VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# IPV6 GEOLOKACE A VIZUALIZACE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. TOMÁŠ SUCHOMEL
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# IPV6 GEOLOKACE A VIZUALIZACE

IPV6 GEOLOCATION AND VISUALIZATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ SUCHOMEL
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MATOUŠEK, Ph.D.
SUPERVISOR

BRNO 2011

# Abstrakt

Tato diplomová práce se zabývá problematikou geolokace IPv6 adres a vizualizace jejich lokalit. Má za úkol prozkoumat oblast IPv6, dále porovnává současné přístupy ke geolokaci. Jejím hlavním cílem je vybudovat systém, který dokáže vytvořit geolokační databázi s pravidelnou aktualizací a který bude schopen zobrazit geografickou lokalitu IPv6 adresy na světové mapě. Geolokační systém nejprve získá geografické adresy pro jednotlivé prefixy IPv6 z veřejně přístupného systému Whois. Tyto adresy transformuje pomocí API Google Maps na kartézské souřadnice, všechny získané údaje poté uloží do databáze. Lokality IPv6 adres jsou zobrazeny pomocí webového rozhraní Google Maps na mapu. Dalším možným výstupem je vizualizace v podobě statistiky zobrazující hustotu rozložení přidělených adresních bloků IPv6 v jednotlivých krajinách.



# Abstract

This master's thesis is focused on geolocation and visualization of IPv6 addresses. Several current approaches to geolocation have been discussed. Its main goal is to design geolocation database with regular updates with ability to visualize geographic location of the IPv6 address on the world map. Implemented system is gathering data about IPv6 prefixes from public Whois database. Gathered geographical addresses are mapped into coordinates by Google Maps Geocoding API. All gathered data are being saved into database together. Locations of IPv6 addresses are visualized via web-based Google Maps API. Moreover, the density mapping of IPv6 address blocks in particular countries can be visualized too.



# Klíčová slova

geolokace, IPv6, Whois, Google Maps

# Keywords

geolocation, IPv6, Whois, geographical mapping, Google Maps

# Citace

Tomáš Suchomel: IPv6 geolokace a vizualizace, diplomová práce, Brno, FIT VUT v Brně, 2011

# IPv6 geolokace a vizualizace

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Tomáš Suchomel
17. května 2011

## Poděkování

Rád bych poděkoval svému vedoucímu Ing. Petru Matouškovi, Ph.D. za jeho vstřícný přístup, ochotu a věcné připomínky. Dále chci vyjádřit poděkování Ondřeji Surému ze sdružení CZ.NIC za užitečné rady a konzultace v průběhu vývoje aplikace. Ing. Matěji Grégrovi a Ing. Pavlu Spáčilovi za poskytnutí hardwarového vybavení za účelem ladění a testování systému. Ing. Petru Novotnému a ostatním spolupracovníkům za toleranci. V neposlední řadě děkuji své rodině, přítelkyni a přátelům za podporu.

© Tomáš Suchomel, 2011.
*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

**Úvod** **3**

**1 IPv6 protokol** **5**
1.1 Historie a důvod vzniku . . . . . 5
1.2 Internet v přechodu na IPv6, problémy . . . . . 6
1.3 IPv6 adresování . . . . . 7
1.4 Shrnutí . . . . . 8

**2 Geolokace** **9**
2.1 Vymezení pojmu geolokace . . . . . 9
2.2 Motivace . . . . . 9
2.3 Možnosti geolokace IP adres . . . . . 10
2.4 Shrnutí . . . . . 11

**3 Rešerše současných přístupů ke geolokaci** **12**
3.1 Technika DNS LOC . . . . . 12
3.2 Dotazování na databázi Whois . . . . . 13
3.3 Geolokace na základě měření nebo vlastností sítě . . . . . 15
3.4 Systém IP2Geo . . . . . 15
3.5 Maximální odhad pravděpodobnosti . . . . . 17
3.6 Geolokace založená na omezení . . . . . 17
3.7 Shrnutí . . . . . 18

**4 Návrh geolokačního systému** **20**
4.1 Získání dat pro IPv6 geolokaci . . . . . 20
4.2 Převod dat do vhodné formy . . . . . 22
4.3 Určení geografické polohy podle adresy IPv6 . . . . . 22
4.4 Převod adresy na geografické údaje . . . . . 23
4.5 Vytvoření geolokační databáze . . . . . 23
4.6 Vizualizace dat . . . . . 24
4.7 Shrnutí . . . . . 24

**5 Implementace systému** **26**
5.1 Získání geolokačních dat . . . . . 26
5.2 Geolokační databáze . . . . . 33
5.3 Vizualizace dat na mapě světa . . . . . 38
5.4 Logování přístupu a dotazů . . . . . 42
5.5 Shrnutí . . . . . 43

**6 Testování systému** **44**
6.1 Testování konzolové aplikace . . . . . 44
6.2 Testování webové aplikace . . . . . 47
6.3 Rychlost a efektivita uložení dat . . . . . 52
6.4 Přesnost geolokačního systému . . . . . 55
6.5 Bezpečnost systému . . . . . 56
6.6 Statistika dotazů . . . . . 57
6.7 Použití systému . . . . . 59
6.8 Shrnutí . . . . . 59

**7 Závěr** **60**

**A Obsah CD** **65**

**B Konfigurace systému** **66**

**C Návod k použití** **68**

# Úvod

## Motivace

V současné době je pojem geolokace na internetu stále populárnější. Proč tomu tak je? Uvědomme si, že například geolokace uživatele na webu může výrazně pomoci zaměřit se na jeho specifické cíle a požadavky. Vyhledávače, informační servery i firmy nabízející služby mohou pružně reagovat na umístění uživatele a nabízet mu tak především to, co se nachází v jeho blízkosti nebo má souvislost s tím, kde se momentálně nachází (informace z regionu, nejbližší pobočky firem, nejbližší hotely, obchody, aj.). Další z důležitých vlastností geolokace je fakt, že na základě umístění uživatele je možné mu zakázat zobrazení určitého obsahu podle lokálních nařízení a regulí, případně v závislosti na vládní politice. Navíc je možné mnohem účinněji detekovat zneužití kreditních karet při internetových platbách, zamezit podvodným platbám, krádežím identity, phishingu a spamu. Geolokace má tedy i svůj bezpečnostní a preventivní význam. K tomu se však váže také negativní stránka, a to uchovat soukromí uživatelů internetu, který má být z principu anonymním komunikačním médiem. Stále je totiž do jisté míry kontroverzní, zda-li není využívání geolokace neoprávněným zásahem do soukromí uživatele, případně na kolik je prozrazení jeho geografické polohy nedovoleným získáním osobních údajů.

Dalším, ještě důležitějším fenoménem dnešní doby v oblasti počítačových sítí a komunikací, je protokol IPv6. Jedná se o nástupnický protokol stávajícího IPv4, který se snaží vyřešit nejvýznamější problémy, které rychlý rozvoj internetu způsobil, a poskytnout perspektivní platformu pro jeho další rozvoj. Vyčerpání adresového prostoru IPv4 bylo avizováno již několikrát, prognóza se vyplnila v únoru 2011, kdy došlo k přidělení posledního volného bloku adres registrátorovi pro Asii a Pacifik.

## Cíle práce

Cílem této diplomové práce je navrhnout a vyvinout systém pro geolokaci IPv6 adres, který bude schopen automaticky sestavit geolokační databázi s pravidelnou aktualizací a poté bude umět zobrazit na mapě světa lokalitu výskytu IPv6 adresy. V praxi použitelná se jeví také statistika hustoty pokrytí jednotlivých krajin IPv6 adresami.

## Struktura práce

Kapitola 1 se zabývá průzkumem oblasti protokolu IPv6, možnostmi současné existence obou protokolů společně a popisuje způsob adresování v IPv6. V kapitole 2 se seznámíme s pojmem geolokace a budeme diskutovat její využití na internetu, dále si přiblížíme možnosti geolokace a provedeme jejich klasifikaci. Kapitola 3 obsahuje vypracovanou rešerši

současných přístupů ke geolokaci a podrobněji popisuje významnější z nich, přičemž jsou diskutovány výhody i omezení jednotlivých přístupů. Vlastní návrh geolokačního systému, který detailně popisuje především způsob získání dat a tvorbu geolokační databáze, je konzultován v kapitole 4. Cílem kapitoly 5 je popsat implementaci navrženého geolokačního systému, přičemž se detailně zaměříme na získání dat, vytvoření geolokační databáze a vizualizace. V kapitole 6 testujeme implementaci a zaměříme se zde také na vyhodnocení výsledků, analýzu použitelnosti a konzultujeme zde také nasazení systému do pilotního provozu. Na závěr diskutujeme dosažené výsledky, hodnotíme přínos práce a zvažujeme její možné rozšíření do budoucna.

# Kapitola 1

# IPv6 protokol

Kapitola si klade za cíl přiblížit a prozkoumat možnosti IPv6, a to zejména co se týče formátu adres. Nejprve stručně popíše protokol IPv6, seznámí nás s jeho historií i vlastnostmi a detailně prozkoumá zejména oblast IPv6 adresování. Znalost IPv6 adresace bude jednou ze základních dovedností potřebných k analýze a návrhu celého geolokačního systému.

## 1.1 Historie a důvod vzniku

Protokol IPv6 začal vznikat v 90. letech minulého století, a to především jako reakce na rychle se vyčerpávající adresový prostor IPv4. Kromě mnohem většího adresního prostoru nabízí oproti stávajícímu IPv4 i nové vlastnosti reagující na vývoj internetu a počítačových sítí v současné době. Je však třeba zdůraznit, že reálný provoz sítí na IPv6 se v současnosti pohybuje v řádu desetin procent. Zatím se tedy nový protokol příliš neujal a zájem o jeho nasazování zaostal za původními předpoklady. To je dáno zejména tím, že hlavním motorem v minulosti pro rozvoj IPv6 byly rychle se vyčerpávající IP adresy. V průběhu času se však hledala řešení na bázi IPv4, ze kterých budeme jmenovat například beztřídní adresování (CIDR) a mechanismy typu NAT pro nahrazení celé lokální sítě jedinou veřejnou IP adresou, což vedlo k razantnímu zpomalení tempa spotřeby adres. To ovšem začalo před pár lety znovu prudce narůstat, a tak začíná opět sílit zájem o IPv6. Prognóza rychlého vyčerpání adres se vyplnila, od února 2011 již sdružení IANA ohlásilo, že neexistují žádné volně přidělitelné IPv4 adresní bloky. [23]. Každý registrátor má ještě IPv4 adresy v zásobě, ale jejich počet velice rychle ubývá. Představíme zde hlavní cíle IPv6:

- dostatečně bohatý adresní prostor
- podpora služeb se zaručenou kvalitou
- bezpečnostní mechanismy integrované přímo v protokolu IPv6
- design odpovídající vysokorychlostním sítím
- automatická konfigurace zařízení a podpora mobilních zařízení
- vzájemná kooperace a současný běh s IPv4 a hladký přechod ze stávajícího protokolu na nový

Největším problémem IPv6 je fakt, že neexistuje zpětná kompatibilita se svým předchůdcem.

## 1.2 Internet v přechodu na IPv6, problémy

Kvůli rychle se ztenčujícím zásobám IPv4 adres bude nutně třeba v blízké budoucnosti přejít v celé síti internetu na protokol IPv6. Ta je však příliš rozsáhlá na to, aby se přechod odehrál skokově. Proto je v současnosti upřednostňovaná varianta postupného přechodu a koexistence obou protokolů a jejich vzájemné spolupráce. Jednou z možností je propojit IPv6 sítě klasickou infrastrukturou, další jsou řešení na bázi komunikace jednoho uzlu IPv4 s protějškem IPv6 přímo. Tři typy metod jsou dvojí zásobník, tunelování a translátory.

### 1.2.1 Tunelování

Tunely slouží ke komunikaci partnerů, kteří používají stejný protokol, avšak ten není podporován sítí ležící mezi nimi. Dnes to obvykle znamená, že je třeba přenášet IPv6 pakety skrze IPv4 internet. Existují dva typy tunelů.

**Explicitně konfigurovaný tunel** vytvoří správce příslušného zařízení. Podporu zaštiťují různé tunelové servery, kde je zájemcům o IPv6 nabídnuto vytvoření tunelu. Po registraci a uvedení základních informací o počítači a připojení obdrží zájemce konfigurační skript, jehož spuštěním vznikne tunel připojující počítač k IPv6 síti [28].

**Automatický tunel** je vytvářen samočinným mechanismem bez zásahu uživatele. Jedním ze zástupců tohoto typu tunelování je *6to4* [15], který na základě jediné IPv4 adresy (přiřazené přístupovému směrovači) vytvoří prefix IPv6 pro adresování celé koncové sítě a zajistí automatické tunelování datagramů mezi ní a zbytkem světa. Z dalších představitelů automatických tunelovacích mechanismů můžeme jmenovat například *6over4* [14], *ISATAP* [30] nebo *Teredo* [22].

### 1.2.2 Translátory

IPv6 překladové mechanismy jsou důležité ke spojení čistě IPv4 klienta se strojem, který podporuje výlučně adresování nové generace IPv6. Úkolem translátorů je překládat data z obou světů, a poskytnout tak vzájemnou domluvu.

*SIIT* [26] je mechanismus, který poskytuje vzájemnou převoditelnost mezi hlavičkami IPv6 a IPv4 paketu. Tento translátor může být použit k řešení situace, kdy IPv6 host bez přiřazené IPv4 adresy chce komunikovat s hostem podporujícím pouze IPv4 adresování. SIIT však definuje jen vlastní překlad a nemá doplňkové informace, které doplňují další mechanismy ze SIIT vycházející. SIIT mimo jiné definuje třídu IPv6 adres zvaných IPv4-přeložené adresy. Ty potom mají prefix *::ffff:0:0:0/96* a mohou být zapsány ve formátu *::ffff:0:a.b.c.d*, ve kterém adresa *a.b.c.d* formátu IPv4 odkazuje na uzel, kde je překladem povoleno IPv6. Výše uvedený prefix byl vybrán za účelem poskytnout nulový kontrolní součet, aby se zabránilo změnám v kontrolním součtu hlavičky transportního protokolu [18].

Velké očekávání vzbuzoval *NAT-PT* [10], což je mechanismus umístěný mezi koncovou IPv6 síť a zbytek IPv4 internetu. Disponuje sadou IPv4 adres, na které mapuje IPv6 adresy z koncové sítě. Navíc provádí konverzi i pro dotazy DNS a odpovědi. Přináší však řadu problémů, byl zavržen a nyní se pracuje na jeho nástupci *NAT64* [12].

Z hlediska praktické použitelnosti nesmíme opomenout zmínit také BIS (pracující v síťové vrstvě OS) a BIA (v aplikačním rozhraní). Ty umožňují aplikacím napsaným výhradně pro IPv4 komunikovat s IPv6 světem, jedná se vlastně o privátní NAT uvnitř počítače. IPv4 komunikace potom probíhá pouze mezi BIS vrstvou a samotnou aplikací.

### 1.2.3 Nedostatek motivace

Přechod na IPv6 je neustále pozdržován nedostatkem motivace, jelikož v současné době je stoprocentní dostupnost všech služeb pro IPv4. Vzhledem k nekompatibilitě obou protokolů tak pravděpodobně nastanou problémy jak pro poskytovatele, tak pro uživatele. Bohužel je novým protokolem poskytováno jen málo služeb, tudíž o něj není zájem. Očekává se však, že s vyčerpáním IP adres se situace změní, s volnými IP adresami se začně obchodovat a poskytovatelé (a tím pádem i uživatelé) budou nuceni přejít na nový protokol.

## 1.3 IPv6 adresování

Adresový prostor u IPv6 je obrovský, délka adresy vzrostla oproti IPv4 na čtyřnásobek, tedy na 128 bitů. Počet všech dostupných IPv6 adres se tedy pohybuje v řádu $10^{38}$, což poskytuje dostatečný počet adres na opravdu hodně dlouhou dobu.

### 1.3.1 Zápis adresy

Vzhledem k délce adresy by nebylo příliš praktické zapisovat adresy ve formátu IPv4, proto se užívá kompaktní zápis v šestnáctkové soustavě. V něm jednotlivé dvojice bytů oddělujeme kvůli přehlednosti dvojtečkou. IPv6 adresa potom může vypadat následovně: *2001:718:802:809::93e5:929*. Počáteční nuly v jednotlivých čtveřicích hexadecimálních čísel se mohou vypustit a také jedna sekvence nul může být nahrazena *::*, avšak dvě dvojtečky mohou být kvůli jednoznačnosti použity pouze jednou. V praxi to znamená, že zápis čtverice *0718* a *718* je ekvivalentní, stejně jako zápis *2001:0000:0000:0000:0000:0000:0000:0512* je ekvivalentní s *2001::512*.

IPv6 adresa je typicky rozdělena na dvě části, prvních 64 bitů tvoří adresa (pod)sítě, zbylých 64 bitů pak adresa počítače, která se vygeneruje buď automaticky z MAC adresy, nebo se přiřazuje sekvenčně.

Prefixy se zapisují obdobně jako je tomu u IPv4 adres ve formátu *adresa/délka*, kde *adresa* určuje začátek adresy (její nepodstatné bity se vynulují) a *délka* říká, kolik bitů je podstatných. Například prefixu *ff00::/8* vyhoví každá adresa, která má na pozici prvních 8 bitů samé jedničky, jinak řečeno adresy začínající *ff* a zbytek může být libovolný.

### 1.3.2 Typy a dělení IPv6 adres

IPv6 adresy se dělí do třech základních kategorií:

- Unicast - označuje jedno rozhraní připojeného zařízení v síti.
- Multicast - představuje adresu skupiny síťových rozhraní, paket bude doručen všem členům skupiny.
- Anycast - adresuje několik rozhraní, ale paket bude doručen jen jednomu z nich, zpravidla nejbližšímu. Anycast adresy umožní realizovat speciální služby, kdy např. klient odešle paket s obecnou adresou, který zpracuje první dostupný server.

Oproti IPv4 zmizely adresy typu broadcast, v IPv6 jsou nahrazeny multicastem pro všechna dostupná zařízení.

## 1.4 Shrnutí

V kapitole je popsána historie a základní vlastnosti protokolu IPv6, dále možnosti přechodu na nový protokol v síti internet souběžně s existujícím protokolem IPv4. Kapitola také prozkoumává oblast adresování IPv6 jako je formát adresy a její zkrácený zápis. Tyto znalosti budou užitečné pro návrh a vývoj geolokačního systému, který pracuje s IPv6 adresami.

# Kapitola 2

# Geolokace

V této kapitole si vymezíme pojem geolokace a budeme diskutovat její možné použití a uplatnění na současném internetu z několika hledisek včetně bezpečnostního. Kapitola dále prozkoumává možnosti geolokace IP adres a stručně popisuje několik více či méně odlišných přístupů k řešení problému, který spočívá v určení co možná nejpřesnější geografické lokality uzlu na internetu. Současně kapitola pojednává i o právních a etických otázkách geolokace. V jejím závěru je provedeno rozdělení geolokačních technik do kategorií, které podrobně prozkoumá až kapitola 3.

## 2.1 Vymezení pojmu geolokace

Internet propojuje počítače z celého světa. Občas je žádoucí vědět, na kterém geograficky vyjádřitelném místě se konkrétní počítač nachází. Pod pojmem *Geolokace na internetu* si tedy představíme problém rozpoznání geografické polohy uživatele na internetu, samozřejmě do určité úrovně přesnosti. Tato technika se také často označuje jako *IP geolokace*, jelikož každý počítač přímo připojený k internetu je identifikován unikátní IP adresou. Rostoucí počet společností spravuje a licencuje databáze, např. (MaxMind: GeoIP [5], Akamai: Edgescape [1], IP2Location: IP-Country-Region-City-Latitude-Longitude Database [3], IPInfoDB: IP address geolocation database [4]), které mapují IP adresy na zeměpisnou lokalitu.

## 2.2 Motivace

Geolokace IP adres na internetu najde uplatnění všude tam, kde je kladen důraz na cílového konzumenta obsahu. Možnost přesného určení geografické polohy cílové stanice v internetu má mnoho praktických využití. Podle aktuální geografické polohy uživatele je možnost mu nabídnout například regionální zpravodajství a aktuální informace o počasí v lokalitě, kde se právě nachází. Mezi nejvíce lukrativní služby patří cílené zobrazování reklamy a dalšího na lokalitě závislého obsahu přes internet. Například, pokud uživatel v Brně otevře webovou stránku, která je schopna rozpoznat jeho geografické umístění, je schopna mu zobrazit obsah relevantní pro brněnské zákazníky. Tento princip je obecně označován jako *geo-targeting*. Kromě toho je možné přizpůsobit webové stránky tak, že se automaticky v závislosti na rozpoznaném regionu nastaví jazyk zobrazovaného obsahu, dále detekovat podvodné pokusy o platby kartou při internetových transakcích a jiná bezpečnostní rizika.

V neposlední řadě geolokace ze své podstaty umožňuje optimálnější rozložení zátěže mezi internetové servery a efektivnější přidělování zdrojů. V tomto případě je IP geolokace

využita pro automatické přesměrování na geograficky nejbližší servery. Příkladem může být společnost Google, která hledaný obsah nabízí právě podle lokalizace IP adresy klienta.

Co se týče bezpečnostních aplikací geolokace, můžeme je rozdělit na dvojí. První je skutečně bezpečnostní význam, kdy dochází ke snížení ztrát kvůli podvodům při transakcích provedených kreditní kartou (výška a počet podvodných plateb se významně liší pro transakce pocházející z určitých částí světa). Dále je to filtrování spamu (vycházíme ze znalostí států zapojených do přeposílání zpráv), zabezpečování vzdálených přihlašování a zákaz šíření digitálního obsahu podléhající regionální jurisdikci. Geolokace údajně také snižuje počet krádeží identity na internetu. Druhý význam bezpečnosti z opačného pohledu spočívá v případech, kdy uživatelé mají motiv se geolokaci vyhnout. To mohou být například důvody osobní nebo humanitární, případně důvody spojené se skrýváním ilegálních aktivit [25].

První a základní otázka zní, zda-li je geolokace vůbec možná a pokud ano, do jaké míry přesnosti lze provést. Ačkoli společnosti nabízející geolokační služby proklamují v jejich marketingu, že je přesnost velká, otázkou zůstává, nakolik je možno provést identifikaci IP adresy přidělené organizaci, anebo přímo koncového uživatele. Také samozřejmě existuje možnost záměrně geolokační techniky obelstít, jako je třeba použití proxy serverů a anonymizačních nástrojů. Druhá otázka je čistě etická - je vhodné používat geolokační techniky, které by mohly být potenciálně zneužity? Odpověď na tuto otázku je vhodnější přenechat právním zástupcům, vzhledem k tomu, že spadá spíše do oblasti občanského práva. Cílem této diplomové práce je zaměřit se především na technickou stránku geolokace.

## 2.3 Možnosti geolokace IP adres

Existuje hned několik různých přístupů, jak přistupovat k problému geolokace IP adres na současném internetu. Jedním z takových přístupů je vytvoření a udržování rozsáhlé distribuované geolokační databáze, která obsahuje mapování IP adres na geografickou lokalitu, konkrétně na zeměpisnou šířku (*Latitude, zkráceně Lat*) a délku (*Longitude, zkráceně Lon*). Mnoho internetových společností, které nabízejí geolokaci, se opírá právě o takové, více či méně přesné, databáze, za účelem rozpoznání lokace koncového uživatele a následně poskytnutí výše zmiňovaných služeb.

Bohužel, tyto databáze jsou často proprietární a manuálně aktualizované, takže jejich konzistence a přesnost je přinejmenším pochybná. Tuto teorii navíc umocňuje i fakt, že při našem průzkumu v oblasti geolokace, jsme na žádost o popis přístupu k vytváření geolokační databáze obdrželi často zamítavou odpověď s dodatkem, že proces budování datábaze a použité přístupy jsou komerční, a tudíž nepublikovatelné. Z takto kontaktovaných společností lze jmenovat například MaxMind [6], která mimo jiné nabízí geolokační služby *GeoIP* a *minFraud*, které si podrobněji popíšeme v následující kapitole 3.

Navíc, s nástupem a schválením protokolu IPv6, bude mnohem složitější tyto databáze udržovat aktuální a konzistentní. Rozsáhlá geolokační databáze se také nedokáže tak snadně adaptovat na častou změnu lokality především u mobilních zařízení, která se stávají fenoménem doby. Dnes má téměř každé takové zařízení přístup k internetu a je jen otázkou času, kdy každé bude mít také vlastní IPv6 adresu. V takových případech už bude v podstatě nemožné a značně neefektivní udržovat databázi aktuální.

