# Přechodové mechanismy k IPv6 (2)

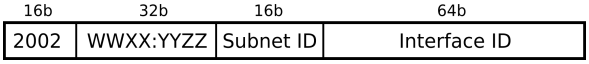
*David Rohleder, ÚVT MU*

V minulém díle jsme si popsali základní principy přechodových mechanismů mezi IPv4 a IPv6 a popsali jsme protokol ISATAP (používaný hlavně v organizacích). V dnešním díle se zaměříme na zbývající dva nejrozšířenější mechanismy, a to 6to4 a Teredo.

## 1 6to4

Technologie 6to4 byla definována v RFC 3056 jako jeden z automatických mechanismů spojování IPv6 ostrůvků přes IPv4 síť. 6to4 používá celý IPv4 internet jako jednu spojovací síť.

6to4 používá pro své adresování IPv6 adres z rozsahu 2002::/16 a zbytek IPv6 adresy se pak vytváří z přidělené IPv4 adresy.




6to4 rozlišuje tři základní druhy uzlů zapojených do tohoto mechanismu:

- *6to4 host* – zařízení ve vnitřní síti, které získá 6to4 IPv6 adresu ze směrovače prostřednictvím standardních mechanismů autokonfigurace (nebo jiným způsobem – DHCPv6, staticky).
- *6to4 router* – uzel s veřejnou IPv4 adresou, kterému může být přímo vytvořena 6to4 IPv6 adresa, a který je připojen k IPv4 internetu.
- *6to4 relay* – uzel, který je jedním rozhraním připojen do IPv4 internetu, druhým do IPv6 internetu a obstarává komunikaci mezi IPv4 internetem a IPv6 internetem.

### 1.1 Přidělení IPv6 adres

Každý IPv6 ostrůvek dostane svůj IPv6 prefix sestavený následovně: 2002:WWXX:YYZZ::/48, kde WWXX:YYZZ jsou čtyři bajty přidělené IPv4 adresy. Prefix /48 je standardizovaný prefix přidělovaný koncovým zákazníkům v IPv6. To umožňuje mít ve vnitřní síti až 65535 IPv6 podsítí, a tedy dostatečnou kapacitu pro adresaci všech vnitřních počítačů. Tam je možné použít standardní mechanismy pro přidělování IPv6 adres, včetně adresy směrovače. Tj. použijeme buď statickou konfiguraci, bezestavovou automatickou konfiguraci SLAAC nebo DHCPv6.

Pokud přidělujeme 6to4 adresu jedinému počítači (6to4 host), pak je možné adresu konstruovat libovolným způsobem odpovídajícím danému prefixu. Např. 2002:WWXX:YYZZ::1. Operační systémy z rodiny Microsoft Windows konstruují adresy jako 2002:WWXX:YYZZ::WWXX:YYZZ.

### 1.2 Směrování

IPv6 adresy máme tedy přiděleny, co nám zbývá, je vyřešit směrování mezi jednotlivými počítači. To je překvapivě jednoduché. Máme tři možnosti podle toho, kde se koncový počítač nachází:

1. Pokud se koncový počítač nachází podle IPv6 adresy ve stejné síti, pak se použije klasické směrování v IPv6.

2. V případě, že nenastal případ 1 a adresa cílového počítače začíná prefixem 2002::/16, znamená to, že koncový počítač leží v jiném IPv6 ostrůvku. Ostrůvky jsou spojené prostřednictvím IPv4 sítě, přičemž z konstrukce 6to4 adresy známe koncovou IPv4 adresu 6to4 routeru/uzlu (ony WWXX:YYZZ). Vezmeme tedy IPv6 paket a zabalíme ho do IPv4 paketu s cílovou adresou WW.XX.YY.ZZ a pak pošleme standardní IPv4 sítí, která ostrůvky propojuje. Cílový 6to4 router/uzel pak IPv6 paket vybalí z IPv4 obálky a pošle do vnitřní IPv6 sítě standardním směrováním.

