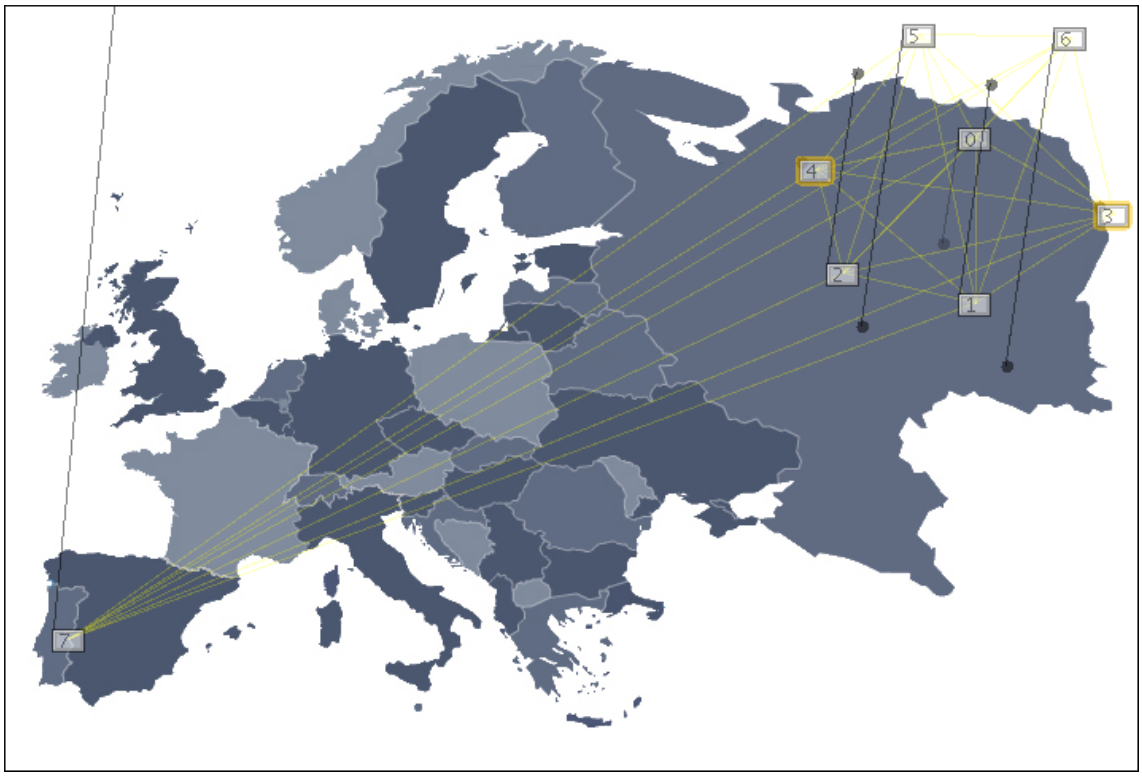
Takto rozložené uzly považujem za najproblematickejšiu variantu, ktorá môže v systéme nastať a mohla by spôsobiť zjavné komplikácie pri rýchlosti a presnosti konvergencie súradnicového systému. V tomto prípade bolo potrebných viac ako 30% presne zameraných uzlov k tomu, aby správne konvergovali aj zvyšné uzly s minimálnou absolútnou chybou, navyše osamotenému uzlu sa darilo minimalizovať túto chybu len po veľmi malých krokoch a mohol by ohroziť funkčnosť a podstatu tohto systému. Preto sa budem v ďalšej práci zaoberať práve takýmto problematickým rozložením uzlov, kedy sú od seba nerovnomerne vzdialené a zoskupené v rôznych počtoch.



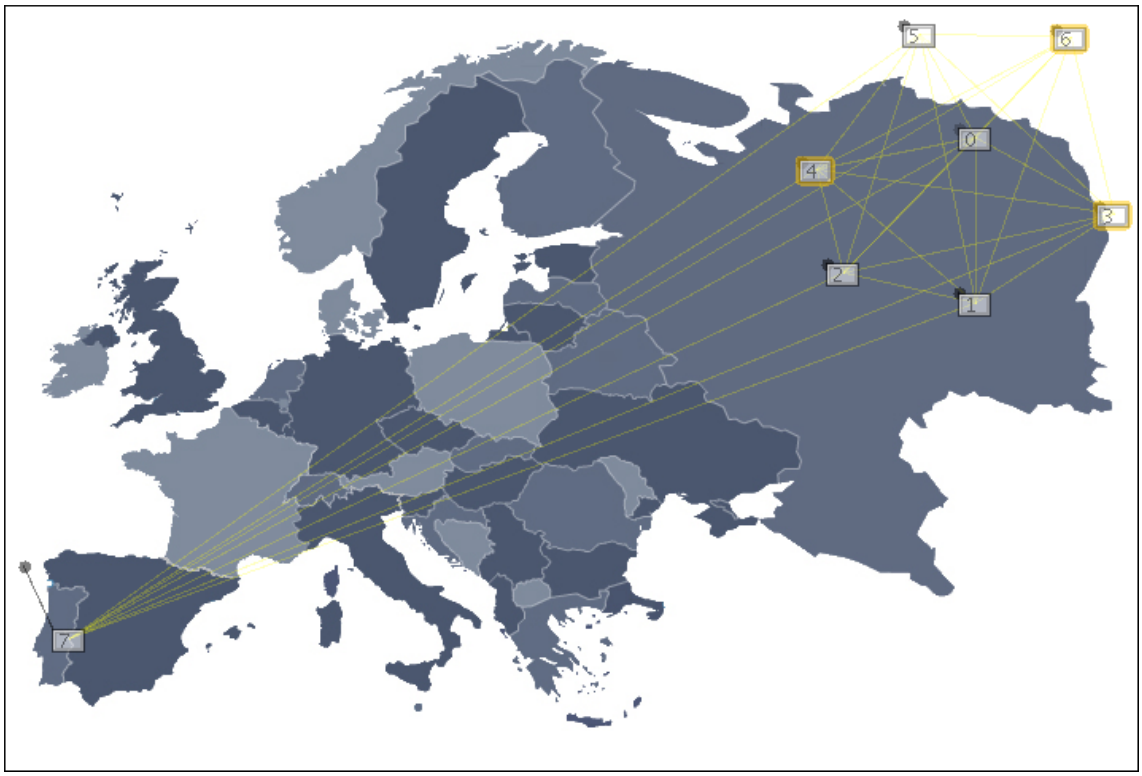
Obrázok 3.11: Simulácia č.3: Ustálenie uzlov po spustení simulácie



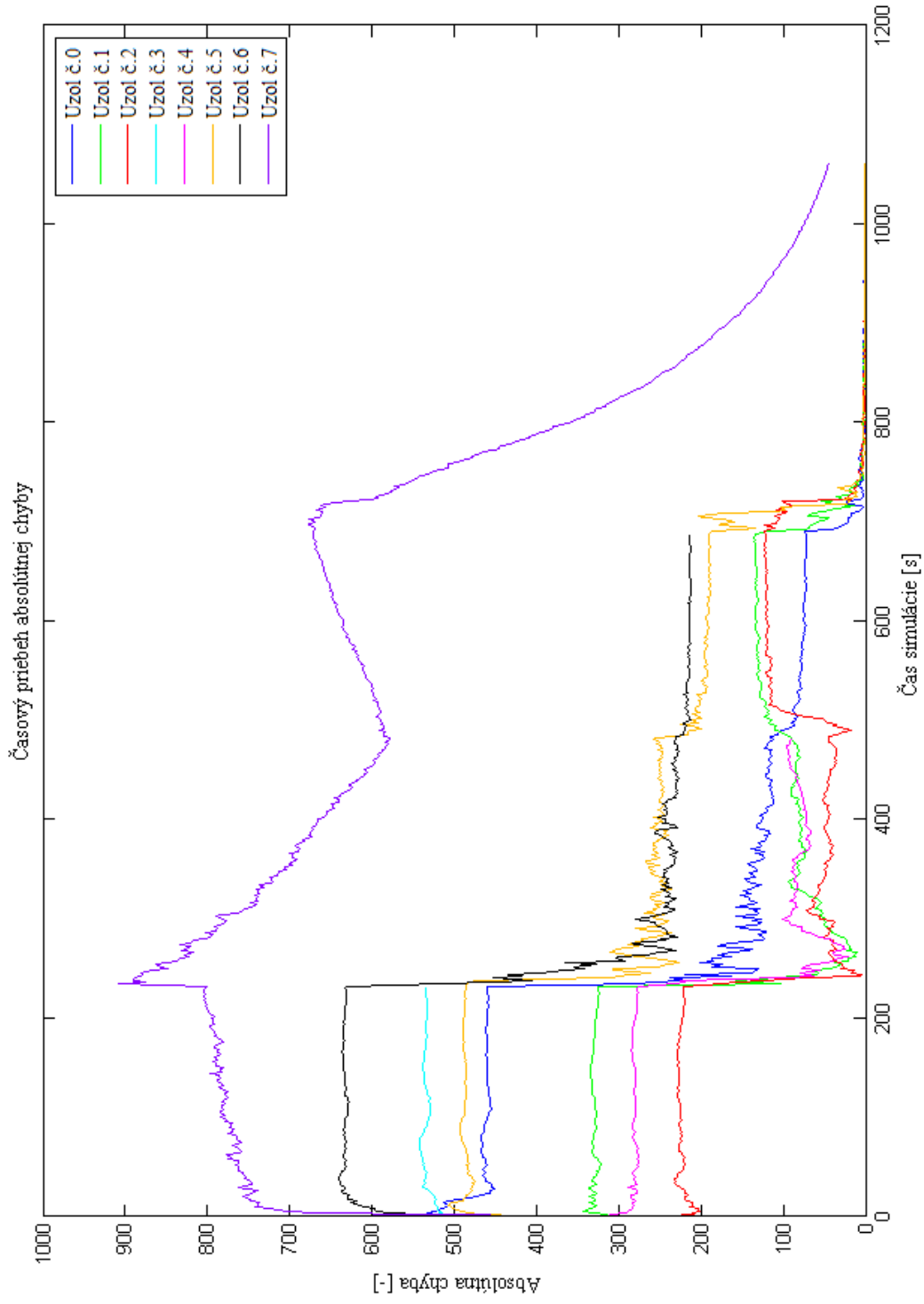
Obrázok 3.12: Simulácia č.3: Nastavenie uzlu č.3 do fixného režimu



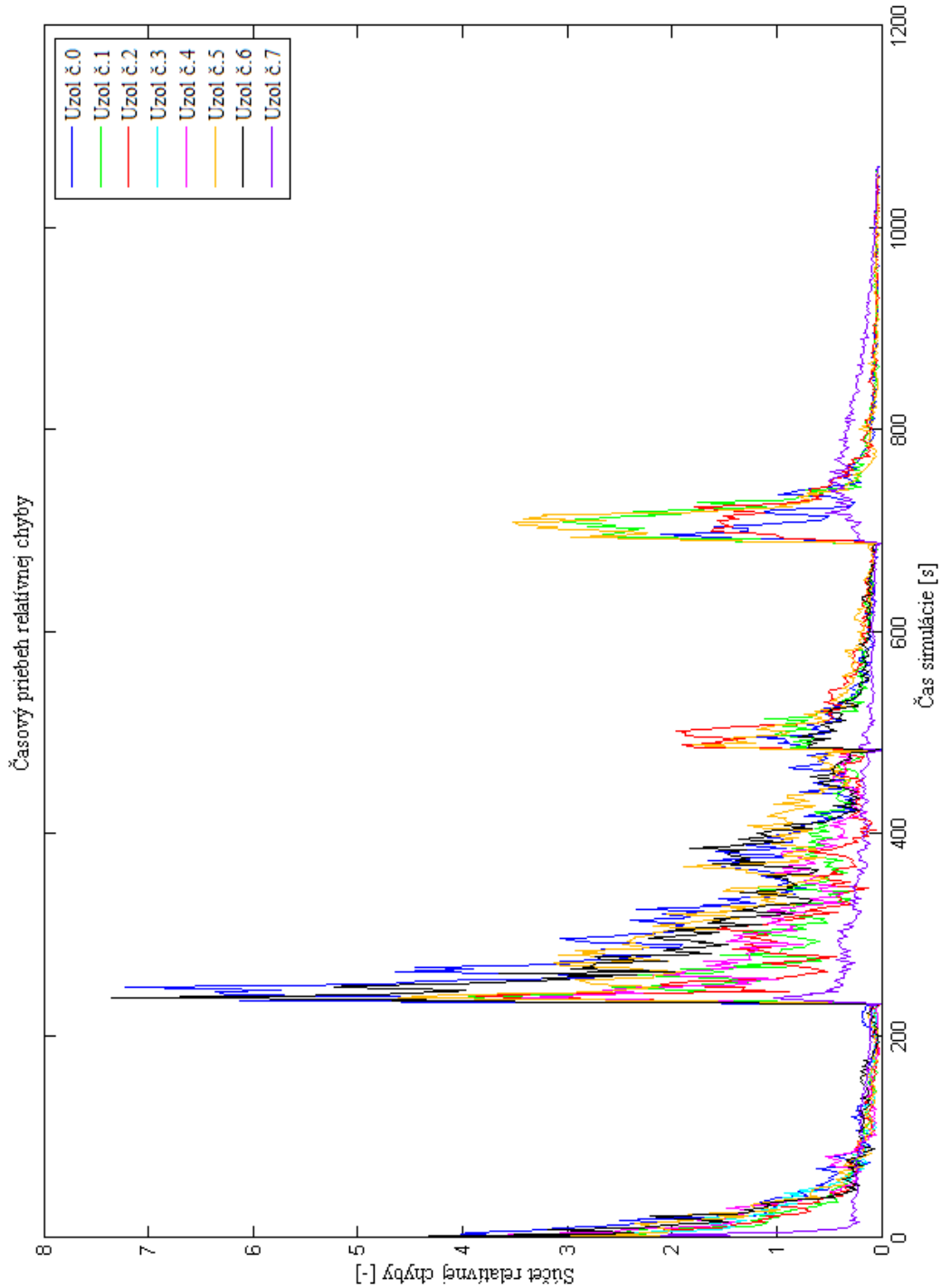
Obrázok 3.13: Simulácia č.3: Nastavenie uzlu č.4 do fixného režimu