Existuje několik různých přístupů, které řeší problém geolokace. Ty rozdělíme ná následující:

- *DNS LOC* [17]- přístup založený na přidání informací o lokalitě (souřadnic) do glo-

bálně používaného systému DNS.

- *Traceroute* - přístup, pomocí něhož získáme jména směrovačů na cestě paketu internetem. Tyto posléze můžeme mapovat na geografické umístění.
- Dotazování na databázi Whois jednotlivých registrátorů k získání geografické adresy zákazníka, ke které je přiřazena IP adresa. Z takto získaných údajů je potom potřeba vytvořit databázi mapující rozsahy IP adres a jejich korespondující geografické umístění v podobě zeměpisných souřadnic. Naše práce spočívá ve vytvoření takové databáze, jejíž efektivita bude uspokojivá. Požadavky definované na geolokační systém budou popsány v kapitole 4.
  - NetGeo [24]
  - IP2LL
- Techniky založené na měření a výpočtech
  - IP2Location [2]
  - MaxMind [6]
  - IP2Geo
  - Technika založená na maximálním odhadu pravděpodobnosti
  - Technika založená na učení
  - Technika založená na omezení
  - Statistická geolokace

Protože je tato problematika rozsáhlá, rozhodli jsme se jí věnovat celou kapitolu 3, v níž je provedena rešerše v současnosti používaných přístupů ke geolokaci, kde je každý uveden v širším kontextu a poté rozebrán a zhodnocen. Zvýšená pozornost je věnována zejména technikám založeným na dotazování se *Whois* a technikám, které využívají empirické měření latencí a derivátům těchto přístupů.

## 2.4 Shrnutí

Tato kapitola přibližuje geolokaci a vysvětluje, proč je ve světě dnešního internetu tak důležitá. Prozkoumává možnosti geolokace IP adres, přičemž se zaměřuje i na adresy IPv6. Popisuje, proč není ideální používat komerční a proprietární databáze a nachází východisko v tvorbě vlastní geolokační databáze. V závěru kapitoly jsou shrnuty a rozčleněny různé techniky pro geolokaci.

# Kapitola 3

# Rešerše současných přístupů ke geolokaci

V rámci analýzy současného stavu jsme prozkoumali oblast v současnosti používaných mechanismů a technik, které se snaží o geografickou aproximaci uzlu v internetu. U každého z prozkoumaných přístupů je nejprve popsán princip činnosti a poté proveden detailní rozbor mechanismu, který zajišťuje geolokaci. Na konci této kapitoly je provedeno závěrečné zhodnocení a porovnání přístupů z několika různých hledisek.

V následujících podkapitolách rozebereme techniky, které se v současné době pro geolokaci používají.

## 3.1 Technika DNS LOC

Jedním z přístupů konzultovaných v [17] je začlenění geografických údajů přímo do struktury záznamu DNS. Toto experimentální rozšíření se nazývá DNS LOC. Formát nového pole Resource Record (RR) pro DNS je definován přímo v [17], rezervuje si odpovídající typ záznamu DNS (LOC) a numerický kód (29). Princip řešení se jeví na první pohled ideální, přesto jeho implementace není v současné době příliš široce rozšířená, nakolik vyžaduje modifikaci stuktury záznamů DNS. Další překážkou v aktivním využívání tohoto přístupu je, že tento údaj LOC musí zadávat administrátoři a navíc neexistuje snadná cesta k ověření přesnosti zadaného lokalitního údaje.

Ke zjištění polohy uzlu potom stačí jednoduše vyhledat přes dotaz nad databází DNS. Dotaz bude vypadat takto:
`dig cybersales.cz ANY`, kde se mimo jiné vrátí záznam LOC. Ten pak obsahuje informace o zeměpisné šířce, délce, nadmořské výšce, velikost bodu, vertikální a horizontální přesnost [17]. Dozvíme se, že dané doménové jméno se dle zadaných údajů geograficky nachází na `50.000N, 14.000E, 365 m.n.m.` a má 1m přesnost ve všech rádiusech). Pro techniku DNS LOC platí následující omezení:

1. Úplnost - Jen velice malý počet uzlů obsahuje záznamy LOC v DNS, je odhadováno, že tento počet je nižší než 1% [25].

2. Nepravost - Informace, které jsou obsaženy v záznamu DNS LOC, jsou neověřené. Jsou zadávány uživatelem, který se může rozhodnout záměrně poskytnout nepravá data.

## 3.2 Dotazování na databázi Whois

### 3.2.1 Dotaz Whois na IP adresu

Informace o konkrétní IP adrese lze nejjednodušeji získat pomocí dotazů na veřejnou databázi Whois [21] registrátorů k získání adresy zákazníka (v geografickém smyslu), jemuž byl přiřazen určitý rozsah IP adres nebo konkrétní IP adresa. Tato databáze mapuje logické identifikátory v internetu (IP adresy, doménová jména, čísla autonomních systémů) na objekty reálného světa. To umožňuje rozpoznat entitu, k níž je daná IP adresa registrována, jako např. poskytovatele připojení k internetu, organizace, koncové uživatele, aj. Kontaktní informace obvykle zahrnují fyzickou (poštovní) adresu, e-mailovou adresu a telefonní kontakt. Z telefonního čísla a poštovní adresy může být zjištěna geografická poloha [25].

Adresový prostor kontroluje společnost ICANN, která řídí společnost IANA. Přehled alokovaných bloků pro IPv6 adresy se dají získat v [8]. ICANN/IANA nyní přiděluje adresové bloky pěti regionálním internetovým registrům (RIR). Ti potom alokují jednotlivé prefixy přímo entitám v jejich regionu. Servery *Whois* jednotlivých registrátorů jsou tyto:

- whois.ripe.net
- whois.arin.net
- whois.apnic.net
- whois.lacnic.net
- whois.afrinic.net

Existuje však několik problémů a omezení pro techniky založené na dotazování databáze Whois:

1. Aktuálnost dat - První z nich je, že informace obsažené v takové databázi mohou být nepřesné či zastaralé. To zahrnuje i možnost jisté nekonzistence mezi servery, které obsahují záznamy korespondující s alokovaným nebo přiděleným blokem IP adres.

2. Geografická přesnost - Druhým nesnadno řešitelným problémem je, že obrovský (a geograficky rozptýlený) blok IP adres může být alokován jednou entitou či organizací a databáze Whois může obsahovat pouze jediný záznam pro celý tento blok. Například, blok IPv4 adres *4.0.0.0/8* je alokován pro Genuity, avšak dotaz na databázi ARIN Whois vrátí jako výsledek lokalitu *Cambridge, Massachusetts* pro jakoukoli IP adresu z tohoto velkého rozsahu. To však může znamenat nepřesnost stovky až tisíce kilometrů při určení cíle. Na druhou stranu, i když schopnost rozpoznat polohu koncového zařízení není nijak zvlášť přesná, pořád to může aproximovat lokalitu dobře v rámci regionu či státu, což je určitý úspěch. Registrátoři už se v současné době snaží o alokování menších rozsahů v rámci obrovského alokačního bloku pro obrovské korporace, nicméně obecně tento typ problému stále ve většině velkých databází přetrvává [27].

3. Falešná data - Data v databázi Whois poskytují registrující entity, ty však mohou poskytnout nekorektní nebo falešné informace.

Příklad takového dotazu: `whois -h whois.ripe.net -- 2001:1488::/32`

### 3.2.2 Dotaz Whois na autonomní systém

Čísla autonomních systémů (AS) jsou dalším logickým identifikátorem užitečným pro oblast geolokace, o které se stará ICANN. Jedná se o 16-bitová čísla používaná směrovacími protokoly jako BGP. Každý RIR má bloky čísel AS, které alokovala IANA. Každá veřejně směrovatelná IP adresa je vždy spojena s nějakým AS. Podle IP adresy můžeme tedy zjistit číslo AS, ve kterém je obsažena a následně získat detaily o konkrétním AS dotazem do veřejné databáze Whois. Jedním ze způsobů, jak zjistit podle IP adresy číslo AS, je podívat se do směrovací tabulky BGP, která obsahuje prefixy IP a sekvence čísel AS. Pro každý uvedený prefix existuje alespoň jedna sekvence čísel AS. Číslo AS popisuje cestu pro doručení paketu s cílovou IP adresou spadající do některého z prefixů. Existují veřejně viditelné tabulky BGP. Pro potřeby vyseparování konkrétního AS můžeme buďto použít příkaz `show ip bgp 83.240.54.45` při připojení přes `telnet` na server `route-views.routeviews.org`, nebo si celou databázi stáhnout a posléze příkazem `grep` konkrétní AS najít. Číslo AS se nám vrátilo 31246. Nejprve pomocí http://www.iana.org/assignments/as-numbers zjistíme, který RIR se stará o číslo AS. Posléze už informace o tomto konkrétním AS můžeme získat dotazem nad databází Whois příslušného RIR, tedy v našem případě RIPE NCC.
Příklad takového dotazu: `whois -h whois.ripe.net -- AS31246`
Omezení jsou v podstatě podobného ražení jako u dotazu na IP adresu. Ne všechny cílové uzly se nacházejí blízko adresy, pro kterou byl blok registrován a také některé velké AS mohou obsahovat širokou škálu prefixů, což může znamenat, že pokrývá rozsáhlou geografickou oblast. Co se týče čerstvosti záznamů, stejně jako v případě IP adresy může být záznam o AS zastaralý, ale je to méně pravděpodobné.

Přístup přes dotazování na databázi *Whois* na serverech registrátorů (RIPE NCC, ARIN, APNIC, LACNIC, AFRINIC) je rozšířen v mnoha nástrojích, například NetGeo [24]. Konkrétní realizace tohoto přístupu bude potom podrobně konzultována v kapitole 4, jelikož se domníváme, že je to jeden z vhodných způsobů pro relativně přesnou geolokaci IPv6 adresních bloků pro reálné použití v provozu. V následujících podkapitolách si shrneme v současnosti používané nástroje, ve kterých se pro geolokaci využívá právě této techniky.

### 3.2.3 Systém NetGeo

Systém **NetGeo** je geografická databáze uzlů v internetu, kterou vyvíjí společnost CAIDA. Používá se na mapování IP adres a čísel autonomních systémů na geografickou polohu. Pro nalezení hodnot souřadnic, jako je zeměpisná šířka a délka, NetGeo hledá nejprve záznam pro danou IP adresu nebo AS ve své vlastní databázi, která využívá systém cache. Databáze tímto způsobem minimalizuje zátěž na serverech *Whois*, protože při nalezení záznamu pro hledaný cíl si informaci uloží do své cache. Pokud takový záznam existuje, NetGeo vrací přímo zeměpisné souřadnice, jinak zkouší dotazovat všechny databáze hlavních světových registrátorů.

Po získání záznamu ze serveru *Whois*, skripty napsané v jazyce Perl rozparsují požadovaný záznam a uloží si informaci o lokalitě společně s datem poslední aktualizace. Parser se snaží rozdělit získané informace z textu na město, okres a stát a využívá k tomu různé metody. Pokud nezíská alespoň jednu důležitou informaci v sekci *address*, pak se snaží provést rozbor telefonních čísel nebo dvoupísmenných TLD z e-mailů uvedených v sekci *contact*, čímž získá minimálně alespoň stát.

Databáze NetGeo dále obsahuje mapování jména lokality nebo poštovních směrovacích čísel na hodnoty zeměpisné šířky a délky. Na závěr se do databáze uloží cílová IP adresa

nebo AS společně s geografickými souřadnicemi. Pak dokáže se získanými daty i dále pracovat, například vizualizovat lokalizovanou IP adresu. Produkt NetGeo je licencován pro společnosti Ixia a je využíván komerčně.

## 3.3 Geolokace na základě měření nebo vlastností sítě

Většina současných pokročilejších technik pro geolokaci je založena na měření doby odezvy mezi „známými uzly“, takzvanými orientačními body a mapování IP adresy na geografickou lokaci. Tyto techniky potom využívají toho, že tyto orientační body (uzly, routery) jsou významné a nemají tendenci se často měnit (a když, tak jsou aktualizovány jejich záznamy DNS), takže podle mapování záznamů DNS těchto uzlů na IP adresy je definována vzdálenost, podle které je možno velice přesně odhadnout lokaci uzlu v internetu. Míra přesnosti samozřejmě závisí na mnoha faktorech, jmenujme například hustotu pokrytí krajiny orientačními body. Při cestě paketu skrze orientační body je potom možno efektivně a rychle určit geografickou vzdálenost, kterou paket urazil, i čas, jak dlouho paket putoval, a na základě těchto informací se dá s určitou přesností lokalizovat polohu původce paketu.

### 3.3.1 Databáze IP2Location

IP2Location [2] je komerční geolokační databáze vytvořená a spravovaná firmou Hexasoft. Obsahuje širokou paletu geolokačních informací jako převod IP adresy na konkrétní stát, také může sloužit k získávání informací o počasí nebo šířce pásma. Jejich databáze *DB5* mapuje IP adresy na stát, region, město, zeměpisnou šířku a délku. Všechny zeměpisné lokality pro IP adresy jsou udávány jako rozsahy, které se liší ve velikosti a v granularitě. Pro mapování se využívá algoritmu sdružování IP adres do takzvaných prezenčních bodů (PoP). Tento algoritmus patří do skupiny geolokačních technik založených na měření zpoždění, protože díky němu se IP adresy sdružují. V rámci jednoho PoP pak můžeme s určitou vysokou pravděpodobností říci, že se v něm geograficky nachází daná IP adresa, případně jejich rozsah [29].

## 3.4 Systém IP2Geo

Tato podkapitola se zabývá průzkumem toho, jestli a jakým způsobem je možné realizovat službu mapování IP adres na geografickou lokaci pro uzly v internetu. Popisuje, rozebírá a diskutuje tři techniky kolektivně označované jako *IP2Geo*, které jsou v současné době používané jako jeden z přístupů k řešení problému geolokace. Jedná se o netriviální problém, poněvadž IP adresa nemusí nutně obsahovat indicie vedoucí k přesné lokalitě, která se k ní váže. Zaměříme se tedy na popis tří odlišných technik, které byly prozkoumány v [27] a které dokáží určit s nízkou mírou chybovosti lokalitu uzlu.

První ze zmiňovaných technik, *GeoTrack*, odvozuje geografickou polohu na základě doménového jména cílového uzlu nebo jemu blízkých uzlů v síti. Záznam DNS v uzlu někdy obsahuje použitelné informace o poloze zkoumaného uzlu. Taková informace může indikovat lokalitu na různých úrovních přesnosti, například na úrovni měst (např., *corerouter1.SanFrancisco.cw.net* indikuje lokalitu města San Francisco), nebo zemí (např., *portal.gov.cz* určuje stát - Českou republiku).

Druhá, *GeoPing*, využívá měření zpoždění v síti v geograficky oddělených lokalitách, z čehož následně odvozuje souřadnice cílového uzlu. Je založena na předpokladu, že latence zjištěná cestováním paketu mezi dvěma uzly v síti je funkce geografického odstupu mezi uzly

(funkce je příbuzná ke vztahu mezi sílou signálu a vzdáleností využívaném v bezdrátových uživatelských polohových systémech). Jedná se zde samozřejmě o jistou míru aproximace, kdy technika silně závisí na empirických výsledcích měření doby odezvy. Tato závislost bude podrobněji diskutována v podkapitole 3.4.2.

Třetí a poslední technika, *GeoCluster*, kombinuje částečně (ne zcela přesné) mapování IP adresy na lokalitu a prefix BGP k určení polohy cíle. Částečná databáze je získána z různých webových zdrojů, přičemž její neúplnost spočívá zejména v relativně nízkém počtu obsažených IP adres, jež jsou namapovány na umístění. Prefix BGP je tedy používán k rozšíření pokrytí takhle získaných dat, kdy jsou identifikovány tzv. clustery IP adres, které se s určitou vysokou pravděpodobností vyskytují ve stejné geografické lokalitě. Tato technika je navíc schopna nabídnout indikaci toho, jak přesně je lokalita odhadnuta.

Každá z těchto technik řeší problém geolokace z různých úhlů pohledu, jako je například hierarchické adresování, korelace mezi latencí a vzdáleností, vliv umístění proxy na přesnost geografické polohy uzlu apod. *IP2Geo* sdružuje geolokační principy, na jejichž základě bylo možno sestrojit pokročilejší techniky. Podrobněji si popíšeme první dvě z nich, *GeoTrack* a *GeoPing*.

### 3.4.1 Technika GeoTrack

*GeoTrack* se snaží extrahovat lokalitu uzlu z doménového jména uzlu nebo jemu blízkých síťových zařízení. Obvykle bývají rozhraní na směrovačích pojmenována podle geografického umístění, což bylo experimentálně zjištěno empirickým testováním v [27].

Směrovač označíme jako *rozpoznatelný*, pokud může být jeho zeměpisné umístění odvozeno z jeho doménového jména. Směrovače, jejichž IP adresa nemůže být namapována na doménové jméno nebo jejichž doménové jméno neobsahuje smysluplnou zeměpisnou polohu, označíme jako *nerozpoznatelné*. Metoda *GeoTrack* tedy dokáže podle doménového jména odhadnout přibližnou polohu směrovače. Nejprve si určí cestu v síti mezi zdrojovým a cílovým uzlem použitím nástroje `traceroute`, čímž získá seznam směrovačů po cestě k cíli. Dále pomocí doménových jmen zjistí jejich geografickou polohu z rozpoznatelných směrovačů. Zde je vhodné poznamenat, že neexistuje žádný standard pro pojmenování směrovačů. Záznam DNS tedy obsahuje podobu kódu, z kterého je nutno informaci o lokalitě určit. To je ne vždy jednoznačné, nicméně jak bylo vyzkoumáno v [27], existuje několik používaných kódu pro směrovače, a to kód města, státu a letiště, podle nichž je možné určit lokalitu směrovače. Například, *brn-pop-r1-vl163.netbox.cz* označuje kódem `brn` město Brno. *GeoTrack* používá speciální algoritmus pro extrakci kódů a jejich převod na zeměpisný název. Tak získá přehled o *geografické cestě* paketu sítí k cílovému uzlu. Nakonec odhadne zeměpisné umístění cíle na základě znalosti polohy posledního rozpoznatelného směrovače, tj. nejblíže k cílovému počítači.

Co se týče výkonnosti, ve [27] byly provedeny testy účinnosti geolokace, a tudíž je možno porovnat techniku *GeoTrack* s již popsanou technikou *NetGeo*, která využívá dotazů na Whois. Medián chyby ve vzdálenosti od skutečného cíle byl 590 km u *GeoTrack*, resp. 650 km u *NetGeo*. Tak velká chyba je zčásti zapříčiněna tím, že mnoho testovaných subjektů je připojeno za proxy.

### 3.4.2 Technika GeoPing

*GeoPing* je technika pokoušející se určit geografickou polohu na základě zkoumání vztahu mezi délkou zpoždění(ms) a vzdáleností(km). Funguje na principu měření doby zpoždění k cílovému uzlu z několika různých zdrojů (jejich lokace je známa) a kombinací měření se

pokouší odhadnout souřadnice cíle. V rozsáhlých sítích však již z principu existuje pouze velmi nízká korelace mezi prodlevou a skutečnou vzdáleností. Jednak je to dáno geografickými oklikami při cestě přes směrovače, druhak některé linky mohou být tzv. „úzkým hrdlem“, takže v síti narůstá zahlcení a tím pádem i prodleva. Nicméně, v dnešní době prudce vzrostl počet a kapacita vysokorychlostních linek v internetu, a také už datagramy díky vyšší hustotě prezenčních bodů (PoP) necestují oklikami. Se zahlcením v síti čekají datagramy ve frontách, což také výrazně narušuje vztah zpoždění - vzdálenost. *GeoPing* zmírňuje tento problém tak, že naměří několik vzorků odezvy mezi dvěma uzly a z těch pak vybere minimální. Bylo experimentálně zjištěno [27], že pro stabilizaci minima je nutné sesbírat 10-15 vzorků zpoždění. Vlastní vyhodnocení pak spočívá ve vytvoření párů [nejnižší zpoždění, geografická vzdálenost] pro každý směrovač na cestě od zdroje k množině cílů, k určení geografické polohy směrovačů je použito techniky *GeoTrack*. Poté se spočte kumulativní rozložení vzdáleností vzhledem k danému zpoždění. Pokud nejsou příliš nevyvážené rychlosti linek v testovací množině, medián odchylky je v tomto ideálním případě okolo 150 km, což je vzhledem k technikám konzultovaným v předchozích podkapitolách jistý úspěch. Nicméně, přítomnost speciálních linek typu dial-up či satelitních spojů degraduje efektivitu této geolokační metody vzhledem k tomu, že ve zmiňovaných případech nezávisí vysoká odezva na geografické vzdálenosti.

## 3.5 Maximální odhad pravděpodobnosti

*Maximum Likelihood Estimation*, zkráceně *MLE*, je poměrně nová technika pro IP geolokaci založená na statistickém modelu latencí v internetu. Použití MLE pro geolokaci cílového uzlu bylo motivováno vykazováním určitých pravděpodobnostních charakteristik u naměřených latencí mezi množinou orientačních bodů za určitý čas. Za tím účelem se používá ICMP `ping`, který změří parametr *round trip delay*, což je čas, za jak dlouho je doručeno potvrzení o přijetí paketu druhou stranou. Z opakovaného měření latencí byla v [11] odvozena funkce hustoty pravděpodobnosti, jakou vzdálenost pro danou latenci urazí paket. Aplikací této pravděpodobnostní funkce byl spočten údaj *MLE* pro množinu naměřených latencí z orientačních bodů k cíli, který odhaduje jeho pravděpodobnou geografickou pozici. Naměřené latence se následně rozdělí do intervalů, které nazveme *vzdálenostní koše*. Počet těchto košů byl určen experimentálně na 40. Všechny koše jsou stejné velikosti a zároveň každý z nich pokrývá určitý rozsah vzdáleností a tím pádem i určitou geografickou oblast, do které s jistým pravděpodobnostním odhadem na základě naměřené latence cíl patří. Kdyby bylo košů podstatně méně, každý z nich by pokrýval větší rozsah vzdáleností, a tak by znepřesnil odhad výskytu uzlu. V rámci každého koše se získá lineární vztah mezi střední hodnotou naměřených hodnot zpoždění a vzdáleností. Tento vztah je dán jako průměrná hodnota všech naměřených hodnot latence vyskytujících se ve vzdálenostním koši. Potom dokážeme říci, s jakou pravděpodobností geolokovaný objekt spadá do kterého koše, resp. do jaké geografické oblasti. Pro efektivitu tohoto statistického přístupu mluví i medián geolokační chyby, jež byla pouze 134 km na testovaném vzorku dat v [11].

## 3.6 Geolokace založená na omezení

*Constraint-based Geolocation* neboli *CBG* je technika, která se pokouší určit geografickou polohu metodou tzv. *multilaterace*. Tak je označen proces, který odhaduje pozici za použití dostatečného počtu vzdáleností ke známým orientačním bodům. Výsledkem procesu je po-

tom souvislý prostor, který se vytvoří na základě vzdáleností z dostatečně velké množiny orientačních bodů k cílovému uzlu v internetu. Multilaterací se tak provede odhad polohy, podobně jako je tomu u systému GPS [20]. Tam je však měření času, který signál urazí k satelitu, naprosto přesné. Naproti tomu v internetu, stejně jako u *GeoPing*, je třeba odhadnout geografickou vzdálenost ze známých orientačních bodů k cíli na základě měření zpoždění, což není příliš přesné (jak bylo řečeno v 3.4.2).

Klíčovým prvkem pro omezující geolokaci je její schopnost přesně transformovat naměřené zpoždění na vzdálenostní omezení. Důležitým faktorem je to, že digitální informace se šíří po optickém kabelu téměř přesně 2/3 rychlosti světla ve vakuu. Tak získáme horní hranici omezení vzdálenosti. Mezi dvěma koncovými body také musí existovat minimální doba nutná pro přenesení paketu. Vzniká zde jisté zkreslení, která se však dá do určité míry odstranit kalibrací naměřených latencí. *CBG* dokáže poté transformovat množinu latencí na rádius, který vychází z fyzikálních omezení pro vzdálenost, ve které se cíl minimálně musí nacházet a za kterou se už nacházet nemůže. Multilaterací se ustanoví odhad, ve které zeměpisné oblasti se nachází hledaný cíl. Z testů provedených v [20] se medián geolokační chyby dostal na uzlech v Evropě dostal pod 25 km. Technika je navíc schopna určit s jistotou region, kde se uzel musí nacházet, což se hodí pro aplikaci využívající geolokaci. Sama se tak může rozhodnout, jestli je tato zjištěná přesnost dostatečná pro její potřeby. Hlavním přínosem geolokace založené na omezení je její schopnost převést dobu naměřené latence na omezení pro vzdálenosti cíle, což je poté použito multilaterací pro přesnější odhad geografické polohy.

