IP Vizualizátor

Bc. Jakub Jančička

11. června 2019

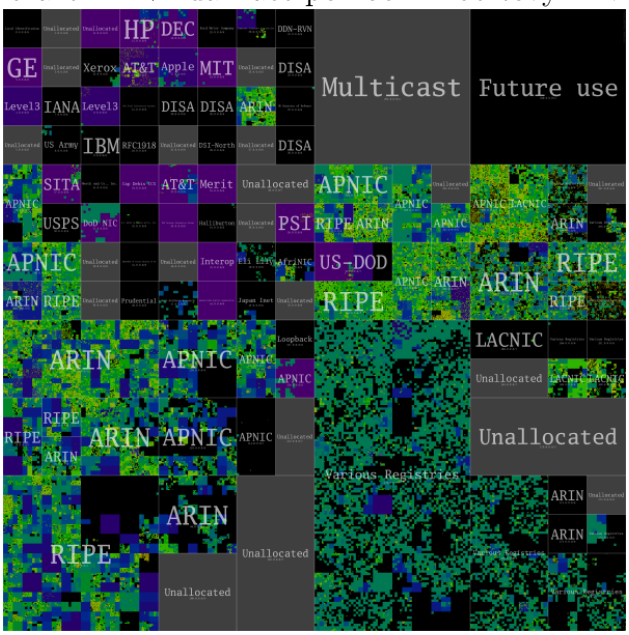
Diplomová práce se bude věnovat navržení přehledné interaktivní vizualizace informací o adresách z IP rozsahu. V rámci použití na Cesnetu se předpokládá zobrazování různé škodlivé aktivity pocházející ze síťových entit z jednotlivých rozsahů.

Jako nejpřehlednější zobrazení se mi zatím vizualizace IPv4 rozsahu pomocí prostor vyplňujících křivek, které řeší lineární pořadí průchodu vícerozměrným prostorem. Známá je například používaná Hilbertova křivka, která se hodí pro zobrazení IPv4 prostoru, protože shlukuje blízké IP adresy a rozsahy (viz obrázek 1). Zajímavou alternativu k Hilbertově křivce mi poskytnula prezentace *Fun with IPv4 Heatmaps* [1], která namítá, že v síti můžeme shlukovat pouze takové rozsahy, které mají stejné číslo sítě (síťový prefix). To Hilbertova křivka ne zcela respektuje – občas shlukuje bloky vertikálně, občas horizontálně. V prezentaci navrhují použít Mortonovu Z-křivku, která umožnuje shlukovat vždy následující blok (viz obrazek 2). Pravděpodobně tedy použiji pro vizualizaci Mortonovu křivku.

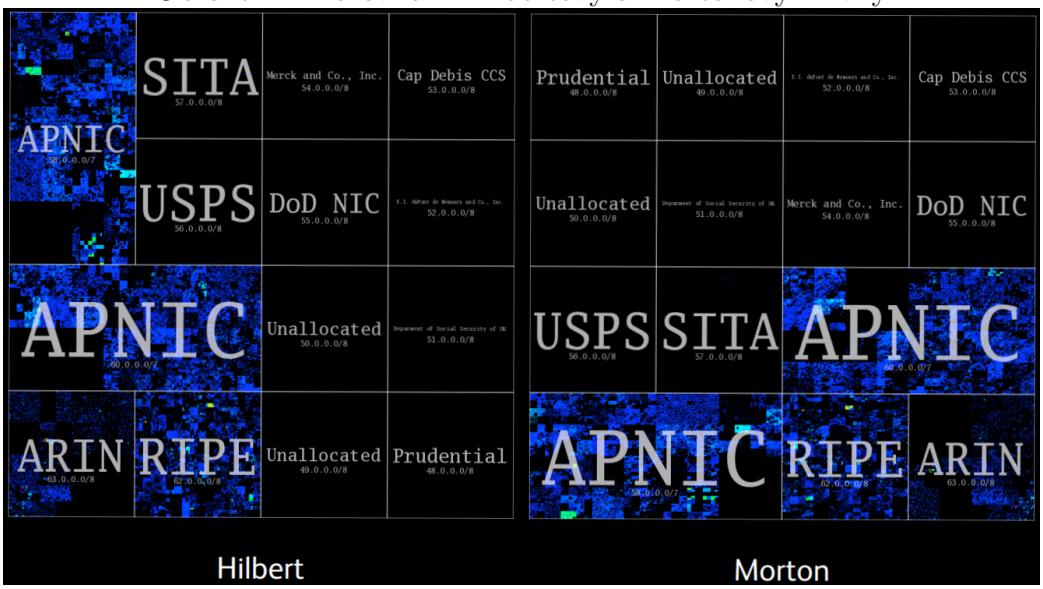
Samotná realizace předpokládá backendovou část aplikace (psanou v jazyce Python), která se bude starat o samotné zpracování příchozích hlášení o závadných síťových entitách pro potřeby daných vizualizací, a frontendovou část (psanou v JavaScriptu), která bude na klientské stanici vytvářet interaktivní vizualizátor, jenž bude umožňovat zoomování na jednotlivé IP rozsahy a zobrazování konkrétnějších informací o IP adresách v daném rozsahu. Tímto se bude realizace lišit od běžně dostupných vizualizátorů, jejichž výstupem je pouze obrázek. Drobným problémem je velký datový objem vizualizátoru – při zhruba 4 miliardách adres v IPv4 nelze klientovi poslat na zobrazení všechna data. Bude tedy nutné používat asynchronní volání

