Univerzitní eduroam pouze na IPv6: mechanismy DNS64 a NAT64

root.cz/clanky/univerzitni-eduroam-pouze-na-ipv6-mechanismy-dns64-a-nat64

Jan Vondráček

Autor: Depositphotos

Na Univerzitě Pardubice jsme se rozhodli nasadit IPv6 only síť na eduroamu. Získané zkušenosti nyní sdílím, protože mi při práci chyběl komplexní návod, který by spojil jednotlivé služby do jednoho řešení.

Letos jsem měl přednášku na Dni IPv6 o našem pokusu na Univerzitě Pardubice nasadit na eduroamu pro klienty IPv6 only síť. Náš pokus sice nedopadl úspěšně a museli jsme zůstat u dual-stacku, ale i tak mi celá ta práce přinesla užitečné zkušenosti.
V těchto článcích vám ukážu, jak jsem celé řešení konfiguroval. Protože první věc, na kterou jsem narazil, byla neexistence nějakého komplexního návodu.

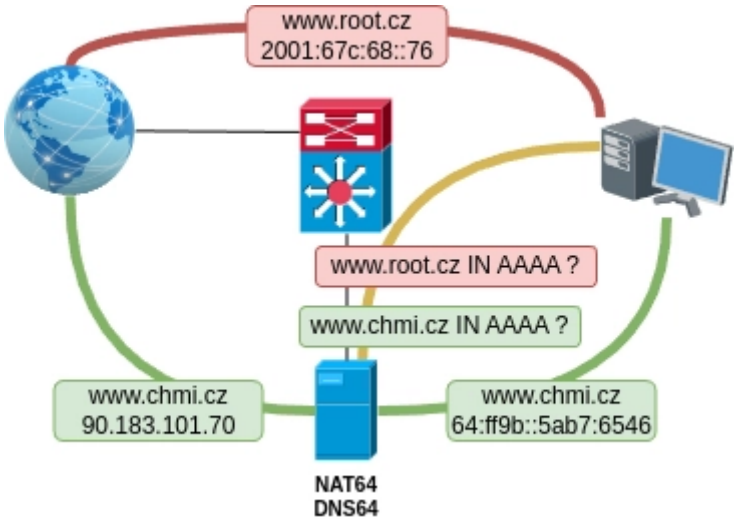
Ano jednotlivé služby sice mají pěkné manuály, ale nikdo to nepopsal dohromady jako **jedno řešení**. Trochu mi to připomíná situaci s internetovými poskytovateli, chceš pořádný internet – staň se lokálním providerem.

Pokud chcete provozovat IPv6 only síť, musíte v současné době stále zajistit dostupnost služeb, které jsou provozovány jen na **IPv4**. To nám umožňuje technologie NAT64 s DNS64.

Přechodový mechanizmus NAT64

Začnu krátkým přehledem co to vlastně ten přechodový mechanizmus NAT64 a DNS64 je.

Princip je ukázaný na obrázku. Máme počítač v IPv6 only síti. K síti patří server rekurzivní dns resolverem, který dělá DNS64. Na serveru také běží NAT64. Počítač přistupuje na dvě webové stránky:



1. DNS dotaz na stránky www.root.cz, jsou k dispozici po IPv6, spojení z počítače je rovnou navázáno přímo na server a stránky se zobrazí,
2. DNS dotaz na stránky www.chmi.cz, stránky mají jen A záznam, nejsou k dispozici po IPv6.

Tady do toho vstupuje finta jménem **DNS64**. Když tohle resolver zjistí, vrátí klientovi modifikovanou adresu. Vezme pool adres 64:ff9b::/96 a na jeho konec přidá IPv4 adresu a pošle zpět klientovi jako odpověď. Klient pak normálně naváže spojení po IPv6 jako v předchozím případě.

