Programování aplikací pro IPv6, IPv4.5 i IPv10

Ondřej Caletka

4. června 2021

TL;DR

- nepoužívejte přímo IP adresy
- pozor na **porovnávání retězců s IP adresami**
- používejte *vysokoúrovňová API*, jsou-li k dispozici
- každé použití BSD soketů by mělo začít funkcí `getaddrinfo(3)`
- serverové služby mají být schopny **používat zároveň několik soketů**
- kompatibilita soketů pro IPv6 s IPv4 **má být vypnuta**

cesnet

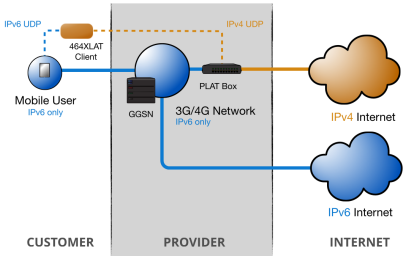
Problém

- svět byl *jednoduchý*, když bylo jen IPv4
- pro nový protokol je třeba **aplikace přepsat**
- prostá náhrada IPv4 za IPv6 **zruší podporu pro IPv4**
- někdo má *jen* IPv4, někdo *i* IPv6, někdo *jen* IPv6 a IPv4 jako službu, někdo má jeden z protokolů *rozbitý*
- aplikace by měla fungovat ve všech kombinacích *bez konfigurace či rekompilace*
- aplikace by měla fungovat bez dalších úprav i s **případným novým protokolem**

cesnet

„Nevadí, naše servery běží jen na IPv4“

- to ale **neznamená**, že všichni klienti také používají IPv4
- sítě s NAT64/DNS64 **jsou realitou**, zejména u mobilních operátorů
- aplikaci bez podpory IPv6-only **nelze publikovat v App store** pro iOS
- koncept 464XLAT, kterým podobné problémy *obchází* Android, **není k dispozici** u ostatních operačních systémů



cesnet

Nepoužívejte přímo IP adresy

Použití nip.io (*jako poslední záchrana*)

```
$ host 1.0.0.1.nip.io
1.0.0.1.nip.io has address 1.0.0.1
1.0.0.1.nip.io has IPv6 address 64:ff9b::100:1
```

Nikdy neporovnávejte uživatelem dodané řetězce s IP adresami

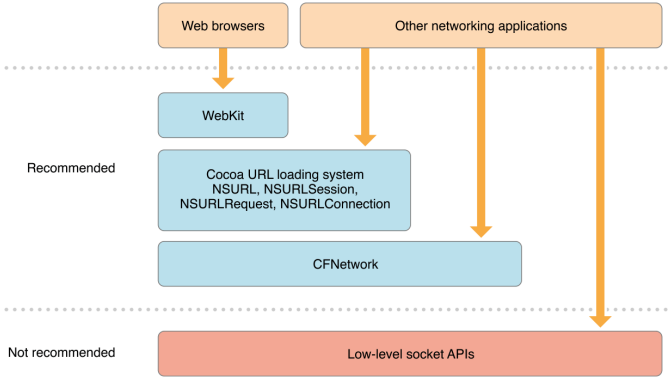
- existuje **víc různých řetězců** reprezentujících stejnou adresu
- časté chyby v knihovnách (např. CVE-2021-29921)

```
8.8.4.4 == 8.8.1028 == 8.525316 == 010.8.04.4
64:ff9b::100:1 == 64:FF9B:0::0:1.0.0.1
```

cesnet

Používejte vysokoúrovňové funkce

- odsouvají problematiku IPv6 do kategorie: „*implementační detail*“
- Cocoa API na Apple, `HttpsURLConnection` na Android
- `libcurl` jako přenositelné řešení



Zdroj: Apple Developer: Supporting IPv6 DNS64/NAT64 Networks

BSD sockets

Server

`socket(2)`	vytvoří soket pro daný protokol
`bind(2)`	připojí soket k dané adrese a portu
`listen(2)`	začne poslouchat příchozí spojení
`accept(2)`	vyrobí připojený soket pro spojení ve frontě
`send(2)`	posílá data
`recv(2)`	přijímá data