Obrázok 3.14: Simulácia č.3: Nastavenie uzlu č.6 do fixného režimu



Obrázok 3.15: Simulácia č.3: Časový priebeh absolútnej chyby



Obrázok 3.16: Simulácia č.3: Časový priebeh relatívnej chyby

3.2 Zhlukovanie uzlov

V predchádzajúcej kapitole boli znázornené simulácie základného algoritmu Vivaldi, ktorý mal adaptívny časový krok. Takýto algoritmus má určité obmedzenia, ktoré boli pozorovateľné z priložených grafov. Našim najväčším problémom pre správnu konvergenciu je nutnosť úplného vzájomného prepojenia uzlov. Toto obmedzenie som sa pokúsil odstrániť výberom určitého počtu náhodných spojení.

Druhý závažný problém takéhoto algoritmu je výskyt chyby výpočtu pri uzloch, ktoré sa nachádzajú od seba v pomerne krátkej vzdialenosti. Problém krátkych vzdialeností už bol popísaný v kapitole 1.3.

Z týchto dôvodov som sa rozhodol vytvoriť vlastný súradnicový systém, ktorý by bol rovnako presný ako samotný Vivaldi, ale za použitia menšieho počtu spojení a hlavne by minimalizoval problém krátkych vzdialeností medzi susednými uzlami.

3.2.1 Popis vlastného algoritmu

Pre zostavenie vlastného algoritmu výpočtu súradníc som ako základ zvolil metódu Vivaldi s adaptívnym časovým krokom a využil som znalosti z metód Pharos [4] a Myth [5].

Vývojový diagram tohto navrhnutého riešenia sa nachádza na obrázku 3.17. V inicializačnej časti je definovaný počet vzájomných spojení medzi susednými uzlami. Táto hodnota je odvodená v nasledujúcej kapitole 3.2.2 na základe vykonaných simulácií.

Zo všetkých pripojených uzlov v sieti sa vykoná definovaný počet náhodných spojení k susedným. Takto vygenerovaný zoznam IP adries sa použije pre výpočet a aktualizáciu súradníc podľa štandardného Vivaldi s adaptívnym časovým krokom, ktorý bol predstavený v kapitole 1.1 aj pomocou algorimtu 3.

Výpočet súradníc, ktorých presnosť je ovplivnená relatívnou chybou, prebieha až do chvíle, keď je relatívna chyba vzdialeného uzlu pod hodnotou 1.0. Po dosiahnutí takejto presnosti sú odoslané súradnice do prvku zhromaždišťa (angl. Rendezvous Point) (RP), ktorý už bol predstavený v kapitole 1.3. Tento prvok rozdeľuje uzly na základe získaných súradníc do zhlukov podľa metódy K-means, ktorá bola predstavená v kapitole 1.5 a môj využitý algoritmus je bližšie naznačený v podkapitole 3.2.1.

Po výpočítaní zhlukov je každému uzlu odoslaný paket, ktorý obsahuje informáciu o identifikátore jeho zhluku a vyjadrenie prípadnej potreby pokračovať v prepočítavaní vytvorených zhlukov. Každý uzol čaká na doručenie takéhoto paketu a na základe získaných informácií sa vygenerujú nové náhodné spojenia. Ak je definovaných N spojení, tak polovica z týchto spojení je realizovaná náhodne medzi

uzlami z rovnakého zhluku a druhá polovica spojení je vytvorená tiež náhodne medzi susednými zhlukmi. Ak je počet požadovaných spojení v rovnakom zhluku menší ako je počet uzlov v tomto zhluku, tak sa vytvoria spojenia na každý takýto uzol a zvyšná časť z N spojení sa vytvorí náhodne na vzdialené zhluky.

Z dôvodu rýchlejšej konvergencie a počiatočného začlenenia uzlov do zhlukov bola definovaná maximálna dovolená relatívna chyba vzdialeného uzlu *max_chyba* s hodnotou 1.0. Po vytvorení zhlukov sa táto tolerovaná chyba zmenší na hodnotu 0.4 a to z dôvodu požadovanej väčšej presnosti pri zameriavaní uzlov medzi vytvorenými zhlukmi.

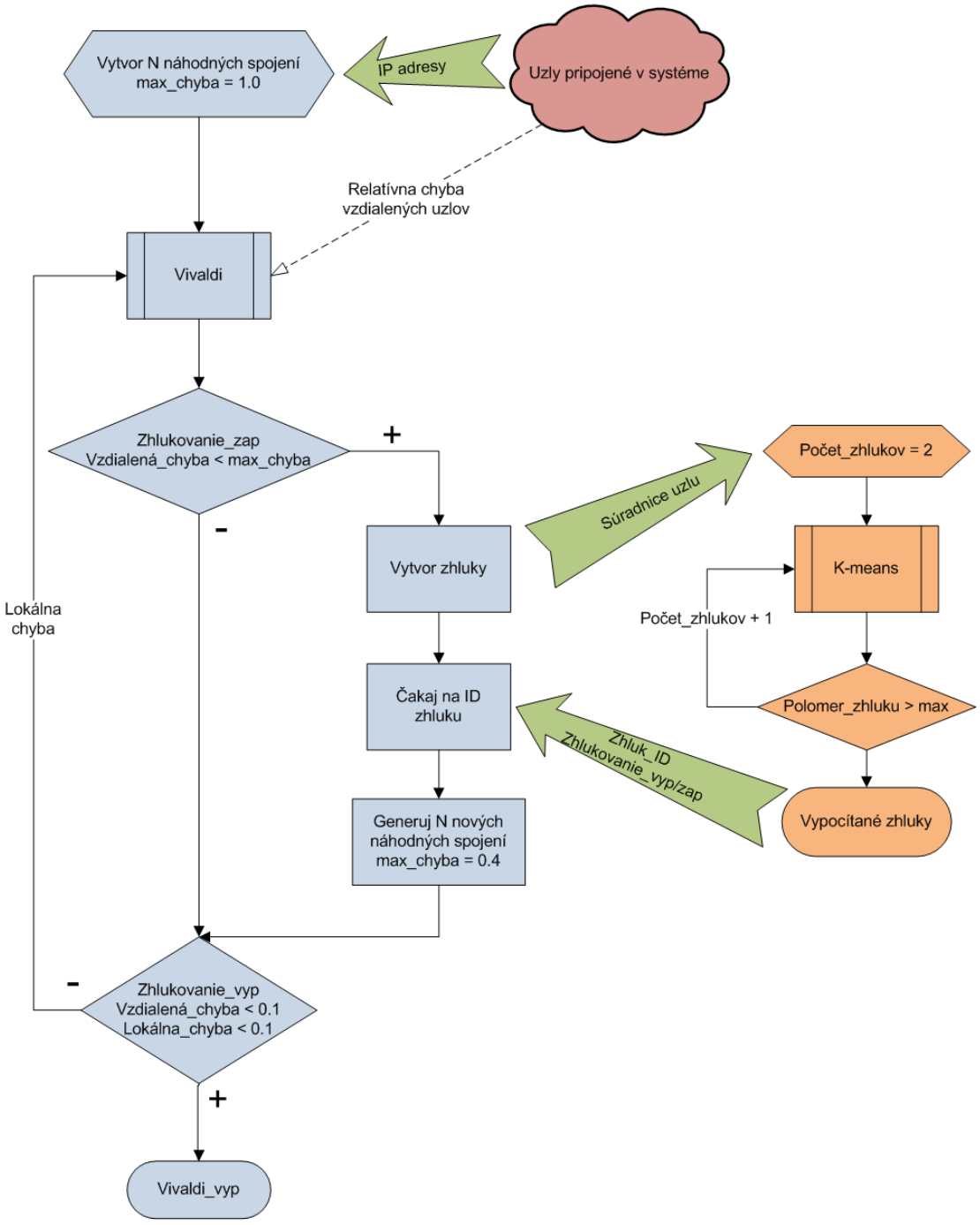
Takýmto spôsobom algoritmus pokračuje až do momentu, kedy prvok zhromaždišťa rozhodne o nepotrebnosti ďalšej aktualizácie zhlukov. V takomto prípade uzly začnú konvergovať na základe algoritmu Vivaldi až do chvíle, keď dosiahnu lokálnu aj vzdialenú relatívnu chybu pod hranicou 0.1. Potom sa celý proces výpočtu ukončí a neprebieha žiadna ďalšia sieťová komunikácia z dôvodu nájdenia požadovaných súradníc.

Určenie čísla K pre algoritmus K-means

Základná myšlienka výpočtu zhlukov je totožná s metódou K-means, ktorá bola predstavená v teoretickej časti pod kapitolou 1.5.1. Najväčším problémom tejto metódy je nájdenie potrebného počtu centroidov, teda určenie čísla K. Tento problém som sa snažil vyriešiť na základe vedomostí o navrhovanom súradnicovom systéme s prihliadnutím na náročnosť výpočtu.

Podľa publikácie [4] môžeme za krátku vzdialenosť, ktorá ovplivňuje výsledky algoritmu Vivaldi, považovať hodnoty do 30ms. Toto je rozhodujúca hodnota pre vytvorenie zhlukov, pretože požadujeme, aby uzly zaradené do rovankého zhluku boli od seba vzdialené najviac týchto 30ms. Na základe tejto informácie bolo odvodené určenie hodnoty K pre algoritmus K-means.

Inicializačná hodnota K je rovná dvom a po vytvorení zhlukov sa skontrolujú vzdialenosti uzlov od centroidu. Na súradnice uzlu, ktorý má najväčšiu vzdialenosť od centroidu a súčasne presahuje hodnotu 30ms, sa pridá nový centroid a hodnota K sa zvýši o jeden. Ďalej sa znova prepočítajú centroidy a vytvoria sa nové zhluky. Takýmto spôsobom algoritmus pokračuje až do chvíle, keď vzdialenosti uzlov od centroidu nepresahujú maximálnu stanovenú hodnotu. Tento postup znázorňuje vývojový diagram na obrázku 3.17.



Obrázok 3.17: Vývojový diagram vlastného algoritmu pre zhlukovanie uzlov

3.2.2 Výsledky simulácií

Navrhnutý algoritmus pre výpočet súradníc v sieti som sa rozhodol simulovať a overiť jeho konštantnosť na základe testov, ktoré boli vykonané pomocou rovnakého simulačného prostredia, ako tomu bolo v kapitole 3.1. Oproti predchádzajúcim simuláciam môžeme z obrázkov pozorovať zmeny v zafarbení jednotlivých uzlov. Týmto farebným označením sa od seba odlišujú uzly, ktoré sú priradené do rôznych zhlukov. Ostatné znázornenia v simulačnom okne sa zhodujú z kapitoly 3.1. Skutočná pozícia uzlu je zobrazená štvorčekom, kde predpovedaná poloha je znázornená čiernym bodom, ktorý je s uzlom spojený čiernou úsečkou. Vzájomné prepojenia uzlov, ktoré sú určené na komunikáciu z dôvodu výpočtu algoritmu, sú znázornené žltou úsečkou.