## 3.7 Shrnutí

Cílem této kapitoly bylo prozkoumat několik různých přístupů ke geolokaci IP adres. Geolokační techniky jsme analyzovali s důrazem na použitelnost při adresách IPv6. Zjistili jsme, že metoda využívající databáze Whois jako zdroj dat nemusí mít data přesná a aktuální, stejně jako může obsahovat neúplná a neplatná data o adresách registrujících entit. Nicméně po pečlivém prozkoumání politiky registrátorů (zejména RIPE) o přidělování prefixů IPv6 jsme dospěli k závěru, že registrující entity jsou prověřovány a kontaktní adresy jsou po zadání entitou ještě ověřeny. S ohledem na nasazení geolokačního systému v praxi při zachování relativní jednoduchosti konstrukce jsme usoudili, že geolokační technika, která získává data z databází Whois, je vhodná, zejména pro její kompatibilitu s IPv6 adresami (databáze Whois totiž obsahují jednotně jak informace o IPv4, tak i o delegovaných prefixech IPv6). Prozkoumali jsme také existující nástroje využívající této techniky. Většina z nich je komerční, avšak obsahují nějakou omezenou funkčnost (například přesnost jen v rámci krajiny). Ostatní techniky jako *MLE* nebo *CBG* považujeme za zajímavé myšlenky s teoreticky větší přesností geolokace v ideálních podmínkách, avšak v praxi jen obtížně realizovatelné. Tyto techniky také trpí neduhy jako použití proxy serverů či anonymizérů, medián nepřesnosti se potom statisticky dá porovnat s chybami, které produkuje databáze Whois (například firma se sídlem v Německu, avšak konektivitou používanou v Lucembursku). Při návrhu geolokačního systému, který vyplývá z výsledku provedené analýzy, se budeme detailněji zabývat dotazováním na databáze Whois a procesem získání relevantních geografických adres pro IPv6 sítě.

| **Technika** | **Vlastnosti** | **Dostupnost** |
|---|---|---|
| DNS LOC | využívá DNS, jednoduchá implementace, začlenění geografických informací přímo do záznamu DNS, není téměř vůbec rozšířená, je přesná při zadání korektních údajů | otevřená |
| Dotazování Whois | využívá veřejně přístupné databáze Whois, kontaktní údaje musí být uvedeny, možnost neaktuálních dat, problém s geografickým rozptýlením IP adres z přiděleného rozsahu, rovnocenná geolokace pro IPv4 i IPv6, přesnost dostatečná obvykle v rámci města i státu | otevřená |
| Měření zpoždění a využití vlastností sítě | využívá „orientační body“ v síťové topologii, seskupuje IP adresy do prezenčních bodů, měří se doba cesty paketu mezi orientačními body, míra přesnosti relativně vysoká, avšak závisí na hustotě pokrytí orientačními body | komerční |
| IP2Geo | extrahuje lokalitu uzlu z doménového jména v kombinaci s měřením latence, medián chyby od 150 do 650 km | komerční |
| MLE | založeno na statistickém modelu latencí v internetu, spočítá se funkce závislosti zpoždění a vzdálenosti, medián chyby 134 km | otevřená |
| CBG | technika pracuje s fyzikálními omezeními v síti, odhad polohy uzlu pomocí multilaterace, medián chyby v Evropě pod 25 km | otevřená |

Tabulka 3.1: Shrnutí vlastností a dostupnosti geolokačních metod

# Kapitola 4

# Návrh geolokačního systému

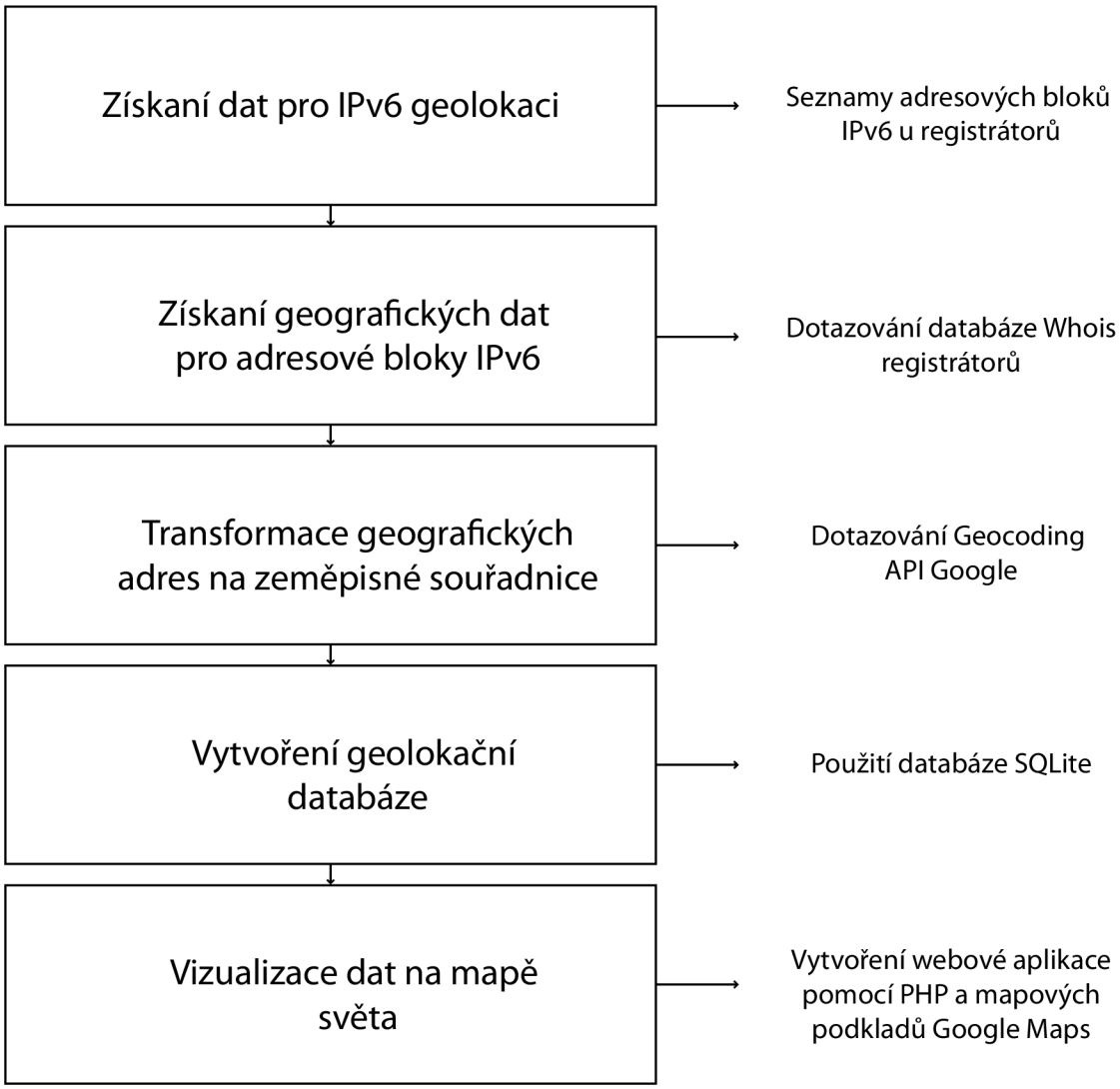
Tato kapitola popisuje návrh systému pro geolokaci s automatickou aktualizací, zejména se zabývá získáváních relevantních dat a vytvářením vlastní geolokační databáze. Databáze by měla být rychlá a umožňovat efektivní vyhledávání. V počáteční fázi návrhu systému pro geolokaci se musíme zamyslet nad vhodným způsobem, jakým geolokaci vůbec provést. V kapitole 3 jsme zhodnotili současné metody, kterými lze geograficky lokalizovat internetové uzly podle IP adresy. Protokol IPv6 ještě v současnosti příliš mnoho geolokačních nástrojů nepodporuje či neuvažuje. Například *MaxMind* [6] sice databázi pro IPv6 poskytuje, ale dokáže geografickou polohu určit pouze na úrovni jednotlivých zemí světa. Na základě námi provedeného průzkumu geolokačních přístupů, odborné konzultace s konzultantem ze společnosti *CZ.NIC* a následném zvážení všech výhod a nevýhod jsme pro získání relevantních dat zvolili metodu založenou na dotazech do databáze Whois jednotlivých registrátorů. Návrh celého systému si rozdělíme na jednotlivé podproblémy, z nichž každý zahrnuje rozbor a vyhodnocení dílčího problému finálně vedoucí k řešení systému jako celku. Budeme diskutovat použitou metodiku a případné problémy, se kterými jsme se při navrhování geolokačního systému setkali. Postup při návrhu systému přibližuje vývojový diagram na obrázku 4.1. Následující podkapitoly se věnují dílčím logickým částem, které jsme při návrhu brali v úvahu. Jedná se o návrh subsystému pro získávání dat a jejich převod do správného formátu, což popisujeme v 4.1. Podstatná část návrhu se bude zabývat vytvořením vlastní geolokační databáze.

## 4.1 Získání dat pro IPv6 geolokaci

Prvním a velice důležitým krokem je vybrat efektivní způsob pro získání dat potřebných ke geolokaci. Protože jsme si vybrali jako vhodnou metodu dotazování nad databázemi *Whois*, je třeba zvážit, jakým způsobem bude toto dolování dat realizováno. Zjistili jsme, že jednotliví registrátoři IP adresních bloků v rámci sítě internet (dále jen *registrátoři*) si udržují soupis aktuálně alokovaných i přiřazených adresních bloků v souborech, které jsou veřejně dostupné na jejich serverech. Tyto seznamy mají standardizovaný obsah a obsahují informace o počtu záznamů pro IPv4 adresy, IPv6 adresy a autonomní systémy. Protože se zabýváme tématem geolokace IPv6 adres, budeme dále uvažovat pouze adresové bloky IPv6. Pro každý delegovaný adresový blok se evidující tyto informace:

- Registrátor, který blok deleguje
- Kód státu, jemuž je daný blok přidělen

- Počáteční adresu IPv6 přiděleného rozsahu
- Masku, která říká, kolik bitů dané adresy je podstatných
- Datum přidělení bloku registrátorem
- Stav adresního bloku, nabývá stavů `allocated` a `assigned`, pro nás jsou podstatné oba dva




Obrázek 4.1: Vývojový diagram při návrhu geolokačního systému

Seznamy delegovaného prostoru jsou aktualizovány každým registrátorem jednou za 24 hodin. Vzhledem k tomu, že náš systém má podporovat automatickou aktualizaci, této vlastnosti využijeme při získávání dat o přidělených adresových blocích od registrátorů. Soubor s delegovanými bloky je sice ukládán každý den pod jiným názvem zohledňující

datum vytvoření, avšak existuje symbolický odkaz se suffixem `latest`, který se odkazuje vždy na nejnovější verzi souboru. Na veřejném serveru FTP registrátora RIPE NCC jsou takto shromážděny a pravidelně aktualizovány seznamy delegovaného prostoru všech pěti registrátorů. To pro nás znamená, že v každém okamžiku vytváření geolokační databáze budeme mít k dispozici aktuální informace o alokovaných či přiřazených adresových blocích IPv6.

## 4.2 Převod dat do vhodné formy

Jelikož seznamy delegovaného prostoru obsahují i pro naše účely spoustu nerelevantních informací, musíme z nich vyseparovat pouze podstatnou část pro vlastní geolokaci, kterou je přidělený rozsah IPv6 v podobě počáteční adresy rozsahu a maska. Jelikož *vhodnou formou* myslíme takový zápis, který podporuje univerzální `whois` klient, musíme získaná data dále upravit. Jak jsme si uvedli v kapitole 1, zkrácený zápis adresy IPv6 se zapisuje ve formátu *adresa/maska*. Dále je pro další postup podstatné vědět, který registrátor deleguje jaký blok. Jako vhodný prostředek pro parsování informací ze souborů a jejich převod do požadované podoby zvolíme skriptovací jazyk Perl. Požadovaným výstupem činnosti skriptu je vytvoření soupisu, který obsahuje úseky alokovaných bloků IPv6 včetně masky. Úseků je celkem pět, na samém počátku každého umístíme zarážku, která nám říká, že následující úsek obsahuje bloky spravované konkrétním registrátorem. Jeho název je uveden na řádku, kde se nachází oddělovač úseků. Protože získané bloky budeme dále zpracovávat pomocí programu `whois`, je důležité vědět, kterého serveru Whois se máme dotazovat na aktuálně zpracovávaný blok.

## 4.3 Určení geografické polohy podle adresy IPv6

V okamžiku, kdy máme připravené všechny aktuálně registrované IPv6 adresní bloky, přistoupíme ke zjištění jejich geografické lokace. Jak jsme naznačili na počátku kapitoly, budeme se dotazovat databází Whois. V těch jsou uloženy informace o registrující entitě, pro kterou byl konkrétní adresový prostor alokován registrátorem. Nejpodstatnější informací, kterou nám server registrátora vrátí jako odpověď, je adresa registrující entity, tedy organizace. V praxi to bude znamenat, že bude potřeba implementovat skript, který se na všechny získané bloky z vytvořeného soupisu (viz předchozí podkapitola) dotáže příslušných serverů *Whois*. Následně tuto adresu vyseparuje a spáruje se zpracovávaným rozsahem. Samozřejmě, jak bylo diskutováno v kapitole 3, žádná geolokační metoda není zcela přesná. Zvláště při získání údajů přes *Whois* může být nepřesnost v tom, že celý alokovaný rozsah se nenachází v těsné blízkosti adresy registrující entity. Na druhou stranu, vzhledem k téměř nevyčerpatelnému rozsahu IPv6 adres jsou bloky přidělovány tak, že není potřeba šetřit adresy jako u protokolu IPv4, a tak se dá očekávat, že většina rozsahů bude registrována korektně, alespoň v rámci jednoho města. Zde je dobré si uvědomit, že cíl této práce nespočívá v invenci co nejpřesnější geolokační metody, nýbrž schopnost podle dodaných reálných dat rozhodnout, do jaké geografické oblasti je daný rozsah přidělen, kde geograficky se konkrétní IPv6 adresa nachází, určit hustotu přidělovaných bloků v rámci regionů. Domníváme se, že medián chyby při geolokaci uzlu tímto způsobem je v rámci rozumných mezí. Verifikace vlastností našeho geolokačního systému bude záležitostí testování při uvedení systému do provozu.

## 4.4 Převod adresy na geografické údaje

Logickým krokem pro efektivní reprezentaci získaných dat se jeví být transformace geografických adres na zeměpisné souřadnice (dle souřadného systému WGS-84), které taktéž jednoznačně identifikují lokalitu. Motivací k tomuto kroku je především formát dat. Zatímco adresa jako poměrně dlouhý řetězec by byla uložena neefektivně, lokalita kódovaná v souřadnicích je uložena jednoduše jako dvě čísla typu float, kde první reprezentuje zeměpisnou šířku, druhé délku. Samotný převod jsme schopni realizovat za použití API, které poskytuje společnost Google.

## 4.5 Vytvoření geolokační databáze

Nyní, když máme získaná data převedená do vhodné formy, vytvoříme samotnou geolokační databázi. S ohledem na efektivitu budoucí implementace je nutné navrhnout si vhodné datové struktury, které nám poskytnou dostatečnou abstrakci nad naší databází. Databázi jsme se rozhodli realizovat jako aplikaci napsanou v jazyce C++, přičemž využíváme knihovny SQLite3. Tento typ databáze jsme vybrali zejména z důvodu efektivního vyhledávání nad velkým objemem dat. Výhodou je také použití knihovny v rámci objektového modelu C++, kde se všemi položkami databáze pracujeme jako s objekty. Databáze SQL bude realizována jako jediný binární soubor, který bude lokálně uložen, bez nutnosti provozovat databázový server SQL. Bylo třeba se důsledně zamyslet nad použitím datových typů, aby byla zachována rychlost zpracování dotazů a požadavek na efektivní uložení dat. Databázi jsme navrhli jako jednu tabulku se šesti sloupci, jejichž datové typy popisuje tabulka 4.1.

| **Sloupec tabulky** | **Datový typ** | **Velikost dat** |
|---|---|---|
| 64bitový síťový prefix IPv6 | `UNSIGNED BIGINT` | 8B |
| maska (shodná délka prefixu) | `UNSIGNED TINYINT` | 1B |
| kód země | `CHAR(2)` | 2B |
| zeměpisná šířka | `FLOAT` | 4B |
| zeměpisná délka | `FLOAT` | 4B |
| geografická adresa | `VARCHAR(200)` | až 200B |

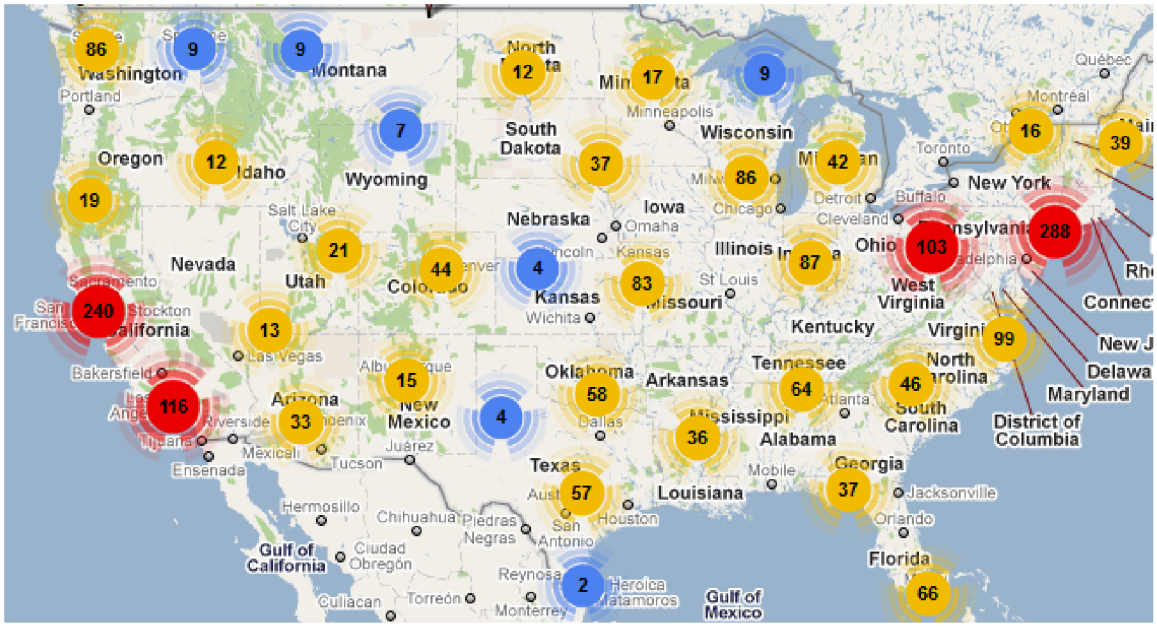
Tabulka 4.1: Struktura záznamu v geolokační databázi

Při budování databáze jsme se zaměřili především na efektivitu vyhledávání podle IPv6 adres, kdy IPv6 adresy jsou reprezentovány polem 64 bitů, přičemž jejich porovnání a ověření, do kterého adresového rozsahu spadají, je díky bitovým operátorům v C++ rychlé. IPv6 adresy i masky jsou převedeny z textových řetězců na bitová pole. Zeměpisné souřadnice budeme ukládat jako čísla s plovoucí desetinnou čárkou. Jediné řetězcové typy jsou kód země a geografická adresa. Kód země ale obsahuje vždy právě dva znaky, takže položka ve sloupci zabere maximálně 2B. Nejvíce prostorově náročné je uložit geografickou adresu, pro kterou jsme zvolili jsme datový typ `VARCHAR(200)`. Oproti typu `CHAR` má tu zvláštnost, že prostor pro uložený řetězec se alokuje dynamicky. Velikost takto uloženého řetězce je potom vždy délka uložené adresy inkrementována o 1.

## 4.6 Vizualizace dat

Takto definovaný geolokační systém je poté možné použít k vizualizaci dat. První možností použití je zobrazit lokalitu konkrétní adresy IPv6 na světové mapě. Toho dosáhneme při propojení našeho systému s API *Google Maps*, kdy vstupem bude soubor XML s geografickými souřadnicemi a popisem místa, a výstupem bude značka bodu (tzv. marker) na světové mapě (umístíme značku do *Google Maps*).

Druhou nepochybně zajímavou možností je vytvářet mapování hustoty pokrytí krajin (vztaženo k mapě) adresovými bloky IPv6. Jedná se o statistickou funkci, která zobrazuje všechny registrované bloky adres IPv6 v dané zemi na mapě. Výstup v podobě bodů bude poté zkombinován s technikou shlukování bodů. To znamená, že čím zeměpisně hustěji budou u sebe přidělené bloky adres IPv6, příslušný (teplejší) odstín barvy se vykreslí do ikony shluku na mapu společně s číslem, které reprezentuje počet bloků IPv6 na území jednoho takového shluku. Funkci mapování hustoty poskytuje rozhraní *MarkerClusterer* v kombinaci s API *Google Maps* pro vykreslení daných bodů. Takové zobrazení je potom škálovatelné z hlediska intenzity barev v závislosti na počtu zobrazovaných položek. Ukázku vizualizace hustoty pokrytí adresovými bloky IPv6 na území Spojených států amerických vidíme na obrázku 4.2.




Obrázek 4.2: Vizualizace hustoty pokrytí USA adresovým prostorem IPv6

## 4.7 Shrnutí

Předmětem této kapitoly bylo analyzovat dostupné informace a navrhnout geolokační systém pro adresy IPv6. Kapitola ukazuje, jakým způsobem se získají data o adresním prostoru IPv6 z veřejné databáze Whois a jak podle nich určíme geografickou polohu. Byl navržen způsob transformace geografické adresy na kartézské souřadnice pomocí Geocoding API Google. Dále jsme provedli návrh geolokační databáze typu SQL včetně formátu jednotlivých sloupců tabulky, zamysleli jsme také se nad použitými datovými typy s ohledem na

velikost databáze. Na závěr kapitoly jsme analyzovali možnosti vizualizace bodů na mapu pomocí API Google Maps, přičemž navržený geolokační systém bude disponovat webovým uživatelským rozhraním.

# Kapitola 5

# Implementace systému

Tato kapitola se zabývá samotnou implementací navrženého geolokačního systému. Kapitola je rozdělěna na tři stěžejní implementační celky, které jsme řešili. První se týká systému pro získávání dat, druhá popisuje implementaci databázového systému a vlastní aplikace, třetí potom vizualiaci. Budeme zde také diskutovat veškeré problémy a zvláštnosti, se kterými jsme se v průběhu implementace museli potýkat. V závěru kapitoly budeme konzultovat efektivitu zvolené implementace a provedeme zde celkové zhodnocení vývoje aplikace.

## 5.1 Získání geolokačních dat

Jak jsme si uvedli v kapitole 4, prvním logickým krokem při vytváření geolokačního systému je sběr dat a jejich vhodná transformace do podoby, ve které budou posléze uložena do databáze. Primární zdrojem dat pro naše účely je databáze *Whois*, kterou provozuje všech pět nadnárodních registrátorů. Jelikož jde především o dolování dat a jejich úpravu, nejvhodnějším kandidátem pro tuto úlohu je skriptovací jazyk. Pro tuto první fázi jsme tedy zvolili implementační jazyk *Perl*, který nabízí především efektivní práci s regulárními výrazy, ale přitom umožňuje provádět i systémová volání. Také umí modifikovat části právě načítaných řádků či celé řádky, čehož posléze využijeme při různých úpravách a vylepšeních činnosti dolovacího skriptu.

Hlavní funkční jednotkou ve skriptu je rutina *datamine($$)*. Tato rutina se vykoná postupně pro všech pět registrátorů. Rutina přijímá dva parametry, a to URL souboru s delegovanými bloky a adresu serveru Whois.

