FIT VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Statický NAT64 překladač

Projekt ISA

2011 Jan Dušek

Obsah

1 Úvod

Cílem tohoto projektu bylo implementovat statickou variantu NAT64, jež je jedním z mechanismů přechodu z IPv4 na IPv6 a zajišťuje překlad paketů jedné verze protokolu na druhou a tím umožňuje komunikaci mezi klienty podporující rozdílné verze IP protokolu.

Projekt je řešen v jazyce C++ jako konzolová aplikace, která dostane jako parametry dvě síťová rozhraní, na kterých v nekonečném cyklu naslouchá a přicházející pakety transformuje a odesílá druhým rozhraním dále do sítě.

2 Analýza

2.1 Uvedení do problematiky

Při vytváření nové verze IP protokolu IPv6 bylo jasné, že bude třeba zpřístupnit uživatelům nové verze protokolu, služby podporující pouze starou verzi protokolu. V RFC 2766[3] byl pro tyto účely definován NAT-PT, který ale byl pro svou nutnost manipulace s DNS záznamy v RFC 4966[4] zavržen. Později byl koncept vzkříšen jako NAT64.

2.2 Transformace paketů

NAT64 je sítový prvek umístěný na rozhraní IPv4 a IPv6 sítě, který transformuje přicházející pakety z IPv4 na IPv6 a naopak. Přesný postup této transformace je definován v RFC 6145[1].

Tato transformace je, až na adresy, celkem jednoduchá, jednotlivé položky v hlavičce se na základě jednoduchých pravidel přetransformují na odpovídající položky transformovaného paketu.

2.3 Překlad adres

Překlad IPv4 adres na IPv6 adresy je v zásadě jednoduchý, jelikož lze využít mnohonásobně většího adresového prostoru jež poskytuje IPv6, takže pro IPv4 konvertované adresy byl vyhrazen takzvaný Well-Known Prefix *64:ff9b::/96,* definován v RFC 6052[2].

Překlad z IPv6 adres na IPv4 adresy je ale komplikován mnohem menším adresovým prostorem IPv4. Tento problém se řeší dvěma způsoby, tzv. statickým nebo dynamickým překladem, oba tyto postupy mají své výhody a nevýhody.

2.3.1 Dynamický překlad

V anglické literatuře označován jako *Stateful* je překlad kdy si překladač uchovává dynamickou tabulku spojení s adresami a porty. Tento mechanizmus funguje podobně jako stávající NAT na IPv4 a je podrobně popsán v RFC 6146[5].

Jelikož je v zadání uveden statický překladač nebudu se dynamickým podrobně zabývat.

2.3.2 Statický překlad

V anglické literatuře označován jako *Stateless* je překlad kdy se IPv6 zdrojové a IPv4 cílové adresy překládají algoritmicky na základě předchozí konfigurace překladače.

Nevýhodou je, že klientům v IPv6 síti musí být přiřazena také IPv4 adresa, které byli ovšem již na celosvětové úrovni vyčerpány a je jich tedy nedostatek. Nespornou výhodou je ovšem fakt, že komunikaci lze navázat z obou stran, lze tak poskytovat služby v IPv6 síti pro celý IPv4 internet.

2.3.3 Výsledný postup

Jak bylo řečeno v zadání je zvolen statický překlad, který je jednodušší na implementaci. Konfigurace se provádí souborem pravidel, kde jsou uvedeny odpovídající dvojce IPv6 a IPv4 adres.

Při transformaci IPv6 na IPv4 adres se zdrojová adresa převede pomocí pravidla ze souboru pravidel. Jelikož IPv6 cílová adresa je IPv4-konvertovaná adresa získá se IPv4 adresa pouhým odstraněním Well-known prefixu.

Transformace IPv4 na IPv6 je obdobná. Zdrojová adresa se převede přidáním Well-known prefixu a cílová adresa použitím pravidla ze souboru pravidel.

2.4 Nesené protokoly

Jelikož překladač má podporovat protokoly TCP a UDP je nutno provést další korekce. Hlavičky těchto protokolů obsahují kontrolní součet, který se počítá z vybraných položek hlavičky, ale také z IP pseudo hlavičky, která obsahuje IP adresy, číslo protokolu a délku paketu neseného protokolu (např. TCP). Transformace ale změní podobu IP adres je tedy nutné kontrolní součty přepočítat.