Najprv som testoval dosiahnutú správnosť výsledkov na základe náhodného generovania rozloženia uzlov v sieti. Po dlhom skúmaní a pátraní sa mi podarilo odladiť algoritmus aj s maximálnymi prahovými hodnotami tak, ako je to vidieť na vývojovom diagrame 3.17. Zvolené prahové hodnoty ako napr. *max_chyba* alebo hodnota 0.1 pre maximálnu dovolenú lokálnu a vzdialenú relatívnu chybu sú odvodené na základe mnohých pokusov. Ak by bola hodota *max_chyba* veľká, tak by algoritmus dosahoval značnej nepresnosti, ktorá by ovplivnila presnosť vytváraných zhlukov. Nízka hodnota zase spôsobuje zbytočné predĺženie doby výpočtu súradníc, ktoré by mohli obsahovať chybu spôsobenú krátkych vzdialeností medzi uzlami a ďalej by spôsobilo oveľa dlhší čas konvergencie. Z toho dôvodu bol hľadaný kompromis správneho nastavnia tejto hodnoty tak, aby algoritmus konvergovl rýchlo a presne.

Preto pri počiatočnom spustení algoritmu nám úplne nezáleží na presnosti, ale na rýchlosti rozprsknutia uzlov z inicializačnej hodnoty a určenie menej presných zhlukov. Až po určení prvých zhlukov je zmenená prísnejšia hodnota relatívnej chyby z hodnoty 1.0 na 0.4. Týmto spôsobom sa vytvorí prvé možné odbúranie nepresnosti vzniknutej z krátkych vzdialeností a získajú sa presnejšie hodnoty pre nové rozdelenie uzlov.

Ak prvok zhrmaždišťa zhlukov zistí, že obsahy uzlov v jednotlivých zhlukoch sa už ďalej nemenia, tak nie je dôvod, aby algortimus pokračoval v ďalšej aktualizácii súradníc za účelom ich zlepšovania. Preto prvok zhrmaždišťa odošle v pakete s číslom zhluku aj informáciu o nepotrebnosti aktualizovať obsahy zhlukov. Z toho dôvodu sa ďalej zameriame len na presnosť vypočítaných súradníc, ktoré musia nadobudbúť hodnoty 0.1 pre lokálnu aj vzdialenú relatívnu chybu. Po dosiahnutí tejto presnosti sa výpočet súradníc zastaví a tým sa ukončí aj celková komunikácia v sieti určená pre tento algoritmus.

Takto zameraný a rozčlenený súradnicový systém je pripravený odolať aj

problému častého a hlavne náhodného odpájania alebo pripájania uzlov do systému.

Pred začiatkom skúmania konkrétnych simulácií som sa najprv oboznámil so spávaním sa navrhnutého algoritmu pre rôzne metódy rozloženia uzlov, podobne ako tomu bolo v kapitole 3.1. Vzhľadom k tomu, že pre naše účely je najdôležitejšie, aby takýto systém konvergoval správne z pohľadu relatívnych vzdialeností, preto už pri ďalších simuláciach nebolo uvažované s nastavením uzlov do pasívneho stavu tak, ako tomu bolo už v spomenutej kapitole.

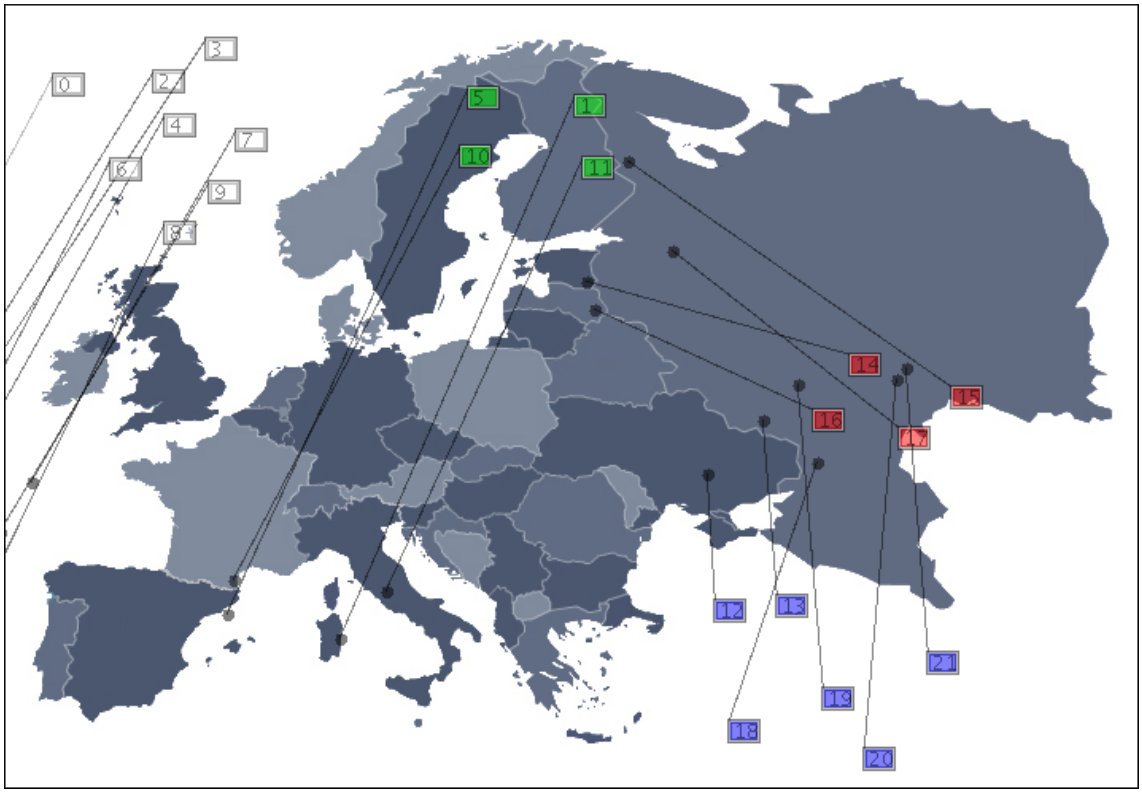
Na základe výsledkov z rôznych vykonaných testov som sa rozhodol podrobne simulovať tri rôzne rozloženia uzlov v systéme, ktoré boli fyzicky zoskupené do štyroch zhlukov. Simulácia bola spustená s očakávaním získania správneho rozdelenia uzlov na zhluky s minimálnou relatívnou chybou. Ako už bolo spomenuté skôr, tak jednou z hlavných nevýhod čistého algoritmu Vivaldi je nutnosť vytvorenia spojení medzi všetkými uzlami zaradených do systému. Mojim cieľom bolo získanie závislosti, ktorá by vystihovala nutný počet spojení tak, aby súradnicový systém konvergoval podľa očakávania.

Vzhľadom k tomu, že celý algoritmus funguje na náhodne generovaných spojeniach k susedným uzlom, tak pri každej simulácii rovnakých počiatočných stavoch boli získané rôzne výsledky. Preto boli jednotlivé simulácie vykonané viackrát a výsledná hodnota znázornená do rozptylu výsledkov. Počet simulácií pre každú nastavenú vzdialenosť sa pohyboval v rozmedzí 15 až 20 opakovaní v závislosti na premenlivosti nameraných hodnôt.

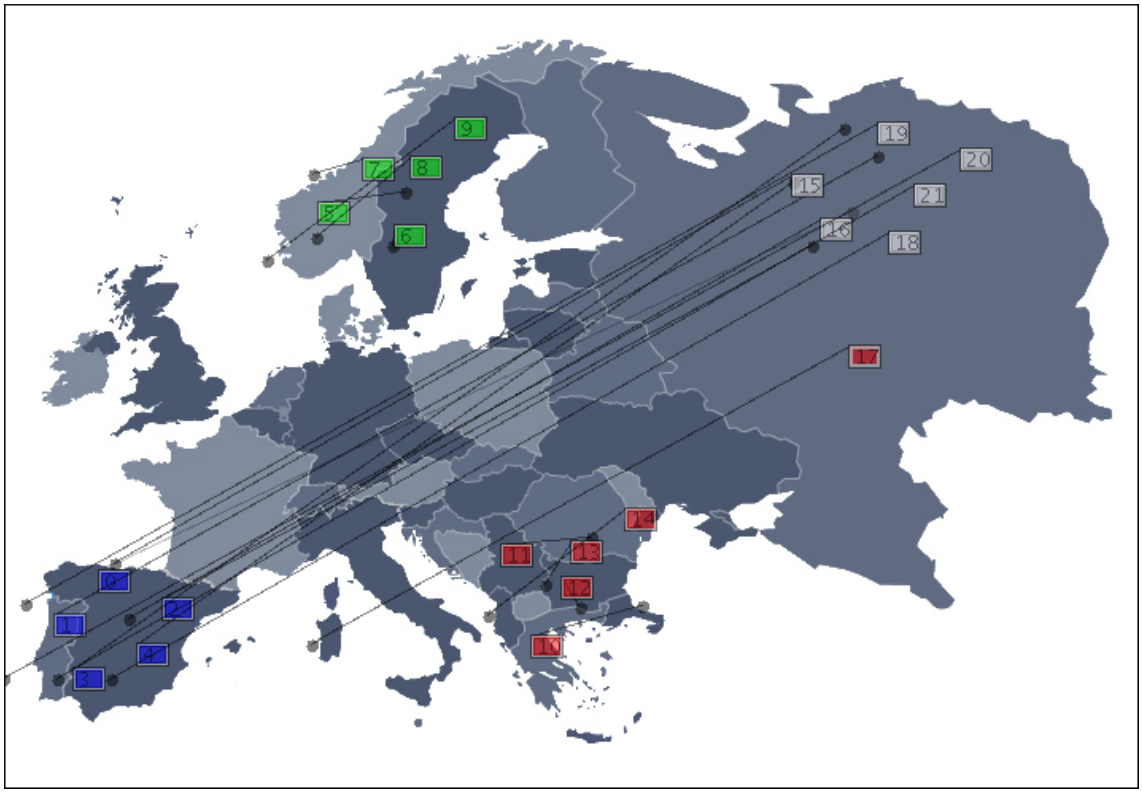
Rozloženie použitých troch verzií rozmiestnenia uzlov v systéme sú viditeľné z nasledujúcich obrázkov 3.18, 3.20 a 3.19. Bolo použitých 22 uzlov pri ktorých sa nastavoval počet nahodných spojení od hodnoty 3 až po maximálne prepojenie každý s každým. Výsledky týchto simulácií sú vynesené do grafov 3.21 a 3.22.

Na prvom grafe môžeme pozorovať závislosť počtu vytvorených zhlukov na základe zvyšujúceho sa počtu vzájomných náhodných spojení. Z vykreslenia pre všetky testované rozloženia si môžeme všimnúť, že so zvyšujúcim sa počtom náhodných spojení sa zlepšuje celkový počet vytvorených zhlukov. Z tohto grafu môžeme vyvodiť tvrdenie, že pre správne vytvorenie zhlukov budeme potrebovať polovicu spojení z celkového počtu uzlov v systéme, aby bol vytvorený rovnaký počet zhlukov ako sa nachádza aj fyzicky.