3. Poslední varianta je ta, že cílový počítač neleží ani v místní síti ani v 6to4 ostrůvku, tj. leží ve standardní IPv6 síti. V této chvíli využijeme tzv. 6to4 relay, který je jedním rozhraním připojen do IPv4 internetu a druhým rozhraním do IPv6 internetu. Zdrojový počítač vyšle IPv6 paket vnitřní sítí k defaultnímu směrovači, který IPv6 paket zabalí do IPv4 paketu a pošle jej na IPv4 adresu 6to4 relaye. 6to4 relay vybalí IPv6 paket a pošle jej standardním mechanismem do IPv6 internetu.

Pro poslední variantu zbývá vyřešit pouze dvě drobnosti. Tou první je, jak zjistit IPv4 adresu 6to4 relaye. K tomu máme tři možnosti:

- IPv4 adresu 6to4 relaye známe a staticky ji nakonfigurujeme do 6to4 routeru/uzlu;
- použijeme DNS. Microsoft windows používá k nalezení 6to4 relaye jméno 6to4.ipv6.microsoft.com;
- použijeme tzv. anycastovou IPv4 adresu. Anycast sice nebyl původně v IPv4 definován, nicméně funguje stejně jako v IPv6. Tj. uzlů s jednou IPv4 adresou může být v Internetu několik a směrovače pošlou paket vždy k tomu nejbližšímu. Pro účely 6to4 tunelu se používá adresa 192.88.99.1.

Druhou překážku představuje cesta z IPv6 internetu zpět k 6to4 relayi. Zpáteční cesta je zajištěna tak, že každá 6to4 relay oznamuje do IPv6 internetu, že se za ním nachází síť 2002::/16. Zpáteční paket do 6to4 ostrůvku si pak vybere nejbližší 6to4 relay automaticky. Tento způsob řešení může mimo jiné způsobit asymetrickou cestu, kdy odesílající 6to4 relay bude jiná než 6to4 relay přijímající zpáteční paket.

Mechanismus 6to4 je tedy relativně jednoduché řešení, které by vždy fungovalo nebýt NAT na straně poskytovatele připojení. Pokud totiž nemá 6to4 router/uzel veřejnou IPv4 adresu, není možné zkonstruovat 6to4 IPv6 adresu tak, aby se zpáteční paket vrátil k původnímu odesilateli. Tento problém řeší až následující protokol.

## 2 Teredo

V dnešní době je Internet kvůli nedostatku IPv4 adres zaplevelen různými druhy NATu, které za jednou IPv4 adresou schovávají celé velké sítě. Internet byl přitom původně založen na principu přímé konektivity mezi koncovými uzly, čemuž NAT úspěšně brání. Většina dříve popsaných přechodových mechanismů nebude na počítačích za NATem fungovat (nebo pouze za zvláštních podmínek).

Aby bylo možné používat IPv6 i v takové situaci (ve které se nachází velká část domácích uživatelů, minimálně v ČR), byl vyvinut protokol Teredo, který si s NATem dokáže většinou poradit.

Před tím, než se pustíme do popisu fungování samotného Teredo mechanismu, musíme se trochu seznámit s fungováním technologie NAT.

### 2.1 NAT

NAT (Network Address Translation) byla původně technologie pro přepisování IPv4 adres v IPv4 paketech. Sloužila hlavně jako dočasné řešení při přečíslování sítě, aby se adresy nemusely měnit na všech počítačích vnitřní sítě hned, ale mohla se využít nějaká dočasná doba, kdy počítače fungovaly v nové síti i se starými adresami. NAT byl provizorní řešení, protože některé protokoly kvůli tomu nefungovaly (FTP). Překlad byl nestavový, a kvůli tomu jej bylo možné provádět vždy pouze v poměru 1:1.

Později vyvstala potřeba překládat adresy ve větším poměru než 1:1, aby se dosáhlo ušetření adresového prostoru. Pro tyto účely byl NAT rozšířen i o použití hlaviček čtvrté vrstvy (tj. např. UDP a TCP). Tímto vznikla čtveřice [src_IP, src_port, dst_IP, dst_port], a díky přemapování zdrojového portu bylo možné detekovat, ke kterému odchozímu paketu patří vracející se pakety. Tento princip se správně nazývá NAPT (Network Address and Port Translation), nicméně dnes se poměrně běžně zaměňuje za NAT, a NAT je takto v novém významu i vnímán. Pro tento způsob NATu je nutné udržovat stavovou tabulku překladů, což může v některých případech (málo výkonný hardware, příliš provozu) způsobovat problémy.