3 Implementace

3.1 Knihovny

Pro vytvoření programu jsem použil dvě knihovny *libpcap* a *libnet*. Obě knihovny jsou zadáním povoleny.

Knihovna *libpcap* slouží pro zachytávání paketů na zadaných síťových rozhraních. Příchozí pakety dovede také filtrovat, používá syntaxi stejnou jako program *tcpdump*. Dle zadání je jedno rozhraní připojeno do IPv4 sítě a druhé do IPv6 sítě, použil jsem filtrování, abych zachytával pouze pakety které je třeba transformovat.

Problém vznikl při nastavení *libpcap* aby zachytával pouze příchozí pakety. Knihovna obsahuje funkci *pcap_setdirection*, která ovšem na testovacím stroji nefunguje, jelikož obsahuje starou verzi knihovny, místo této funkce jsem použil filtr *not ether src MAC*, který odfiltruje pakety se zdrojovou MAC adresou stejnou jako je MAC adresa síťového rozhraní.

Knihovnu *libnet* jsem použil pro vytváření a posílání transformovaných paketů.

3.2 Architektura

Program je napsán v C++ a využívá OOP. Základními třídami jsou: *Sniffer*, který je obálkou nad *libpcap* a zachytává pakety. *Sender*, který je obálkou nad *libnet* a stará se o posílání paketů. *SnifferService*, což je třída, u které se zaregistrují sniffery a ona je pak obsluhuje, když přijde paket třída zavolá uživatelem definovaný callback. *Nat*, který se stará o transformaci paketů mezi verzemi IP protokolu, zároveň je mozkem tj. vytváří sniffery, registruje je u služby atp.

3.3 Naslouchání na více rozhraních

Knihovna *libpcap* neumožňuje jednoduše naslouchat na více rozhraních zároveň bylo třeba použít takový malý trik, *libpcap* používá socket, je tedy možné získat file descriptor tohoto socketu pomocí *pcap_fileno*. *SnifferService* obsahuje základní nekonečný cyklus, který se spouští její metodou *start*, tento cyklus používá systémové volání *select* na file descriptorech registrovaných snifferů a tak je možno pohodlně obsluhovat několik snifferů naráz.

Alternativou by bylo pro každý sniffer vytvořit separátní vlákna a v něm použít funkci *pcap_loop*, myslím ale že je zvolená cesta vhodnější, jelikož odpadá práce s více vlákny a pro rozumný počet snifferů je tato cesta jednodušší a efektivnější.

4 Použití

Program se spouští jako

```
nat64 -i <INTERFACE> -o <INTERFACE> -r <RULES>
```

-i INTERFACE je rozhraní napojené na IPv6 síť
-o INTERFACE je rozhraní napojené na IPv4 síť
-r RULES je soubor s pravidly pro mapování IPv6 adres na IPv4 adresy

4.1 Pravidla

Soubor s pravidly obsahuje na každém řádku právě jednu dvojci IPv6 adresa a IPv4 adresa oddělené bílými znaky.

5 Testování

Projekt byl testován pomocí virtuálního počítače, kde běžel NAT a dvěma virtuálními síťovými rozhraními na hostitelském počítači, které sloužili jako IPv4 klient a IPv6 klient pro počítač s natem. Testování probíhalo pomocí programů *sendip*, *netcat* a webového serveru *Apache*.

Reference

[1] *IP/ICMP Translation Algorithm* [online]. [cit. 27. listopadu 2011]. Dostupné na: <http://tools.ietf.org/html/rfc6145>.

[2] *IPv6 Addressing of IPv4/IPv6 Translators* [online]. [cit. 27. listopadu 2011]. Dostupné na: <http://tools.ietf.org/html/rfc6052>.

[3] *Network Address Translation - Protocol Translation (NAT-PT)* [online]. [cit. 27. listopadu 2011]. Dostupné na: <http://tools.ietf.org/html/rfc2766>.

[4] *Reasons to Move the Network Address Translator - Protocol Translator(NAT-PT) to Historic Status* [online]. [cit. 27. listopadu 2011]. Dostupné na: <http://tools.ietf.org/html/rfc4966>.

[5] *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers* [online]. [cit. 27. listopadu 2011]. Dostupné na: <http://tools.ietf.org/html/rfc6146>.