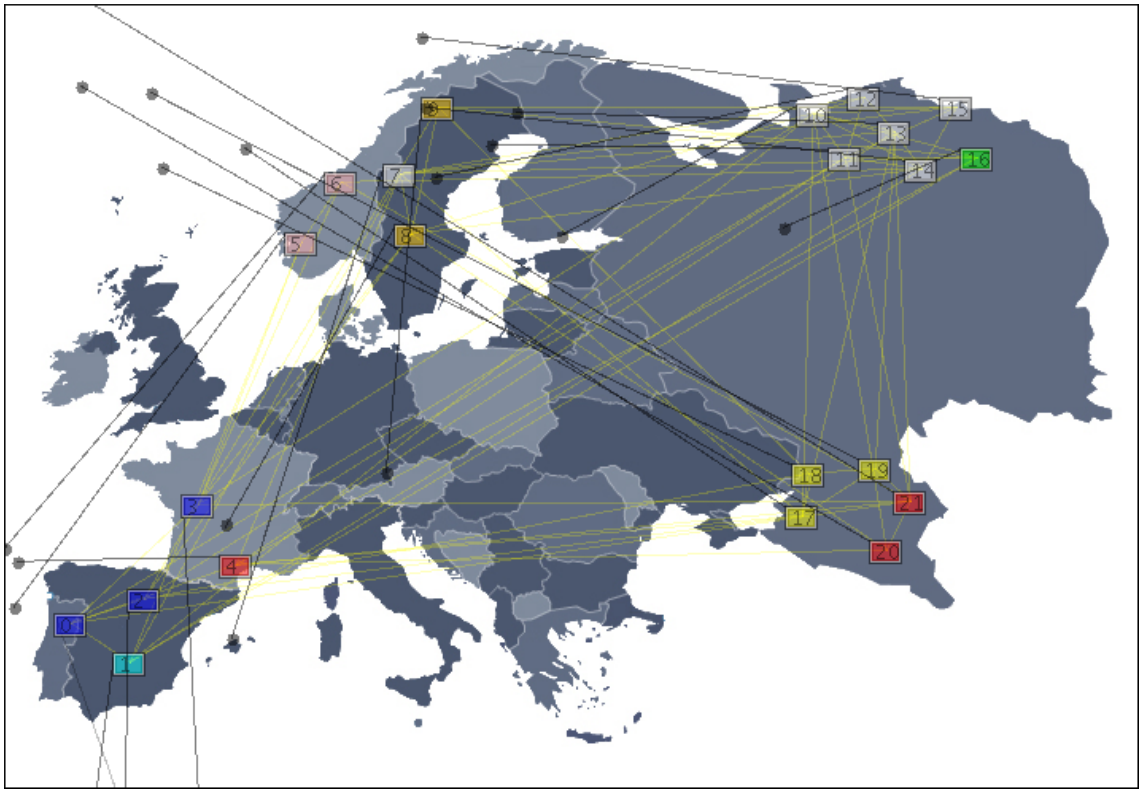
Z druhého grafu stojí za povšimnutie percentuálny podiel správne ukončených simulácií pri zvyšujúcom sa počte spojení. Za správne ukončenú simuláciu sa považoval výsledný stav, ktorý mal správny počet vytvorených zhlukov a súčastne boli blízke uzly priradené do rovnakého zhluku tak, ako je to znázornené na obrázku 3.18. Chybný výsledok, keď je správny počet zhlukov, ale nie sú do nich správne zaradené uzly, je zase znázornený na obrázku 3.19.



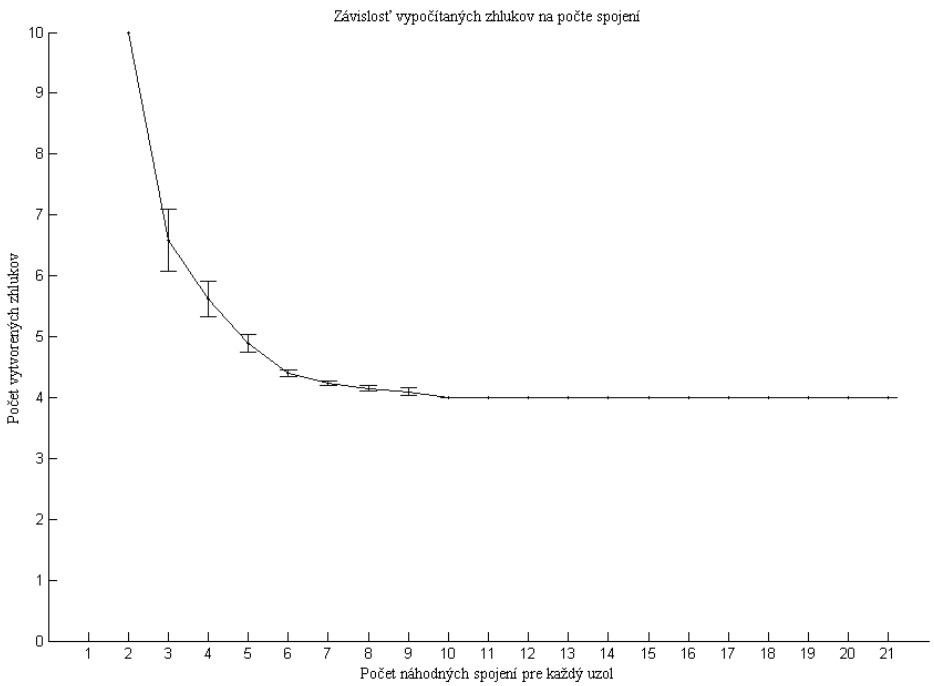
Obrázok 3.18: Správne vytvorené zhluky pri optimálnom počte vzájomných spojení



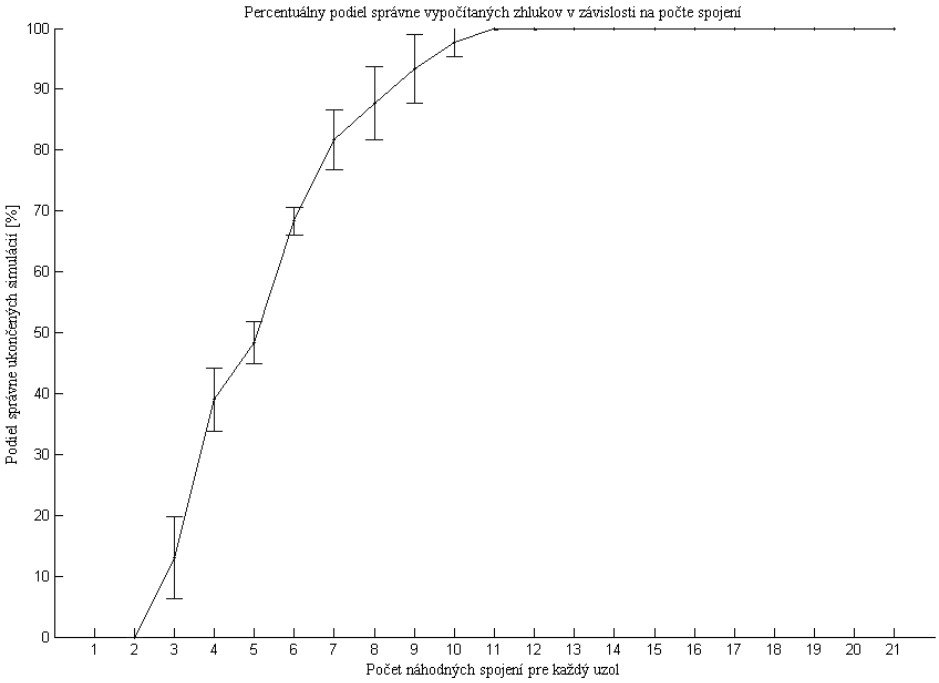
Obrázok 3.19: Nesprávne zamerané uzly v zhlukoch



Obrázok 3.20: Priebeh výpočtu zhlukov pri nízkom počte vzájomných spojení



Obrázok 3.21: Závislosť vyočítaných zhlukov na počte náhodných spojení



Obrázok 3.22: Percentuálny podiel správne zaradených uzlov do zhlukov

3.3 Výpočet súradníc na základe meraní reálnej siete

V predchádzajúcich kapitolách som sa zaoberal výpočtom súradníc na základe vykonaných simuláciach, kotré predpokladali vytvorené virtuálne spojenia medzi uzlami. Simulačné prostredie slúžilo hlavne pre testovacie účely navrhnutého súradnicového systému. Vzhľadom k tomu, že sa potrebné RTT vzdialenosti medzi uzlami počítali na základe ich skutočnej vzdialenosti, tak vstupné hodnoty pre vytvorený algoritmus pôsobili príliš ideálne a bez ohľadu na možnosť rôznych rýchlostí prenosových ciest.

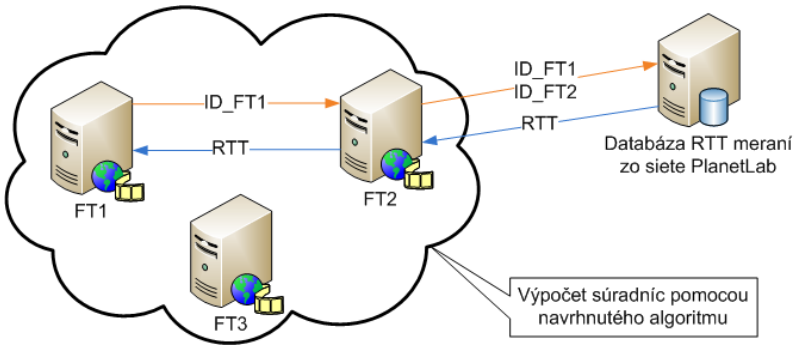
Hlavnou výhodou takýchto meraní bola možnosť sledovať výpočet súradníc na základe skutočného rozmiestnenia uzlov. Po ukončení simulácie, prípadne aj v jej priebehu bolo možné zhodnotiť funkčnosť a užitočnosť zamerania uzlov pomocou umelých súradníc. Všetky tieto poznatky a skutočnosti z meraní boli popísané v predchádzajúcich kapitolách.

V ďalšom postupe testovania navrhnutého súranicového systému som sa rozhodol ďalej už nezaoberať sa presnosťou určených súradníc z pohľadu dopadu na skutočné rozloženie. Hľadal som určitý spôsob ako otestovať funkčnosť výpočtu súradníc

na reálnej sieti. Jedna z možností by bolo použiť niekoľko desiatok počítačov zapojených v dostatočne rozsiahlej sieti, ktoré by medzi sebou komunikovali a aktualizovali svoje súradnice podľa navrhnutej metódy 3.17. Takéto riešenie by si vyžadovalo značné úpravy a ďalší potrebný vývoj pre implementáciu aktuálneho riešenia do podoby vhodnej pre nasadenie v reálnej počítačovej sieti.

Oveľa jednoduchšie bolo zachovať vytvorenú architektúru a zaoberať sa len problémom RTT vzdialeností. Pre získanie databázy reálnych hodnôt som si vybral experimentálnu počítačovú sieť PlanetLab, ktorá tvorí globálnu výzkumnú sieť pre podporu nových sieťových služieb. Využíva sa k vývoju nových technológií pre distribuované ukladanie, mapovanie sietí, peer-to-per systémov, distribuovaných rozptylových tabuliek a spracovávanie front. Uzly tejto siete sú rozložené globálne po celej planéte a umožňujú testovať v reálnom prostredí celosvetového Internetu. Projekt PlanetLab začal pôsobiť v roku 2002 medzi niekoľkými americkými univerzitami a v súčasnosti má unikátna sieť konsorcia PlanetLab štatút celosvetovej laboratórie pre sieťové aplikácie s 1006 uzlami distribuovanými do všetkých častí sveta. Vďaka združeniu CESNET majú do tejto siete prístup aj odborníci z Českej republiky.

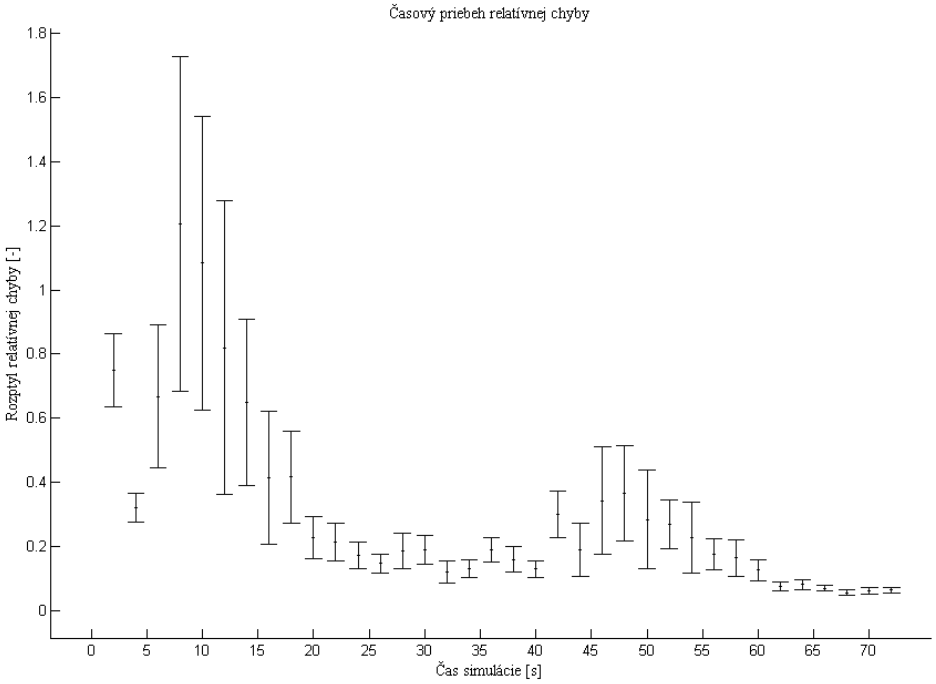
Získaný súbor nameraných hodnôt RTT vzdialeností zo siete PlanetLab bol použitý ako referenčný zdroj dát pre vytvorené simulačné prostredie s navrhnutým algoritmom. Pri spustení výpočtu súradníc už neprebieha meranie RTT vzdialenosti medzi vzájomným virtuálnym spojením uzlov v systéme, ale každý uzol, ktorý obdrží paket zo žiadosťou na zmeranie RTT vzdialenosti, odošle v odpovedi hodnotu, ktorá bola získaná z databáze meraní PlanetLab na základe identifikačných čísiel uzlov. Tento proces demonštruje aj nasledujúci obrázok 3.23.