Použití informací z vyšších vrstev hlaviček paketů v NATu představuje pro standardní IPv6 tunelovací mechanismy problém, protože NAT obvykle pracuje pouze s protokoly TCP a UDP, jiným protokolům většina NAT směrovačů nerozumí. Ostatní IPv6 přechodové mechanismy totiž nepoužívají TCP ani UDP a pro zapouzdření IPv6 paketu do IPv4 paketu používají protokol číslo 41, ve kterém není nic, co by se dalo označit za číslo portu.

Teredo tedy zapouzdřuje IPv6 paket do UDP paketu, který snadněji projde NATem.

Pro přesnější členění různých druhů NATu rozlišujeme následující typy (podle RFC 3489):

- *cone NAT* (trychtýřovitý NAT) – vytvoří vazbu mezi vnitřní IP adresou a portem a vnější adresou a portem. Všechny pakety, které pak přicházejí na vnější adresu a port zvenku, jsou přeposílány na vnitřní adresu a vnitřní port

(tj. je možné komunikovat z libovolné vnější adresy na správný vnitřní port vnitřního počítače, pokud před tím byla zevnitř vytvořena vazba vnitřní adresy a portu na vnější adresu a port. Tato vazba se vytvoří vysláním paketu směrem zevnitř ven).
- *restricted NAT* – podobná situace jako cone NAT, pouze s tím omezením, že s vnitřní adresou a portem může komunikovat pouze vnější počítač, na který bylo navázáno toto spojení zevnitř, nezávisle na čísle portu.
- *symetrický NAT* – mapuje vnitřní IP a port na různé vnější IP a porty, s Teredo mechanismem funguje pouze v případě, že za takovým NATem je maximálně jeden z komunikujících partnerů.

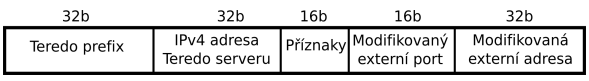
### 2.2 Součásti Teredo mechanismu

Teredo rozlišuje tři součásti nutné ke správnému fungování:

- Teredo klient – koncový uzel podporující Teredo mechanismus;
- Teredo server – uzel připojený jedním rozhraním do IPv4 internetu a druhým rozhraním do IPv6 internetu. Teredo server slouží k inicializaci spojení mezi Teredo klienty nebo Teredo klientem a IPv6 uzlem;
- Teredo relay – uzel, který dokáže ukončovat Teredo tunely a směrovat pakety mezi Teredo klienty na IPv4 internetu a IPv6 uzly na IPv6 internetu.

### 2.3 Teredo adresace

IPv6 adresa Teredo klienta vypadá následovně:




Teredo prefix: aby bylo možné rozeznat, že se jedná o protokol Teredo, je pro tento protokol vyhrazen prefix 2001:0000::/32.

Teredo server IPv4 adresa: adresa Teredo serveru, který spolupracoval na vytvoření Teredo adresy na klientovi.

Příznaky: náhodná data sloužící jako ochrana proti některým druhům útoků.

Modifikovaný externí port: je upravené číslo externího portu, na který se mapuje vnitřní IPv4 adresa. Tento port se systém dozví až po první komunikaci s Teredo serverem, protože mezitím prošel IPv4 paket NATem, který původní vnitřní port změnil.

Modifikovaná externí adresa: upravená externí IPv4 adresa, na kterou se mapuje vnitřní IPv4 adresa. Tuto adresu se klient dozví až po první komunikaci s Teredo serverem.

Jak si Teredo klient zjistí svoji IPv6 Teredo adresu? Celkem překvapivě jednoduše pomocí téměř standardního IPv6 ohlášení směrovače. Klient vytvoří IPv6 paket obsahující zprávu Router Solicitation, zabalí ji do UDP IPv4 paketu s cílovou adresou Teredo serveru. Tento paket projde přes NAT, dorazí na Teredo server, a ten odpoví upravenou RA zprávou, která v sobě obsahuje informaci o externí adrese a portu klienta za NATem. Klient pak tyto informace vezme a vytvoří si z nich podle výše uvedeného návodu svoji vlastní IPv6 Teredo adresu.