[1] http://www.iepg.org/2007-12-ietf70/3dheatmaps.pdf

Obrázek 1: Vizualizace pomocí Hilbertovy křivky



Obrázek 2: Porovnání Hilbertovy a Mortonovy křivky



na backend (např. s využitím AJAX), kdy se klientovi budou posílat pouze relevantní data pro danou hloubku zanoření (zoom) a zobrazené IP rozsahy.

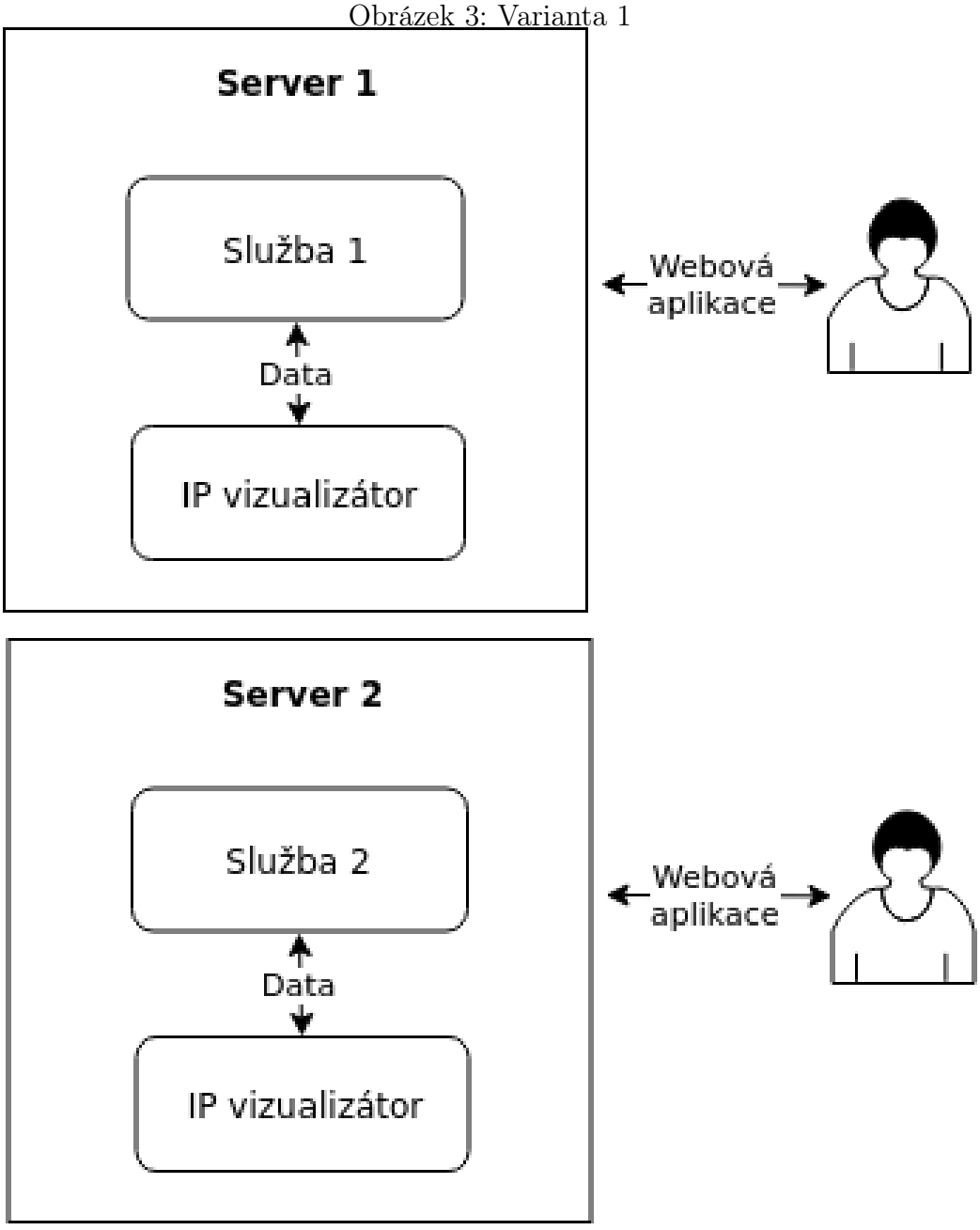
Co se týká nasazení aplikace, tak se zatím rozhoduji mezi třemi variantami. První (na obr. 3), kdy backend, který zpracovává vizualizaci, běží na jednom stroji spolu s danou službou, která chce data vizualizovat (např. s NERDem). Výhodou je, že se zpracovávají hlášení přijatá službou na daném stroji, nedochází k dalšímu posílání po síti. Nevýhodou je, že náročné zpracování vizualizace může vytěžovat daný stroj a tím ovlivňovat i chod služby. Další nevýhodou je horší možnost aktualizace aplikace, která by v tomto případě běžela v mnoha instancích na různých strojích.