Obrázok 3.23: Meranie RTT vzdialeností na základe hodnôt získaných zo siete PlanetLab

Je zrejmé, že pri takomto výpočte súradníc nemôžeme pozerať na výsledok simulácie podľa rozloženia uzlov v simulačnom okne aplikácie. Vzhľadom k tomu, že nepoznáme skutočné pozície uzlov, tak nás na výsledku zaujíma len priebeh relatívnej chyby, ktorá by sa mala postupne znižovať na minumálnu hodnotu.

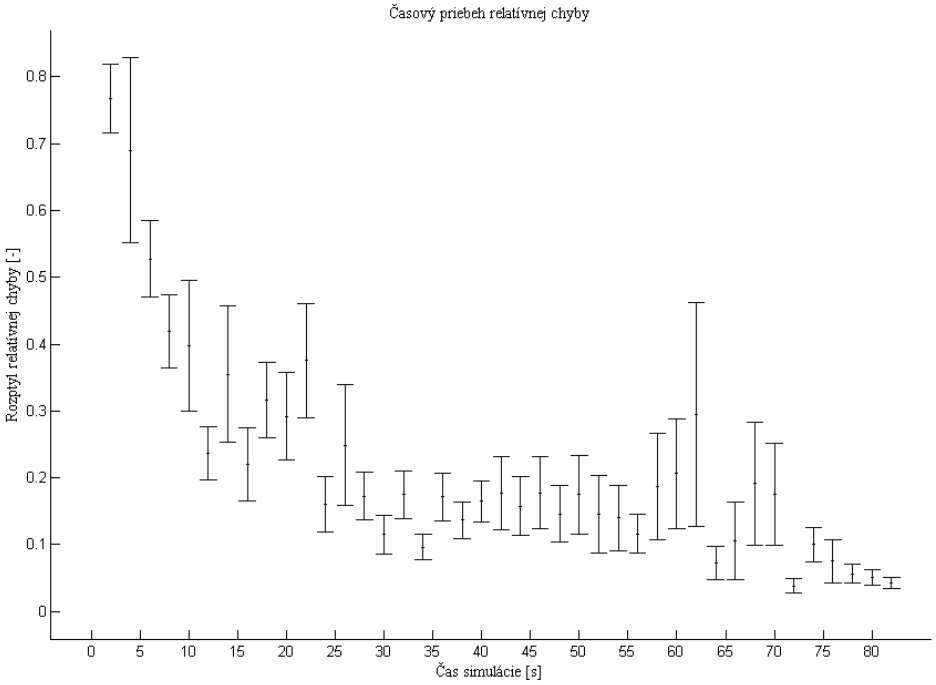
Na začiatku simulácie boli uzlom priradené náhodné inicializačné súradnice, ktoré však nemali nijaký vpliv na priebeh výpočtu. Z výsledkov môžeme aspoň sledovať uzly zaradené do rovnakých zhlukov a na základe toho usúdiť, že sa s najväčšou pravdepodobnosťou nachádzajú blízko seba aj v reálnom prostredí celosvetového Internetu. Túto skutočnosť demonštrujú výsledky simulačných okien po ukončení zameriavania uzlov s reálnymi hodnotami RTT, ktoré sú na obrázkoch 3.28 a 3.29, prípadne 3.26 a 3.27. Na základe databáze zo siete PlanetLab by bolo možné dohľadať názvy a umiestnenie jednotlivých uzlov a výsledky porovnať s predpovedanými súradnicami.



Obrázok 3.24: Priebeh relatívnej chyby pri náhodnom spojení šiestich uzlov

Pretože pre naše účely je najdôležitejšia predpoveď polohy uzlu z pohľadu vzdialenosti, tak som sa zaoberal len relatívnou chybou predikcie. Pre zjednodušenie vykreslenia som v grafoch vyznačil rozptyl hodnôt na základe relatívnej chyby jednotlivých uzlov.

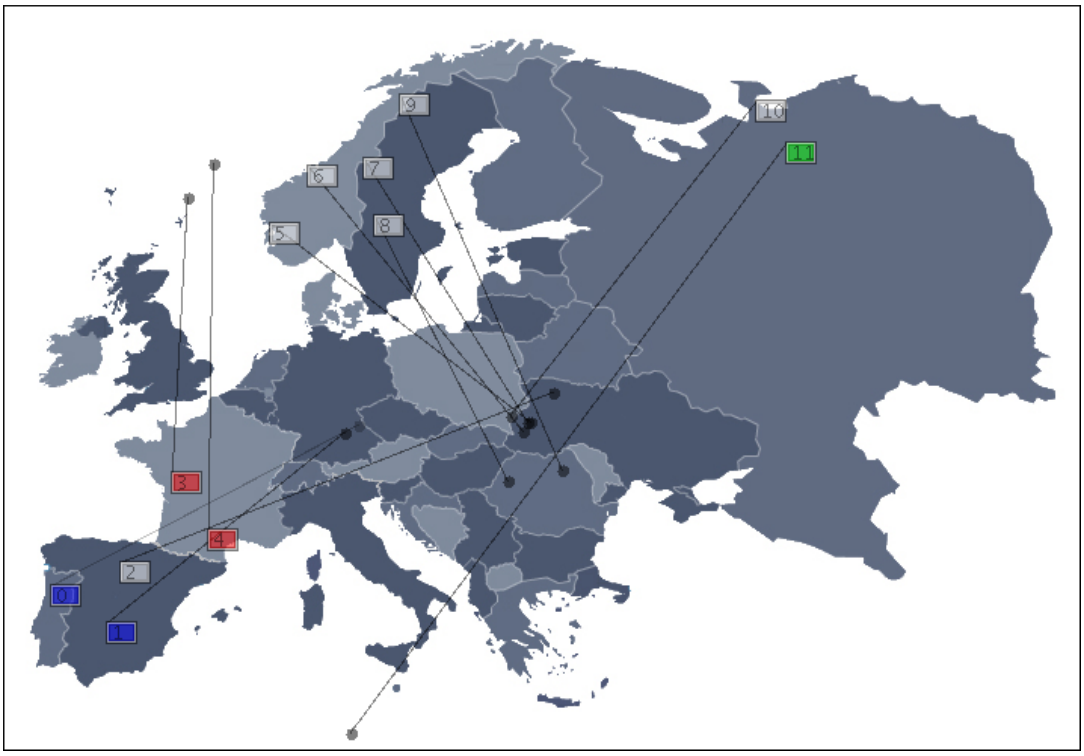
Pre porovnanie boli vykonané dva výpočty súradníc rovnakých vstupných hodnôt RTT vzdialensotí siete PlanetLab. Výsledky priebehu relatívnej chyby pri úplnom vzájomnom prepojení uzlov sú znázornené v grafe 3.25. Ako už bolo dokázané skôr na základe simulácií z kapitoly 3.2.2, tak navrhnutý algoritmus funguje správne aj pri menšom počte náhodných prepojení uzlov. Z toho dôvodu bolo vykonané meranie na reálnej sieti s polovičným počtom spojení, ktoré je zobrazené v grafe 3.24.



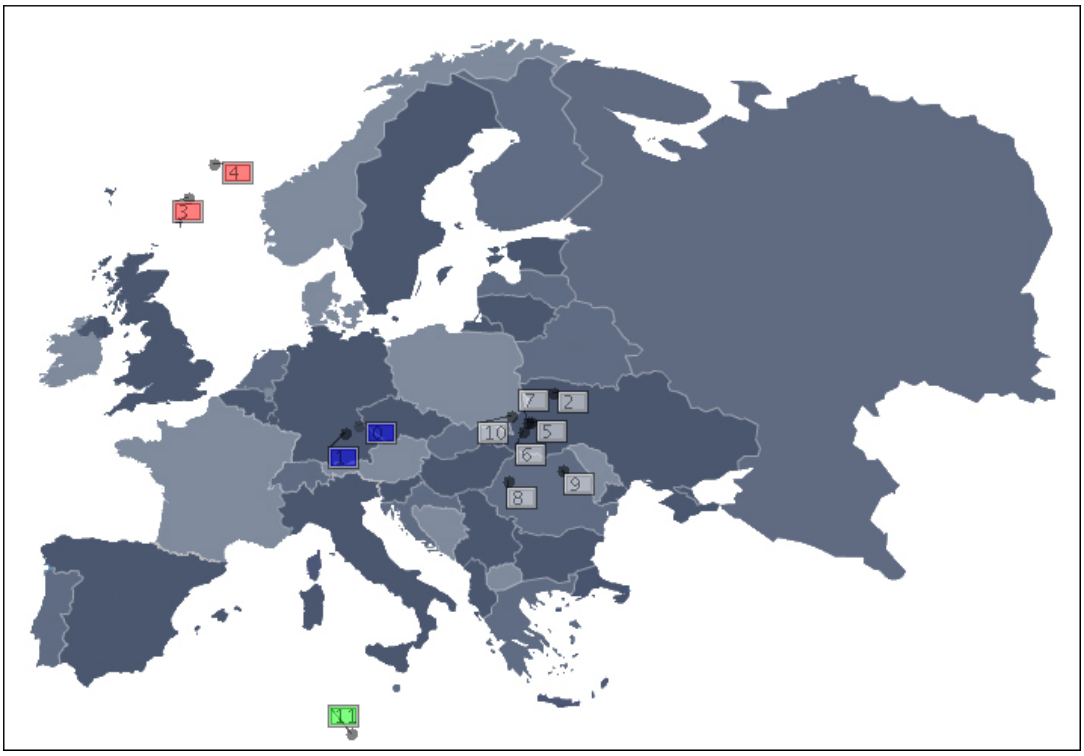
Obrázok 3.25: Priebeh relatívnej chyby pri úplnom vzájomnom spojení medzi uzlami

Z vykrelsených priebehov relatívnej chyby môžme pozorovať približne rovnaké výsledky. Jedine pri polovičnom počte spojení stojí za povšimnutie väčší rozptyl chyby na začiatku výpočtov, ktorý však nemal vpliv na celkový čas konvergencie uzlov, pretože v oboch prípadoch súradnice uzlov konvergovali k požadovaným hodnotám v priebehu jedenj minúty. Po dosiahnutí požadovanej presnosti sa aktualizácia súradníc preruší a tak v sieti neprebieha žiadna komunikácia z dôvodu réžie výpočtov.