Tímto postupem si Teredo klient vytvořil také patřičný záznam v NAT tabulkách, takže je možné, aby Teredo server komunikoval s Teredo klientem (už může posílat pakety dovnitř, což se nám bude za chvíli hodit).

Mechanismus Teredo používá techniku, které se říká „NAT punching“ – proděravění NATu. Komunikace mezi Teredo klientem A a Teredo klientem B vypadá potom následovně:

1. Klient A, který chce komunikovat s klientem B, vyšle Teredo paket na externí adresu NATu klienta B (tuto IPv4 adresu odvodí z IPv6 adresy klienta B). Tím také vytvoří v NAT tabulkách díru, díky které bude možné v budoucnosti přijímat pakety od klienta B. NAT zařízení před klientem B tento paket zahodí, protože v jeho NAT tabulkách neexistuje žádný záznam, který by odpovídal danému paketu. Ale to nevadí, protože následuje:
2. Klient A vyšle paket (říká se mu bublina) Teredo serveru klienta B (IPv4 adresu tohoto serveru opět získá z IPv6 adresy klienta B). Teredo server klienta B už může komunikovat s klientem B díky již existující vazbě v NAT tabulkách zařízení klienta B.

3. Teredo server B předá tuto bublinu klientovi B. Klient B se tedy právě dozvěděl, že s ním chce komunikovat klient A.
4. Klient B vyšle bublinu přímo směrem ke klientovi A (jeho externí IPv4 adresu a port odvodí z jeho IPv6 adresy). Tím se vytvoří na NAT zařízení B záznam pro komunikaci od klienta B ke klientovi A.
5. Protože v prvním kroku jsme vytvořili na NAT zařízení A záznam pro komunikaci s klientem B, tak NAT zařízení A propustí paket až ke klientovi A. Tímto je komunikace sestavena a klienti A a B si nyní mohou vyměňovat pakety navzájem.

Komunikace mezi Teredo klientem A a IPv6 only klientem C probíhá odlišně:

1. Teredo klient A vyšle ping přes Teredo server ke klientovi C.
2. Teredo server přepošle bublinu klientovi C.
3. Klient C odpoví, paket dojde na Teredo relay nejbližší ke klientovi C.
4. Teredo relay vyšle bublinu pro klienta A přes Teredo server klienta A (Teredo relay zatím nemůže komunikovat s klientem A napřímo, Teredo server už ale ano). Odpověď na ping paket zatím relay zařadí do fronty na vyřízení.
5. Teredo server přepošle bublinu klientovi A.
6. Klient A z bubliny zjistí IPv4 adresu Teredo relaye pro klienta C (relaye jsou pro různé IPv6 počítače různé). Klient A vyšle bublinu směrem k Teredo relayi a vyrobí tím tedy díru do NATu.
7. Teredo relay dostane bublinu, vezme odpověď na ping z fronty a pošle ji na IPv4 adresu NAT zařízení klienta A. NAT už má vytvořenou vazbu a paket přepošle. Tímto byla komunikace mezi A a C ustanovena a uzly tak mohou komunikovat mezi sebou.

### 2.4 Bezpečnostní rizika

O NATu se často říká, že je jistá forma bezpečnosti, protože na počítače není možné navazovat spojení z vnější sítě. Jak jsme si předvedli v případě mechanismu Teredo, není to pravda. Komunikace dokonce může probíhat mezi dvěma uzly, které oba jsou za NATem.

Obecně všechny přechodové mechanismy mohou představovat jisté bezpečnostní riziko, protože se často tvoří samy automaticky, bez nutnosti zásahu administrátora. Firewally nejsou většinou na takovou tunelovanou komunikaci připraveny a může se stát, že propustí pakety, které by v IPv4 zahodily.

O nebezpečnosti tohoto mechanismu hovoří už samotný název. Původně se totiž tento mechanismus jmenoval Shipworm, nicméně byl přejmenován na Teredo, protože worm (červ) nemá v počítačových sítích zrovna dobré konotace. Autoři vtipně přejmenovali mechanismus na Teredo, což je latinský název pro anglický shipworm, v češtině *sášeň lodní* (drobný mlž dělající díry do dřevěných lodí). Je vidět, že i autoři suchých technických specifikací mají smysl pro humor.

## Literatura

[1] Toredo overview. `http://www.microsoft.com/technet/network/ipv6/teredo.mspx` □