Druhou variantou (na obr. 4) je běh backendu na jednom dedikovaném stroji, který by se staral o zpracování vizualizace na základě dat poslaných z dané služby. Daná služba by buď posílala pravidelně zachycené hlášení o škodlivém provozu nebo by odeslala už předzpracovaná data (např. NERD by poslal některé informace, které má uložené v databázi k daným entitám). Aplikace by data zpracovala a vrátila by výsledek, který by frontend na dané službě zobrazil. Výhodou je, že backend běží na jednom stroji, nezpomaluje ostatní služby a aplikace se dá jednoduše aktualizovat. Nevýhodou je, že dochází k přenášení většího množství dat po síti směrem od služby k aplikaci a zpátky. Předpokládám, že u této varianty by byla i časová prodleva mezi tím, než aplikace data zpracuje a než je zobrazí u klienta.

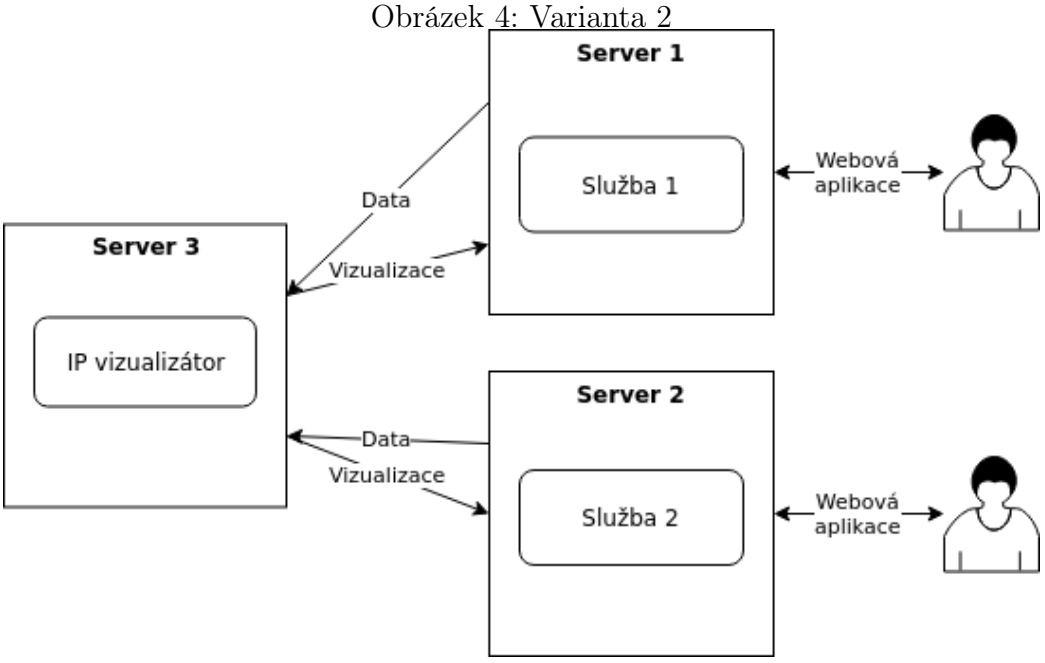
Poslední uvažovaná varianta (na obr. 5) je podobná předchozí, tedy aplikace by běžela na dedikovaném serveru a sama by byla napojena na nějaký zdroj dat (např. NERD). Službě, která by chtěla vizualizátor používat, by stačilo na stránku přidat javascriptovou knihovnu a pomocí API specifikovat, jak má daná vizualizace vypadat (za jaké období, jaké konkrétní typy škodlivých entit se mají vizualizovat atd.). Aplikace na základě dotazu vrátí výslednou vizualizaci (tato varianta by fungovala na podobném principu, na jakém fungují např. vložené Google mapy na webových stránkách). Tato varianta má výhody předchozí, nedochází k přenášení tolika dat a také aplikace může udržovat na základě dat už vhodně předpřipravené výsledky, a tedy by nenastávala delší časová prodleva mezi posláním dotazu na aplikaci a vrácením výsledku. Nevýhodou může být, že služba nemůže jednoduše aplikaci poslat vlastní data, která by chtěla zpracovat.

Zatím jako nejlepší varianty ke zpracování vidím variantu 1 – služba se specifickými daty by si aplikaci mohla nasadit u sebe a daná data vizualizovat a variantu 3 – pokud službě stačí zobrazit již sbíraná data jinou službou (např. NERDem), stačilo by poslat dotaz na server a dostala by výsledek.

Obrázek 3: Varianta 1



Obrázek 4: Varianta 2



Obrázek 5: Varianta 3