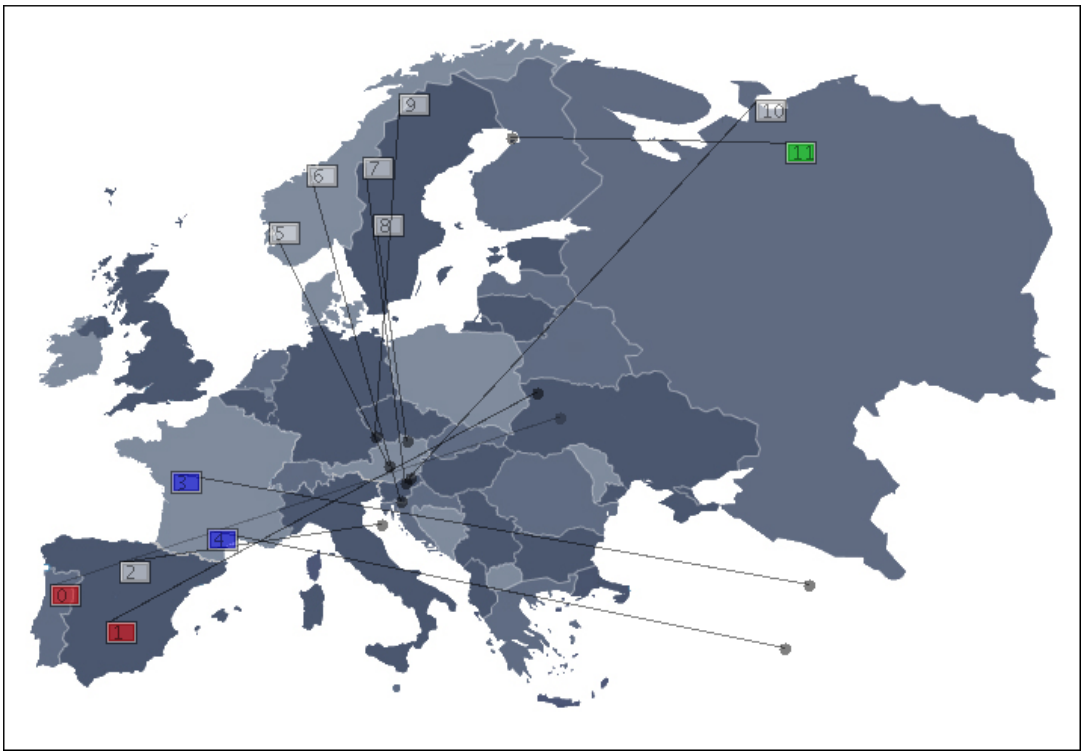
Z doziahnutých výsledkov sa nám uplatnili tvrdenia vyjadrené na základe simulácií aj pre reálne siete.



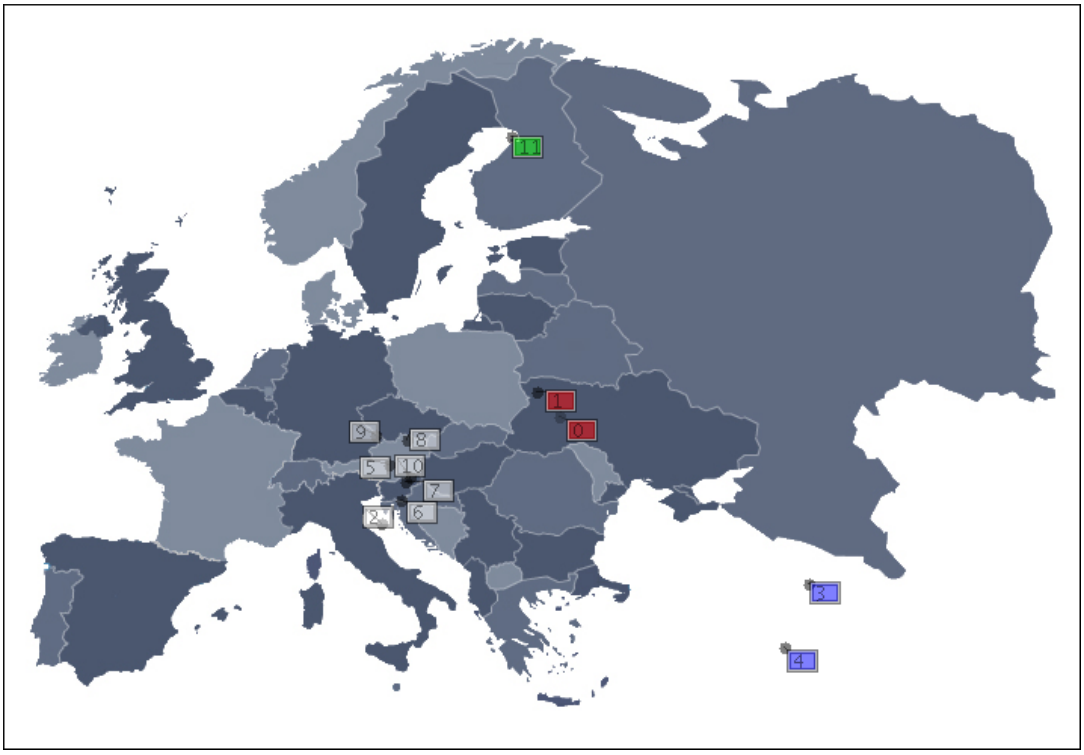
Obrázok 3.26: Výsledok simulácie pre polovičný počet náhodných spojení



Obrázok 3.27: Predpokladané zoskupenie uzlov pri polovičnom počte spojení



Obrázok 3.28: Výsledok simulácie pre plný počet vzájomných spojení



Obrázok 3.29: Predpokladané zoskupenie uzlov pre plný počet vzájomných spojení

3.4 Návrh implementácie systému

Táto práca by mala prispieť k prebiehajúcemu fakultnému výzkumu v oblasti prenosu signalizácie IPTV systému. Predchádzajúce kapitoly sa zaoberali návrhom vhodného súradnicového systému, ktorý by mohol byť využitý pre potreby hierarchyckej agregácie, ktorá bola popísaná v kapitole 2.

Navrhnutá a vytvorená komunikácia za účelom aktualizácie súradníc bola doposiaľ použitá v simulačnom prostredí, pomocou ktorého sa mi podarilo pochopiť vlastnosti takéhoto systému. Toto prostredie bolo užitočné aj pri odlaďovaní a celkovom vývoji súradnicového systému, či už s použitím simulačných dát alebo hodnôt z reálnej siete.

Z časovej a obsahovej náročnosti nebolo možné v rámci tejto práce nasadiť navrhnutý súradnicový systém do reálnej hierarchyckej stromovej štruktúry podľa 2.1. Z toho dôvodu sa budem v tejto časti zaoberať teoretickým návrhom takejto implementácie.

Na obrázku 2.1 môžeme vidieť prvok v sieti označený ako cieľ spätnej väzby (angl. Feedback Target) (FT). Z kapitoly 2.1 plynie potreba rovnomerného rozmiestnenia týchto prvkov v mieste prijímačov, preto navrhovaný súradnicový systém bude použitý práve na týchto prvkoch. FT v reálnom sieťovom prostredí bude vyzerať ako samostatný dedikovaný server, ktorý bude vyťažený zberom RR alebo RSI správ a následnou agregáciou do novej RSI správy [1].

Z popísaného dôvodu je potrebné, aby bol takýto server nevyťažený žiadnou inou aplikáciou ako je jeho pôvodný účel. Pretože pre správnu hierarchickú agregáciu je potrebná správne určená poloha jednotlivých FT, tak pri spustení každého ďalšieho budú najprv vypočítané súradnice za použitia navrhnutého algoritmu, ktorý bol predstavený na vývojovom diagrame 3.17.

Ďalším zavedeným prvkom v systéme podľa [1] je FT manažér (FTM), ktorého úlohou je rozmiestniť FT stanice tak, aby vytvorili stromovú štruktúru a súčasne nedochádzalo k zbytočnému plýtvaniu prostriedkov siete. FTM aplikácia bude spustená na samostatnom dedikovanom serveri a jeho súčasťou bude aj vytváranie zhlukov na základe vypočítaných súradníc, ktoré bolo predstavené v kapitole 3.2.1 ako prvok zhromaždišťa (RP).

Pre inicializáciu polohy jednotlivých FT uzlov bude potrebné určiť počet náhodných spojení medzi uzlami, ktorý by mal dosahovať hodnoty polovičného počtu prvkov v systéme. Počas výpočtu súradníc nie je vhodné, aby bola súčasne spustená primárna aplikácia týchto sieťových prvkov, pretože bude značne vyťažená dostupná šírka pásma. Po dosiahnutí požadovanej relatívnej presnosti pošle každý FT informáciu o svojich súradniciach do FTM, na ktorom bude mimo iného spustený aj RP algoritmus pre určenie zhlukov. FTM bude jednotlivým FT uzlom odosielať

informáciu o priradenom zhluku a súčasne aj o prípadnej nepotrebnosti ďalšieho výpočtu súradníc, tak ako to bolo pojednávané v kapitole 3.2.1.

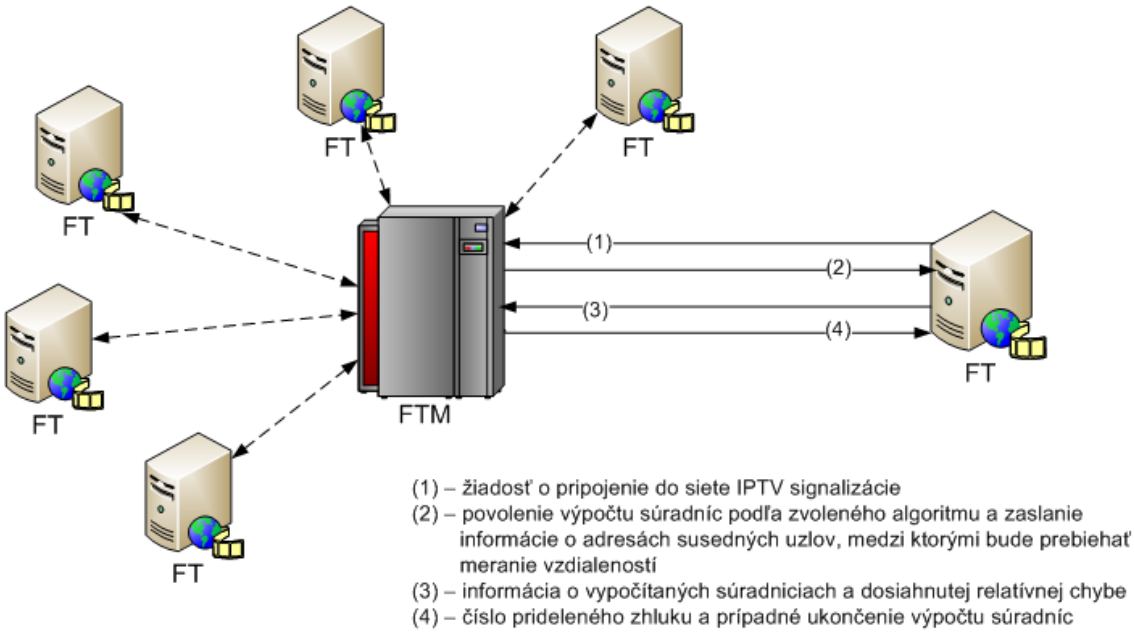
Dôležitým opatrením v porovnaní so simulačným prostredím je možnosť vzniku kolízie na sieti pri komunikácii FT uzlov smerom k FTM. Ak FT vypočíta svoje súradnice s požadovanou presnosťou, tak ich preposiela do FTM. Kedže bude v sieti zapojený veľký počet FT uzlov, tak je potrebné vyhnúť sa stavu, kedy všetky uzly budú posielať požadované dáta do FTM v rovnaký čas. V opačnom prípade by hrozilo preťaženie komunikačného kanálu smerom k FTM. Tento problém je možné riešiť použitím náhodnej hodnoty časovača z definovaného intervalu pre odoslanie súradníc tesne po ich výpočte. Interval pre tento časovač by sa mohol pohybovať v rozmedzí 0 až 5 sekúnd.

Celkový výpočet súradníc v systéme by mal konvergovať rýchlo, pretože počas tejto doby nebude možné využívať primárnu aplikáciu FT uzlov. Vykonané simulácie ukázali, že čas konvergenicie sa pohybuje v rádoch jednotiek minút a tak by tento výpočet nemal vážne ovplivniť primárnu funkčnosť týchto uzlov.

Predchádzajúce úvahy vychádzali z predpokladu, že v systéme sa nenachádzal v aktívnom stave žiadny FT uzol a neprebiehalo posielanie signalizačných správ. V tomto prípade išlo len o inicializačné spustenie celého systému. Je zrejmé, že takýto stav nemôžeme považovať za častý a preto treba navrhnúť postupné prídávanie FT uzlov do spusteného systému tak, aby nebolo nutné obmedzovať prevádzku.