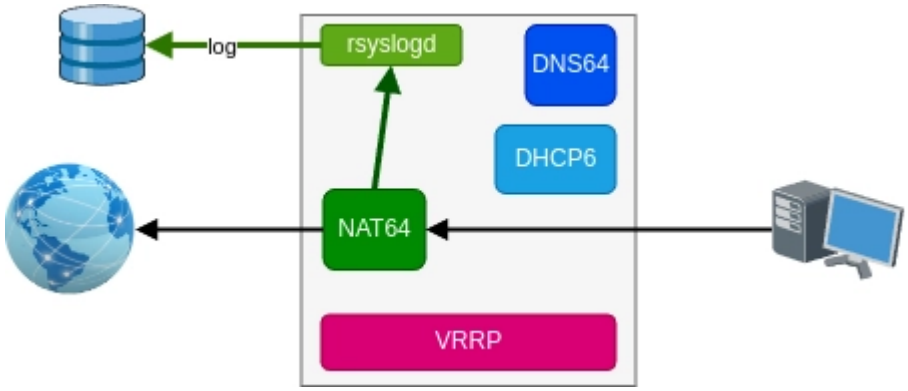
Na klientovi se toho víc neděje. Zato v síti dále ano. Pool 64:ff9b::/96 je jeden z vyhrazených rozsahů pro překladové mechanizmy. Stejně jako privátní pool, který se nepoužívá na internetu, ale jen pro překlady, podobně jako privátní rozsah IPv4 10.0.0.0/8. Překladový pool je na routeru nasměrován na **NAT64 server**, který provede překlad na IPv4 a tak odbaví provoz. Výraz překlad není rozhodně přesný, používám ho jen jako analogii kvůli snazšímu pochopení.

Zde ještě stojí za zmínku **DNSSEC**. Pokud resolver provádí validaci, vrací tyto modifikované odpovědi jako **validní**. Však jsou, on si je před odesláním odpovědi validoval. Ale problém nastane, když paranoidní klient bude chtít validovat také. Modifikovaná adresa mu

validací neprojde. Přechodový mechanizmus NAT64 je krásný a funkční, ale má svá omezení. Stejně jako to, že takto překládat lze jen protokoly TCP, UDP a ICMP.

Servery pro NAT64

Na obrázku je pohled dovnitř serveru, na jeho služby. Pro NAT64 použijeme Jool a pro DNS64 Bind9. Dále si ještě musíme rozchodit další pomocné služby: DHCPv6, použijeme Kea. Především z důvodu lepší kompatibility s klienty je třeba mít v síti jak stavovou, tak bezstavovou konfiguraci IP adres. Jako VRRP HA nasadíme démona keepalived. Pak musíme také řešit logování.



Poznámka

Když jsem hledal, jaké řešení na NAT64 použiji, rovnou jsem zamítl hardwarové řešení společnosti Cisco jako moc drahé. Použiji linuxový server s nějakým překladovým řešením. Při hledání jsem vycházel z přednášky kolegy Jana Pokorného, který prezentoval měření výkonnosti NAT64 řešení na jednom z minulých dní IPv6. Bohužel jeho přednáška už není k dispozici, měl tam moc pěkné obrázky na vysvětlení principu. Ale jeho seminární práce stále k dispozici je.

Na překlad jsem použil servery s CPU AMD EPYC 7313P, s 32GB RAM, 10GB síťovými kartami a Debian linuxem. Servery jsou dva, ve dvou lokalitách a běží jako HA v módu primary-backup. Překladový software je Jool z mexického NICu. Je pro mě v podstatě jediný použitelný, protože je součástí Debianu a Tayga je už zastaralá. Ono

vlastně na výběr moc není. Existuje sice ještě Tundra od Víta Labudy, která je výkonnější než Jool, ale já řešení s vlastní kompilací do provozu nechci, nezvládal bych to udržovat.

Konfigurace sítě

Router pro síť je Cisco VSS, fyzické boxy a servery jsou v různých lokalitách. Tím pádem nám stačí připojení serveru jedním rozhraním. Samozřejmě je možné použít zapojení se dvěma rozhraními. Pozor na routy pro překladové pooly IPv4 a IPv6, vše routujte na VRRP IP adresy. Adresace a routování jsou v obrázku.