### 5.1.1 Data o delegovaném adresním prostoru IPv6

Prvotní úlohou je získat od každého registrátora seznam alokovaných a přiřazených adresních bloků IPv6. Pomocí systémového volání `system()` je zavolán program *wget*, který požadovaný soubor stáhne. Tento soubor následně prochází po řádcích a za relevantní považuje pouze řádky obsahující klíčové slovo `ipv6`, mimo řádku s počtem záznamů o IPv6. Řádky s informacemi o delegovaném adresovém prostoru IPv6 potom vypadají následovně:

```
ripencc|SA|ipv6|2a02:d70::|32|20090421|allocated|
```

Protože pro geolokaci potřebujeme znát rozsah adres, masku a kód země, záznam přetransformujeme pomocí kódu

```
$_ =~ /(\w+)\|(\w+)\|(\w+)\|(.[^|]+)\|(\d+)/;
```

do následující podoby:

```
2a02:d70::/32|SA
```

Každý takto vytvořený řádek uložíme do hash tabulky se záznamy adres IPv6. Tato datová struktura byla zvolena proto, aby bylo možné efektivně kontrolovat, zda-li už se nějaký záznam v množině nachází, či nikoli. Tuto vlastnost později využijeme. Také provedeme náhodnou rotaci záznamů v poli. Testováním bylo zjištěno, že při sekvenčním procházení záznamů se může stát, že narazíme na velmi mnoho rychle zpracovaných dotazů na prefixy hned za sebou, a tím by teoreticky mohlo dojít k překročení limitu získaných dat o objektech. Vytvořené pole poté procházíme a pro každý záznam se spojíme pomocí roury s programem *whois*, kterým se dotážeme na server Whois aktuálně zpracovávaného registrátora. Otevře se tak datový stream, do které proudí informace z výstupu dotazu Whois a skript je přímo čte. Odpověď serveru Whois se skládá z pojmenovaných objektů, z nichž každý má různé povinné a nepovinné položky. Informace o formátu databáze Whois byly čerpány především z [13], částečně také z [21].

### 5.1.2 Dotaz Whois na podřízené prefixy IPv6

Zde je dobré zmínit fakt, že každý z registrátorů má vlastní formát uložení informací v systému Whois, proto bylo nutné při zpracování dat brát ohled na různorodost i nestandardnost některých záznamů, které Whois vrací. Pro RIPE NCC, které (jako jediný z registrátorů) podporuje přímé dotazy na všechny hierarchicky podřízené podsítě spadající pod určitý prefix [13], bylo třeba implementovat zvláštní typ dotazování. Nejprve se systém dotáže na server Whois s parametrem `-- -r -M`, který vypíše podřízené rozsahy adres pod aktuálně zpracovávaným prefixem. Zpravidla se jedná o situace, kdy je alokován jeden velký /32 blok, který je dále rozčleněn na administrativně menší bloky. Tyto bloky (obvykle /48 nebo /64) se potom přiřazují přímo jednotlivým zákazníkům. Parametr `-r` je do dotazu zakomponován z důvodu limitu na počet získaných citlivých informacích o subjektech od RIPE NCC[16]. Tento parametr zakazuje rekurzivní prohledávání a nedoluje citlivé informace o objektech typu `person`, tudíž se na tímto způsobem položený dotaz nevztahují limity. Protože jsme ve fázi, kdy nás zajímají pouze IPv6 adresy všech podřízených podsítí a dvoupísmenný ISO kód země [9], nepotřebujeme zatím citlivé informace jako adresa, majitel, apod.

V odpovědi se adresa bloku se nachází na řádku začínajícím `inet6num:`. Ke každému z těchto bloků se také získá informace o kódu země. Tato informace je obsažena na řádku začínajícím `country:`. Skript tak na konec pole záznamů přidá všechny zjištěné podřízené rozsahy ve správném formátu a následně pokračuje ve zpracovávání aktuálního záznamu. Ovšem řádek s novým rozsahem přidáme pouze tehdy, pokud již jednou není v poli obsažen. Toto je nutné kontrolovat z důvodu vzniku případných duplicit. Pokud je nalezen řádek s `%ERROR:101:`, znamená to, že žádný podřízený rozsah není k záznamu zaznamenán, a tedy se pokračuje také ve zpracování aktuálního záznamu. V praxi to potom znamená, že v databázi budou uloženy jak hlavní delegované prefixy IPv6, tak i rozsahy spadající pod ně. Řádky obsahující klíčová slova `descr:` a `remarks:` ihned bez čtení přeskakujeme, jelikož testováním bylo zjištěno, že mohou obsahovat zavádějící informace, jako například v poznámce klíčové slovo `inet6num`, apod. Přesnou specifikaci formátu objektů RPSL používaných v databázi *Whois* RIPE NCC jsme pro implementaci čerpali z [16]. V nalezených informacích ISO kód země transformujeme na kapitálky a v IPv6 adrese převedeme veškerá písmena na malá. Tuto úpravu jsme provedli z důvodu jednotnosti informací vkládaných do geolokační databáze.

```
% Information related to '2001:1488:1001::/48'

inet6num:       2001:1488:1001::/48
netname:        CZ-NIC-NETWORK
descr:          CZ.NIC, z.s.p.o
country:        CZ
admin-c:        CZ-RIPE
tech-c:         CZ-RIPE
status:         ASSIGNED
mnt-by:         CZ-NIC-MNT
source:         RIPE # Filtered

% Information related to '2001:1488:1010::/48'

inet6num:       2001:1488:1010::/48
netname:        CZ-NIC-NETW2
descr:          CZ.NIC, z.s.p.o
country:        CZ
admin-c:        CZ-RIPE
tech-c:         CZ-RIPE
status:         ASSIGNED
mnt-by:         CZ-NIC-MNT
source:         RIPE # Filtered
```

Obrázek 5.1: Odpověď na query: whois -h whois.ripe.net −− -r -M 2001:1488::/32

### 5.1.3 Dotaz Whois na konkrétní prefix IPv6

Nyní se tedy skript dotáže stejným způsobem (ovšem již bez parametrů) serveru Whois na aktuální záznam v poli. Protože účelem skriptu je získat informace o adrese pro každý záznam, snažíme se ve výstupním proudu adresu klasifikovat pokud možno co nejpřesněji. Ve whois odpovědi existuje několik typů objektů s adresou (`person`, `role`, `organisation`), přičemž ne každý objekt musí být nutně přítomen v každé odpovědi. Testováním bylo zjištěno, že nejvyšší přesnost geolokace je při adrese pod objektem `person`, který popisuje zodpovědnou osobu za přiřazený adresní blok. Skript byl proto navržen tak, že se nejprve snaží najít veškeré řádky začínající klíčovým slovem `address:` pod objektem `person`. Pokud neuspěje (například chybí objekt `person`, případně chybí adresa v tomto objektu), snaží se vyhledat jakýkoli výskyt klíčového slova `address:` v odpovědi. Adresu získáváme ze všech řádků za prvotním klíčovým slovem `address:`, nezávisle na tom, jestli se na dalších řádcích klíčové slovo vyskytuje. Formát objektu podporuje obě možné notace [13]. Jednotlivé části získané adresy jsou oddělovány čárkou a postupně konkatenovány s dosud získaným adresním řádkem. V případě ARIN jsou záznamy uloženy v jiném formátu, než jsou RPSL objekty RIPE NCC. Bylo nutné implementovat odlišné parsování adresních řádků. Aby byla geolokace v rámci Severní Ameriky přesná, je nutné zpracovat také řádky začínající `city:` a `stateProv:`, které obsahují město, resp. stát v rámci Kanady a Spojených států.

```
% Information related to '2001:718::/32'

inet6num:       2001:718::/32
netname:        CZ-TEN-34-20010521
descr:          CESNET Sub-TLA block
country:        CZ
org:            ORG-CA9-RIPE
status:         ALLOCATED-BY-RIR
mnt-by:         RIPE-NCC-HM-MNT
mnt-lower:      TENCZ-MNT
mnt-routes:     AS2852-MNT
source:         RIPE # Filtered

organisation:   ORG-CA9-RIPE
org-name:       CESNET
org-type:       LIR
address:        CESNET z.s.p.o.
                Zikova 4
                160 00 Prague 6
                Czech Republic
mnt-ref:        TENCZ-MNT
mnt-by:         RIPE-NCC-HM-MNT

person:         Pavel Vachek
address:        CESNET, z.s.p.o.
address:        Zikova 4
address:        Praha 6
address:        160 00
address:        The Czech Republic
org:            ORG-CA9-RIPE
```

Obrázek 5.2: Odpověď na query: whois -h whois.ripe.net 2001:718::/32

### 5.1.4 Úprava získaného adresního řádku

Nyní máme pro alokovaný prefix IPv6 získaný adresní řádek. Ten sestává z několika položek, oddělených čárkou, nebo také jen z jediné, v závislosti na formátu, v jakém byla adresa do databáze Whois vložena žadatelem. Protože je to právě onen adresní řádek, který slouží jako vstup transformaci na souřadnice, bylo třeba experimentálně zjistit, jaká je přesnost převodu na souřadný systém ze získané adresy. V následující tabulce uvádíme úpravy provedené v adresním řádku pro jednotlivé registrátory.

Dále bylo nutné upravit veškeré adresní řádky tak, že pokud obsahují znak `'`, pak je tento znak nahrazen dvojicí stejných znaků `''`. Toto opatření je provedeno kvůli pozdějšímu vkládání záznamů do databáze. V databázi SQLite, tak jako v každé databázi SQL, je znak `'` znakem pro uvození, resp. ukončení řetězce. Zdvojený výskyt tohoto znaku je převeden na normální apostrof v rámci textového řetězce.

Co se týče adresních řádků z databáze Whois APNIC, existují takové záznamy, které

| server Whois | Provedená úprava |
|---|---|
| whois.ripe.net | Odstranění znaků & a #, odstranění čísel z konce řádku, převod kódování z ISO-8859-2 na Unicode |
| whois.arin.net | Odstranění znaků # |
| whois.apnic.net | Odstranění znaků & a #, odstranění sekvencí čísel za znaky + nebo - |
| whois.afrinic.net | Odstranění znaků & a #, odstranění čísel z konce řádku, |
| whois.lacnic.net | Odstranění znaků & a #, odstranění všeho kromě města mezi dvěma pomlčkami, převod kódování z ISO-8859-2 na Unicode |

Tabulka 5.1: Úprava získaných adres pro jednotlivé RIR

nemají adresu uvedenou vůbec. V tom případě se jako geolokační údaj použije ISO kód státu, který je přítomen. Některé kódy států jsou však chybně vyhodnoceny API Google Maps, tudíž bylo nutné provést další úpravy. Adresní řádky i samotné kódy států jsou kontrolovány a je na nich provedena následující úprava:

- "IN"nahradíme "India", chybně vyhodnoceno jako "Indiana, US"
- "ID"nahradíme "Indonesia", chybně vyhodnoceno jako "Idaho, US"
- "DE"nahradíme "Germany", chybně vyhodnoceno jako "Delaware, US"
- "MD"nahradíme "Moldavia", chybně vyhodnoceno jako "Maryland, US"

### 5.1.5 Transformace adresy na souřadnice

Postupem popsaných v předchozích podkapitolách jsme získali finální verzi adresního řádku, pomocí kterého se pokusíme získat souřadnice, jmenovitě zeměpisnou šířku a délku. Na počátku jsme si vhodně nainicializovali proměnné `lat` a `lng` na hodnotu `0.0`, čehož posléze využijeme při neúspěšném pokusu o transformaci.

Vlastní převod provedeme dotazem na Geocoding API Google Maps [19]. Dotaz položíme pomocí protokolu HTTP, kdy žádost ve správném formátu (dle dokumentace v [19] doplníme o adresní řádek získaný ve fázi dolování dat a také o formát výstupu. V našem případě jsme zvolili jako výstupní formát soubor XML, který se stáhne ze serveru Googlu jako odpověď na naši žádost.

Získaný soubor otevřeme pro čtení a snažíme se v něm najít výskyt souřadnic zeměpisné šířky a délky. To znamená, že vyextrahujeme hodnoty, které jsou uzavřeny mezi tagy `<lat>` a `</lat`, resp. `<lng>` a `</lng>`. Čtení souboru ukončíme, pokud narazíme na tag `</location>`. Nyní, v případě, že byl převod adresy na souřadnice úspěšný, máme v proměnných `lat` a `lng` uloženy zeměpisnou šírku a délku.

Převod však nemusí být vždy úspěšný. V případě, že po prvním dotazu nezískáme souřadnice, spustíme cyklus, v jehož krocích postupně manipulujeme adresou a zkoušíme znovu dotaz s upravenou adresou. V prvním kroku cyklu z adresního řádku odstraníme veškeré číslice. Pokud jsme stále nezískali uspokojivou odpověď, postupně adresní řádek redukujeme. V každém dalším kroku cyklu vždy odstřihneme část adresního řádku po první čárku včetně. Čárka slouží jako oddělovač jednotlivých řádků s adresou v odpovědi serveru Whois, takže je jistá šance, že čím se stává adresa obecnější, tím je vyšší pravděpodobnost

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
 <status>OK</status>
 <result>
  <type>street_address</type>
  <formatted_address>Zikova 1905/4, 160 00 Prague 6-Dejvice, Czech Republic
  </formatted_address>
  <address_component>
   <long_name>4</long_name>
   <type>street_number</type>
  </address_component>
  <address_component>
   <long_name>Zikova</long_name>
   <type>route</type>
  </address_component>
  <address_component>
   <long_name>Prague 6</long_name>
   <type>sublocality</type>
  </address_component>
  <address_component>
   <long_name>Prague</long_name>
   <type>locality</type>
  </address_component>
  <address_component>
   <long_name>Czech Republic</long_name>
   <short_name>CZ</short_name>
   <type>country</type>
  </address_component>
  <geometry>
   <location>
    <lat>50.1021773</lat>
    <lng>14.3884772</lng>
   </location>
   <location_type>ROOFTOP</location_type>
  </geometry>
 </result>
</GeocodeResponse>
```


Obrázek 5.3: Výstup XML s odpovědí na transformační dotaz

úspěšného převodu pomocí Geocoding API Google Maps. Tuto vlastnost jsme ověřili v praxi testováním. V případě, že už je poslední zbývající část adresy dále nedělitelná, nicméně jsme stále neuspěli s převodem na souřadnice, znamená to, že k prefixu IPv6 v databázi Whois byl jako adresa vložen pravděpodobně nesmysl, jako například pouze název firmy, nějaká neidentifikovatelná zkratka, apod. Pokud se tedy skutečně přes všechna provedená vylepšení nepodařilo převést adresu na souřadnice, přistoupíme k převodu podle ISO kódu země. Takový převod uspěje vždy, i když je částečně nepřesný. Pořád se však z pohledu

geolokační databáze jedná o úspěšný převod, jelikož jsme prefix IPv6 dokázali jednoznačně identifikovat alespoň v rámci celého státu.

### 5.1.6 Výstup skriptu

Nyní jsme se dostali k poslednímu kroku, a to je výpis získaných dat. Skript vypisuje na standardní výstup, což je efektivní, protože ho jednoduše můžeme přesměrovat do souboru. Této vlastnosti později využijeme při automatickém vytváření databáze pomocí bash skriptu.

Výstup skriptu pro jeden prefix IPv6 nazvěme geolokační řádek. Ten má následující podobu, ve které jsou obsaženy všechny podstatné informace pro uložení do databáze, a to: prefix IPv6 (počáteční adresa), maska, dvoupísmenný ISO kód státu, zeměpisná šířka, zeměpisná délka, získaná vydolovaná adresa. Výsledný výstup pro jeden prefix potom vypadá takhle:

```
2620:0:910::|48|US|39.9533130|-75.1721933|1900 Market Street,Philadelphia,PA
```

### 5.1.7 Limity, omezení a jejich řešení

Při získávání dat pro geolokační databázi jsme narazili hned na několik více či méně závažných problémů. Prvním z nich bylo, že maximální počet dotazů na *Google Maps Geocoding API* je 2500 denně. Po překročení této hodnoty server vrací v XML souboru pouze status `OVER_QUOTA` a při opakovaném dotazování může být IP adrese žadatele zablokován přístup. Naše řešení daného omezení spočívá v pokládání dotazu na konverzní API pouze jedenkrát za 35 sekund, což v praxi znamená, že za 24 hodin je skript schopen geokódovat přibližně 2400 adresních řádků. Hodnotu 35 sekund jsme spočítali následovně:

```
60s * 60min * 24h = 86400s za den
86400s / 2500 dotazů = 34.70 dotazů/s
```

Z výpočtu vidíme, že při tomto tempu dotazování máme ještě nějakou rezervu, takže denní limit nepřekročíme. Z toho však vyplývá další problém, a to, že geokódování bude tímto tempem příliš pomalé a než se vytvoří celá databáze, trvalo by to i týdny. Proto jsme implementovali další optimalizaci, a to v podobě vytvoření hash tabulky, kam ukládáme již jednou geolokované adresy. Vždy před samotnou transformací na souřadnice ověřujeme, zda-li je adresní řádek unikátní, anebo zda taková adresa už existuje v hash tabulce. V případě, že adresa je unikátní, převedeme ji na souřadnice a tuto trojici (adresní řádek, zeměpisná délka, zeměpisná šířka) si uložíme do hash tabulky, jejímž klíčem je právě adresní řádek. V případě, že adresa unikátní není, pouze si k odpovídajícímu klíči v tabulce najdeme záznam o souřadnicích a celý geolokační řádek vypíšeme. To je poměrně velká optimalizace, poněvadž nemusíme pro již převedené adresy vůbec dotazovat API Google. Z testování vyplynulo, že počet stejných adres je (převážně pro jemně členěné subdelegované rozsahy adres spadající pod jeden prefix) relativně velký, a proto tímto vylepšením ušetříme přibližně 40% všech dotazů na API Google. Podobnou techniku jsme aplikovali i tehdy, pokud se záznam nepodaří transformovat na souřadnice a je třeba provést geolokaci podle kódu státu. Jednotlivé kódy státu si držíme jako klíče v hash tabulce a k nim máme odpovídající již transformované souřadnice. Potom například pro neexistující adresu v Itálii jen zkontrolujeme hash a vezmeme klíč `IT` a k němu souřadnice. Tímto řešením ušetříme necelých 5% dotazů.

Rozestup 35 sekund mezi dotazy na API Google řešíme pomocí systémového volání `sleep()`, které běžící proces na požadovaný čas uspí.

Další omezení je nastaveno na serverech Whois. To se vztahuje pouze na rekurzivní dotazování (mezi které se řadí i bežné dotazy), tedy takové, které v odpovědi na výstup vypisují i RPSL objekty `person`, `role`, `organisation`. Přesný limit není stanoven a nepodařilo se nám ho získat ani při e-mailové komunikaci přímo s administrátorem databáze Whois RIPE NCC. Nicméně testování ukázalo, že je to přibližně 30000 dotazů během 24 hodin, přičemž toto množství se reguluje podle rychlosti dotazů. Abychom se tomuto, pro činnost skriptu klíčovému, omezení dokázali vyhnout, před každým dotazem na server Whois uspáváme skript na jednu sekundu. U adres uložených v mezipaměti uspáváme skript na 6 sekund. Také provádíme některé další optimalizace, jelikož jsme zjistili, že adresový prostor spravovaný společnosti Six Access (SixXS) je rozčleněn na několik desítek tisíc prefixů s maskou /48 až /64, kde ze všech těchto prefixů jsme schopni získat pouze jedinou adresu. Tím, že známe adresu, geolokujeme ji pouze jednorázově a při výskytu klíčového slova `mnt-by:` v nerekurzívním dotazu Whois ověřujeme, zda za ním následuje slovo `SIXXS-MNT`. V případě, že ano, pouze vypíšeme příslušný geolokační řádek bez nutnosti se dotazovat serveru Whois kvůli adrese. Tato optimalizace nám ušetří přibližně 30000 zbytečných dotazů na databázi Whois RIPE NCC, které produkují stále stejný výsledek.

## 5.2 Geolokační databáze

Postupem uvedeným v podkapitole 5.1 jsme získali soubor dat obsahující geolokované všechny dosud existující přiřazené prefixy IPv6. Tato data bylo potřeba zpracovat tak, aby byla uložena efektivně s ohledem na velikost a aby v nich bylo možné rychle vyhledávat. Jak vyplývá z analýzy a návrhu, rozhodli jsme se geolokační databázi realizovat jako aplikaci napsanou v jazyce C++ s využitím knihovny SQLite. Tato databázová knihovna byla vybrána jednak z důvodu efektivity při vyhledávání nad velkým objemem dat, jednak pro rychlost vytváření databáze a v neposlední řadě pro její relativně jednoduché, avšak efektivní použití v kombinaci s objektovým modelem C++. V rámci implementace pak bylo důležité zamyslet se nad návrhem tabulek SQL i nad dotazy do takové databáze, aby byla zachována rychlost a efektivita. Neméně důležité bylo zvážit použití vhodných datových struktur a typů, nad kterými aplikace i databáze operují. Vytvořením databáze se rozumí vytvoření binárního datového souboru `geo.db`, který bude uložen lokálně.

### 5.2.1 Návrh databáze jako tabulky SQL

Geolokační databázi jsme navrhli jako jedinou tabulku SQL s šesti sloupci. Prvním sloupcem tabulky je `ip_range`, který nese informaci o počáteční adrese prefixu IPv6, tedy rozsahu adres. Datový typ byl zvolen `UNSIGNED BIGINT`, který nabývá maximální velikosti 64 bitů. Druhým sloupcem je `mask`. Do tohoto sloupce je ukládána maska prefixu. Protože maximální velikost masky je 64, zvolili jsme datový typ `UNSIGNED TINYINT`, který má velikost 8 bitů. Další sloupec, `cc`, má datový typ `CHAR(2)`, což znamená, že má pevnou velikost 2 znaky. Tyto nesou dvoupísmenný ISO kód státu. Čtvrtý, resp. pátý sloupec obsahuje souřadnice, tedy zeměpisnou šířku, resp. délku. Oba tyto sloupce `lat`, `lon` jsou datového typu `FLOAT`, což charakterizuje datový typ s plovoucí desetinnou čárkou. Posledním, šestým sloupcem, je `address`. V něm je uložen vydolovaný adresní řádek. Zvolili jsme datový typ `VARCHAR(200)`. Oproti typu `CHAR` má tu zvláštnost, že prostor pro uložený řetězec se alokuje dynamicky. Velikost takto uloženého řetězce je potom vždy délka uložené adresy inkrementována o 1.

V tabulce SQL potřebujeme, co se týče uložených geolokačních dat, mít unikátní spojení počáteční IPv6 adresy rozsahu (prefixu) a síťové masky. To je vyžadováno zejména z důvodu existence více stejných prefixů s odlišnými maskami. V praxi totiž často dochází k situacím, kdy je např. pro poskytovatele internetového připojení alokován velký blok s maskou /32, který je potom dále rozdělený na několik menších prefixů, např. s maskou /48. Může tedy existovat např. prefix *2a02:3c0::/32* i prefix *2a02:3c0::/48*. Proto vytvoříme *složený primární klíč* ze sloupců `ip_range` a `mask`. Tak zajistíme unikátnost pro každý vkládaný řádek do tabulky. Na položkách primárního klíče jsou v SQLite zároveň automaticky vytvářeny indexy, což vede k rychlejšímu prohledávání databáze zejména při velkých tabulkách.

### 5.2.2 Vytvoření databáze

Vytvoření geolokační databáze vyplývá přímo z návrhu v předešlé podkapitole. Vlastní tabulka `geodb` se vytvoří pomocí následujícího příkazu SQL:

```
CREATE TABLE IF NOT EXISTS geodb
                                (ip_range UNSIGNED BIGINT,
                                 mask UNSIGNED TINYINT,
                                 cc CHAR(2),
                                 lat FLOAT,
                                 lon FLOAT,
                                 address VARCHAR(200),
                                 PRIMARY KEY(ip_range, mask));
```

### 5.2.3 Načítání dat do databáze

Tato podkapitola popisuje implementaci metod, které slouží pro načítání dat do databáze. V rámci aplikace jsme vytvořili samostatnou třídu pro databázi a samostatnou třídu pro záznam. Pro databázi i záznam pak vytváříme v hlavním programu vždy instanci objektu dané třídy. Pro každou třídu jsme implementovali metody, které pracují nad příslušnými objekty.

Datový soubor načítáme do databáze tak, že zpracováváme postupně jeho jednotlivé řádky získané činností skriptu. Každý jeden řádek nejprve zpracujeme (viz dále), tedy převedeme na vhodný formát a poté provedeme zápis do databáze příkazem `INSERT`. Protože by ale každá operace `INSERT` znamenala zároveň i `COMMIT` do databáze, celý proces načítání databáze by byl velmi pomalý. Využijeme tedy toho, že v SQL databázi můžeme používat transakce, a všechny operace typu `INSERT` zaobalíme mezi databázové příkazy `BEGIN TRANSACTION` a `END TRANSACTION`. V praxi to potom znamená, že všechny operace typu `INSERT` se provedou současně pouhým jediným zápisem typu `COMMIT` do databáze.

Geolokační data v rámci jednoho řádku jsou konkatenována pomocí znaku |, který slouží jako oddělovač. Celý řádek tedy rozdělíme pomocí metody `Record::Tokenize()` do vektoru řetězců na jeho jednotlivé složky a ty následně zpracujeme v metodě `Record::Parse()`. Nejpodstatnější z hlediska rychlosti vyhledávání v databázi je převedení prefixové adresy IPv6 na posloupnost bitů. Protože IPv6 adresa je složena ze síťové části identifikující síť nebo podsíť (horních 64 bitů adresy) a části identifikující koncové síťové zařízení (spodních 64 bitů), můžeme si dovolit uvažovat při zpracování pouze horních 64 bitů. Adresa ve tvaru řetězce je tedy převedena pomocí bitové aritmetiky na datový typ `uint64_t`, který dokáže obsáhnout všech podstatných 64 bitů prefixu IPv6. Maska i souřadnice jsou převedeny na číslo. Adresa zůstává ve tvaru řetězce, kód země je převeden na pole dvou znaků. Řádky,

které neobsahují masku v intervalu $\langle 0, 64\rangle$ jsou vynechány. V praxi se jedná zejména o kořenový uzel databáze Whois 0::/0, který existuje pouze kvůli hierarchii, ovšem tuto vlastnost jsme implementovali především kvůli garanci bezchybnosti vytváření databáze.

Každý takto zpracovaný řádek je vložen do databáze pomocí metody `Record::CreateQueryInsert()`, která transformuje získané údaje z geolokačního řádku na následující dotaz typu SQL:

```
INSERT INTO geodb VALUES(ip_range,
                         mask,
                         cc
                         lat,
                         lon,
                         address);
```

Desetinná čísla v souřadnicích jsou předtím zformátována na tvar *.7f*, tak aby byla zachována přesnost na sedm míst za desetinnou čárkou. Při použití tohoto způsobu nedochází k zaoukrouhlování oproti číslu získanému z odpovědi od Google API.

### 5.2.4 Vlastní SQL funkce IsInNet

Protože vstupem do databáze je konkrétní IPv6 adresa nebo příslušná prefixová část, bylo nutné zajistit, aby v rámci vyhledávání byla nalezena maximální možná shoda s geolokovanými prefixy IPv6. To znamená provést porovnání vstupu (konkrétní IPv6 adresy) s jednotlivými řádky tabulky a zjistit, zda se daná IPv6 adresa nachází v nějaké síti specifikované prefixem uloženým v databázi. SQL databáze však žádnou takovou funkci porovnání nemá. Z toho důvodu jsme implementovali zcela novou funkci SQL, která je schopna výše uvedené spočítat, přičemž jsme kladli důraz na efektivitu zvoleného řešení.

Funkce `IsInNet` přijímá tři parametry, a to v následujícím pořadí: prefix uložený jako sloupec `ip_range`, síťovou masku uloženou ve sloupci `mask` a danou IPv6 adresu, pro níž chceme provést geolokaci. Protože obě IPv6 adresy i masku jsme uložili jako posloupnost bitů, pro efektivní porovnání můžeme použít operace bitové aritmetiky, které jsme považovali pro tento účel za nejvhodnější. V databázi jsou tyto hodnoty uloženy jako datový typ `SQLITE_INTEGER`, přičemž pro přímé efektivní porovnávání extrahujeme IPv6 adresu a prefix jako `sqlite3_value_int64`. Samotné porovnání provádíme tak, že vymaskujeme vstupní IPv6 adresu vždy konkrétní maskou v řádku tabulky, přičemž výsledek této operace porovnáme s prefixem uloženým v daném řádku. Pokud se tato dvě čísla rovnají (jejich bity si přesně odpovídají), pak je výsledek porovnání kladný, v opačném případě záporný. V rámci SQLite vracíme jako výstup z vlastní funkce hodnotu

```
sqlite3_result_int(context, 1);
```

kde číslo `1` jako druhý parametr určuje, že porovnání bylo úspěšné. V případě neúspěšného porovnání funkce vrací jako druhý parametr číslo `0`.

Samotná funkce je součástí námi implementované metody `Database::Open()`, kdy současně při každém použití databáze je vlastní funkce `IsInNet` inicializována v kontextu SQLite následujícím funkcí:

```
sqlite3_create_function
   (database, "ISINNET", 3, SQLITE_UTF8, NULL,
    &Database::isInNetFunc, NULL, NULL)
```

Tímto způsobem jsme vytvořili vlastní dotaz SQL, který je nám schopen zjistit příslušnost IPv6 adresy do konkrétní podsítě. Díky němu lze v databázi efektivně vyhledávat tyto příslušnosti do subnetů, čehož je využito při téměř všech operacích nad databází, které naše knihovna implementuje.

### 5.2.5 Vyhledávání v databázi

Pro práci s databází je implementováno API, s jehož použitím je přímý přístup k databázi pro uživatele zcela transparentní. Implementovali jsme několik základních operací pro vyhledávání v databázi a přidali také jedno rozšíření. Všechny podporované operace si nyní popíšeme.

#### Vyhledávání podle konkrétní IPv6 adresy/prefixu

První a základní implementovanou operací bylo vyhledávání v geolokační databázi, kde jako kritérium je uživatelem zadaná IPv6 adresa, případně prefix. Uživatelem zadaná maximálně 128bitová IPv6 adresa jako argument programu je nejprve ořezána na její síťovou část. Potom je vytvořen dotaz SQL, který využívá funkce `IsInNet` a porovnává, do které podsítě geolokovaná IPv6 adresa spadá. Pokud spadá do více podsítí, je vybrána ta s nejdelší shodou v prefixu. To znamená, že maska určující podstatné bity v IP adrese je nejvyšší. Data do dotazu SQL jsou transformována metodou `Record::CreateQueryInInet()`. Zmiňovaný dotaz má podobu:

```
SELECT * FROM geodb
   WHERE ISINNET(ip_range, mask, ipv6) = 1
   ORDER BY mask DESC LIMIT 1;
```

Za povšimnutí stojí tvorba dotazu SQL způsobem, aby byl efektivní a byl proveden v co nejkratším možném čase. Vyhnuli jsme se vnořeným dotazům SQL SELECT i použití agregačních funkcí (například `MAX()`). Namísto toho jsme použili klíčové slovo `ORDER BY`. Jak už samo toto klíčové slovo napovídá, slouží pro třídění výstupu. Třídění výstupních řádků podle hodnot určitého sloupce nebo sloupců je velmi užitečné a často využívané. Sloupce, podle kterých chceme třídit, uvedeme jako seznam za `ORDER BY`. V našem případě provádíme třídení dle masky a naším požadavkem je pouze jediný řádek takový, který má současně hodnotu masky nejvyšší. Použití klauzule `DESC` říká, že se použije řazení podle hodnot ve sloupci sestupně a `LIMIT 1` určuje, kolik řádků se na výstup vypíše. Tímto způsobem dosáhneme požadovaného nejpřesnějšího prefixu pro zadanou IPv6 adresu.

#### Vyhledávání podle kódu státu

Další z podporovaných operací nad geolokační databází je vyhledávání veškerých alokovaných IPv6 prefixů v zemi identifikované podle jejího dvoupísmenného kódu ISO. Uživatelem zadaný kód státu je potom vstupem do dotazu SQL, který například pro vyhledání veškerých adresních bloků IPv6 v České republice bude mít následující podobu:

```
SELECT * FROM geodb WHERE cc='CZ';
```

Na výstup jsou tak vypsány všechny takové řádky z databáze, které obsahují ve sloupci *cc* uživatelem zadaný kód státu.

#### Fulltextové vyhledávání v adrese

Implementovaným rozšířením je potom možnost fulltextově vyhledávat v geolokační databázi ve sloupci geografické adresy. Tuto možnost jsme shledali velice užitečnou, protože například je možné vyhledávat podle měst a obcí, společností, libovolně dlouhé části adresy apod. Při implementaci jsme využili výhody databáze SQLite, a transformovali uživatelem zadaný text pro vyhledávání na dotaz SQL ve tvaru:

```
SELECT * FROM geodb WHERE address LIKE '%hledaný text%';
```

Operátor `LIKE` hledá shodu v daném sloupci *address*, znak `%` na obou stranách hledaného textu značí libovolné znaky ve výrazu zleva i zprava. Hledaný text tedy může být umístěn kdekoli v rámci celého sloupce s adresou. Výstupem takového vyhledávání jsou potom všechny řádky obsahující v získaném adresním řádku hledaný výraz.

### 5.2.6 Filtrování výstupu

Vhodnou vlastností, která bude později využita při vizualizaci, je filtrování výstupu na zadané sloupce. Při použití databáze v konzolové aplikaci si může uživatel nastavit, jak má být výstup omezen. Například je možné zobrazit pouze zeměpisné souřadnice, pouze adresu, nebo kombinaci obou požadavků. Také je možné omezit výstup pouze na kód státu, ve kterém je hledaná adresa IPv6 geograficky umístěna. Právě filtrování je následně využito při exportu výstupu a jeho transformaci do souboru .KML, nebo pro tvorbu výstupu pro php skript při použití interaktivního webového rozhraní.

### 5.2.7 Ovládání konzolové aplikace

Geolokační knihovnu je možné použít i jako konzolovou aplikaci, která má svá specifika při ovládání z příkazového řádku. Přijímá několik parametrů jako argumenty příkazové řádky, které jsou popsány zde:
Ovládání aplikace (povinné parametry):

- `./ip2geo -c`
  vytvoření databáze ze souboru geolocation
- `./ip2geo -q IPv6 adresa/prefix`
  geolokace konkrétní IPv6 adresy
- `./ip2geo -s "část adresy"`
  fulltextové vyhledávání v DB podle části adresy
- `./ip2geo -z "CZ"`
  vyhledávání v databázi podle kódu státu
- `./ip2geo -h`
  zobrazí nápovědu a popis ovládání

Filtrování výstupu (volitelné parametry):

- `-f`
  zobrazí pouze zeměpisné souřadnice pro hledný záznam
- `-a`
  zobrazí pouze adresní řádek pro hledaný záznam

### 5.2.8 Automatická aktualizace databáze

Geolokační databáze je aktualizována automaticky. Pro tuto funkci byl implementován skript *dbdaemon.sh*, jehož činností je periodické opakování následující množiny příkazů:

1. Spuštění skriptu na dolování dat, jehož výstup je uložen do souboru *geolocation*.
2. Zkopírování vytvořeného souboru do adresáře aplikace.
3. Změna aktuálního adresáře do adresáře aplikace.
4. V případě, že již databázový soubor existuje, dojde k jeho smazání.
5. Z výstupu skriptu je vytvořena databáze SQLite v souboru *geo.db*.
6. Je provedena archivace souboru obsahujícího geolokační databázi. Archivace je nastavena s parametrem *numbered*, který zajistí, že i při opakovaném přepisování souboru bude vytvářena záloha s vyšším číselným indexem. V praxi to znamená, že bude k dispozici historie záloh databáze. Současně je provedeno zálohování výstupu dolovacího skriptu se stejným parametrem.
7. Změna aktuálního adresáře zpět na původní
8. Vymazání dočasných souborů

Tato činnost se opakuje v nekonečné smyčce automaticky po spuštění skriptu, který se spouští příkazem

```
nohup ./dbdaemon.sh 2> chyby &
```

Příkaz *nohup* zajistí, že skript bude získávat data i po odhlášení, současně & zajišťuje spuštění na pozadí. Standardní chybový výstup je přesměrován do souboru *chyby*. Typickým příkladem chybového výstupu může být například nefunkční přesměrování uvnitř dotazování Whois, "Spojení odmítnuto", "Síť nedostupná"apod. Tyto chyby jsou generovány zejména u serveru Whois registrátora pro Severní Ameriku, ARIN, přičemž však nemají vliv na proces získávání dat.

Databáze se tedy udržuje stále v aktuálním stavu. Proces získávání dat pro databázi trvá přibližně 5 dní, přičemž obsahuje data pro vytvoření databáze s počtem více než 60000 unikátních záznamů (prefixů). Geolokační systém je tak zcela autonomní jednotkou bez nutnosti zásahů.

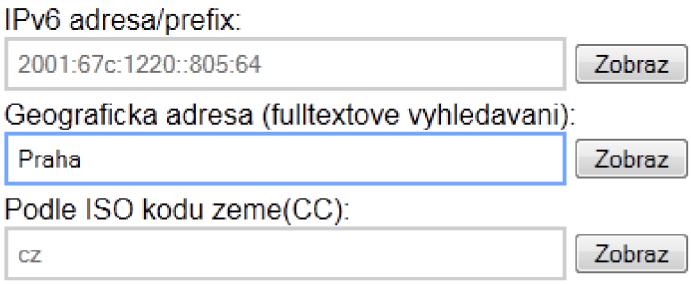
## 5.3 Vizualizace dat na mapě světa

Součástí implementovaného geolokačního systému bylo zpřístupnit uživatelsky přívětivé rozhraní, které by výstup dotazu na geolokační databázi vykreslilo do mapy světa. Pro práci s mapovými podklady jsme se rozhodli použít API *Google Maps*, které podporuje mnoho užitečných funkcí pro vykreslování bodů a manipulaci s mapou. Rozhodli jsme se vytvořit intuitivní a jednoduchou, přitom však účelnou webovou aplikaci. Za implementační jazyk jsme vybrali PHP ve verzi 5.3 a JavaScript s rozšířením jQuery. Díky nové verzi PHP5, která podporuje několik nových funkcí, jako je práce s formátem JSON, jsme tak mohli implementovat vylepšené přeposílání dat z geolokační databáze přímo do webové aplikace. Webový server Apache, který zajišťuje běh této aplikace, může být umístěn na libovolném

počítači v internetu. Při každém novém spuštění webové aplikace je zobrazena mapa celého světa, po zadání vstupu je v případě existujícího výsledku vždy zobrazena lokalita s bodem nebo více body a dalšími informacemi o těchto bodech.

### 5.3.1 Webová aplikace

Navržená webová aplikace obsahuje jeden plovoucí element s mapou, která je do něj umístěna jako samostatná vrstva Google Maps ve verzi 3. Dále obsahuje tři formuláře s potvrzovacím tlačítkem "Zobraz". V každém z nich může uživatel aplikace vyhledávat podle příslušných kritérií. Těmi jsou vyhledávání podle IPv6 adresy či prefixu sítě, fulltextové vyhledávání v geografické adrese a vyhledávání podle kódu státu. Po potvrzení volby ve formuláři dojde automaticky k vykreslení mapy bez nutnosti obnovení celé stránky pomocí technologie AJAX. Do mapy je v závislosti na volbě vyhledávání vykreslen také bod zájmu, nebo množina těchto bodů. Každý bod zájmu nese kromě svojí ikony také další informace v tzv. informačním okénku. Implementační podrobnosti budou zmíněny v následujících podkapitolách, které se věnují konkrétně jednotlivým funkcím mapy a vyhledávání.




Obrázek 5.4: Ovládací panel webové geolokační aplikace

### 5.3.2 Komunikace s databází

Díky uživatelskému rozhraní geolokační webové aplikace je získán vstup od uživatele. K tomuto účelu slouží příslušný formulář, který po stisku tlačítka data odešle metodou *POST*. Uživatelský vstup je následně předán databázové aplikaci, která se nachází na stejném serveru ve svém adresáři. To zajišťuje skript PHP, který zavolá aplikaci pomocí metody `exec()`, s příslušným parametrem v závislosti na požadavku uživatele. Samotný uživatelský vstup připojí také jako argument programu. Samozřejmě, proces spouštění aplikace s parametry je chráněn od veškerých nelegálních vstupů. K jejich ošetření jsme zvolili ochranné metody jazyka PHP, konkrétně k filtrování spuštění unixového příkazu jsme využili metodu `escapeshellcmd()` a k filtrování uživatelského vstupu (argumentu) metodu `escapeshellarg()`. Tyto odstraní ze vstupu veškeré potenciálně nebezpečné konstrukce, které by dovolily útočníkovi spustit další příkazy v prostředí unixového shellu. Také prověřujeme délku vstupu uživatele a znovu ještě samotný vstup. Nesmí mít nikdy nulovou délku, nesmí se v ní vyskytovat znaky jako %, ' a podobné, které by mohly způsobit nekonzistenci pozdějšího dotazu SQL. Dále odstraňujeme zbytečné dotazy, jako nelegální délku vstupu u požadavku na hledání podle dvoupísmenného kódu země ISO. Také je rozhodně správné opatření povolit démonu Apache komunikaci pouze uvnitř složky WWW s obsahem webu a databázovou aplikací. Operací `exec()` získáme čistý výstup aplikace na standardní

výstup. Tento si uložíme v PHP jako řetězec a kontrolujeme délku. Pokud je nulová, vracíme uživateli "Nenalezeno". Jelikož jsou sloupce tabulky ve výstupu z geolokační databáze rozděleny oddělovačem |, získaná data rozdělíme pomocí metody `explode()` a uložíme je do pole do příslušných proměnných. V případě, že je výsledkem dotazu do databáze více než jeden řádek, všechny tyto řádky si uložíme do pomocného pole a potom pro každý z nich provedeme výše uvedenou transformaci. Nyní známe IPv6 prefix, masku, zeměpisné souřadnice, geografickou adresu a kód státu. Získané pole dat zakódujeme do formátu JSON kvůli přenosu dat přes AJAX.

### 5.3.3 Zobrazování lokalit do mapy

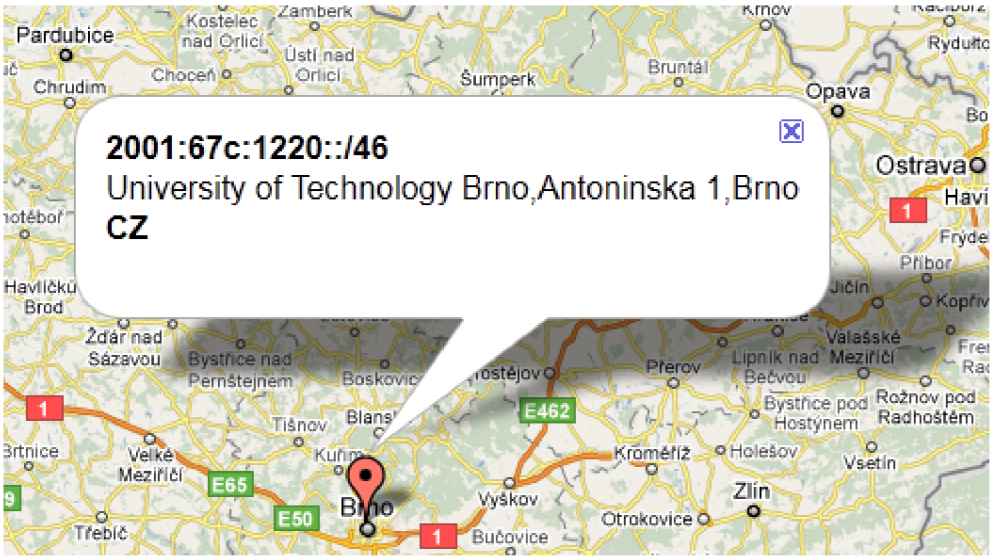
Po zadání vstupu tedy získáme odpověď od geolokační databáze a zkonvertujeme ji na JSON formát, který díky technologii AJAX přeneseme a dále zpracováváme v JavaScriptu. Nejprve zkontrolujeme, jaká data nám přišla. Pokud jsou označena štítky "Nenalezeno" nebo "Nesprávný vstup", uživateli tuto událost oznámíme prostřednictvím výstražného okénka metodou `alert()`. V případě, že se nám vrátilo správně dekódované JSON pole dat, dále s ním pracujeme v závislosti na typu dotazu uživatele.

#### Zobrazení bodu dle konkrétní IPv6 adresy/prefixu

Výsledkem zpracování vstupu z prvního formuláře je jediný řádek z databáze, který obsahuje geolokační záznam pro zadanou konkrétní IPv6 adresu nebo prefix. Tento řádek je reprezentován polem dat. Nejprve vždy provedeme vyčištění mapy od předchozích dotazů. Proces vyčištění je implementován metodou `cleanMap()`. To znamená, že všechny vykreslené body jsou smazány a všechna otevřená informační okna jsou zavřena. Nejprve si uložíme do speciální proměnné `LatLng` obě souřadnice, tedy zeměpisnou délku i šířku. Tato proměnná je objektem API Google Maps v3. Pomocí ní mapu vycentrujeme tak, že zeměpisná poloha je přesně v aritmetickém středu mapy. Měřítko mapy nastavíme na hodnotu 8, aby byl vidět bod i jeho blízké okolí a abychom mohli ihned odhadnout, v jaké lokalitě se daný bod nachází. Do proměnné `content` si uložíme informace o zobrazovaném bodu a naformátujeme je pomocí HTML. Mezi ně patří informace o prefixu včetně masky, který má nejdelší míru shody s hledanou adresou IPv6 nebo prefixem, geografické adrese a kódu státu. Bod je na mapu poté vykreslen pomocí metody `createMarker()`. K tomuto bodu je zároveň přidruženo informační okénko, které obsahuje bližší informace o bodu. Tato dvojice (vytvořený bod, informační okno) je poté uložena do pole zobrazovaných lokalit. Při zobrazování jediného bodu jako v tomto případě je explicitně nastaveno, aby se informační okno automaticky otevřelo již expandované. V praxi to znamená, že při potvrzení uživatelské volby v prvním formuláři se automaticky změní mapa, zobrazí se na ní geografická lokalita dané adresy IPv6, zacentruje se a přiblíží a otevře se malé okénko s detaily zobrazovaného bodu.

#### Zobrazení více bodů do mapy

V případě hledání všech prefixů IPv6 registrovaných pod kódem země ISO nebo fulltextového vyhledávání nad sloupcem *address* v databázi může být výstupem několik řádků. Pole dat z každého řádku databáze je zaobaleno do vnějšího pole, které sdružuje všechny řádky výstupu. Opět před každým zobrazováním bodů nejprve vyčistíme mapu. Změnou oproti zobrazení jednoho bodu je centrování mapy a nastavení měřítka přiblížení. Při implementaci bylo použito tzv. hraničních bodů, které v závislosti na přidávaných souřadnicích každého



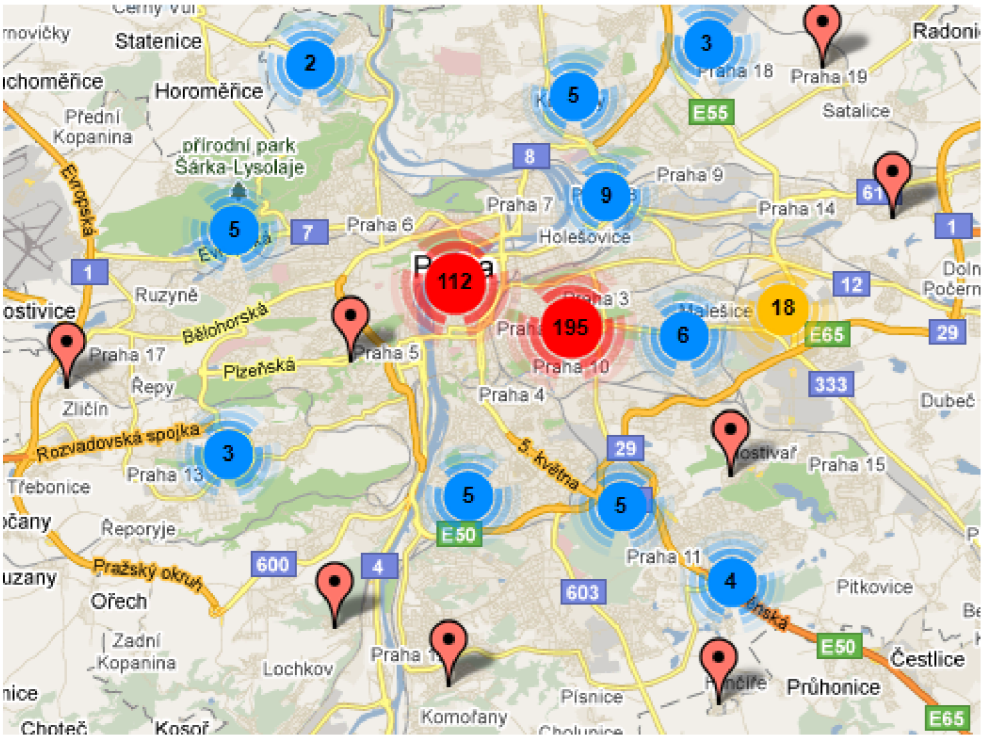

Obrázek 5.5: Zobrazení bodu do mapy při hledání dle IPv6 adresy/prefixu

bodu z množiny vždy upravují vystředění mapy a měřítko takovým způsobem, aby bylo možné vidět na mapě všechny zobrazené body. Postupně procházíme celé pole dat a pro každý bod vykreslíme do mapy značku v příslušné geografické lokalitě. Informační okénka všech bodů jsou implicitně zavřena. Při otevírání informačních oken u více vykreslených bodů jsme brali v úvahu zejména přehlednost zobrazení, proto jsme se rozhodli implementovat mechanismus, který dovoluje mít otevřené nejvýše jedno okno s podrobnostmi bodu na mapě. Vždy také přizpůsobíme zobrazení mapy právě přidávanému bodu. Se zobrazováním několika stovek, ba dokonce tisíc bodů na mapě, nastávají v zásadě dva problémy. Prvním z nich je rychlost zobrazování. Při počtu 10000 vykreslovaných bodů je spotřebováno spousta procesorového času a odezva webové aplikace není úplně ideální. Druhý problém je potom přehlednost mapy. Při menším měřítku není patrné, kolik bodů na nějakém místě existuje a je těžké od sebe jednotlivé značky rozeznat.

### Shlukování bodů na mapě

Z výše uvedených důvodů jsme se rozhodli implementovat metodu zvanou seskupování bodů, anglicky *clustering*. Jedná se o metodu sdružování skupiny bodů, které se nacházejí v rámci jednoho čtverce na mřížce. Tato mřížka pokrývá celou mapu světa. Velikost rozměrů mřížky je nastavitelná tak, aby nedocházelo k seskupování na příliš rozsáhlém území, anebo naopak, aby nevznikalo příliš mnoho shluků. Za pomoci externího API JavaScript *MarkerClusterer* jsme tuto metodu implementovali tak, že vždy při zobrazení více bodů na mapě jsou body shlukovány v rámci čtverců o velikosti 50. V praxi to potom znamená, že namísto například 40 zobrazených bodů v rámci jedné lokality s vysokou hustotou pokrytí bloky IPv6 uvidíme dva shluky, jeden třeba o 25 bodech a druhý o 15 bodech. Uprostřed ikony označující shluk bodů je číselně vyznačen počet bodů, které se nacházejí na mapě na území toho konkrétního shluku. Shluky jsou také dynamicky přepočítávány při přibližování nebo oddalování mapy. Při kliknutí myší na ikonu konkrétního shluku se mapa automaticky vycentruje a přiblíží tak, aby pokrývala rozsah tohoto shluku. Shluky se zobrazují až do úrovně přiblížení rovné hodnotě 12. Poté jsme již považovali zobrazování samostatných zna-

ček bodů za natolik přehledné, že není potřeba body dále shlukovat. Obvykle se také na tak malém území již tolik bodů nenachází. Intenzita shlukovaných bodů je vyjádřena pomocí barevných tónů ikon shluků. Čím více bodů se nachází v rámci shluku, tím je barva ikony teplejší (fialová, červená) a naopak, při nízké hustotě bodů přechází barva do chladných tónů (žlutá, světle modrá). Výhodou tohoto způsobu zobrazování mnoha bodů současně je především přehlednost a názornost. V neposlední řadě je pozitivně ovlivněna rychlost. Protože je ikon shluků méně než všech bodů, jsou vykresleny do mapy rychleji. Body jsou vykreslovány až v rámci určitého přiblížení mapy, a tudíž nemůže nastat situace, kdy je vykreslováno několik tisíc bodů současně.




Obrázek 5.6: Zobrazení více bodů do mapy při použití shlukování

## 5.4 Logování přístupu a dotazů

Při implementaci jsme mysleli také na zaznamenávání a statistiky přístupu uživatelů a jejich dotazů. Taková funkce je vhodná zejména pro účely statistické, kdy můžeme odhadnout, kdo a na co aplikaci používá. Také plní bezpečnostní funkci, kdy pokud dojde k bezpečnostnímu incidentu, lze z logů zpětně dohledat, proč k němu došlo.

V rozhraní webové aplikace zaznamenáváme veškeré vstupy uživatelů přes určená textová pole. Vždy je zaznamenána IP adresa uživatele, typ dotazu a vstup dotazu. Tyto informace jsou přes PHP zapisovány do souboru přímo na server, kde je spuštěn geolokační systém. K takovému souboru má potom přístup správce systému nebo jeho provozovatel. Více o výsledcích logování bude řečeno v kapitole 6.6.

## 5.5 Shrnutí

V této kapitole jsme popsali veškeré náležitosti týkající se implementace geolokačního systému jako celku.

Nejprve jsme se zabývali metodou pro získání relevantních dat pro geolokaci a jejich převod do podoby vhodné pro další zpracování. Ukázali jsme využití systému Whois při získávání adres pro přiřazené prefixy IPv6 a také jsme uvedli omezení a limity, se kterými bylo třeba při dolování dat počítat a se kterými jsme se implementačně vypořádali. Byl vysvětlen proces převodu geografických adres na souřadnice pomocí Geocoding API Google a byly zmíněny úpravy těchto adres takovým způsobem, aby byla jejich transformace na souřadnice co nejpřesnější.

Demonstrovali jsme způsob vytvoření geolokační databáze ze získaných dat, a to včetně popisu implementace databáze v jazyce C++ s využitím knihovny SQLite3, tedy i příkazů jazyka SQL. Vysvětlili jsme námi implementovanou funkci SQL `IsInNet`, která spočítá z dané IPv6 adresy jí příslušející prefix sítě IPv6. Také byly detailně popsány operace nad geolokační databází, stejně jako výstupy takových dotazů.

V poslední části kapitoly jsme si potom ukázali, jakým způsobem lze zobrazovat geografickou lokalitu hledané adresy nebo sítě IPv6. Popsali jsme vytvoření webové aplikace jako interaktivního uživatelského rozhraní pro naši geolokační knihovnu. Vysvětlili jsme, jakým způsobem je volána příslušná aplikace a jak je transformován výstup z databáze v geografické značky v mapě světa. Zabývali jsme se také problémem nepřehlednosti zobrazení více bodů na mapě, přičemž jsme detailně popsali použití shlukovacích metod při zobrazení značek.

Výstupem implementace je tedy interaktivní webová aplikace s intuitivním uživatelským rozhraním, která získává vstup od uživatele a ihned ho transformuje do podoby geografických značek do světové mapy.

# Kapitola 6

# Testování systému

Tato kapitola se věnuje testování geolokačního systému. Ukazuje, jak aplikace reaguje na různé vstupy, jak správné, tak nekorektní. Ověřuje také schopnost systému reagovat na libovolný vstup. Výstupy jsou demonstrovány pomocí vhodných obrázků, které lépe znázorňují činnost systému. Také je zde měřena doba reakce na různé vstupy, rychlost zpracování dotazu na databázi a rychlost vytvoření geolokační databáze. Také jsme se zaměřili na vyhodnocení správnosti a přesnosti geolokace IPv6. Protože jsme v systému implementovali logování dotazů, můžeme zde také zpracovat statistiku dotazů v rámci uvedení geolokačního systému do pilotního provozu.

## 6.1 Testování konzolové aplikace

V této podkapitole si ukážeme funkčnost konzolové aplikace. Ačkoli při spuštění v terminálu neukazuje lokalitu na mapě, může být využita ke geolokaci většího množství adres IPv6, například za souboru. Její výstup může být přesměrován do souboru, nebo do nějaké další aplikace, která pouze využívá geolokačních funkcí.

### 6.1.1 Geolokace konkrétní adresy IPv6

#### Reakce na korektní vstup

Jako testovací vstup pro geolokaci jedné adresy/sítě jsme zvolili IPv6 adresu, o které víme, že je adresou serveru v rámci FIT VUT v Brně.
Vstup: *2001:67c:1220:8b0::93e5:b0fb*
Výstup:

```
[xsucho00@qwe db]$ ./ip2geo -q "2001:67c:1220:8b0::93e5:b0fb"
2001:67c:1220::
46
CZ
49.2015407
16.603619
University of Technology Brno,Antoninska 1,Brno
```

Výstupem je tedy vždy šest bloků, kde je postupně vypsán registrovaný nejdelší shodný prefix IPv6, maska podsítě, kód země, zeměpisná šířka, zeměpisná délka a na závěr blok obsahující geografickou adresu. Výstup pro jeden vstup je vypsán v podobě jediného řádku,

kde jsou jednotlivé bloky odděleny pomocí |. Nicméně pro větší přehlednost v tomto textu jsme jednotlivé výstupní bloky oddělili.

#### Reakce na neexistující/nekorektní vstup

Pokud je zadána IPv6 adresa, jejíž prefix není aktuálně žádným registrátorem přidělen, na výstup není vypsáno nic. Pokud totiž není nalezena v databázi taková kombinace prefixu IPv6 a masky, se kterou se vstupní IPv6 adresa shodne, dotaz do databáze vrátí nulovou shodu ve funkci `IsInNet()`. Dále pokud je zadán jakýkoli nelegální vstup (pouze znaky, jiný formát než IPv6 adresa/prefix apod.), tento vstup je také odfiltrován a na výstup není vráceno nic.
Vstup: *2005:1000::*
Výstup:

```
[xsucho00@qwe db]$ ./ip2geo -q "2005:1000::"
```

Vstup: *neexistujici_vstup?!=*
Výstup:

```
[xsucho00@qwe db]$ ./ip2geo -q "neexistujici_vstup?!="
```

### 6.1.2 Geolokace s fulltextovým vyhledáváním

#### Reakce na korektní vstup

Jako korektní testovací vstup při dotazu, který vyhledá všechny záznamy se shodou v adresním řádku, jsme vybrali řetězec „CZ.NIC“. Tento řetězec je vyhledáván fulltextově, to znamená, že může být obsažen kdekoli v adresním řádku, nezávisle na obsahu před ním a za ním. To vede k určitým drobným nedostatkům, které budou diskutovány později. Nicméně funkce možného vyhledávání v databázi získaných adresních řádků může být v mnoha případech užitečná i přes její částečnou nedokonalost.

```
[xsucho00@qwe db]$ ./ip2geo -s "CZ.NIC"
2001:1488::|32|CZ|50.0729504|14.4376125
           |CZ.NIC, z.s.p.o. Americka 23 12000 Praha 2 Czech Republic
2001:1488:1001::|48|CZ|50.0724941|14.4381016
           |CZ.NIC, z.s.p.o.,Americka 23,Praha 2,120 00,The Czech Republic
2001:1488:1010::|48|CZ|50.0724941|14.4381016
           |CZ.NIC, z.s.p.o.,Americka 23,Praha 2,120 00,The Czech Republic
2001:678:1::|48|CZ|50.0729504|14.4376125
           |CZ.NIC, z.s.p.o. Americka 23 12000 Praha 2 Czech Republic
2001:678:f::|48|CZ|50.0729504|14.4376125
           |CZ.NIC, z.s.p.o. Americka 23 12000 Praha 2 Czech Republic
2001:678:10::|48|CZ|50.0729504|14.4376125
           |CZ.NIC, z.s.p.o. Americka 23 12000 Praha 2 Czech Republic
2001:678:11::|48|CZ|50.0729504|14.4376125
           |CZ.NIC, z.s.p.o. Americka 23 12000 Praha 2 Czech Republic
```

Výstupem je vždy takový počet řádků z databázové tabulky, kolik je nalezeno shod se zadaným řetězcem v adresním řádku. Jinými slovy, jsou vypsány veškeré prefixy IPv6, jejichž adresní řádek obsahuje zadaný řetězec.

#### Reakce na nekorektní vstup

Reakce na nekorektní vstupy jsou identické jako u jiných typů dotazů, popsány jsou v podkapitole 6.1.1.

#### Vlastnosti a nedostatky

Protože máme vyhledávání jako fulltextové, při některých typech dotazů jako např. *Berlin*, jsou nalezeny i takové záznamy, které toto zadané slovo obsahují, např. *Berlinerstrasse* a další. Toto chování teoreticky není na škodu. Funkce fulltextového vyhledávání v adresách byla implementována zejména ze statistických důvodů, když chceme vědět, kolik prefixů IPv6 je registrovaných na určitém místě, nebo na sídlo nějaké firmy. Nutno ale dodat, že ne ve všech získaných adresních řádcích je jméno majitele uvedeno. Také zde narazíme na nejednotnost adres kvůli tomu, že některé záznamy obsahují ve svém adresním řádku např. *Praha*, zatímco jiné *Prague*. Proto při takovém hledání neobsáhneme všechny prefixy IPv6 skutečně lokalizované v Praze. Přesto je tato vesměs experimentální funkce zajímavá a může být užitečná pro rychlé vyhledání adresních bloků IPv6 v rámci nějaké lokality.

### 6.1.3 Geolokace podle kódu země

Zde si popíšeme testování statistické funkce, kdy vstup je kód země dle normy ISO [9] a výstupem je potom seznam všech IPv6 prefixů sítí, které jsou registrovány v dané zemi. Korektním vstupem je pouze dvoupísmenný kód, na ostatní vstupy program nereaguje, tedy na výstup nevypíše nic. Jako testovací vstup jsme zvolili kód státu *UG*, protože chceme zjistit lokalitu přidělených prefixů IPv6 pro Ugandu.

#### Reakce na korektní vstup