Každý práve pripojený FT kontaktuje FTM so žiadosťou o pripojenie do siete. FTM bude obsahovať databázu jednotlivých FT mimo iné aj s informáciou o ich súradniciach a pridelených zhlukoch. Všetky FT budú čakať na spustenie výpočtu súradníc podľa navrhnutého algoritmu aj s informáciou o počte potrebných náhodných spojení. Ak ešte nebudú vytvorené žiadne zhluky a vypočítané súradnice, tak bude potrebné zvoliť časový interval počas ktorého FTM bude očakávať ďalšie žiadosti pripojenia FT uzlov do systému. Ďalej je potrebné stanoviť minimálny počet uzlov, ktoré musia prejaviť záujem o pripojenie do siete. Ak budú tieto podmienky splnené, tak FTM pošle jednotlivým FT odpoveď, ktorá bude obsahovať počet potrebných náhodných spojení, prípadne adresy uzlov na ktoré môžu FT posielať požiadavky spojené s výpočtom súradníc. Celý tento priebeh komunikácie demonštruje nasledovný obrázok 3.30.

Druhou spomenutou možnosťou je pripojenie FT uzlu do rozbehnutého systému, ktorý bude plniť svoju úlohu v rámci šírenia signalizácie v IPTV sieti. Takýto nový FT bude potrebné zamerať voči ostatným. FTM novému prvku povolí výpočet svojich súradníc, ale už s použitím pripojených FT, ktoré v predpovedaných súradniciach vykazovali najmenšiu relatívnu chybu. Takto sa podarí zameranie nového uzlu v pomerne krátkom časovom intervale aj s určením správneho zhluku v ktorom sa bude nachádzať.



Obrázok 3.30: Pripájanie nového FT do systému

Po krátkom časovom intervale, ktorý by mal byť rádovo niekoľko minút, bude pripojený FT zameraný a v sieti bude môcť plniť svoju primárnu úlohu v rámci šírenia signalizácie. Nie je už ďalej potrebné, aby sa každý FT zaoberal aktualizáciou svojich súradníc, preto môže byť po dosiahnutí požadovanej presnosti algoritmus automaticky vypnutý, prípadne výpočet súradníc môže pokračovať, ale už len s obmedzeným zaťažením siete. To by predstavovalo napríklad možnosť vyslať jeden požiadavok na susedný FT v časovom intervale 60 sekúnd za účelom získania polohy a predikovanej chyby.

Suradnicový systém na prijímačoch by mal byť zvolený iným spôsobom ako tomu bolo v prípade FT uzlov. Pre zameranie prijímačov by mohol byť použitý algoritmus GNP [17], ktorý môže za referenčné body (angl. landmarks) využívať vypočítané súradnice FT uzlov. Určenie súradníc na prijímačoch prekračuje rozsah tejto práce a preto som sa touto problematikou podrobnejšie nezaoberal.

3.5 Vyvesenie vytvorených aplikácií na webové stránky

Pre potreby rozšírenia vytvorených aplikácií som sa rozhodol využiť jeden zo školských serverov. Vzhladom k tomu, že aplikácie boli vytvorené v programovacom jazyku Java, tak existujú dve reálne možnosti ako takéto aplikácie distribuovať. Môžu to byť aplety alebo využitie Java Web Start (JWS) technológie.

Vybral som si druhú zo spomínaných možností a to z dôvodu jednoduchej distribúcie a aktualizácie. Koncový užívateľ klikne na spúšťací skript JWS (súbor JNLP) a aplikácia sa sama nainštaluje a spustí. Web Start aplikácie môžu vyžadovať rôzne verzie Javy. Táto verzia sa automaticky nainštaluje a odpadá tak problém s kompatibilitou jednotlivých verzií.

JWS je klasická aplikácia a nemusí mať žiadneho predka typu Applet a jej prevádzka nie je závislá na webovom prehliadači. Pre činnosti, ktoré vyžadujú ďalšie práva musíme použiť špecálne API. Aplikácia je cachovaná v prostredí JWS a preto sa spúšťa veľmi rýchlo. Pred spustením je tak treba skontrolovať či nie je k dispozícii novšia verzia, ktorá by sa potom automaticky nainštalovala.

Protokol pre JWS sa používa už spomenutý JNLP (Java Network Launch Protocol). JWS je referenčná implementácia, ktorá je zdarma poskytovaná spolu s JRE/JDK.

Vytvorené aplikácie sú prístupné na webových stránkach `http://adela.utko.feec.vutbr.cz/projects/` v sekcii Internet coordinate systems.

3.5.1 Vytvorenie aplikácie

Vytvorené aplikácie pre Java Web Start je nutné distribuovať v súboroch jar, ktoré sú archívom zip. Jedinou nutnosťou je vytvorenie JNLP súboru, ktorý funguje ako spôšťací skript pre našu aplikáciu. Finálny odkaz na našu aplikáciu nie je na súbor jar ale na JNLP, ktorý zaistí spustenie Java Web Start.

JNLP je súbor vo formáte XML, v ktorom sa poskytujú rôzne informácie o JWS ako napríklad o ceste k súborom aplikácie, informácie o autorovi, bezpečnosti a môže tu byť uvedená spúšťacia trieda.

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="http://adela.utko.feec.vutbr.cz/multicastssm/vivaldi2/"
    href="Vivaldi.jnlp">

<information>
  <title>Vivaldi Simulation</title>
  <vendor>Brno University of Technology</vendor>
  <homepage href="http://adela.utko.feec.vutbr.cz/projects/"/>
  <description>Simulation of Vivaldi algorithm</description>
  <description kind="short">Vivaldi Simulation</description>
  <icon href="icon.gif"/>
  <icon kind="splash" href="splash.jpg"/>
  <offline-allowed/>
</information>

<resources>
  <j2se version="1.6+"/>
  <jar href="http://adela.utko.feec.vutbr.cz/multicastssm/vivaldi2/Vivaldi.jar" main="true" />
  <jar href="http://adela.utko.feec.vutbr.cz/multicastssm/vivaldi2/lib/commons-lang.jar"/>
  <jar href="http://adela.utko.feec.vutbr.cz/multicastssm/vivaldi2/lib/jcommon-1.0.10.jar"/>
  <jar href="http://adela.utko.feec.vutbr.cz/multicastssm/vivaldi2/lib/log4j-1.2.14.jar"/>
</resources>