/etc/network/interfaces

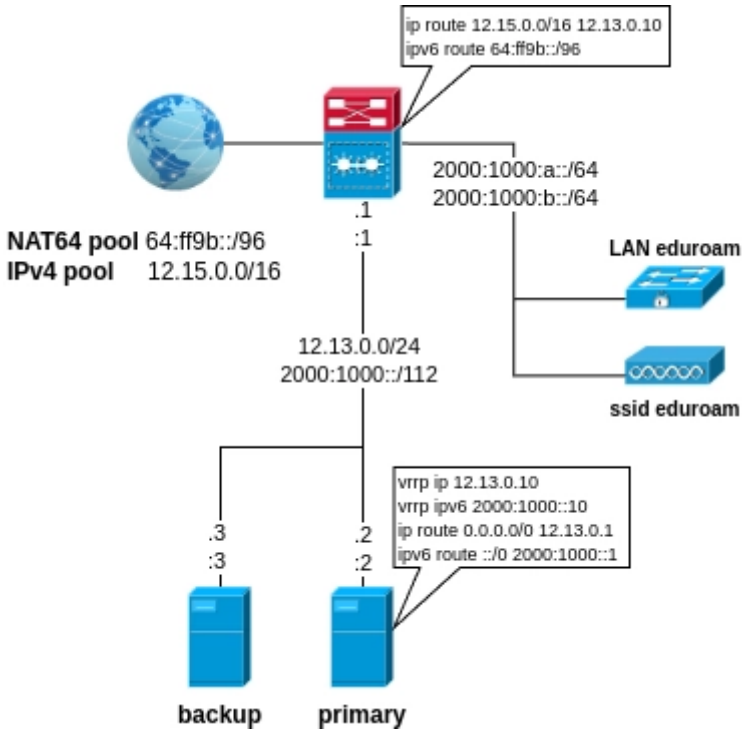
```
allow-hotplug eth0
iface eth0 inet static
  address 12.13.0.2/24
  gateway 12.13.0.1
iface eth0 inet6 static
  address
2000:1000::2/112
  gateway 2000:1000::1
```

/etc/sysctl.conf

```
net.ipv4.conf.all.forwar

net.ipv6.conf.all.forwar

net.ipv6.conf.default.au
```



Serverům přidělíme IP adresy a povolíme routování pro oba protokoly. U IPv6 na serverech je lepší vypnout podporu autokonfigurace, je to kvůli bezpečnosti. K adresaci samozřejmě lze použít network-manager, pokud ho preferujete.

/etc/nftables.conf

```
chain input {
  type filter hook input priority 0; policy drop;
  iifname "lo" accept
  ct state invalid drop
  icmp type { echo-request, time-exceeded } limit rate
50/second accept
  icmpv6 type { echo-request, nd-neighbor-solicit, nd-neighbor-
advert } limit rate 50/second accept
  tcp dport { 22 } ip saddr { x.x.x.x } accept #management
serveru
  ip saddr { 12.13.0.3 } ip daddr { 224.0.0.18 } accept #vrrp
  ip6 saddr { 2000:1000::3 } ip6 daddr { ff02::12, ff08::64 }
#vrrp a jool synchronizace
  ct state { established, related } accept
}
```

Lokální firewall není nutný, ale je bezpečnější ho použít.

VRRP

Je to technologie, která umožňuje **sdílet** jednu IP adresu více fyzickým zařízením. VRRP adresa je aktivní jen na primárním zařízení, ostatní jsou v záloze pro případ výpadku. Velice často se VRRP používá, aby v koncové síti pro výchozí bránu nebyl jen jeden router (na IPv6 to ale jde řešit také pomocí oznámení směrovače). Na Linuxu se VRRP realizuje pomocí démona keepalived. Má velmi mnoho různých schopností v oblasti HA, ale my ho použijeme na přehazování HA adres.