Klient

`socket(2)`	vytvoří soket pro daný protokol
`connect(2)`	naváže spojení s danou adresou
`send(2)`	posílá data
`recv(2)`	přijímá data

cesnet

Správné použití BSD sockets

- textové řetězce adresy a portu předáme funkci `getaddrinfo(3)`
- získáme seznam parametrů soketů seřazený podle preference
- jako klient **zkoušíme jeden po druhém**
- jako server **posloucháme na všech**
- adresu připojeného soketu zjišťujeme funkcí `getnameinfo(3)`

Zastaralé funkce

- `gethostbyname(3)`
- `inet_aton(3)`
- `inet_ntoa(3)`
- `gethostbyname2(3)`
- `gethostbyaddr(3)`
- `inet_pton(3)`
- `inet_ntop(3)`

cesnet

Zpětná kompatibilita soketů pro IPv6

- soket pro IPv6 může *za jistých okolností* pracovat i s provozem IPv4
- provoz IPv4 se mapuje pomocí IPv6 adres `::ffff:0.0.0.0/96`
- nebezpečí **zneužití pro odrazy z IPv6 do IPv4**
- kompatibilita nefunguje na systémech s **vypnutou podporou IPv6**
- správně napsaná aplikace používá samostatné sokety, **kompatibilitu vypíná**
- kompatibilita **se vypíná** aktivací volby soketu `IPV6_V6ONLY`
  - na *velmi starých* systémech volba neexistuje – kompatibilitu **nelze vypnout**
  - na Linuxu a macOS je kompatibilita ve výchozím stavu **zapnutá**
  - na Windows a BSD je kompatibilita ve výchozím stavu **vypnutá**
  - na OpenBSD je volba *read-only* – kompatibilitu **nelze zapnout**

cesnet

Happy Eyeballs

- paralelní navazování více spojení pro **skrytí problémů s nefunkčním IPv6**
- implementováno především ve webových prohlížečích
- funguje **jen na dual-stacku**
- funguje dobře jen s **krátkodobými spojeními** (např. HTTP)
- *noční můra* technické podpory provozovatele sítě

Happy Eyeballs 2.0 (RFC 8305)

- k dispozici ve **vysokoúrovňových funkcích** macOS a iOS
- větší náskok pro IPv6 za účelem **úspory spojení přes CGN**
- funkčnost i na **IPv6-only sítích** s NAT64/DNS64

cesnet

Podpora pro NAT64/DNS64 v macOS a iOS

```
>>> import socket
>>> socket.getaddrinfo("1.1.1.1", "https", type=socket.SOCK_STREAM)
[(<AddressFamily.AF_INET: 2>, <SocketKind.SOCK_STREAM: 1>, 6, '',
                                                    ('1.1.1.1', 443))]
>>> import urllib.request; urllib.request.getproxies()
{}
>>> socket.getaddrinfo("1.1.1.1", "https", type=socket.SOCK_STREAM)
[(<AddressFamily.AF_INET6: 30>, <SocketKind.SOCK_STREAM: 1>, 6, '',
                                    ('64:ff9b::101:101', 443, 0, 0))]
```

curl 7.77.0 na macOS

```
$ curl -I -v https://1.1.1.1
*   Trying 64:ff9b::101:101:443...
* Connected to 1.1.1.1 (64:ff9b::101:101) port 443 (#0)
…
*  SSL certificate verify ok.
```

Noví programátoři opakují staré chyby

Nejčastější omyly

- IPv4 je *vždy* k dispozici
- soket pro IPv6 *vždy* podporuje i IPv4
- server obsluhuje *jen jeden* poslouchající soket
- server běžící *jen* na IPv4 znamená klienty připojící se *jen* přes IPv4

Python http.server

- až do Python 3.7 IPv4-only
- od Python 3.8 IPv6-first
- stále používá **jen jeden poslouchající soket**

cesnet

Děkuji za pozornost

Ondřej Caletka
Ondrej.Caletka@ripe.net
https://**Ondřej.Caletka.cz**

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

cesnet

Prezentace je již nyní k dispozici ke stažení.