<application-desc main-class="cz.vutbr.feec.vivaldi.gui.VivaldiSimulation"/>

</jnlp>
```

V XML súbore je ako prvý uvedený tag s verziou použitého XML a typom kódovania. Ďalší tag *<jnlp>* s atribútom *spec*, kde je určená podporovaná verzia JNLP, *codebase* určuje cestu k jar súboru aplikácie a nakoniec *href*, ktorý pridáva ikonu do Application Manageru.

Nasledujúci tag *<information>* poskytuje informácie o autorovi a aplikácii, ktoré sa zobrazia pri načítaní. Tag *<offline–allowed>* dovoluje spustiť aplikáciu aj keď počítač práve nie je pripojený k internetu.

Predposledný tag *<resources>* umožňuje určiť požadovanú verziu Javy a meno jar súboru, ktorý obsahuje aplikáciu. V poslednom tagu je určená požadovaná spúšťacia trieda.

Podpísanie JAR archívu

Pre naše účely je možné podpísať jar súbory self-signed certifikátom, ktorý nepodpísala žiadna certifikačná autorita, ale je podpísaný sám sebou. Všetky utility ktoré k tomu potrebujeme sú súčasťou JDK. Pre reálne aplikácie je lepšie použiť certifikát od niektorej známej autority.

Najprv je potrebné vytvoriť úložisko do ktorého budú pripojené naše certifikáty. Keytool sa bude dotazovať na základné informácie a heslo. Ďalším krokom je vytovrenie self-signed certifikátu. Nakoniec musíme podpísať týmto certifikátom všetky jar súbory, ktoré patria k aplikácii.

- *keytool -genkey -keystore myKeystore -alias myself*
- *keytool -selfcert -alias myself -keystore myKeystore*
- *jarsigner -keystore myKeystore Vivladi.jar myself*

4 ZÁVER

Na základe vykonaných simulácií boli pozorované výsledky algoritmu Vivaldi s adaptívnym časvým krokom a zhodnotené jeho vlastnosti v popísaných troch simulačných scenároch. Bolo zistené, že použitie Vivaldi nemá veľký vpliv na počet uzlov pripojených v systéme, ale skôr na spôsobe ich rozmiestnenia. Práca vyjadruje chybu, ktorá vzniká vždy pri použití takéhoto systému a spôsob ako túto chybu minimalizovať. Zatiaľ čo veľká relatívna chyba napomáha k presnejšej konvergencii súradníc zúčastnených uzlov, tak absolútnu chybu nemôžme do výpočtu vôbec zahrnúť a pri simuláciach mala len informatívny charakter, aby bola viditelná rýchlosť konvergencie súradníc k reálnym hodnotám.

Z pozorovaných simulácií bolo zistené, že ak sa na súradnicový systém pozeráme z pohľadu výberu najbližšieho uzlu, tak potrebujeme dosiahnuť nízku relatívnu chybu, rádovo postačujú jednotky desatín. V systéme s nízkym počtom uzlov je to dosiahnuteľné v rádoch jednotiek minút. Pri väčšom počte uzlov bude tento čas oveľa dlhší z dôvodu komunikácie spôsobom každý s každým. Tento problém bol vyriešený vlastným algoritmom výpočtu súradníc, ktorý ako základ využíva Vivaldi s adaptívnym časovým krokom a metódu zhlukovania uzlov. Pre každý uzol bola zvolená konečná množina susedných s ktorými sa zameriava. Takto bol dosiahnutý krátky čas konvergencie súradníc s minumálnou relatívnou chybou aj pre rozsiahle systémy.

Druhým hodnoteným kritériom bola absolútna chyba, ktorá sa minimalizuje pomerne ťažko a to len za predpokladu, že zameriavaný uzol používa pre výpočet súradnice susedných, ktoré už majú túto chybu pomerne malú. Pre takéto presné zameranie by bolo potrebné, aby uzol komunikoval aspoň s 25-timi až 30-timi percentami uzlov s malou absolútnou chybou. Pri pripájaní nového uzlu do systému bude vhodné ak sa za jeho susedné uzly vyberú také, ktoré majú nízku relatívnu chybu a tak budú vystupovať ako dôverihodné zdroje pre výpočet vlastných súradníc. Absolútna chyba už nebola ďalej pri návrhu algoritmu zohľadňovaná, pretože pre naše účely použitia súradnicového systému sú dôležité relatívne vzdialenosti.

Pri simuláciach bol predpoklad, že každý uzol pošle žiadosť k susednému uzlu približne každých 110 ms. Z tohto vzťahu sa odvíja aj spotrebovaná šírka pásma pre prenos synchronizačných paketov, ktorá sa pohybuje okolo hodnoty 100 kb/s. Z toho dôvodu bola pri návrhu vlastného algoritmu zohľadňovaná rýchlosť konvergencie a pri dosiahnutí požadovanej presnosti sa výpočet súradníc ukončí. Výpočtové a sieťové prostriedky tak budú uvoľnené pre primárne aplikácie uzlov.

Táto práca nadviazala na fakultný výzkum v oblasti súradnicových systémov a simulačné prostredie s navrhnutým algoritmom pre výpočet súradníc bolo vystavené na príslušných webových stránkach.

LITERATÚRA

[1] Burget, R.; Komosny, D.; Muller, J.; aj.: Tree transmission Feedback: Scalable Signaling for IPTV Systems. Technická zpráva, Faculty of electrical Engineering and Communication, Brno Universitry of Technology, Brno Czech Republic, 2008.

[2] Burget, R.; Komosný, D.; Müller, J.: Best Effort Hierarchical Aggregation Tree for IPTV Signaling. In *International Journal of Computer Science and Network Security*, 2008, ISSN 1738-7906, s. 1–5.

[3] Burget, R.; Komosný, D.; Novotný, V.: Integration of Host Position Prediction into Hierarchical Aggregation. In *In Seventh International Conference on Networking*, 2008, ISBN 978-0-7695-3106-9, s. 740–745.

[4] Chen, Y.; Xiong, Y.; Shi, X.; aj.: Pharos: A Decentralized and Hierarchical Network Coordinate System for Internet Distance Pediction. In *In Proceeding of the 50th Annual IEEE Global Telecommunications Conference*, Washington, D.C., USA, November 2007, ISBN 978-1-4244-1043-9, s. 421–426, doi:10.1109/GLOCOM.2007.85.
URL `http://www.net-glyph.org/~chenyang/Globecom07_Pharos.pdf`

[5] Chen, Y.; Zhao, G.; Li, A.; aj.: Myth: An Accurate and Scalable Network Coordinate System under High Node Churn Rate. In *In Proceeding of the 15th IEEE International Conference on Networks*, Adelaide, Australia, November 2007, ISBN 978-1-4244-1230-3, ISSN 1556-6463, s. 143–148, doi:10.1109/ICON.2007.4444076.
URL `http://www.net-glyph.org/~chenyang/ICON07_Myth.pdf`

[6] Chu, Y.-H.; Ganjam, A.; Ng, T. S. E.; aj.: Early deployment experience with an overlay based Internet broadcasting system. In *in Proc. USENIX Annual Technical Conference*, 2004.

[7] Dabek, F.; Cox, R.; Kaashoek, F.; aj.: Vivaldi: a decentralized network coordinate system. In *SIGCOMM Comput. Commun. Rev.*, ročník 34, New York, NY, USA: ACM, 2004, ISSN 0146-4833, s. 15–26, doi:http://doi.acm.org/10.1145/1030194.1015471.

[8] Dabek, F.; Li, J.; Sit, E.; aj.: Designing a DHT for low latency and high throughput. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, Berkeley, CA, USA: USENIX Association, 2004, s. 7–7.

[9] Hamerly, G.; Elkan, C.: Learning the k in k-means. In *In Neural Information Processing Systems*, MIT Press, 2003, str. 2003.

[10] Jiawei, H.; Micheline, K.: *Data Mining: Concepts and Techniques*. Academic Press, 2001, ISBN 1-55860-489-8.

[11] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; aj.: An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 24, č. 7, July 2002: s. 881–892, ISSN 0162-8828, doi: http://dx.doi.org/10.1109/TPAMI.2002.1017616.
URL `http://dx.doi.org/10.1109/TPAMI.2002.1017616`

[12] Komosný, D.; Burget, R.; Novotný, V.: Tree Transmission Protocol for Feedback Distribution in IPTV Systems. In *In Proceedings of the Seventh IASTED International Conference on Communication Systems and Networks*, Palma de Mallorca, Spain, 2008, ISBN 978-0-88986-758-1, s. 1–7.

[13] Komosný, D.; Novotný, V.: Analysis of bandwidth redistribution algorithm for single source multicast. In *Sixth International Network Conference*, United Kingdom, 2006, ISBN 1-84102-157-1, s. 45–52.

[14] Komosný, D.; Novotný, V.: Tree Structure for Source-Specific Multicast with feedback Aggregation. In *in ICN07 - The Sixth International Conference on Networking*, Martinique, 2007, ISBN 0-7695-2805-8, s. 85–85.

[15] Michálek, M.: Zhlukovacie algoritmy, 2008, Ústav informatiky a softvérového inžinierstva, FIIT, STU, Bratislava.
URL `http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0809/michalek/index.html`

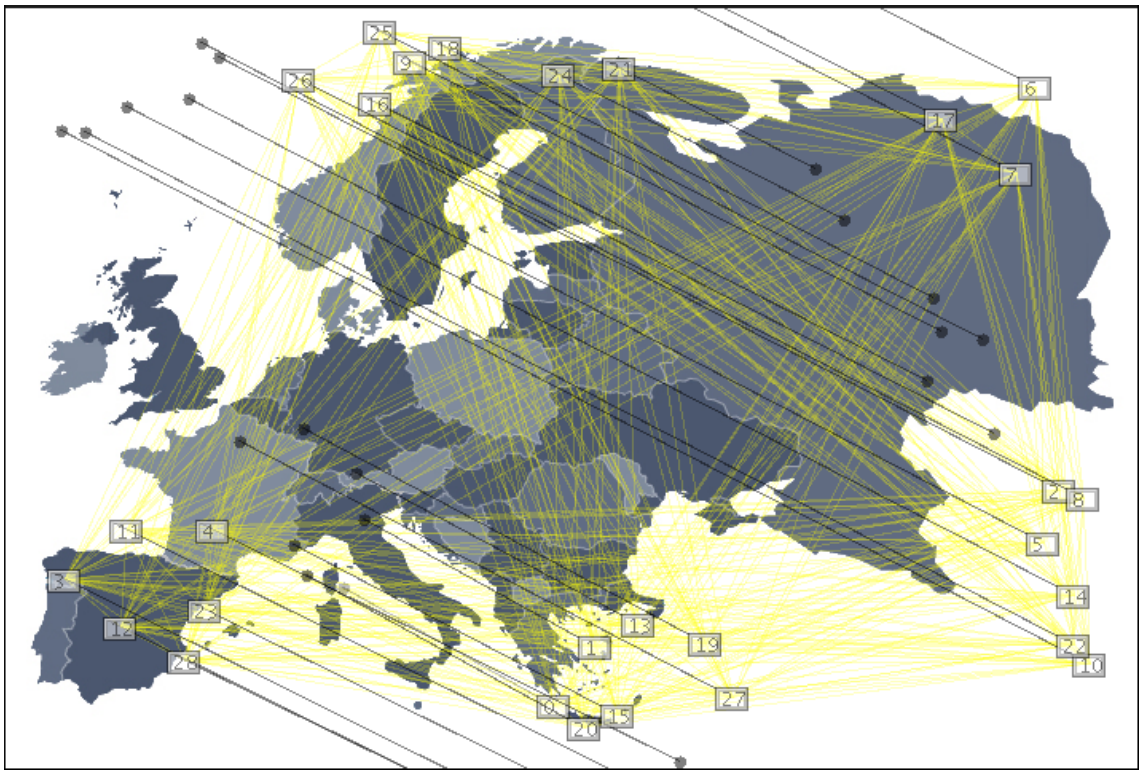
[16] Müller, J.; Komosný, D.; Burget, R.; aj.: Advantage of Hierarchical Aggregation. In *International Journal of Computer Science and Network Security*, 2008, ISSN 1738-7906, s. 1–7.

[17] Ng, T.; Zhang, H.: Predicting Internet network distance with coordinates-based approaches. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, ročník 1, New York, NY, USA, June 2002, ISBN 0-7803-7476-2, ISSN 0743-166X, s. 170–179, doi:10.1109/INFCOM.2002.1019258.
URL `http://www.cs.rice.edu/~eugeneng/papers/INFOCOM02.pdf`

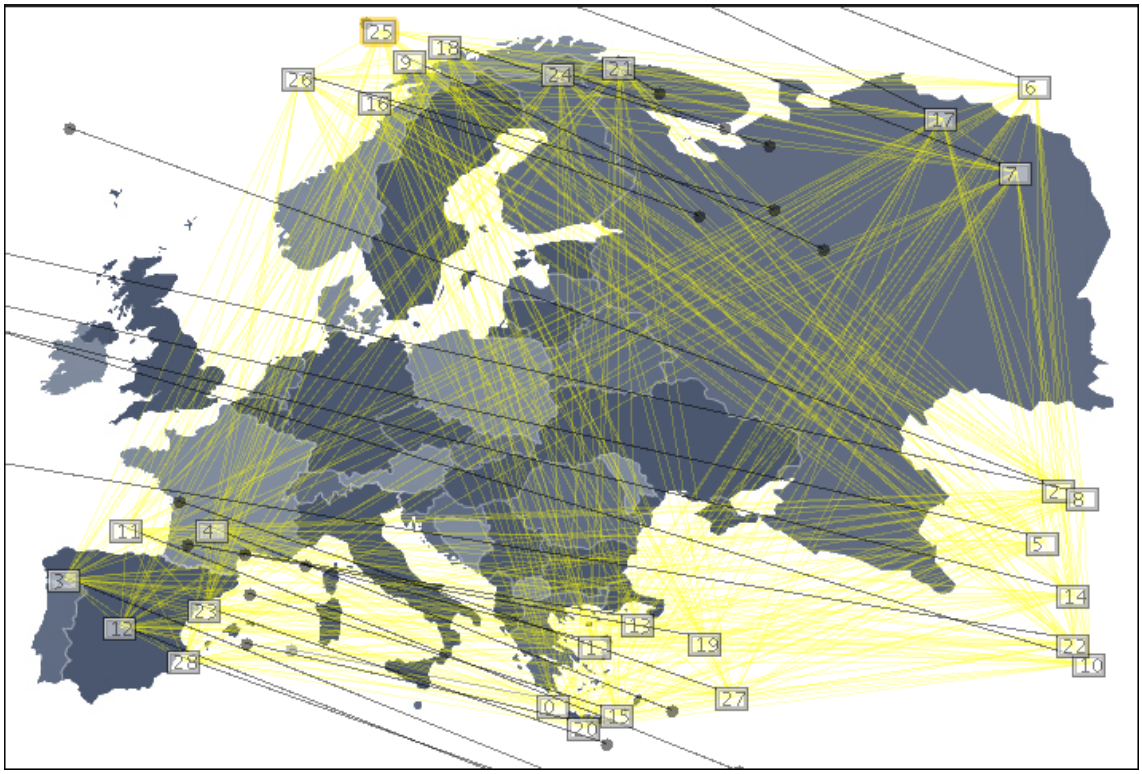
[18] Novotný, V.; Komosný, D.: Optimization of Large-Scale RTCP Feedback Reporting. In *ICWMC 2007 - The Third International Conference on Wireless and Mobile Communications*, Guadeloupe, 2007, ISBN 0-7695-2796-5.

[19] Wagstaff, K.; Cardie, C.; Rogers, S.; aj.: Constrained K-means Clustering with Background Knowledge. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, ISBN 1-55860-778-1, s. 577–584.

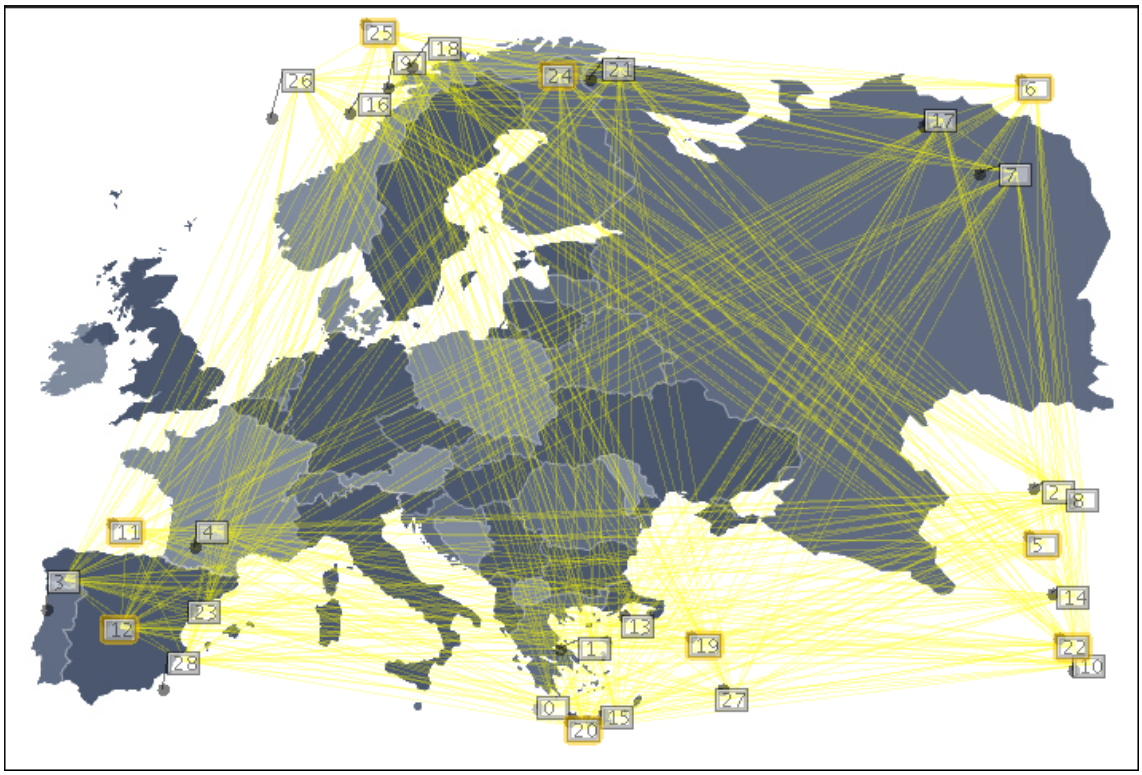
A SIMULÁCIE S VÄČŠÍM POČTOM PRIPOJENÝCH UZLOV



Obrázok A.1: Ustálenie uzlov po spustení simulácie



Obrázok A.2: Nastavenie uzlu č.25 do fixného režimu



Obrázok A.3: Nastavenie uzlov č.6, 11, 12, 19, 20, 22, 24 do fixného režimu