```
[xsucho00@qwe db]$ ./ip2geo -z UG
2001:43b8::|32|UG|0.3136111|32.5811111
           |Celtel House, 40 Wampewo Avenue,P.O.Box 8373, Kampala,Uganda
2001:43f8:130::|48|UG|0.3136111|32.5811111
           |Uganda Online,P.O. Box 12510,Kampala,Uganda
2c0f:fe10::|32|UG|0.3136111|32.5811111
           |MTN Uganda,22, Hannington Road,P.O. Box 24624,Kampala,Uganda
2c0f:fe70::|32|UG|0.3252777|32.5975193
           |Plot 16/17 Block Office 2,Nyonyi Gardens Kololo,Kampala 256,Uganda
2c0f:fec0::|32|UG|0.3136111|32.5811111
           |P.O Box 1624,Kampala,Uganda
2c0f:ff98::|32|UG|-1.2833333|36.8166667
           |Tangerine Limited,P O BOX 47859-00100,Nairobi,Kenya
2c0f:ffa0::|32|UG|-1.2796134|36.8161358
           |P O Box 17670,00100,Nairobi, Kenya
```

Výstupem je zde opět několik řádků z databáze, které obsahují informace o delegovaných prefixech IPv6. Za povšimnutí stojí dva poslední záznamy. Ačkoli jsou přiděleny pro kód státu *UG*, adresa uvedená v databázi Whois pod objektem `person` se nachází v hlavním městě Keni, Nairobi. Možnosti proč tomu tak je jsou v zásadě dvě. První z nich je špatné označení kódu země žadatele o prefix IPv6 už u registrátora (v tomto konkrétním případě AFRINIC), druhá je potom, že registrující entita má uvedenou adresu v jiné zemi. Druhá

možnost se dá ještě dále rozdělit na situaci, kdy má registrující subjekt na uvedené adrese pouze sídlo firmy a konektivita IPv6 je používána jinde, anebo na situaci, kdy právě poskytne do databáze Whois korektní lokalitu, kde se bude nacházet daná síť. Při zvoleném způsobu geolokace se už z principu vyskytují nepřesnosti, protože veškerá přesnost je určena údaji z databází Whois, kam je zadává žadatel. Pro téma přesnosti geolokace jsme vyhradili samostatnou podkapitolu.

#### Reakce na nekorektní vstup

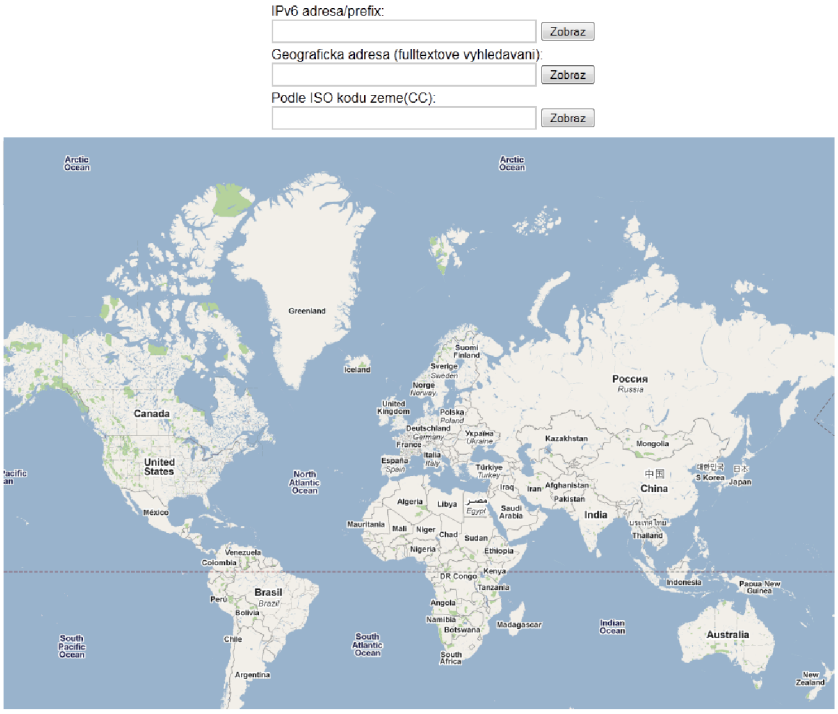
Reakce na nekorektní vstupy jsou identické jako u jiných typů dotazů, popsány jsou v podkapitole 6.1.1.

## 6.2 Testování webové aplikace

Tato podkapitola si klade za cíl demonstrovat činnost výsledné webové aplikace při použití geolokačního systému v pilotním provozu na reálném serveru, kam mají přístup libovolní uživatelé. Výstupy aplikace s vizualizací na mapě zde budou prezentovány ve formě obrázků.

### 6.2.1 Uživatelské rozhraní, použitelnost

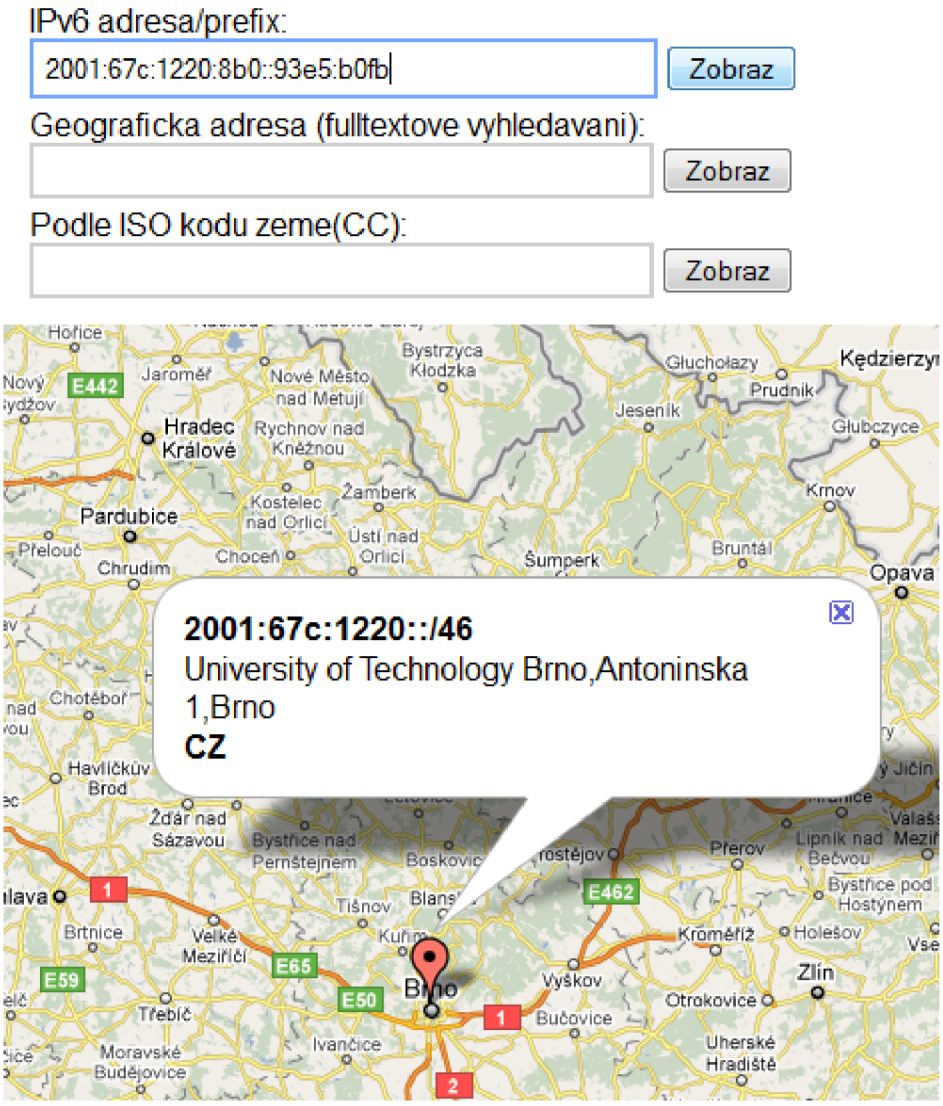
Při návrhu uživatelského rozhraní jsme se přiklonili k jednoduchému a intuitivnímu ovládání v podobě třech formulářů s potvrzovacími tlačítky. Zbytek okna prohlížeče je vyplněn mapou světa, do které jsou ihned zobrazovány výsledky hledání v geolokační databázi. Při testování byl kladen důraz na provozní použitelnost aplikace, a to jak pro administrátora, tak pro širokou veřejnost. Základní pohled při spuštění rozhraní webové aplikace je možno vidět na obrázku 6.1.




Obrázek 6.1: Uživatelské rozhraní geolokačního systému

### 6.2.2 Geolokace adresy IPv6 s vizualizací

V této podkapitole jsme otestovali schopnost webové aplikace reagovat na uživatelské vstupy a zabývali jsme se zejména úspěšnou lokalizací se zobrazením na mapu pro konkrétní adresu IPv6 nebo prefix sítě. Jako korektní testovací vstup jsme zvolili adresu IPv6 *2001:67c:1220:8b0::93e5:b0fb* stejně jako při testování konzolové aplikace, aby bylo možné porovnat reálný výstup při vizualizaci. Přitom víme, že tato adresa IPv6 patří serveru na půdě FIT VUT v Brně. Výstupem při tomto typu dotazu je mapa světa se zobrazením jedi-



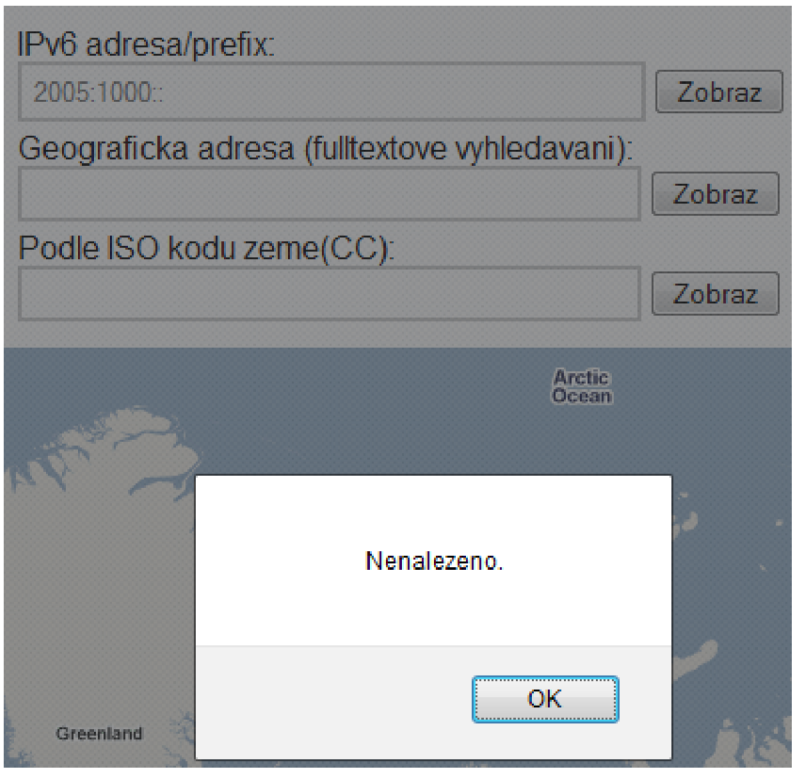

Obrázek 6.2: Vizualizace lokality adresy IPv6 2001:67c:1220:8b0::93e5:b0fb

ného bodu, optimálně přiblížená a vycentrovaná na zobrazovanou lokalitu, jak je popsáno v kapitole 5.3 a jak demonstruje obrázek 6.2. Zadaná IPv6 adresa je lokalizována relativně správně, na sídlo registrujícího subjektu, tedy na geografické adrese rektorátu VUT v Brně, na Antonínské ulici. Zároveň se zobrazením bodu je otevřeno informační okno, kde je možné nalézt podrobnosti o hledané IPv6 adrese, jako je její prefix sítě, kód země, ve které je tento

prefix zaregistrován, a samozřejmě i geografická adresa. Odchylka od reálného umístění serveru je cca 3km, což je hodnota velice nízká a akceptovatelná. Celý proces geolokace včetně vizualizace byl proveden v podstatě ihned, kdy výstup je zobrazen bez povšimnutelného zpoždění.

#### Vizualizace nekorektního vstupu

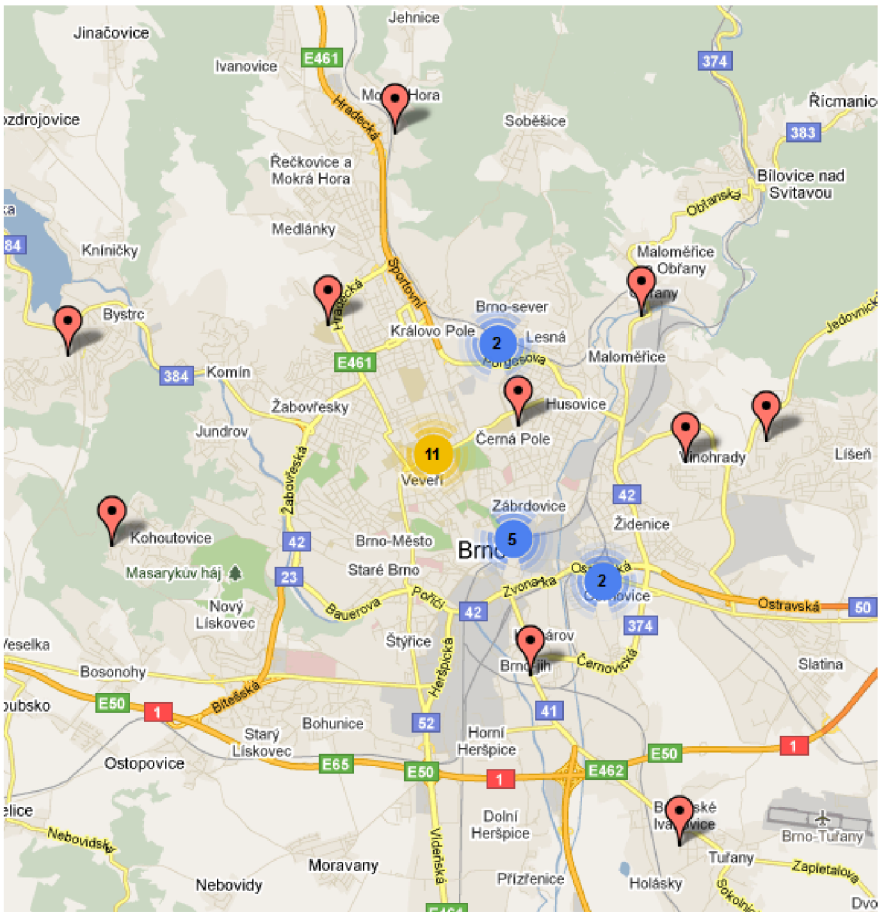
Také jsme testovali schopnost reagovat v rámci uživatelského rozhraní na vstup, který nelze nalézt v databázi nebo který je jiným způsobem nekorektně zadán. Pro testovací účely byl zvolen neexistující prefix IPv6 *2005:1000::*. V okně aplikace dojde k zatmavení obsahu, což zvýrazňuje na popředí umístěné okno s informací, že odpověď na hledaný dotaz v databázi není nalezena, viz obrázek 6.3.




Obrázek 6.3: Vizualizace nekorektního vstupu 2005:1000::

### 6.2.3 Geolokace s fulltextovým vyhledáváním

Tato podkapitola testuje druhou funkci geolokačního systému, a to vizualizaci lokalit, které se fulltextově vyhledávají v získaných adresních řádcích. Jak již bylo zmíněno, tato funkce je inovátorská a experimentální, přesto funguje relativně korektně a její využití vidíme především ve statistické funkci, kdy získáme zevrubný přehled o počtu registrovaných prefixů IPv6 v hledané lokalitě (městě, apod.). Testovaným řetězcem bylo *Brno*, výstup demonstruje obrázek 6.4. Jak vidíme z výstupu, v lokalitě Brna se nachází celkem 30 unikátních přidělených adresových bloků IPv6. Výstup do mapy je opět korektně vycentrován a přiblížen, aby obsáhl všechny relevantní lokality výskytu prefixů IPv6. Podklad mapy je




Obrázek 6.4: Vizualizace lokalit prefixů IPv6 v Brně

rozdělen na čtverce o určité ploše. Pokud je v takovém čtverci koncentrováno více bodů než jeden, dochází automaticky ke shlukování. Při kliknutí na jednotlivé červeně označené body se zobrazí jejich informační okénko, při kliknutí na shluk se mapa přiblíží na další úroveň a zobrazí přiblížený výřez čtverce reprezentovaného shlukem. Tento čtverec zase může obsahovat shluky i body. Při dosažení určité úrovně přiblížení se již shluky rozpadnou na samostatné body.

Co se týče přesnosti lokalizovaných prefixů IPv6 ve městě Brně, skutečně byly zobrazeny všechny takové, které jsou registrované u RIPE NCC a mají v jednom ze záznamů v databázi Whois uvedené slovo *Brno*. Samozřejmě při potřebě geolokovat konkrétní adresu sáhneme raději po předchozí funkci k tomu účelu určené, avšak z hlediska rychle proveditelné statistiky s grafickým výstupem přímo na mapu shledáváme funkci fulltextového vyhledávání v adresách velice užitečnou. Jedná se v podstatě o seznam sítí IPv6 nacházející se v určité oblasti, oproti konzolové verzi však graficky znázorněn na mapě včetně všech informací o každé registrované síti.

### 6.2.4 Geolokace podle kódu země

Poslední implementovanou funkcí s vizualizací přes uživatelské rozhraní je statistická funkce, která zobrazuje všechny prefixy IPv6 registrované v hledané zemi. Země se do příslušného políčka zadá jako dvoupísmenný kód ISO3166-*alpha2* [9]. Vstup je omezen pouze na dva

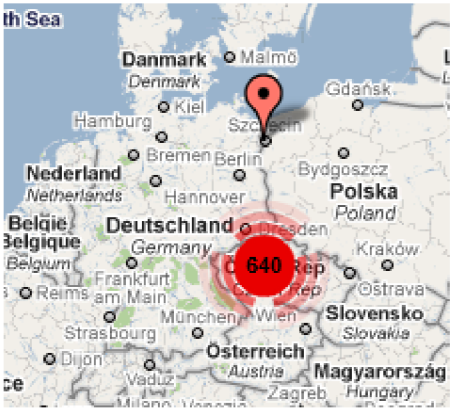
znaky, jakýkoli jiný vstup se bere jako nesprávně zadaný vstup, o čem je uživatel informován pomocí výstražného okna. Vstup může být zadaný jak malými písmeny, tak kapitálkami. Geolokační databáze přijímající kód země jako parametr si písmena zkonvertuje automaticky na kapitálky. Našimi testovacími daty byla Česká republika (kód *CZ*) a Spojené státy americké (kód US). Výstupem je vždy vizualizovaná množina bodů na mapě, jenž reprezentují přidělené prefixy sítí IPv6, které jsou registrovány na území dané země. Při velkém počtu výstupních záznamů jsou záznamy shlukovány. Funkce je užitečná zejména pro administrátory a jiné pověřené osoby, kteří chtějí rychle zjistit hustotu pokrytí nějaké země adresami IPv6. Díky tomu, že protokol IPv6 nyní zažívá prudký rozmach, je zajímavé sledovat, jak se v jednotlivých zemích vyvíjí hustota pokrytí adresními bloky IPv6 v čase.

Při velkém počtu zobrazených záznamů může proces vykreslování chvíli trvat. V takovém případě se uživateli na chvíli zobrazí rámeček s informací „Prosím strpení…“, přičemž zbytek celé aplikace se zatmaví. Takové chování vidíme na obrázku 6.5.

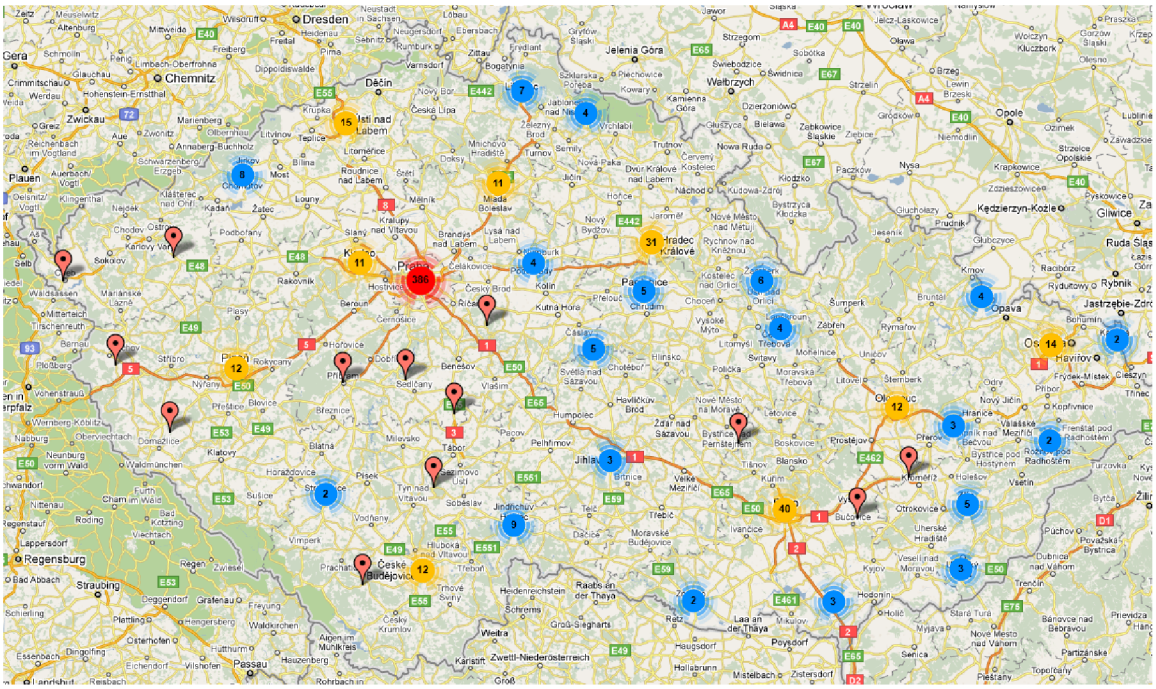
Obrázek 6.5: Čekání při vizualizaci velkého množství dat

Prvním výstupem po zadání korektního vstupu *CZ* je mapa, která obsahuje pohled tak, aby byly vidět všechny body registrované u RIPE NCC pod kódem *CZ*. Na obrázku 6.6 vidíme výřez z mapy, který zobrazuje, že v České republice se nachází 640 zaregistrovaných unikátních prefixů IPv6. Pohled na celou mapu následně odhalil, že osamocené prefixy IPv6 se vyskytují na pěti jiných místech než v České republice. To je způsobeno jednak tím, že byl nějaký prefix registrován s chybným kódem země, a jednak principiální nepřesností při tomto způsobu geolokace, kdy adresa může být zadána v podstatě jakákoli, takže firma s adresovým prostorem využívaným v ČR může uvést do databáze Whois svoje sídlo například v USA.

Po kliknutí na ikonu shluku v ČR se vystředí mapa tak, aby byly vidět veškeré body na území shluku. Na výstupu (jak vidíme na obrázku 6.7) se zobrazí mapa celé České republiky mapující hustotu pokrytí republiky adresními bloky IPv6. Lze zde jasně vidět, že ve větších městech a jejich okolí je bloků registrovaných nejvíce, přičemž absolutně dominantní v počtu sítí IPv6 je hlavní město Praha. Hustota shluků a bodů se mění v závislosti na poloze mapy




Obrázek 6.6: Vizualizace statistiky prefixů IPv6 v České republice bez přiblížení




Obrázek 6.7: Vizualizace lokalit všech prefixů IPv6 v České republice s přiblížením

a jejím přiblížení s důrazem na přehledné zobrazení.

## 6.3 Rychlost a efektivita uložení dat

Jedním z důležitých faktorů při testování funkčnosti celého geolokačního systému bylo měření rychlosti dotazů na databázi, odezvy webového rozhraní po zadání vstupu a rychlost aktualizace samotné databáze. Dále se tato kapitola zabývá hodnocením efektivity uložením dat v databázi, tedy aby data zabírala pokud možno co nejméně prostoru vzhledem k jejich množství.

### 6.3.1 Rychlost provedení dotazu do databáze

Rychlost zpracování dotazů do databáze jsme testovali na serveru, kde byla nasazena provozní verze kompletního geolokačního systému. Změřili jsme rychlost všech zmiňovaných tří typů dotazů, přičemž v geolokaci konkrétní IPv6 adresy dochází navíc k výpočtu a ověřování, do jakého nejdelšího síťového prefixu zadaná adresa patří, aby byl výsledkem korektní prefix s nejdelší bitovou shodou.

První test ukazuje rychlost dotazu do databáze, který porovnává zadanou IPv6 adresu s celou databází prefixů. Zjistí, pod který prefix adresa patří a na výstup vypíše příslušný řádek z databáze. Měření probíhalo nástrojem `time`, který je standardní součást unixových operačních systémů.

```
time ./ip2geo -q "2001:67c:1220:805::6af3:2ff"
real  0m0.036s
user  0m0.031s
sys   0m0.004s
```

Dotaz byl tedy zpracován v čase 0,03 sekundy. Nutno podotknout, že celá tabulka databáze obsahuje přes 60000 řádků dat s unikátními kombinacemi prefixů IPv6 a masky.

Druhý test ukazuje rychlost zpracování geolokačního dotazu podle kódu státu pro Německo, jehož výstupem je přes 7500 řádků z databáze.

```
time ./ip2geo -z "DE"
real  0m0.103s
user  0m0.087s
sys   0m0.009s
```

Tento test jsme zvolili, protože Německo obsahuje nejvíce unikátních prefixů IPv6 ze všech krajin v celé databázi. Z výsledku je patrné, že rychlost není nijak degradována ani velkým množstvím výstupů na dotaz, což měl test dokázat.

### 6.3.2 Rychlost vytvoření geolokační databáze

Třetím testem bylo změřit, jak dlouho trvá vytvořit kompletní geolokační databázi. Měření probíhalo sledováním cyklické činnosti vytváření databáze na serveru při pilotním provozu celého systému, což odpovídá reálnému prostředí při nasazení systému v praxi. Byla zaznamenávána data počátku a konce procesu. Průměrná doba zpracování databáze ze čtyř sledovaných cyklů bylo 6 dní a 20 hodin. Vezmeme-li v potaz, že máme omezení na pouze 2500 transformačních dotazů denně na API Google Maps a zároveň se nesmíme příliš rychle za sebou dotazovat na databázi Whois, při celkovém počtu 62150 získaných geolokačních záznamů byla průměrná rychlost tvoření databáze více než 8800 záznamů denně. Lze tedy říci, že optimalizace v rámci získávání dat byly úspěšné, protože jsme denně získali o cca 350% dat více, než kolik stanovuje limit u API Google Maps.

Samotné vytvoření binárního souboru s vlastní databází ze získaných dat potom trvá méně než 1 sekundu.

Vhodnou periodou pro aktualizací databáze je přibližně jeden měsíc, protože data v databázi se příliš nemění. Znamená to, že databáze je spíše konzervativního charakteru. Nabízí se tedy varianta přírůstkových aktualizací, ale vzhledem k charakteru přidělování adresního prostoru (adresové bloky IPv6 je možné měnit, přesouvat mezi zákazníky nebo přemapovávat na jiné země) jsme se při implementaci rozhodli od této varianty upustit a použít metodu

znovutvoření celé databáze od základů. Démon pro tvorbu databáze iteruje v nekonečné smyčce, takže ihned po dokončení tvorby databáze začíná tvořit novou verzi.

Ačkoli tvorba jedné verze databáze trvá přibližně jeden týden, což je relativně dlouhá doba, vzhledem k požadované aktuálnosti dat je tato rychlost naprosto dostačující. Pro ilustraci, průměrný přírůstek do každé nové verze geolokační databáze byl přibližně 500 nových unikátních zaregistrovaných prefixů týdně.

Do budoucna se počítá s postupným nárůstem adresových bloků IPv6, ale protože bloky jsou přidělovány zejména větším organizacím, poskytovatelům internetu a telekomunikačním společnostem, není očekáván extrémní nárůst unikátních prefixů IPv6, které by bylo nutné geolokovat. I kdyby počet registrovaných adresních bloků vzrostl na 250 000, databáze je schopna se v takovém případě vytvořit nanejvýš za jeden měsíc, což pořád lze považovat za rozumnou dobu s ohledem na aktuálnost dat v databázi. Při extrémně rychlém rozvoji protokolu IPv6 by bylo vhodné se zamyslet nad lepší formou aktualizace databáze, například s využitím časových razítek u záznamů v databázi a aktualizovat jen ty, u kterých by bylo datum novější.

### 6.3.3 Rychlost odezvy webového rozhraní

Čtvrtým a zároveň posledním testem rychlosti bylo změření odezvy komunikace mezi zadáním uživatelského vstupu do webového rozhraní geolokačního systému a vizualizací výsledku na mapě světa. Měřili jsme pomocí časových razítek v PHP, která byla zaznamenána dvakrát. Poprvé při počátku spouštění databázové aplikace a podruhé při ukončení funkce, která vytváří pole zobrazovaných bodů na mapu. Tato metoda měření není zcela přesná a nelze změřit zcela přesně, protože záleží na rychlosti klientského systému. Na tom totiž závisí rychlost zpracování výstupu přes JavaScript v okně klientského prohlížeče. Pro testovací účely jsme použili průměrně rychlý kancelářský počítač (procesor Intel Core2 Duo 2GHz, 2GB RAM).

Byly učiněny dva různé pokusy. Prvním je rychlost zobrazení jednoho bodu do mapy při geolokaci IPv6 adresy. Časová razítka jsme od sebe odečetli a získali jsme čas 0,181 sekundy.

Druhým pokusem bylo změřit odezvu při zobrazení všech prefixů IPv6 ve Spojených státech (kód země *US*), což čítá přibližně 2200 zobrazovaných bodů. Naměřili jsme čas 3,486 sekundy.

### 6.3.4 Efektivita uložení dat

Snahou této podkapitoly je zhodnotit efektivitu reprezentace dat v geolokační databázi. Aktuálně má databáze 62150 řádků v tabulce a její velikost na disku činí 6,7 MB. Z toho vyplývá, že průměrná velikost jednoho datového záznamu je přibližně 115 bajtů. Taková hodnota je v souladu s návrhem databáze, kde velikost všech sloupců kromě adresy činí 19 bajtů (8 B rozsah IPv6, 1 B maska, 4+4 B souřadnice, 2 B kód země). Prostorově nejvýraznější položku tak tvoří řetězec s geografickou adresou, který může nabývat hodnoty maximálně 200 znaků, tedy 200 B. Průměrná délka adresy v jednom řádku tabulky potom vychází na 96 bajtů. Velikost databáze je navíc o něco zvýšena kvůli indexům, které se nad databází automaticky vytvoří. Efektivní datovou reprezentaci tedy implementovaná databáze splňuje, a to zejména proto, že jsme vybrali vhodné datové typy pro adresy IPv6 a masku, jež zapisujeme v binárním tvaru.

## 6.4 Přesnost geolokačního systému

Další důležitou vlastností, která byla otestována, je přesnost námi implementované geolokační techniky. Na úvod je třeba poznamenat, že přesnost geografické lokalizace je poměrně obtížné charakterizovat. Co se týče nepřesností obecně, jsou dvě možnosti, jak geografickou nepřesnost chápat.

### 6.4.1 Rozdělení chyb v přesnosti geolokace

Jak jsme již zmínili v předchozí podkapitole, první je nepřesnost zadání údajů do databáze Whois, kterou nemůžeme ovlivnit. Jedná se o principiální chybu tohoto přístupu ke geolokaci. K těmto nepřesnostem řadíme i chybu tzv. „krátkých prefixů“, kde konkrétně pro Latinskou Ameriku je registrován například prefix s maskou 16 pro Brazílii, který se našemu systému podařilo lokalizovat do města Sao Paulo, přičemž tento prefix již není dále podrobně rozčleněn na delší, které jsou přiřazeny konkrétním zákazníkům či společnostem. Přesnost geolokace takových záznamů je potom vázána na důslednost členění registrátorů. Z našeho testování vyplynulo, že nejlepší členění prefixů IPv6 provádí RIPE NCC, potom také i ARIN a APNIC.

Druhou chybou v přesnosti je potom špatné vyhodnocení získaných adres z databází Whois. Tato chyba se v naprosté většině případů vztahuje k špatné transformaci adresy na souřadnice, kdy zejména v málo rozvinutých zemích či oblastech GeoCoding API Google Maps špatně vyhodnotí adresu nebo část adresy, a tím pádem nesprávně lokalizuje daný blok IPv6, který má v databázi Whois zadanou adresu korektně. Naším cílem bylo tuto druhou variantu nepřesnosti omezit na minimum přidáním několika funkcí úprav pro získaných adresní řádky. Z výsledků dosažených testováním zejména na adresách v Asii, Africe a malých státech Evropy (Albánie, Moldávie, Černá Hora, Rumunsko) lze konstatovat, že se nám to z větší části podařilo, avšak tyto typy chyb se nedají ošetřit systémově a úplně. Testování úspěšnosti úprav jsme prováděli vždy před a po přidání každé úpravy uvedené v kapitole 5.1.4. Velké zlepšení v geolokaci pramenilo také z postupné geolokace jednotlivých částí adresního řádku, ze kterého se opakovaně postupně odebírala jednotlivá slova, která potenciálně mohla znepřesňovat geolokaci.

### 6.4.2 Testování přesnosti při geolokaci bloků adres IPv6 v ČR

Zajímavým testem, ve kterém můžeme poukázat na různé typy chyb, bylo testování geolokace pro kód země *CZ*. Při zadání tohoto kódu České republiky se na mapě světa objevilo 667 záznamů o unikátních prefixech, přičemž 663 jich bylo v rámci hranic České republiky. Další čtyři body značící adresové bloky IPv6 byly lokalizovány na území jiných států. Prvním z nich je prefix *2a01:5f0:1021::/48*, který má adresu registrovanou v Polsku a skutečně se nachází v Polsku. Chybou v tomto případě je mylné označení státu v databázi Whois. Dalším ze stejnou chybou je *2001:41a8:a00::/48*. Tento prefix má registrovaný společnost Telecom Italia se sídlem v Římě. Třetí nepřesností je prefix *2a01:5f0:1012::/48*, který je zaregistrován u českého poskytovatele Coolhousing, s.r.o., ačkoli společnost uvedla jako kontaktní údaj sídlo ve státě Maryland, USA. Vzhledem k tomu, že se jedná o firmu registrující si konektivitu v Česku, je zcela evidentní nesprávné zadání, podobně jako u čtvrtého prefixu *2a02:26f0:2::/48*, který vlastní společnost Akamai. V tomto případě se pravděpodobně jedná o pobočku firmy sídlící v ČR, která opět uvedla jako kontaktní údaj sídlo hlavní společnosti ve státě Maine, USA.

Pokud přesnost vztáhneme konkrétně na ČR, zjistíme, že medián geolokační chyby se pohybuje rozhodně hluboko pod 50 km, což je relativně slušná schopnost lokalizace. Je samozřejmě možné, že pokud má poskytovatel pro všechny svoje přidělené prefixy sídlo na jedné jeho adrese, například v Praze 9, potom geolokace i těchto koncových prefixů je omezena na jeho sídlo. Na druhou stranu je ale vysoce pravděpodobné, že poskytovatel pokrývá oblast Prahy, takže chyba není nikterak závažná.

### 6.4.3 Charakteristika nepřesností v závislosti na regionu

Samozřejmě přesnost obecně kolísá v závislosti na státu nebo regionu, o který se stará nadnárodní registrátor. Nízkou chybu (průměrně pod 100 km) vykazují všechny státy pod RIPE NCC a ARIN. U AfriNIC se chyba pohybuje od cca 100 do 200 km. Pro Asii a Pacifik jsou chyby větší, ale záleží na lokalitě. Dobře spravované státy jako Čína, Japonsko, Austrálie, Nový Zéland mají chybu blížící se státům v Evropě, tj. pod 50 km. Zdaleka nejhorší přesnost geolokace vykazují státy spravované pod LACNIC, tzn. státy Latinské Ameriky. Tam je geolokační přesnost omezena pouze na město a navíc registrátor přiděluje pouze krátké prefixy, které nejsou dále rozčleněny.

## 6.5 Bezpečnost systému

Tato podkapitola pojednává o bezpečnosti implementovaného systému. Prvním bodem je privátnost používaných dat, druhým pak robustnost a odolnost celého systému při nasazení v reálném provozu.

### 6.5.1 Privátnost dat

Protože geolokace pracuje na principu dolování dat, je dobré si něco říci o způsobu jejich získávání. Veškerá data (adresy, popř. jména, jsou-li uvedena v rámci kontaktních údajů) jsou získávána z veřejně dostupných databází systému Whois, na který se může dotázat kdokoli. Tato data však podléhají určitým pravidlům, která musí každý uživatel databáze Whois ctít a respektovat. Jmenujme například rozesílání nevyžádané pošty na kontaktní e-mailové adresy, nebo vytváření databází telefonních čísel pro reklamní účely a další. Samozřejmě kontaktní údaje jako adresy registrujících subjektů lze považovat za citlivé informace (pojednání o morálních aspektech geolokace se nachází v kapitole 2), nicméně jejich získání za účelem vytvoření geolokační databáze není v rozporu s pravidly databáze Whois [7].

Geolokační systém tedy neposkytuje možnosti ke zneužití privátních dat, jediná možnost zneužití geolokační databáze tkví v systematickém získávání geografických (a tedy poštovních) adres registrovaných subjektů, na než potom potenciální útočník může rozesílat nevyžádanou poštu. V tomto případě je však pro útočníka mnohem výhodnější použít přímo databázi Whois, ve které kromě adres jsou i kontaktní telefonní čísla a e-mailové adresy.

### 6.5.2 Odolnost proti chybám

Tato sekce se věnuje odolnosti systému vůči chybám, popř. jejich nápravě. Systém nasazený na serveru je při získávání dat odolný proti krátkodobým výpadkům internetu nebo serverů Whois. Je toho docíleno proaktivní kontrolou návratové hodnoty při volání aplikace *wget*. Pokud je stav vyhodnocen jako chybný (vypršelo spojení, žádná odpověď, odmítnutí), pak systém automaticky čeká 50 sekund při dotazech na Whois a 200 sekund při dotazech na

GeoCoding API Google. Časová lhůta pro pokus o znovuspojení je nastavitelná ve zdrojovém souboru dolovacího skriptu. Po uplynutí této doby se znovu pokusí provést stažení odpovědi na dotaz na daný server. Skript tedy dokáže vyčkat, než se spojení obnoví, a pokračovat ve vytváření databáze. Nedojde ke znehodnocení databáze například zapsáním prázdných řádek, ani k pádu dolovacího skriptu. Při spuštění skriptu, který periodicky obnovuje databázi, je chybový výstup přesměrován do souboru *chyby*, odkud vlastník nebo správce geolokačního systému má možnost zjistit, co bylo příčinou chyby, a v případě neobnovitelné chyby poznat, na čem konkrétně a proč dolování přestalo fungovat. Dolování má tedy schopnost částečné automatické obnovy při výpadku například internetové konektivity nebo dotazovaných serverů, a také při přetížení. Nicméně například pokud systém ukončí dolovací skript, je nutno jeho činnost obnovit manuálně. Absolutní bezpečnost a neomylnost systému však nebylo účelem této diplomové práce, proto se při provozování systému počítá s občasnou manuální kontrolou korektnosti. Při nasazení do provozu systém však nejeví žádné známky chyb a pravidelně databázi úspěšně obnovuje již déle než jeden měsíc.

### 6.5.3 Odolnost proti útokům

V implementaci webového rozhraní byla brána v úvahu i bezpečnost při provozu systému. Proto jsou veškeré vstupy kontrolovány a ošetřovány k tomu určenýmu funkcemi jazyka PHP. Nehrozí tak napadení příkazové řádky útočníkem, ani útok typu SQL injection.

Co se týče konzolové aplikace, veškeré vstupy jsou při zadávání kontrolovány na správný formát, ale ne tak důkladně jako u webové aplikace. Je totiž pravděpodobné, že konzolovou aplikace bude používat administrátor lokálně, kde není velké riziko, zatímco webovou aplikaci spuštěnou na serveru může používat široká veřejnost.

## 6.6 Statistika dotazů

Tato část testování se zabývá vyhodnocením logování dotazů a přístupů uživatelů přes webové rozhraní při nasazení do provozu. Jde zde provedena statistika přístupů, kdy jsme zjistili, odkud (z jakých IP adres) se uživatelé nejčastěji ptají. Nejčastěji používané třídní rozsahy IP adres, které se v logu objeví, jsou přes Whois přeloženy a jsou vyneseny do tabulky. Neméně důležité je také, na co se uživatelé nejčastěji ptají. V rámci testování byla provedena analýza dotazů a její výsledek je opět vynesen do tabulky. Testování proběhlo v období od 20.4.2011 do 15.5.2011. Struktura logu je zobrazena na obrázku 6.8.