/etc/keepalived/keepalived.conf

```
vrrp_instance ipv6 {
  interface eth0
  state MASTER
  priority 200
  virtual_router_id 6
  unicast_src_ip 2000:1000::2
  use_vmac
  virtual_ipaddress {
    2000:1000::10/112
    }
  notify_master /etc/keepalived/master.sh
  notify_backup /etc/keepalived/backup.sh
  }

vrrp_instance ipv4 {
  interface eth0
  state MASTER
  priority 200
  virtual_router_id 4
  unicast_src_ip 12.13.0.2
  use_vmac
  virtual_ipaddress {
    12.13.0.10/24
    }
  }
```

Pro každý protokol musíme mít vlastní instanci, ta nám podle *virtual_router_id* vytvoří vlastní subinterface. Zde je ukázaná konfigurace primárního nodu, záložní má ve *state* nastaveno BACKUP a v prioritě nižší číslo, třeba 100. Záložních nodů může být více než jeden.

```
~# ip address
.....
9: vrrp.4@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:5e:00:01:04 brd ff:ff:ff:ff:ff:ff
    inet 12.13.0.2/24 scope global vrrp.4
       valid_lft forever preferred_lft forever
10: vrrp.6@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue state UP group default qlen 1000
    link/ether 00:00:5e:00:02:06 brd ff:ff:ff:ff:ff:ff
    inet6 2000:1000::2/112 scope global nodad
       valid_lft forever preferred_lft forever
```

Důležitá je položka *use_vmac*. Způsobí, že HA IP adresa nesdílí MAC adresu fyzického rozhraní, ale vytvoří si vlastní. To je užitečné, protože při převodu provozu mezi servery se v tabulce sousedů nemění MAC adresa HA IP adresy a převod provozu je tak rychlejší.

Master nod posílá multicastové zprávy hlásící: jsem tady a pracuji. Ve chvíli, kdy zprávy přestanou chodit, master pravděpodobně umřel a backup s nejvyšší prioritou se stává novým masterem. Nakonfiguruje si HA ip adresy a začne vysílat zprávy: jsem tady a pracuji. MAC zůstane stejná jako byla na původním masteru.

Unicast_src_ip je adresa, která se používá jako zdrojová pro multicastové zprávy, nezapomeňte ji povolit ve firewallu. Keepalived sice sám nějak lokální firewall konfiguruje, ale já na tyto automatické konfigurace nespoléhám a bohužel jsem nepřišel na způsob, jak je vypnout.

Ve chvíli, kdy se mění stav nodu z MASTER na BACKUP a obráceně, spouští se notifikační skript. To může být velmi užitečné a využil jsem to při konfiguraci DHCP démona Kea, ale o tom až později.

NAT64

Jool je prostředek na **překládání provozu** mezi světem IPv4 a IPv6. Umožňuje realizovat spoustu různých scénářů, statické překlady IPv4 na IPv6 a podobně, ale na to se podívejte v jejich moc pěkné dokumentaci. Nás bude zajímat NAT64, schovat naši IPv6 only síť za pool IPv4 adres.

```
apt-get install jool-dkms jool-tools
```

Balíček Jool obsahuje jaderný modul, distribuovaný pomocí Dynamic Kernel Module Support, a pomocné utility. Při instalaci si apt sám zkompiluje nový modul a funguje i s automatickými aktualizacemi. Modul se automaticky zavede při spuštění služby jool.

Jool se nastavuje buď jen z řádky nebo se konfigurace načte ze souboru ve formátu JSON a spustí se jako služba *jool.service*. Přes řádku se získávají informace o překladové tabulce *jool session display*.

/etc/jool/jool.conf

```
{
  "instance": "default",
  "framework": "netfilter",
  "global": {
    "pool6": "64:ff9b::/96",
    "logging-session": true,
    "ss-enabled": true,
    "f-args": "8"
    },
  "pool4": [
    {
    "protocol": "TCP",
    "prefix": "12.15.0.0/16",
    "port range": "1-65535"
    }, {
    "protocol": "UDP",
    "prefix": "12.15.0.0/16",
    "port range": "1-65535"
    }, {
    "protocol": "ICMP",
    "prefix": "12.15.0.0/16"
    }
  ]
}
```

Konfigurace je krátká a jednoduchá. *Pool6* je síť, za kterou se nám v naší IPv6 only síti maskuje IPv4 svět. Musí být nastavena stejná ve všech službách (routování, jool, dns64…). Je to jedna ze sítí, které jsou vyhrazeny pro použití v NAT64. Není používaná v internetu a můžete ji použít pro každou instalaci NAT64.

Velice podstatný je také parametr *f-args*, je to nastavení hashovacího algoritmu, který vybírá z IPv4 poolu adresu pro odchozí spojení. Hodnota 8 (pozor výchozí je 11) nám zajistí, že klient je v internetu maskován za jednu veřejnou IPv4 adresu po celou dobu provozu. Je to užitečné hlavně kvůli službám Video On Demand, kde se klientovi nesmí měnit veřejná IP adresa, ze které na službu přistupuje. Pěkně se o tom zmiňuje Ondra Caletka ve své přednášce na CSNOG 2024.

Pool4 - jool neumí do IPv4 překládat všechny protokoly, umí jen TCP, UDP a ICMP. Pro každý z nich se překladový IPv4 pool a rozsah portů nastavuje samostatně. Já jsem tuto možnost nepotřeboval, všude mám jen jeden pool.

Synchronizace Jool

Je to už součást HA řešení, (pokud HA nepoužijete, tuto část přeskočte), předává dalším strojům informace z překladové tabulky. Každý záznam v tabulce vyvolá synchronizační paket. Musí se spustit pomocný joold jako démon, jádro totiž nemůže navazovat síťová spojení. Informace se tak předá démonu a ten to multicastem pošle na další stroje. Multicastová adresa je částečně volitelná, musí zůstat začátek ff08::/16, ale konec je volitelný, stejně jako UDP port.

/etc/jool/netsocket.json

```
{"multicast address": "ff08::64",
"multicast port": "6464",
"in interface": "eth0",
"out interface": "eth0",
"reuseaddr": 1,
"ttl": 3}
```

/etc/jool/modsocket.json

```
{"instance": "default"}
```

Pozor na hodnotu instance, musí odpovídat instanci z `jool.conf` a také tam musí být zapnuto *ss-enabled*.

/etc/systemd/system/joold.service

```
[Unit]
Description=Jool synchronization daemon.
After=network.target
ConditionPathExists=/etc/jool/netsocket.json

[Service]
ExecStart=/usr/bin/joold /etc/jool/netsocket.json
ExecStop=/bin/kill $MAINPID
KillMode=process

[Install]
WantedBy=multi-user.target
```

Služba do systemd byla vyrobena klasickým způsobem `systemctl edit --full --force joold.service`.

DNS64

Démon Bind9 překlad DNS64 normálně umí. Konfigurace je hodně jednoduchá, vezměte klasickou konfiguraci pro rekurzivní resolver a přidejte tohle.

/etc/bind/named.conf

```
acl eduroam-klienti {
  2000:1000:a::/64;
  2000:1000:b::/64;
  };

acl nat64-prekladac {
  127.0.0.1;
  ::1/128;
  2000:1000::/112;
  };

match-clients {
  2000:1000:a::/64;
  2000:1000:b::/64;
  };
dnssec-validation auto;

dns64 64:ff9b::/96 {
  clients { !nat64-prekladac; eduroam-klienti; };
  };

zone "." {
  type hint;
  file "/usr/share/dns/root.hints";
  };
```

Při konfiguraci si dejte pozor na překladový IPv6 pool v sekci DNS64 – musí být stejný jako máme v konfiguraci sítě a joolu. Adresy serveru musí být z funkce DNS64 vyjmuty.

V dalším článku si probereme funkci CLAT, DHCPv6 a také logování takto upraveného provozu.

(Autorem obrázků je Jan Vondráček.)

Softwarová sklizeň (6. 11. 2024): mrkněte na předpověď počasí

Vstoupit do diskuse (14 názorů)

Byl pro vás článek přínosný?

+31

Autor článku

Jan Vondráček

V současné době pracuje jako správce linuxových systémů na Univerzitě Pardubice.

Témata:

- IPv6

tak db8 je "di b(i) ejt" - tedy debate, debatovat...6. 11. 2024, 15:51 editováno autorem komentáře

PavelM