```
81.30.244.4:     z: CH
147.229.208.148: z: CZ
83.240.54.165:   q: 2a03:1600::/32
147.229.208.148: s: Zlin
83.240.22.42:    q: 2001:ae8:3::/48
89.103.130.79:   z: TW
89.103.130.79:   s: egypt
```

Obrázek 6.8: Ukázka části souboru *log* se statistikou dotazů

### 6.6.1 Vyhodnocení přístupu uživatelů

Ve výše uvedeném časovém období byly zaznamenávány přístupy uživatelů do systému přes webové rozhraní. Byla zaznamenána IP adresa uživatele při každém dotazu na systém. IP adresy byly na konci testování třídně klasifikovány a pět nejčastěji přistupujících rozsahů adres IP bylo vyneseno do tabulky 6.1.

| Prefix IPv4 | Počet dotazů | ISP | Geografická lokalita |
|---|---|---|---|
| 83.240.*.* | 141 | Netbox | Brno, ČR |
| 147.229.*.* | 128 | VUT Brno | Brno, ČR |
| 89.103.130.* | 101 | UPC Brno | Brno, ČR |
| 81.30.244.* | 83 | Ha-Vel | Ostrava, ČR |
| 91.127.*.* | 60 | Slovak Telecom | Bratislava, SK |

Tabulka 6.1: IP adresy s nejvyšší četností dotazů na geolokační systém

Jak vidíme z tabulky 6.1, nejčastěji přistupující rozsah IP adres byl registrován od poskytovatele Netbox v Brně. Druhý nejčastější logovaný adresní prostor byl z rozsahu registrovaného na VUT v Brně. Vyhodnocením této statistiky jsme dospěli k závěru, že nejčastěji byl používán geolokační systém zřejmě k účelům testování a vyhodnocení přesnosti geolokace autorem a účastníky testování, dále studenty a pracovníky VUT v Brně. Pro komplexnější statistiku by bylo třeba vyhodnotit log z dlouhodobě provozovaného systému, což v našem případě nebylo možné.

### 6.6.2 Vyhodnocení dotazů

Současně s IP adresami uživatelů byly zaznamenány i podrobnosti jednotlivých dotazů jako je typ a obsah dotazu. Nejčastější položky, které byly obsahem dotazů v rámci testovacího období, jsou zaznamenány do tabulky 6.3 společně s celkovou četností jejich výskytů v tabulce 6.2. Celkový počet dotazů na systém v testovacím období byl 705.

| Typ dotazu | Počet dotazů | Procento dotazů z celku |
|---|---|---|
| Geolokace dle kódu země ISO | 302 | 42,8% |
| Geolokace s hledáním dle části adresy | 262 | 37,2% |
| Geolokace dle IPv6 adresy | 141 | 20% |

Tabulka 6.2: Vyhodnocení četnosti dotazů podle typu dotazu

Z tabulek vidíme, že nejvyšší procento dotazů bylo zobrazení četnosti adresního prostoru IPv6 podle kódu země. Naopak nejméně uživatelů se dotazovalo na konkrétní adresu/prefix IPv6. To nám říká, že implementované statistické funkce jsou často využívány. Dále, nejvíce uživatelů (64) se dotazovalo na kód země *CZ*, zatímco 48 jich hledalo sítě IPv6 nacházející se v Brně. Opět jako u vyhodnocení přístupu uživatelů do systému zde platí, že pro relevantní statistické informace by bylo třeba vyhodnotit log při dlouhodobějším nasazení geolokačního systému.

| Typ dotazu | Obsah dotazu | Počet dotazů |
|---|---|---|
| Geolokace dle kódu země ISO | CZ | 62 |
| Geolokace s hledáním dle části adresy | Brno | 48 |
| Geolokace dle kódu země ISO | SK | 47 |
| Geolokace dle kódu země ISO | US | 24 |
| Geolokace dle IPv6 adresy | 2001:67c:1220:: | 16 |

Tabulka 6.3: Vyhodnocení nejčastějších dotazů na geolokační systém

## 6.7 Použití systému

Implementovaný geolokační systém má široké využití. Počínaje běžným uživatelem, který je s jeho pomocí schopen zjistit geografickou lokalitu libovolné internetové IPv6 adresy, přes vývojáře internetových aplikací využívající jeho aplikační rozhraní pro IPv6 geolokaci například pro cílenou reklamu nebo poskytování lokalitně závislých služeb, až po síťové administrátory, kteří dokáží sledovat, odkud pochází nebo kam je směrován provoz na síti. Systém má využití také u poskytovatelů internetového připojení, popřípadě národních registrátorů, kdy jsou schopni vizualizovat přidělené IPv6 prefixy jak v rámci libovolného státu, tak v rámci určité konkrétní lokality díky fulltextovému vyhledávání. Jistě zajímavou možností je potom použítí projektu pro sledování rozvoje protokolu IPv6 ve světě a vytváření statistik pro hustotu pokrytí IPv6 adres v jednotlivých regionech či kontinentech. Dále je možné geolokační systém využít pro účely výuky díky snadno pochopitelné vizualizaci. Společnost CZ.NIC v současnosti provozuje IPv6 geolokační systém na vlastním serveru http://staging.labs.nic.cz/ip2geo/.

## 6.8 Shrnutí

V této kapitole byla popsána metodika testování geolokačního systému a prezentovány výsledky testů. Zaměřili jsme se jak na testování konzolové, tak i webové aplikace. Testy byly doplněny vhodnými obrázky. Dále jsme se zaměřili na rychlost zpracování dotazů a rychlost zobrazení výsledku při použití webového rozhraní. S rychlostí souvisí i efektivita uložení dat v databázi, o níž jsme se také zmínili. Vyhodnotili jsme přesnost IPv6 geolokace a podrobněji jsme se zaměřili na příčiny i důsledky nepřesností. V závěru kapitoly jsme diskutovali bezpečnost a použitelnost celého systému a provedli jsme vyhodnocení statistiky dotazů na aplikaci, kdy cílem bylo zjistit kdo a na co se nejčastěji ptá. S výhledem do budoucna předpokládáme rozvoj protokolu IPv6, proto očekáváme, že počet registrovaných adresních bloků IPv6 bude i nadále růst. S postupným přechodem na IPv6 bude databáze dále růst. Současný přírůstek do databáze je asi 500 unikátních prefixů IPv6 týdně.

# Kapitola 7

# Závěr

Cílem této diplomové práce bylo navrhnout a vyvinout systém pro geolokaci adres IPv6, který umí vizualizovat geografickou lokalitu adres IPv6 na světové mapě. Systém disponuje nástrojem pro automatické vytváření geolokační databáze s pravidelnou aktualizací.

Součástí práce je také rešerše současných přístupů ke geolokaci, která poskytuje ucelený přehled geolokačních metod a principů, z nichž každý je podrobněji diskutován. Největší pozornost je potom věnována geolokačním technikám založeným na dotazování do databáze Whois, protože se jedná o přístup, který jsme se rozhodli použít pro návrh vlastního geolokačního systému.

Implementovaný systém se skládá z několika částí. Prvním z nich je nástroj pro získávání dat z veřejné databáze Whois, který současně transformuje získané adresy na zeměpisné souřadnice. Tento nástroj je realizován ve skriptovacím jazyce Perl za pomoci API Google Maps.

Další částí je databázový systém, který je implementován v jazyce C++ a využitím knihovny SQLite. Tato aplikace ze získaných dat dokáže vytvořit geolokační databázi a obsluhovat veškeré dotazy do této databáze. Při její implementaci jsme se zaměřili zejména na rychlost vyhledávání nad databází a efektivní uložení dat.

Poslední část systému tvoří webová aplikace, která poskytuje přehledné grafické uživatelské rozhraní pro práci s geolokační databází a která dokáže vizualizovat výsledky databázových dotazů na mapu. Aplikace je napsána v jazycích PHP a JavaScript s využitím komponent AJAX a jQuery, přičemž pro práci s mapou bylo použito API Google Maps.

Geolokační systém podporuje tři typy dotazů. Prvním je dotaz na konkrétní IPv6 adresu nebo prefix, jehož výstupem je vizualizace lokality na mapě. Druhým je fulltextové vyhledávání v adresách, které vykreslí do mapy množinu bodů, jejichž geografická adresa obsahuje daný výraz. Posledním typem dotazu je statistika hustoty pokrytí jednotlivých krajin bloky adres IPv6, jejímž výstupem je vizualizace lokalit všech bloků IPv6 registrovaných v zadané zemi.

Velice důležitou část tvoří testování celého systému. Zaměřili jsme se zejména na přesnost geografické lokalizace, rychlost systému, schopnost vyrovnat se s nekorektními vstupy a v neposlední řadě také na bezpečnost a použitelnost. Testováním bylo zjištěno, že přesnost geolokačního systému je na vysoké úrovni. Omezením přesnosti v tomto případě je dáno správností kontaktních údajů v databázi Whois a také kvalitním delegováním adresového prostoru IPv6 jednotlivými registrátory. Přesnost systému je mnohem vyšší v Evropě, Severní Americe a Austrálii, než v Africe a Jižní Americe.

Podařilo se nám tedy vytvořit zcela autonomní systém pro geolokaci IPv6, který disponuje, narozdíl od komerčních produktů, i automatickou vizualizací na mapě světa a použitel-

nými statistickými funkcemi a geolokací podle části adresy. Nasazení do pilotního provozu bylo po domluvě s konzultantem ze společnosti CZ.NIC provedeno dne 20.4.2011, přičemž demo webové aplikace je možné najít na http://qwe.fit.vutbr.cz/geolocation/. Projekt byl také představen začátkem května na jarní konferenci RIPE NCC v Amsterdamu http://ripe62.ripe.net/presentations/67-IPv6-20110503-OS-RIPE62.pdf. Společnost CZ.NIC v současnosti provozuje IPv6 geolokační systém na vlastním serveru http://staging.labs.nic.cz/ip2geo/.

Další vývoj projektu by mohl zahrnovat ještě větší optimalizace při získávání adres z databází Whois, a to zejména v méně rozvinutých zemích, kde by potom mohla být přesnost geolokace ještě o něco vyšší. Principiálně však není možné pomocí této (a ani žádné současně známé) techniky dosáhnout absolutní přesnosti geolokace.

# Literatura


[1] Akamai Edgescape: Distributed data-gathering database. [Online; navštíveno 21.11.2011].
URL http://www.akamai.com/html/technology/products/edgescape.html

[2] IP address Location: Geolocation to Identify Geographical Location. [Online; navštíveno 15.12.2010].
URL http://www.ip2location.com

[3] IP2Location: IP-Country-Region-City-Latitude-Longitude Database. [Online; navštíveno 22.11.2011].
URL http://www.ip2location.com/ip-country-region-city-latitude-longitude.aspx

[4] IPInfoDB: IP address geolocation database. [Online; navštíveno 23.11.2011].
URL http://ipinfodb.com/ip_database.php

[5] MaxMind: GeoIP Country® Database. [Online; navštíveno 22.11.2011].
URL http://www.maxmind.com/app/country

[6] MaxMind: Geolocation and Online Fraud Prevention. [Online; navštíveno 14.12.2010].
URL http://www.maxmind.com

[7] RIPE Database Terms and Conditions. [Online; navštíveno 20.4.2011].
URL http://www.ripe.net/data-tools/support/documentation/db-tc

[8] IPv6 Unicast Address Assignments. IANA, srpen 2008.
URL http://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xml

[9] International Organization for Standardization: ISO 3166-1-alpha-2 code elements. ISO, Únor 2011.

[10] Aoun, C.; Davies, E.: Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966, únor 2007.

[11] Arif, M. J.; Karunasekera, S.; Kulkarni, S.; aj.: Internet Host Geolocation Using Maximum Likelihood Estimation Technique. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, AINA '10*, ročník 24, Washington, DC, USA: IEEE Computer Society, 2010, ISBN 978-0-7695-4018-4, ISSN 1550-445X, s. 422–429.

[12] Bagnulo, M.: NAT64/DNS64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. Behave WG, září 2008.
URL http://tools.ietf.org/html/draft-bagnulo-behave-nat64-00

[13] Bukowy, M.; Snabb, J.: RIPE NCC Database Documentation Update To Support RIPE DB ver. 2.2.1. Leden 1999, [Online; navštíveno 22.12.2010].
URL http://www.ripe.net/ripe/docs/ripe-189/at_download/pdf

[14] Carpenter, B.: Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529, březen 1999.

[15] Carpenter, B.: Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, únor 2001.

[16] Damas, J. L. S.; Robachevsky, A.; Walker, D.: RIPE Whois Database Query Reference Manual. Říjen 2005, [Online; navštíveno 4.1.2011].
URL http://www.ripe.net/ripe/docs/ripe-358/at_download/pdf

[17] Davis, C.: A Means for Expressing Location Information in the Domain Name System. RFC 1876, leden 1996.

[18] Gleeson, B.: A Framework for IP based Virtual private networks. RFC 2764, únor 2000.

[19] Google: The Google Geocoding API: Google API Web Services - Google Code. Leden 2011, [Online; navštíveno 20.1.2011].
URL http://code.google.com/intl/cs/apis/maps/documentation/geocoding/#GeocodingRequests

[20] Gueye, B.; Ziviani, A.; Crovella, M.; aj.: Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions on Networking*, ročník 16, č. 6, prosinec 2006: s. 1219–1232, ISSN 1063-6692.

[21] Harrenstien, K.; Stahl, M.; Feinler, E.: NICNAME/WHOIS. RFC 954, říjen 1985.

[22] Huitema, C.: Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380, únor 2006.

[23] Huston, G.: IPv4 Address Pool Exhaustion Prognosis. Listopad 2010, [Online; navštíveno 13.11.2010].
URL http://www.potaroo.net/tools/ipv4/index.html

[24] Moore, D.: Where in the World is netgeo.caida.org? 2000, [Online; navštíveno 11.12.2010].
URL http://www.caida.org/publications/papers/2000/inet_netgeo

[25] Muir, J. A.; van Oorschot, P. C.: Internet geolocation: Evasion and counterevasion. *ACM Computing Surveys*, ročník 42, December 2009: s. 4:1–4:23, ISSN 0360-0300.

[26] Nordmark, E.: Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765, únor 2000.

[27] Padmanabhan, V. N.; Subramanian, L.: An investigation of geographic mapping techniques for internet hosts. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, New York, NY, USA: ACM, 2001, ISBN 1-58113-411-8, s. 173–185.

[28] Satrapa, P.: Přechod od IPv4 k IPv6. Listopad 2008, [Online; navštíveno 13.11.2010].
URL https://www.ipv6.cz/Co_je_IPv6

[29] Shavitt, Y.; Zilberman, N.: A Study of Geolocation Databases. *Computing Research Repository at Cornell University article library*, červenec 2010.
URL http://arxiv.org/abs/1005.5674

[30] Templin, F.: Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 5214, březen 2008.

# Příloha A

# Obsah CD

- /src/ - zdrojové soubory geolokačního systému
- /doc/projekt.pdf - text technické zprávy ve formátu PDF

# Příloha B

# Konfigurace systému

V této kapitole je uvedeno, jaké jsou požadavky pro nasazení geolokačního systému na reálný server při uvedení do provozu. Je zde detailně popsána konfigurace serveru, na kterém byl systém vyvíjen a testován.

**Požadavky na systém**

- OS - libovolný unixový systém (testováno na CentOS 5.6 + Debian)
- HW - testováno na serveru v následující konfiguraci:
  - Intel® Pentium® processor G6950 (2.8GHz, 3MB cache)
  - 4GB DDR3 s taktovací frekvencí 1600MHz
  - 500GB SATA2, 7200rpm

Spotřeba systémových prostředků při provozu vyplývající z testování je následující:

- maximálně 90 MB systémové paměti při 7-denní činnosti skriptu
- využití procesoru max. 0,1%
- celá instalace systému včetně databáze zabírá přibližně 8 MB, přičemž pro každou další databázi (zálohují se staré verze) je potřeba dalších cca 7 MB

**Požadavky na SW vybavení**

Na serveru je potřebné mít k provozu geolokačního systému nainstalované následující softwarové vybavení:

- Webový server Apache
- PHP ve verzi 5.3 a vyšší
- Perl ve verzi 5.8 a vyšší
- `whois` klient - obvykle standardní součástí unixových systému, na OS Fedora a jeho derivátech přítomen klient `jwhois`
- sqlite3 - databázový systém a knihovna pro C++, pro správnou funkčnost je potřeba nainstalovat následující balíky:

- sqlite3 (vlastní interpret SQL databáze)
- libsqlite3-dev (knihovna pro C++, nutná pro překlad zdrojového kódu databázové aplikace)

### Instalace systému

K instalaci systému na server a uvedení systému do provozu je nutné následující:

- Nainstalovat potřebné SW vybavení uvedeno v předchozím odstavci
- Základní konfigurace serveru Apache a PHP (porty, vypnout *safe_mode*)
- Zkopírovat zdrojové soubory systému na server, přičemž vznikne následující souborová hierarchie:
  - *datamine/* - složka, obsahuje skript pro dolování dat a pro automatickou tvorbu geolokační databáze s pravidelnými aktualizacemi
  - *db/* - složka, obsahuje databázovou knihovnu včetně aktuální verze databáze a souboru *Makefile*
  - index.php - zdrojový kód webové aplikace
  - jquery.min.js - knihovna pro podporu jQuery
  - markerclusterer.js - knihovna pro práci se shlukováním bodů na mapě
  - phpinfo.php - soubor s informacemi o nainstalované verzi PHP a jeho konfigurací
- Kompilace zdrojových souborů ve složce *db/* příkazem `make`
- Spuštění skriptu *dbdaemon.sh* ve složce *datamine* příkazem

  ```
  nohup dbdaemon.sh 2> chyby &
  ```

Co se týče varianty pouze konzolové aplikace bez webového rozhraní, potom není třeba instalovat Apache ani PHP. Při instalaci potom vynechat druhý krok, přičemž ostatní kroky zůstávají zachovány. Potom je možné provozovat geolokační systém lokálně bez webového rozhraní, přičemž jeho výstupem je geolokace v textové podobě jako výpis z databáze ve vhodném formátu.

### Nastavení oprávnění

Co se týče oprávnění při provozování systému s webovým rozhraním, je potřeba explicitně nastavit pouze právo pro přístup ke složce *db* skupině *others*. Tuto volbu provedeme pomocí příkazu `chmod o+rw db/`. Toto nastavení je potřebné z důvodu ukládání logu do této složky webovým serverem. Na vytvořený soubor *log* má totiž právo uživatel `apache`, který ho zároveň vytvořil a zapisuje do něj při veškerých dotazech uživatelů na webové rozhraní. Tak zajistíme funkčnost logování v systému.

# Příloha C

# Návod k použití

Tato kapitola popisuje, jak se geolokační systém používá z pohledu uživatele. Rozlišujeme použití rozhraní příkazové řádky a webového grafického rozhraní s vizualizací na mapu. Existují tři typy základní typy dotazů pro obě implementovaná rozhraní.

### Příkazová řádka

- `./ip2geo -q IPv6 adresa/prefix`
  geolokace konkrétní IPv6 adresy (konkrétní IPv6 adresa ve standardním tvaru a prefix ve tvaru končící :: bez masky)
- `./ip2geo -s "část adresy"`
  fulltextové vyhledávání v DB podle části adresy
- `./ip2geo -z "CZ"`
  vyhledávání v databázi podle kódu státu

Dále můžeme konzolové aplikaci zadat další parametry:

- `./ip2geo -c`
  vytvoření databáze ze souboru geolocation
- `./ip2geo -h`
  zobrazí nápovědu a popis ovládání

Filtrování výstupu (volitelné parametry):

- `-f`
  zobrazí pouze zeměpisné souřadnice pro hledný záznam
- `-a`
  zobrazí pouze adresní řádek pro hledaný záznam

### Webové rozhraní

Při použití webové aplikace uživatel systému zadává dotazy do okna příslušného formuláře podle typu dotazu. Pro geolokaci konkrétní IPv6 adresy nebo prefixu zadá dotaz do prvního formuláře, pro fulltextové vyhledávání v adresách geolokační databáze použije druhý formulář. Pro zobrazení všech adresových bloků v konkrétní krajině uživatel zadá kód dané krajiny do třetího formuláře. Každý formulář obsahuje popis jeho funkce. Demo geolokačního systému s webovým rozhraním je možné najít na http://qwe.fit.vutbr.cz/geolocation/.